Hewlett Packard Enterprise

Operations Guide

This guide provides a list of useful procedures for managing your HPE Helion OpenStack 8 cloud. The audience is the admin-level operator of the cloud.

Contents

4 Managing Identity **49**

1 Operations Overview

A high-level overview of the processes related to operating a HPE Helion OpenStack 8 cloud.

1.1 What is a cloud operator?

When we talk about a cloud operator it is important to understand the scope of the tasks and responsibilities we are referring to. HPE Helion OpenStack defines a cloud operator as the person or group of people who will be administering the cloud infrastructure, which includes:

- Monitoring the cloud infrastructure, resolving issues as they arise.

- Managing hardware resources, adding/removing hardware due to capacity needs.

- Repairing, and recovering if needed, any hardware issues.

- Performing domain administration tasks, which involves creating and managing projects, users, and groups as well as setting and managing resource quotas.

1.2 Tools provided to operate your cloud

HPE Helion OpenStack provides the following tools which are available to operate your cloud:

Operations Console

Often referred to as the Ops Console, you can use this console to view data about your cloud infrastructure in a web-based graphical user interface (GUI) to make sure your cloud is operating correctly. By logging on to the console, HPE Helion OpenStack administrators can manage data in the following ways:

- Triage alarm notifications in the central dashboard

- Monitor the environment by giving priority to alarms that take precedence

- Manage compute nodes and easily use a form to create a new host

- Refine the monitoring environment by creating new alarms to specify a combination of metrics, services, and hosts that match the triggers unique to an environment

- Plan for future storage by tracking capacity over time to predict with some degree of reliability the amount of additional storage needed

For more details on how to connect to and use the Operations Console, see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.1 "Operations Console Overview"*.

Dashboard

Often referred to as Horizon or the Horizon dashboard, you can use this console to manage resources on a domain and project level in a web-based graphical user interface (GUI). The following are some of the typical operational tasks that you may perform using the dashboard:

- Creating and managing projects, users, and groups within your domain.

- Assigning roles to users and groups to manage access to resources.

- Setting and updating resource quotas for the projects.

For more details, see the following pages:

- *Section 4.3, "Understanding Domains, Projects, Users, Groups, and Roles"*

- *Book "User Guide", Chapter 3 "Cloud Admin Actions with the Dashboard"*

Command-line interface (CLI)

Each service within HPE Helion OpenStack provides a command-line client, such as the novaclient (sometimes referred to as the python-novaclient or nova CLI) for the Compute service, the keystoneclient for the Identity service, etc. There is also an effort in the OpenStack community to make a unified client, called the openstackclient, which will combine the available commands in the various service-specific clients into one tool. By default, we install each of the necessary clients onto the hosts in your environment for you to use.

You will find processes defined in our documentation that use these command-line tools. There is also a list of common cloud administration tasks which we have outlined which you can use the command-line tools to do. For more details, see *Book "User Guide", Chapter 4 "Cloud Admin Actions with the Command Line"*.

There are references throughout the SUSE OpenStack Cloud documentation to the HPE Smart Storage Administrator (HPE SSA) CLI. HPE-specific binaries that are not based on open source are distributed directly from and supported by HPE. To download and install the SSACLI utility, please refer to: https://support.hpe.com/hpsc/swd/public/detail?swItemId=MTX_3d16386b418a443388c18da82f ↗

1.3   Daily tasks

- **Ensure your cloud is running correctly**: HPE Helion OpenStack is deployed as a set of highly available services to minimize the impact of failures. That said, hardware and software systems can fail. Detection of failures early in the process will enable you to address issues before they affect the broader system. HPE Helion OpenStack provides a monitoring solution, based on OpenStack's Monasca, which provides monitoring and metrics for all OpenStack components and much of the underlying system, including service status, performance metrics, compute node, and virtual machine status. Failures are exposed via the Operations Console and/or alarm notifications. In the case where more detailed diagnostics are required, you can use a centralized logging system based on the Elasticsearch, Logstash, and Kibana (ELK) stack. This provides the ability to search service logs to get detailed information on behavior and errors.

- **Perform critical maintenance**: To ensure your OpenStack installation is running correctly, provides the right access and functionality, and is secure, you should make ongoing adjustments to the environment. Examples of daily maintenance tasks include:

  - Add/remove projects and users. The frequency of this task depends on your policy.

  - Apply security patches (if released).

  - Run daily backups.

1.4   Weekly or monthly tasks

- **Do regular capacity planning**: Your initial deployment will likely reflect the known near to mid-term scale requirements, but at some point your needs will outgrow your initial deployment's capacity. You can expand HPE Helion OpenStack in a variety of ways, such as by adding compute and storage capacity.

To manage your cloud's capacity, begin by determining the load on the existing system. OpenStack is a set of relatively independent components and services, so there are multiple subsystems that can affect capacity. These include control plane nodes, compute nodes, object storage nodes, block storage nodes, and an image management system. At the most basic level, you should look at the CPU used, RAM used, I/O load, and the disk space used relative to the amounts available. For compute nodes, you can also evaluate the allocation of resource to

hosted virtual machines. This information can be viewed in the Operations Console. You can pull historical information from the monitoring service (OpenStack's Monasca) by using its client or API. Also, OpenStack provides you some ability to manage the hosted resource utilization by using quotas for projects. You can track this usage over time to get your growth trend so that you can project when you will need to add capacity.

1.5   Semi-annual tasks

- **Perform upgrades**: OpenStack releases new versions on a six-month cycle. In general, HPE Helion OpenStack will release new major versions annually with minor versions and maintenance updates more often. Each new release consists of both new functionality and services, as well as bug fixes for existing functionality.

☞ Note

If you are planning to upgrade, this is also an excellent time to evaluate your existing capabilities, especially in terms of capacity (see Capacity Planning above).

1.6   Troubleshooting

As part of managing your cloud, you should be ready to troubleshoot issues, as needed. The following are some common troubleshooting scenarios and solutions:

How do I determine if my cloud is operating correctly now?: HPE Helion OpenStack provides a monitoring solution based on OpenStack's Monasca service. This service provides monitoring and metrics for all OpenStack components, as well as much of the underlying system. By default, HPE Helion OpenStack comes with a set of alarms that provide coverage of the primary systems. In addition, you can define alarms based on threshold values for any metrics defined in the system. You can view alarm information in the Operations Console. You can also receive or deliver this information to others by configuring email or other mechanisms. Alarms provide information about whether a component failed and is affecting the system, and also what condition triggered the alarm.

How do I troubleshoot and resolve performance issues for my cloud?: There are a variety of factors that can affect the performance of a cloud system, such as the following:

- Health of the control plane

- Health of the hosting compute node and virtualization layer

- Resource allocation on the compute node

If your cloud users are experiencing performance issues on your cloud, use the following approach:

1. View the compute summary page on the Operations Console to determine if any alarms have been triggered.

2. Determine the hosting node of the virtual machine that is having issues.

3. On the compute hosts page, view the status and resource utilization of the compute node to determine if it has errors or is over-allocated.

4. On the compute instances page you can view the status of the VM along with its metrics.

How do I troubleshoot and resolve availability issues for my cloud?: If your cloud users are experiencing availability issues, determine what your users are experiencing that indicates to them the cloud is down. For example, can they not access the Dashboard service (Horizon) console or APIs, indicating a problem with the control plane? Or are they having trouble accessing resources? Console/API issues would indicate a problem with the control planes. Use the Operations Console to view the status of services to see if there is an issue. However, if it is an issue of accessing a virtual machine, then also search the consolidated logs that are available in the ELK stack or errors related to the virtual machine and supporting networking.

1.7   Common Questions

To manage a cloud, how many administrators do I need?

A 24x7 cloud needs a 24x7 cloud operations team. If you already have a NOC, managing the cloud can be added to their workload.

A cloud with 20 nodes will need a part-time person. You can manage a cloud with 200 nodes with two people. As the amount of nodes increases and processes and automation are put in place, you will need to increase the number of administrators but the need is not linear. As an example, if you have 3000 nodes and 15 clouds you will probably need 6 administrators.

What skills do my cloud administrators need?

Your administrators should be experienced Linux admins. They should have experience in application management, as well as experience with Ansible. It is a plus if they have experience with Bash shell scripting and Python programming skills.

In addition, you will need networking engineers. A 3000 node environment will need two networking engineers.

What operations should I plan on performing daily, weekly, monthly, or semi-annually?

You should plan for operations by understanding what tasks you need to do daily, weekly, monthly, or semi-annually. The specific list of tasks that you need to perform depends on your cloud configuration, but should include the following high-level tasks specified in the *Chapter 2, Tutorials*

2 Tutorials

This section contains tutorials for common tasks for your HPE Helion OpenStack 8 cloud.

2.1 HPE Helion OpenStack Quickstart Guide

2.1.1 Introduction

This document provides simplified instructions for installing and setting up a HPE Helion OpenStack. Use this quickstart guide to build testing, demonstration, and lab-type environments., rather than production installations. When you complete this quickstart process, you will have a fully functioning HPE Helion OpenStack demo environment.

🎓 Note

These simplified instructions are intended for testing or demonstration. Instructions for production installations are in *Book "*Installing with Cloud Lifecycle Manager*".*

2.1.2 Overview of components

The following are short descriptions of the components that HPE Helion OpenStack employs when installing and deploying your cloud.

Ansible. Ansible is a powerful configuration management tool used by HPE Helion OpenStack to manage nearly all aspects of your cloud infrastructure. Most commands in this quickstart guide execute Ansible scripts, known as playbooks. You will run playbooks that install packages, edit configuration files, manage network settings, and take care of the general administration tasks required to get your cloud up and running.

Get more information on Ansible at https://www.ansible.com/ ↗.

Cobbler. Cobbler is another third-party tool used by HPE Helion OpenStack to deploy operating systems across the physical servers that make up your cloud. Find more info at http://cobbler.github.io/ ↗.

Git. Git is the version control system used to manage the configuration files that define your cloud. Any changes made to your cloud configuration files must be committed to the locally hosted git repository to take effect. Read more information on Git at https://git-scm.com/ ↗ .

2.1.3 Preparation

Successfully deploying a HPE Helion OpenStack environment is a large endeavor, but it is not complicated. For a successful deployment, you must put a number of components in place before rolling out your cloud. Most importantly, a basic HPE Helion OpenStack requires the proper network infrastrucure. Because HPE Helion OpenStack segregates the network traffic of many of its elements, if the necessary networks, routes, and firewall access rules are not in place, communication required for a successful deployment will not occur.

2.1.4 Getting Started

When your network infrastructure is in place, go ahead and set up the Cloud Lifecycle Manager. This is the server that will orchestrate the deployment of the rest of your cloud. It is also the server you will run most of your deployment and management commands on.

Set up the Cloud Lifecycle Manager

1. **Download the installation media**
   Obtain a copy of the HPE Helion OpenStack installation media, and make sure that it is accessible by the server that you are installing it on. Your method of doing this may vary. For instance, some may choose to load the installation ISO on a USB drive and physically attach it to the server, while others may run the IPMI Remote Console and attach the ISO to a virtual disc drive.

2. **Install the operating system**

   a. Boot your server, using the installation media as the boot source.

   b. Choose "install" from the list of options and choose your preferred keyboard layout, location, language, and other settings.

   c. Set the address, netmask, and gateway for the primary network interface.

   d. Create a root user account.

Proceed with the OS installation. After the installation is complete and the server has rebooted into the new OS, log in with the user account you created.

3. **Configure the new server**

   a. SSH to your new server, and set a valid DNS nameserver in the `/etc/resolv.conf` file.

   b. Set the environment variable `LC_ALL`:

   ```
   export LC_ALL=C
   ```

   You now have a server running SUSE Linux Enterprise Server (SLES). The next step is to configure this machine as a Cloud Lifecycle Manager.

4. **Configure the Cloud Lifecycle Manager**
   The installation media you used to install the OS on the server also has the files that will configure your cloud. You need to mount this installation media on your new server in order to use these files.

   a. Using the URL that you obtained the HPE Helion OpenStack installation media from, run `wget` to download the ISO file to your server:

   ```
   wget INSTALLATION_ISO_URL
   ```

   b. Now mount the ISO in the `/media/cdrom/` directory

   ```
   sudo mount INSTALLATION_ISO /media/cdrom/
   ```

   c. Unpack the tar file found in the `/media/cdrom/ardana/` directory where you just mounted the ISO:

   ```
   tar xvf /media/cdrom/ardana/ardana-x.x.x-x.tar
   ```

   d. Now you will install and configure all the components needed to turn this server into a Cloud Lifecycle Manager. Run the `ardana-init.bash` script from the uncompressed tar file:

   ```
   ~/ardana-x.x.x/ardana-init.bash
   ```

The `ardana-init.bash` script prompts you to enter an optional SSH passphrase. This passphrase protects the RSA key used to SSH to the other cloud nodes. This is an optional passphrase, and you can skip it by pressing Enter at the prompt.

The `ardana-init.bash` script automatically installs and configures everything needed to set up this server as the lifecycle manager for your cloud.

When the script has finished running, you can proceed to the next step, editing your input files.

5. **Edit your input files**

Your HPE Helion OpenStack input files are where you define your cloud infrastructure and how it runs. The input files define options such as which servers are included in your cloud, the type of disks the servers use, and their network configuration. The input files also define which services your cloud will provide and use, the network architecture, and the storage backends for your cloud.

There are several example configurations, which you can find on your Cloud Lifecycle Manager in the `~/openstack/examples/` directory.

a. The simplest way to set up your cloud is to copy the contents of one of these example configurations to your `~/openstack/mycloud/definition/` directory. You can then edit the copied files and define your cloud.

```
cp -r ~/openstack/examples/CHOSEN_EXAMPLE/* ~/openstack/my_cloud/definition/
```

b. Edit the files in your `~/openstack/my_cloud/definition/` directory to define your cloud.

6. **Commit your changes**

When you finish editing the necessary input files, stage them, and then commit the changes to the local Git repository:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My commit message"
```

7. **Image your servers**

Now that you have finished editing your input files, you can deploy the configuration to the servers that will comprise your cloud.

a. Image the servers. You will install the SLES operating system across all the servers in your cloud, using Ansible playbooks to trigger the process.

b. The following playbook confirms that your servers are accessible over their IPMI ports, which is a prerequisite for the imaging process:

```
ansible-playbook -i hosts/localhost bm-power-status.yml
```

c. Now validate that your cloud configuration files have proper YAML syntax by running the `config-processor-run.yml` playbook:

```
ansible-playbook -i hosts/localhost config-processor-run.yml
```

If you receive an error when running the preceeding playbook, one or more of your configuration files has an issue. Refer to the output of the Ansible playbook, and look for clues in the Ansible log file, found at `~/.ansible/ansible.log`.

d. The next step is to prepare your imaging system, Cobbler, to deploy operating systems to all your cloud nodes:

```
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

e. Now you can image your cloud nodes. You will use an Ansible playbook to trigger Cobbler to deploy operating systems to all the nodes you specified in your input files:

```
ansible-playbook -i hosts/localhost bm-reimage.yml
```

The `bm-reimage.yml` playbook performs the following operations:

1. Powers down the servers.

2. Sets the servers to boot from a network interface.

3. Powers on the servers and performs a PXE OS installation.

4. Waits for the servers to power themselves down as part of a successful OS installation. This can take some time.

5. Sets the servers to boot from their local hard disks and powers on the servers.

6. Waits for the SSH service to start on the servers and verifies that they have the expected host-key signature.

8. **Deploy your cloud**

   Now that your servers are running the SLES operating system, it is time to configure them for the roles they will play in your new cloud.

   a. Prepare the Cloud Lifecycle Manager to deploy your cloud configuration to all the nodes:

   ```
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

   NOTE: The preceding playbook creates a new directory, `~/scratch/ansible/next/ardana/ansible/`, from which you will run many of the following commands.

   b. *(Optional)* If you are reusing servers or disks to run your cloud, you can wipe the disks of your newly imaged servers by running the `wipe_disks.yml` playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible/
   ansible-playbook -i hosts/verb_hosts wipe_disks.yml
   ```

   The `wipe_disks.yml` playbook removes any existing data from the drives on your new servers. This can be helpful if you are reusing servers or disks. This action will not affect the OS partitions on the servers.

   ☞ Note

   The `wipe_disks.yml` playbook is only meant to be run on systems immediately after running `bm-reimage.yml`. If used for any other case, it may not wipe all of the expected partitions. For example, if `site.yml` fails, you cannot start fresh by running `wipe_disks.yml`. You must `bm-reimage` the node first and then run `wipe_disks`.

   c. Now it is time to deploy your cloud. Do this by running the `site.yml` playbook, which pushes the configuration you defined in the input files out to all the servers that will host your cloud.

   ```
   cd ~/scratch/ansible/next/ardana/ansible/
   ansible-playbook -i hosts/verb_hosts site.yml
   ```

The `site.yml` playbook installs packages, starts services, configures network interface settings, sets iptables firewall rules, and more. Upon successful completion of this playbook, your HPE Helion OpenStack will be in place and in a running state. This playbook can take up to six hours to complete.

9. **SSH to your nodes**

Now that you have successfully run `site.yml`, your cloud will be up and running. You can verify connectivity to your nodes by connecting to each one by using SSH. You can find the IP addresses of your nodes by viewing the `/etc/hosts` file.

For security reasons, you can only SSH to your nodes from the Cloud Lifecycle Manager. SSH connections from any machine other than the Cloud Lifecycle Manager will be refused by the nodes.

From the Cloud Lifecycle Manager, SSH to your nodes:

```
ssh <management IP address of node>
```

Also note that SSH is limited to your cloud's management network. Each node has an address on the management network, and you can find this address by reading the `/etc/hosts` or `server_info.yml` file.

2.2 Log Management and Integration

2.2.1 Overview

HPE Helion OpenStack uses the ELK (Elasticsearch, Logstash, Kibana) stack for log management across the entire cloud infrastructure. This configuration facilitates simple administration as well as integration with third-party tools. This tutorial covers how to forward your logs to a third-party tool or service, and how to access and search the Elasticsearch log stores through API endpoints.

2.2.2　The ELK stack

The ELK logging stack consists of the Elasticsearch, Logstash, and Kibana elements:

- **Logstash.** Logstash reads the log data from the services running on your servers, and then aggregates and ships that data to a storage location. By default, Logstash sends the data to the Elasticsearch indexes, but it can also be configured to send data to other storage and indexing tools such as Splunk.

- **Elasticsearch.** Elasticsearch is the storage and indexing component of the ELK stack. It stores and indexes the data received from Logstash. Indexing makes your log data searchable by tools designed for querying and analyzing massive sets of data. You can query the Elasticsearch datasets from the built-in Kibana console, a third-party data analysis tool, or through the Elasticsearch API (covered later).

- **Kibana.** Kibana provides a simple and easy-to-use method for searching, analyzing, and visualizing the log data stored in the Elasticsearch indexes. You can customize the Kibana console to provide graphs, charts, and other visualizations of your log data.

2.2.3　Using the Elasticsearch API

You can query the Elasticsearch indexes through various language-specific APIs, as well as directly over the IP address and port that Elasticsearch exposes on your implementation. By default, Elasticsearch presents from localhost, port 9200. You can run queries directly from a terminal using `curl`. For example:

```
curl -XGET 'http://localhost:9200/_search?q=tag:yourSearchTag'
```

The preceding command searches all indexes for all data with the "yourSearchTag" tag.

You can also use the Elasticsearch API from outside the logging node. This method connects over the Kibana VIP address, port 5601, using basic http authentication. For example, you can use the following command to perform the same search as the preceding search:

```
curl -u kibana:<password> kibana_vip:5601/_search?q=tag:yourSearchTag
```

You can further refine your search to a specific index of data, in this case the "elasticsearch" index:

```
curl -XGET 'http://localhost:9200/elasticsearch/_search?q=tag:yourSearchTag'
```

The search API is RESTful, so responses are provided in JSON format. Here's a sample (though empty) response:

```
{
    "took":13,
    "timed_out":false,
    "_shards":{
        "total":45,
        "successful":45,
        "failed":0
    },
    "hits":{
        "total":0,
        "max_score":null,
        "hits":[]
    }
}
```

2.2.4   For More Information

You can find more detailed Elasticsearch API documentation at https://www.elastic.co/guide/en/elasticsearch/reference/current/search.html ↗.

Review the Elasticsearch Python API documentation at the following sources: http://elasticsearch-py.readthedocs.io/en/master/api.html ↗

Read the Elasticsearch Java API documentation at https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/index.html ↗.

2.2.5   Forwarding your logs

You can configure Logstash to ship your logs to an outside storage and indexing system, such as Splunk. Setting up this configuration is as simple as editing a few configuration files, and then running the Ansible playbooks that implement the changes. Here are the steps.

1. Begin by logging in to the Cloud Lifecycle Manager.

2. Verify that the logging system is up and running:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts logging-status.yml
```

When the preceding playbook completes without error, proceed to the next step.

3. Edit the Logstash configuration file, found at the following location:

```
~/openstack/ardana/ansible/roles/logging-server/templates/logstash.conf.j2
```

Near the end of the Logstash configuration file, you will find a section for configuring Logstash output destinations. The following example demonstrates the changes necessary to forward your logs to an outside server (changes in bold). The configuration block sets up a TCP connection to the destination server's IP address over port 5514.

```
# Logstash outputs
    output {
      # Configure Elasticsearch output
      # http://www.elastic.co/guide/en/logstash/current/plugins-outputs-
elasticsearch.html
      elasticsearch {
        index => "%{[@metadata][es_index]}
        hosts => ["{{ elasticsearch_http_host }}:{{ elasticsearch_http_port }}"]
        flush_size => {{ logstash_flush_size }}
        idle_flush_time => 5
        workers => {{ logstash_threads }}
      }
      # Forward Logs to Splunk on TCP port 5514 which matches the one specified in
 Splunk Web UI.
      tcp {
        mode => "client"
        host => "<Enter Destination listener IP address>"
        port => 5514
      }
    }
```

Note that Logstash can forward log data to multiple sources, so there is no need to remove or alter the Elasticsearch section in the preceding file. However, if you choose to stop forwarding your log data to Elasticsearch, you can do so by removing the related section in this file, and then continue with the following steps.

4. Commit your changes to the local git repository:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "Your commit message"
```

5. Run the configuration processor to check the status of all configuration files:

```
ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Run the ready-deployment playbook:

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Implement the changes to the Logstash configuration file:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts logging-server-configure.yml
```

Please note that configuring the receiving service will vary from product to product. Consult the documentation for your particular product for instructions on how to set it up to receive log files from Logstash.

2.3   Integrating Your Logs with Splunk

2.3.1   Integrating with Splunk

The HPE Helion OpenStack 8 logging solution provides a flexible and extensible framework to centralize the collection and processing of logs from all nodes in your cloud. The logs are shipped to a highly available and fault-tolerant cluster where they are transformed and stored for better searching and reporting. The HPE Helion OpenStack 8 logging solution uses the ELK stack (Elasticsearch, Logstash and Kibana) as a production-grade implementation and can support other storage and indexing technologies.

You can configure Logstash, the service that aggregates and forwards the logs to a searchable index, to send the logs to a third-party target, such as Splunk.

For how to integrate the HPE Helion OpenStack 8 centralized logging solution with Splunk, including the steps to set up and forward logs, please refer to *Section 3.1, "Splunk Integration"*.

2.4 Integrating HPE Helion OpenStack with an LDAP System

You can configure your HPE Helion OpenStack cloud to work with an outside user authentication source such as Active Directory or OpenLDAP. Keystone, the HPE Helion OpenStack identity service, functions as the first stop for any user authorization/authentication requests. Keystone can also function as a proxy for user account authentication, passing along authentication and authorization requests to any LDAP-enabled system that has been configured as an outside source. This type of integration lets you use an existing user-management system such as Active Directory and its powerful group-based organization features as a source for permissions in HPE Helion OpenStack.

Upon successful completion of this tutorial, your cloud will refer user authentication requests to an outside LDAP-enabled directory system, such as Microsoft Active Directory or OpenLDAP.

2.4.1 Configure your LDAP source

To configure your HPE Helion OpenStack cloud to use an outside user-management source, perform the following steps:

1. Make sure that the LDAP-enabled system you plan to integrate with is up and running and accessible over the necessary ports from your cloud management network.

2. Edit the `/var/lib/ardana/openstack/my_cloud/config/keystone/keystone.conf.j2` file and set the following options:

   ```
   domain_specific_drivers_enabled = True
   domain_configurations_from_database = False
   ```

3. Create a YAML file in the `/var/lib/ardana/openstack/my_cloud/config/keystone/` directory that defines your LDAP connection. You can make a copy of the sample Keystone-LDAP configuration file, and then edit that file with the details of your LDAP connection. The following example copies the `keystone_configure_ldap_sample.yml` file and names the new file `keystone_configure_ldap_my.yml`:

   ```
   ardana > cp /var/lib/ardana/openstack/my_cloud/config/keystone/
   keystone_configure_ldap_sample.yml \
     /var/lib/ardana/openstack/my_cloud/config/keystone/keystone_configure_ldap_my.yml
   ```

4. Edit the new file to define the connection to your LDAP source. This guide does not provide comprehensive information on all aspects of the `keystone_configure_ldap.yml` file. Find a complete list of Keystone/LDAP configuration file options at: https://github.com/openstack/keystone/blob/stable/pike/etc/keystone.conf.sample ↗

The following file illustrates an example Keystone configuration that is customized for an Active Directory connection.

```
keystone_domainldap_conf:

    # CA certificates file content.
    # Certificates are stored in Base64 PEM format. This may be entire LDAP server
    # certificate (in case of self-signed certificates), certificate of authority
    # which issued LDAP server certificate, or a full certificate chain (Root CA
    # certificate, intermediate CA certificate(s), issuer certificate).
    #
    cert_settings:
      cacert: |
        -----BEGIN CERTIFICATE-----

        certificate appears here

        -----END CERTIFICATE-----

    # A domain will be created in MariaDB with this name, and associated with ldap
 back end.
    # Installer will also generate a config file named /etc/keystone/domains/
keystone.<domain_name>.conf
    #
    domain_settings:
      name: ad
      description: Dedicated domain for ad users

    conf_settings:
      identity:
         driver: ldap


      # For a full list and description of ldap configuration options, please refer
 to
      # http://docs.openstack.org/liberty/config-reference/content/keystone-
configuration-file.html.
      #
      # Please note:
      #  1. LDAP configuration is read-only. Configuration which performs write
 operations (i.e. creates users, groups, etc)
```

```
        #       is not supported at the moment.
        #  2. LDAP is only supported for identity operations (reading users and groups
from LDAP). Assignment
        #       operations with LDAP (i.e. managing roles, projects) are not supported.
        #  3. LDAP is configured as non-default domain. Configuring LDAP as a default
domain is not supported.
        #

        ldap:
          url: ldap://YOUR_COMPANY_AD_URL
          suffix: YOUR_COMPANY_DC
          query_scope: sub
          user_tree_dn: CN=Users,YOUR_COMPANY_DC
          user : CN=admin,CN=Users,YOUR_COMPANY_DC
          password: REDACTED
          user_objectclass: user
          user_id_attribute: cn
          user_name_attribute: cn
          group_tree_dn: CN=Users,YOUR_COMPANY_DC
          group_objectclass: group
          group_id_attribute: cn
          group_name_attribute: cn
          use_pool: True
          user_enabled_attribute: userAccountControl
          user_enabled_mask: 2
          user_enabled_default: 512
          use_tls: True
          tls_req_cert: demand
          # if you are configuring multiple LDAP domains, and LDAP server certificates
are issued
          # by different authorities, make sure that you place certs for all the LDAP
backend domains in the
          # cacert parameter as seen in this sample yml file so that all the certs are
combined in a single CA file
          # and every LDAP domain configuration points to the combined CA file.
          # Note:
          # 1. Please be advised that every time a new ldap domain is configured, the
single CA file gets overwritten
          # and hence ensure that you place certs for all the LDAP backend domains in
the cacert parameter.
          # 2. There is a known issue on one cert per CA file per domain when the
system processes
          # concurrent requests to multiple LDAP domains. Using the single CA file
with all certs combined
          # shall get the system working properly.

          tls_cacertfile: /etc/keystone/ssl/certs/all_ldapdomains_ca.pem
```

5. Add your new file to the local Git repository and commit the changes.

```
ardana > cd ~/openstack
ardana > git checkout site
ardana > git add -A
ardana > git commit -m "Adding LDAP server integration config"
```

6. Run the configuration processor and deployment preparation playbooks to validate the YAML files and prepare the environment for configuration.

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Keystone reconfiguration playbook to implement your changes, passing the newly created YAML file as an argument to the `-e@FILE_PATH` parameter:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml \
  -e@/var/lib/ardana/openstack/my_cloud/config/keystone/keystone_configure_ldap_my.yml
```

To integrate your HPE Helion OpenStack cloud with multiple domains, repeat these steps starting from *Step 3* for each domain.

3 Third-Party Integrations

3.1 Splunk Integration

This documentation demonstrates the possible integration between the HPE Helion OpenStack 8 centralized logging solution and Splunk including the steps to set up and forward logs.

The HPE Helion OpenStack 8 logging solution provides a flexible and extensible framework to centralize the collection and processing of logs from all of the nodes in a cloud. The logs are shipped to a highly available and fault tolerant cluster where they are transformed and stored for better searching and reporting. The HPE Helion OpenStack 8 logging solution uses the ELK stack (Elasticsearch, Logstash and Kibana) as a production grade implementation and can support other storage and indexing technologies. The Logstash pipeline can be configured to forward the logs to an alternative target if you wish.

This documentation demonstrates the possible integration between the HPE Helion OpenStack 8 centralized logging solution and Splunk including the steps to set up and forward logs.

3.1.1 What is Splunk?

Splunk is software for searching, monitoring, and analyzing machine-generated big data (https://en.wikipedia.org/wiki/Machine-generated_data) , via a web-style interface. Splunk captures, indexes and correlates real-time data in a searchable repository from which it can generate graphs, reports, alerts, dashboards and visualizations. It is commercial software (unlike Elasticsearch) and more details about Splunk can be found at https://www.splunk.com .

3.1.2 Configuring Splunk to receive log messages from HPE Helion OpenStack 8

This documentation assumes that you already have Splunk set up and running. For help with installing and setting up Splunk, refer to Splunk Tutorial (http://docs.splunk.com/Documentation/Splunk/latest/SearchTutorial/Systemrequirements) .

There are different ways in which a log message (or "event" in Splunk's terminology) can be sent to Splunk. These steps will set up a TCP port where Splunk will listen for messages.

1. On the Splunk web UI, click on the Settings menu in the upper right-hand corner.

2. In the *Data* section of the Settings menu, click *Data Inputs*.

3. Choose the *TCP* option.

4. Click the *New* button to add an input.

5. In the *Port* field, enter the port number you want to use.

> ### 🎓 Note
>
> If you are on a less secure network and want to restrict connections to this port, use the *Only accept connection from* field to restrict the traffic to a specific IP address.

6. Click the *Next* button.

7. Specify the Source Type by clicking on the *Select* button and choosing `linux_messages_syslog` from the list.

8. Click the *Review* button.

9. Review the configuration and click the *Submit* button.

10. A success message will be displayed.

3.1.3   Forwarding log messages from HPE Helion OpenStack 8 Centralized Logging to Splunk

When you have Splunk set up and configured to receive log messages, you can configure HPE Helion OpenStack 8 to forward the logs to Splunk.

1. Log in to the Cloud Lifecycle Manager.

2. Check the status of the logging service:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts logging-status.yml
   ```

   If everything is up and running, continue to the next step.

3. Edit the logstash config file at the location below:

   ```
   ~/openstack/ardana/ansible/roles/logging-server/templates/logstash.conf.j2
   ```

At the bottom of the file will be a section for the Logstash outputs. Add details about your Splunk environment details.

Below is an example, showing the placement in bold:

```
# Logstash outputs
#-----------------------------------------------------------------------------
output {
  # Configure Elasticsearch output
  # http://www.elastic.co/guide/en/logstash/current/plugins-outputs-
elasticsearch.html
  elasticsearch {
    index => %{[@metadata][es_index]}
    hosts => ["{{ elasticsearch_http_host }}:{{ elasticsearch_http_port }}"]
    flush_size => {{ logstash_flush_size }}
    idle_flush_time => 5
    workers => {{ logstash_threads }}
  }
  # Forward Logs to Splunk on the TCP port that matches the one specified in Splunk
 Web UI.
 tcp {
   mode => "client"
   host => "<Enter Splunk listener IP address>"
   port => TCP_PORT_NUMBER
 }
}
```

📖 Note

> If you are not planning on using the Splunk UI to parse your centralized logs, there is no need to forward your logs to Elasticsearch. In this situation, comment out the lines in the Logstash outputs pertaining to Elasticsearch. However, you can continue to forward your centralized logs to multiple locations.

4. Commit your changes to git:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "Logstash configuration change for Splunk integration"
```

5. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Complete this change with a reconfigure of the logging environment:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts logging-configure.yml
```

8. In your Splunk UI, confirm that the logs have begun to forward.

3.1.4 Searching for log messages from the Spunk dashboard

To both verify that your integration worked and to search your log messages that have been forwarded you can navigate back to your Splunk dashboard. In the search field, use this string:

```
source="tcp:TCP_PORT_NUMBER"
```

Find information on using the Splunk search tool at http://docs.splunk.com/Documentation/Splunk/6.4.3/SearchTutorial/WelcometotheSearchTutorial ↗ .

3.2 Nagios Integration

HPE Helion OpenStack cloud operators that are using Nagios or Icinga-based monitoring systems may wish to integrate them with the built-in monitoring infrastructure of HPE Helion OpenStack. Integrating with the existing monitoring processes and procedures will reduce support overhead and avoid duplication. This document describes the different approaches that can be taken to create a well-integrated monitoring dashboard using both technologies.

🎓 Note

This document refers to Nagios but the proposals will work equally well with Icinga, Icinga2, or other Nagios clone monitoring systems.

3.2.1　HPE Helion OpenStack monitoring and reporting

HPE Helion OpenStack comes with a monitoring engine (Monasca) and a separate management dashboard (Operations Console). Monasca is extremely scalable, designed to cope with the constant change in monitoring sources and services found in a cloud environment. Monitoring agents running on hosts (physical and virtual) submit data to the Monasca message bus via a RESTful API. Threshold and notification engines then trigger alarms when predefined thresholds are passed. Notification methods are flexible and extensible. Typical examples of notification methods would be emails generated or creating alarms in PagerDuty.

While extensible, Monasca is largely focused on monitoring cloud infrastructures rather than traditional environments such as server hardware, network links, switches, etc. For more details about the monitoring service, see *Section 12.1, "Monitoring"*.

The Operations Console (Ops Console) provides cloud administrators a clear web interfaces to view alarm status, management alarm workflow, and configure alarms and thresholds. For more details about the Ops Console, see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.1 "Operations Console Overview"*.

3.2.2　Nagios monitoring and reporting

Nagios is an industry leading open source monitoring service with extensive plugins and agents. Nagios checks are either run directly from the monitoring server or run on a remote host via an agent and with results submitted back to the monitoring server. While Nagios has proven extremely flexible and scalable, it requires significant explicit configuration. Using Nagios to monitor guest virtual machines becomes more challenging because virtual machines can be ephemeral which means new virtual machines are created and destroyed regularly. Configuration automation (Chef, Puppet, Ansible etc) can create a more dynamic Nagios setup but they still require the Nagios service to be restarted every time a new host is added.

A key benefit of Nagios style monitoring is that it allows for HPE Helion OpenStack to be monitored externally, from a user or service perspective. For example, checks can be created to monitor availability of all the API endpoints from external locations or even to create and destroy instances to ensure the entire system is working as expected.

3.2.3   Adding Monasca

Many private cloud operators already have existing monitoring solutions such as Nagios and Icinga. We recommend that you extend your existing solutions into Monasca or forward Monasca alerts to your existing solution to maximize coverage and reduce risk.

3.2.4   Integration Approaches

Integration between Nagios and Monasca can occur at two levels, at the individual check level or at the management interfaces. Both options are discussed in the following sections.

Running Nagios-style checks in the Monasca agents

The Monasca agent is installed on all HPE Helion OpenStack servers and includes the ability to execute Nagios-style plugins as well as its own plugin scripts. For this configuration check, plugins need to be installed on the required server then added to the Monasca configuration under `/etc/monasca/agent/conf.d`. Care should be taken as plugins that take a long time (greater than 10 seconds) to run can result in the Monasca agent failing to run its own checks in the allotted time and therefore stopping all client monitoring. Issues have been seen with hardware monitoring plugins that can take greater than 30 seconds and any plugins relying on name resolution when DNS services are not available. Details on the required Monasca configuration can be found at https://github.com/openstack/monasca-agent/blob/master/docs/ Plugins.md#nagios-wrapper ↗.

Use Case:

- Local host checking. As an operator I want to run a local monitoring check on my host to check physical hardware. Check status and alert management will be based around the Operations Console, not Nagios.

Limitation

- As mentioned earlier, care should be taken to ensure checks do not introduce load or delays in the Monasca agent check cycle. Additionally, depending on the operating system the node is running, plugins or dependencies may not be available.

Using Nagios as a central dashboard

It is possible to create a Nagios-style plugin that will query the Monasca API endpoint for an alarm status to create Nagios alerts and alarms based on Monasca alarms and filters. Monasca alarms appear in Nagios using two approaches, one listing checks by service and the other listing checks by physical host.

In the top section of the Nagios-style plugin, services can be created under a dummy host, `monasca_endpoint`. Each service retrieves all alarms based on defined dimensions. For example the `ardana-compute` check will return all alarms with the compute (Nova) dimension.

In the bottom section, the physical servers making up the HPE Helion OpenStack cluster can be defined and checks can be run. For example, one could check the server hardware from the Nagios server using a third party plugin and the another could retrieve all monasca alarms related to that host.

To build this configuration, a custom Nagios plugin (Please see example plugin at: https://github.com/openstack/python-monascaclient/tree/stable/pike/examples ↗) was created with the following options:

```
check_monasca —c CREDENTIALS -d DIMENSION -v VALUE
```

Examples:

To check alarms on `test-ccp-comp001-mgmt` you would use:

```
check_monasca —c service.osrc —d hostname —v test-ccp-comp001-mgmt
```

To check all Network related alarms, you would use:

```
check_monasca —c service.osrc —d service —v networking
```

Use Cases:

- Multiple clouds, integrating HPE Helion OpenStack monitoring with existing monitoring capabilities or viewing Monasca alerts in Nagios, fully integrating Monasca alarms with Nagios alarms and workflow.

- In a predominantly Nagios or Icinga-based monitoring environment, Monasca alarm status can be integrated into existing processes and workflows. This approach works best for checks associated with physical servers running the HPE Helion OpenStack services.

- With multiple HPE Helion OpenStack clusters, all of their alarms can be consolidated into a single view, the current version of Operations Console is for a single cluster only.

Limitations

- Nagios has a more traditional configuration model that requires checks to belong to predefined services and hosts, this is not well suited in highly dynamic cloud environments where the lifespan of virtual instances can be very short. One possible solution is with Icinga2 which has an API available to dynamically add host and service definitions, the check plugin could be extended to create alarm definitions dynamically as they occur.

  The key disadvantage is that multiple alarms can appear as a single service. For example, suppose there are 3 warnings against one service. If the operator acknowledges this alarm and subsequently a 4th warning alarm occurs, it would not generate an alert and could get missed.

  Care has to be taken that alarms are not missed. If the defined checks are only looking for checks in an ALARM status they will not report undetermined checks that might indicate other issues.

Using Operations Console as central dashboard

Nagios has the ability to run custom scripts in response to events. It is therefore possible to write a plugin to update Monasca whenever a Nagios alert occurs. The Operations Console could then be used as a central reporting dashboard for both Monasca and Nagios alarms. The external Nagios alarms can have their own check dimension and could be displayed as a separate group in the Operations Console.

Use Cases

- Using Operations Console the central monitoring tool.

Limitations

- The alarm could not be acknowledged from the Operations Console so Nagios could send repetitive notifications unless configured to take this into account.

HPE Helion OpenStack-specific Nagios Plugins

Several OpenStack plugin packages exist (see https://launchpad.net/ubuntu/+source/nagios-plugins-openstack ↗) that are useful to run from external sources to ensure the overall system is working as expected. Monasca requires some OpenStack components to be working in order to work at all. For example, if Keystone were unavailable, Monasca could not authenticate client or console requests. An external service check could highlight this.

3.2.5   Common integration issues

Alarm status differences

Monasca and Nagios treat alarms and status in different ways and for the two systems to talk there needs to be a mapping between them. The following table details the alarm parameters available for each:

System	Status	Severity	Details
Nagios	OK		Plugin returned OK with given thresholds
	WARNING		Plugin returned WARNING based on thresholds
	CRITICAL		Plugin returned CRITICAL alarm
	UNKNOWN		Plugin failed
Monasca	OK		No alarm triggered
	ALARM	LOW	Alarm state, LOW impact
	ALARM	MEDIUM	Alarm state, MEDIUM impact
	ALARM	HIGH	Alarm state, HIGH impact
	UNDETERMINED		No metrics received

In the plugin described here, the mapping was created with this flow:

```
Monasca OK -> Nagios OK
Monasca ALARM ( LOW or MEDIUM ) -> Nagios Warning
Monasca ALARM ( HIGH ) -> Nagios Critical
```

Alarm workflow differences

In both, system alarms can be acknowledged in the dashboards to indicate they are being worked on (or ignored). Not all the scenarios above will provide the same level of workflow integration.

3.3 Operations Bridge Integration

The HPE Helion OpenStack 8 monitoring solution (Monasca) can easily be integrated with your existing monitoring tools. Integrating HPE Helion OpenStack 8 Monasca with Operations Bridge using the Operations Bridge Connector simplifies monitoring and managing events and topology information.

The integration provides the following functionality:

- Forwarding of HPE Helion OpenStack Monasca alerts and topology to Operations Bridge for event correlation

- Customization of forwarded events and topology

For more information about this connector please see https://software.microfocus.com/en-us/ products/operations-bridge-suite/overview ↗ .

3.4 Monitoring Third-Party Components With Monasca

3.4.1 Monasca Monitoring Integration Overview

Monasca, the HPE Helion OpenStack 8 monitoring service, collects information about your cloud's systems, and allows you to create alarm definitions based on these measurements. Monasca-agent is the component that collects metrics such as metric storage and alarm thresholding and forwards them to the monasca-api for further processing.

With a small amount of configuration, you can use the detection and check plugins that are provided with your cloud to monitor integrated third-party components. In addition, you can write custom plugins and integrate them with the existing monitoring service.

Find instructions for customizing existing plugins to monitor third-party components in the *Section 3.4.4, "Configuring Check Plugins"*.

Find instructions for installing and configuring new custom plugins in the *Section 3.4.3, "Writing Custom Plugins"*.

You can also use existing alarm definitions, as well as create new alarm definitions that relate to a custom plugin or metric. Instructions for defining new alarm definitions are in the *Section 3.4.6, "Configuring Alarm Definitions"*.

You can use the Operations Console and Monasca CLI to list all of the alarms, alarm-definitions, and metrics that exist on your cloud.

3.4.2   Monasca Agent

The Monasca agent (monasca-agent) collects information about your cloud using the installed plugins. The plugins are written in Python, and determine the monitoring metrics for your system, as well as the interval for collection. The default collection interval is 30 seconds, and we strongly recommend not changing this default value.

The following two types of custom plugins can be added to your cloud.

- **Detection Plugin**. Determines whether the monasca-agent has the ability to monitor the specified component or service on a host. If successful, this type of plugin configures an associated **check plugin** by creating a YAML configuration file.

- **Check Plugin**. Specifies the metrics to be monitored, using the configuration file created by the detection plugin.

Monasca-agent is installed on every server in your cloud, and provides plugins that monitor the following.

- System metrics relating to CPU, memory, disks, host availability, etc.

- Process health metrics (process, http_check)

- HPE Helion OpenStack 8-specific component metrics, such as apache rabbitmq, kafka, cassandra, etc.

Monasca is pre-configured with default check plugins and associated detection plugins. The default plugins can be reconfigured to monitor third-party components, and often only require small adjustments to adapt them to this purpose. Find a list of the default plugins here: https://github.com/openstack/monasca-agent/blob/master/docs/Plugins.md#detection-plugins ↗

Often, a single check plugin will be used to monitor multiple services. For example, many services use the `http_check.py` detection plugin to detect the up/down status of a service endpoint. Often the `process.py` check plugin, which provides process monitoring metrics, is used as a basis for a custom process detection plugin.

More information about the Monasca agent can be found in the following locations

- Monasca agent overview: https://github.com/openstack/monasca-agent/blob/master/docs/Agent.md ↗

- Information on existing plugins: https://github.com/openstack/monasca-agent/blob/master/docs/Plugins.md ↗

- Information on plugin customizations: https://github.com/openstack/monasca-agent/blob/master/docs/Customizations.md ↗

3.4.3 Writing Custom Plugins

When the pre-built Monasca plugins do not meet your monitoring needs, you can write custom plugins to monitor your cloud. After you have written a plugin, you must install and configure it.

When your needs dictate a very specific custom monitoring check, you must provide both a detection and check plugin.

The steps involved in configuring a custom plugin include running a detection plugin and passing any necesssary parameters to the detection plugin so the resulting check configuration file is created with all necessary data.

When using an existing check plugin to monitor a third-party component, a custom detection plugin is needed only if there is not an associated default detection plugin.

Check plugin configuration files

Each plugin needs a corresponding YAML configuration file with the same stem name as the plugin check file. For example, the plugin file `http_check.py` (in `/usr/lib/python2.7/site-packages/monasca_agent/collector/checks_d/`) should have a corresponding configuration file, `http_check.yaml` (in `/etc/monasca/agent/conf.d/http_check.yaml`). The stem name `http_check` must be the same for both files.

Permissions for the YAML configuration file must be **read** + **write** for `mon-agent` user (the user that must also own the file), and **read** for the `mon-agent` group. Permissions for the file must be restricted to the **mon-agent** user and **monasca** group. The following example shows correct permissions settings for the file `http_check.yaml`.

```
ardana > ls -alt /etc/monasca/agent/conf.d/http_check.yaml
-rw-r----- 1 monasca-agent monasca 10590 Jul 26 05:44 http_check.yaml
```

A check plugin YAML configuration file has the following structure.

```
init_config:
```

```
    key1: value1
    key2: value2

instances:
    - name: john_smith
      username: john_smith
      password: 123456
    - name: jane_smith
      username: jane_smith
      password: 789012
```

In the above file structure, the `init_config` section allows you to specify any number of global **key:value** pairs. Each pair will be available on every run of the check that relates to the YAML configuration file.

The `instances` section allows you to list the instances that the related check will be run on. The check will be run once on each instance listed in the `instances` section. Ensure that each instance listed in the `instances` section has a unique name.

Custom detection plugins

Detection plugins should be written to perform checks that ensure that a component can be monitored on a host. Any arguments needed by the associated check plugin are passed into the detection plugin at setup (configuration) time. The detection plugin will write to the associated check configuration file.

When a detection plugin is successfully run in the configuration step, it will write to the check configuration YAML file. The configuration file for the check is written to the following directory.

```
/etc/monasca/agent/conf.d/
```

Writing process detection plugin using the ServicePlugin class

The monasca-agent provides a `ServicePlugin` class that makes process detection monitoring easy.

Process check

The process check plugin generates metrics based on the process status for specified process names. It generates `process.pid_count` metrics for the specified dimensions, and a set of detailed process metrics for the specified dimensions by default.

The ServicePlugin class allows you to specify a list of process name(s) to detect, and uses **psutil** to see if the process exists on the host. It then appends the `process.yml` configuration file with the process name(s), if they do not already exist.

The following is an example of a `process.py` check `ServicePlugin`.

```
import monasca_setup.detection

class MonascaTransformDetect(monasca_setup.detection.ServicePlugin):
    """Detect Monasca Transform daemons and setup configuration to monitor them."""
    def __init__(self, template_dir, overwrite=False, args=None):
        log.info("     Watching the monasca transform processes.")
        service_params = {
            'args': {},
            'template_dir': template_dir,
            'overwrite': overwrite,
            'service_name': 'monasca-transform',
            'process_names': ['monasca-transform','pyspark',
                              'transform/lib/driver']
        }
        super(MonascaTransformDetect, self).__init__(service_params)
```

Writing a Custom Detection Plugin using Plugin or ArgsPlugin classes

A custom detection plugin class should derive from either the Plugin or ArgsPlugin classes provided in the `/usr/lib/python2.7/site-packages/monasca_setup/detection` directory.

If the plugin parses command line arguments, the `ArgsPlugin` class is useful. The ArgsPlugin class derives from the Plugin class. The ArgsPlugin class has a method to check for required arguments, and a method to return the instance that will be used for writing to the configuration file with the dimensions from the command line parsed and included.

If the ArgsPlugin methods do not seem to apply, then derive directly from the Plugin class.

When deriving from these classes, the following methods should be implemented.

- *_detect - set self.available = True when conditions are met that the thing to monitor exists on a host.*

- *build_*config - writes the instance information to the configuration and return the configuration.

- dependencies_installed (default implementation is in ArgsPlugin, but not Plugin) - return true when python dependent libraries are installed.

The following is an example custom detection plugin.

```
import ast
import logging

import monasca_setup.agent_config
import monasca_setup.detection
```

```
log = logging.getLogger(__name__)


class HttpCheck(monasca_setup.detection.ArgsPlugin):
    """Setup an http_check according to the passed in args.
       Despite being a detection plugin this plugin does no detection and will be a noop
 without   arguments.
       Expects space separated arguments, the required argument is url. Optional
 parameters include:
       disable_ssl_validation and match_pattern.
    """

    def _detect(self):
        """Run detection, set self.available True if the service is detected.
        """
        self.available = self._check_required_args(['url'])

    def build_config(self):
        """Build the config as a Plugins object and return.
        """
        config = monasca_setup.agent_config.Plugins()
        # No support for setting headers at this time
        instance = self._build_instance(['url', 'timeout', 'username', 'password',
                                         'match_pattern', 'disable_ssl_validation',
                                         'name', 'use_keystone',
 'collect_response_time'])

        # Normalize any boolean parameters
        for param in ['use_keystone', 'collect_response_time']:
            if param in self.args:
                instance[param] = ast.literal_eval(self.args[param].capitalize())
        # Set some defaults
        if 'collect_response_time' not in instance:
            instance['collect_response_time'] = True
        if 'name' not in instance:
            instance['name'] = self.args['url']

        config['http_check'] = {'init_config': None, 'instances': [instance]}

        return config
```

Installing a detection plugin in the OpenStack version delivered with HPE Helion OpenStack

Install a plugin by copying it to the plugin directory (`/usr/lib/python2.7/site-packages/ monasca_agent/collector/checks_d/`).

The plugin should have file permissions of **read + write** for the root user (the user that should also own the file) and **read** for the root group and all other users.

The following is an example of correct file permissions for the **http_check.py** file.

```
-rw-r--r-- 1 root root 1769 Sep 19 20:14 http_check.py
```

Detection plugins should be placed in the following directory.

```
/usr/lib/monasca/agent/custom_detect.d/
```

The detection plugin directory name should be accessed using the `monasca_agent_detection_plugin_dir` Ansible variable. This variable is defined in the `roles/monasca-agent/vars/main.yml` file.

```
monasca_agent_detection_plugin_dir: /usr/lib/monasca/agent/custom_detect.d/
```

Example: Add Ansible `monasca_configure` task to install the plugin. (The `monasca_configure` task can be added to any service playbook.) In this example, it is added to `~/openstack/ardana/ansible/roles/_CEI-CMN/tasks/monasca_configure.yml`.

```
---
- name: _CEI-CMN | monasca_configure |
    Copy Ceilometer Custom plugin
  become: yes
  copy:
    src: ardanaceilometer_mon_plugin.py
    dest: "{{ monasca_agent_detection_plugin_dir }}"
    owner: root
    group: root
    mode: 0440
```

Custom check plugins

Custom check plugins generate metrics. Scalability should be taken into consideration on systems that will have hundreds of servers, as a large number of metrics can affect performance by impacting disk performance, RAM and CPU usage.

You may want to tune your configuration parameters so that less-important metrics are not monitored as frequently. When check plugins are configured (when they have an associated YAML configuration file) the agent will attempt to run them.

Checks should be able to run within the 30-second metric collection window. If your check runs a command, you should provide a timeout to prevent the check from running longer than the default 30-second window. You can use the `monasca_agent.common.util.timeout_command` to set a timeout for in your custom check plugin python code.

Find a description of how to write custom check plugins at https://github.com/openstack/monasca-agent/blob/master/docs/Customizations.md#creating-a-custom-check-plugin ↗

Custom checks derive from the AgentCheck class located in the `monasca_agent/collector/checks/check.py` file. A check method is required.

Metrics should contain dimensions that make each item that you are monitoring unique (such as service, component, hostname). The hostname dimension is defined by default within the AgentCheck class, so every metric has this dimension.

A custom check will do the following.

- Read the configuration instance passed into the check method.

- Set dimensions that will be included in the metric.

- Create the metric with gauge, rate, or counter types.

Metric Types:

- gauge: Instantaneous reading of a particular value (for example, mem.free_mb).

- rate: Measurement over a time period. The following equation can be used to define rate.

    rate=delta_v/float(delta_t)

- counter: The number of events, increment and decrement methods, for example, zookeeper.timeouts

The following is an example component check named SimpleCassandraExample.

```
import monasca_agent.collector.checks as checks
from monasca_agent.common.util import timeout_command

CASSANDRA_VERSION_QUERY = "SELECT version();"


class SimpleCassandraExample(checks.AgentCheck):

    def __init__(self, name, init_config, agent_config):
        super(SimpleCassandraExample, self).__init__(name, init_config, agent_config)

    @staticmethod
    def _get_config(instance):
        user = instance.get('user')
        password = instance.get('password')
        service = instance.get('service')
```

```
        timeout = int(instance.get('timeout'))

        return user, password, service, timeout

    def check(self, instance):
        user, password, service, node_name, timeout = self._get_config(instance)

        dimensions = self._set_dimensions({'component': 'cassandra', 'service': service},
instance)

        results, connection_status = self._query_database(user, password, timeout,
CASSANDRA_VERSION_QUERY)

        if connection_status != 0:
            self.gauge('cassandra.connection_status', 1, dimensions=dimensions)
        else:
            # successful connection status
            self.gauge('cassandra.connection_status', 0, dimensions=dimensions)

    def _query_database(self, user, password, timeout, query):
        stdout, stderr, return_code = timeout_command(["/opt/cassandra/bin/vsql", "-U",
user, "-w", password, "-A", "-R",
                                                       "|", "-t", "-F", ",", "-x"],
timeout, command_input=query)
        if return_code == 0:
            # remove trailing newline
            stdout = stdout.rstrip()
            return stdout, 0
        else:
            self.log.error("Error querying cassandra with return code of {0} and error
{1}".format(return_code, stderr))
            return stderr, 1
```

Installing check plugin

The check plugin needs to have the same file permissions as the detection plugin. File permissions must be **read** + **write** for the root user (the user that should own the file), and **read** for the root group and all other users.

Check plugins should be placed in the following directory.

```
/usr/lib/monasca/agent/custom_checks.d/
```

The check plugin directory should be accessed using the `monasca_agent_check_plugin_dir` Ansible variable. This variable is defined in the `roles/monasca-agent/vars/main.yml` file.

```
monasca_agent_check_plugin_dir: /usr/lib/monasca/agent/custom_checks.d/
```

3.4.4 Configuring Check Plugins

Manually configure a plugin when unit-testing using the monasca-setup script installed with the monasca-agent

Find a good explanation of configuring plugins here: https://github.com/openstack/monasca-agent/blob/master/docs/Agent.md#configuring ⬈

SSH to a node that has both the monasca-agent installed as well as the component you wish to monitor.

The following is an example command that configures a plugin that has no parameters (uses the detection plugin class name).

```
root # /usr/bin/monasca-setup -d ARDANACeilometer
```

The following is an example command that configures the apache plugin and includes related parameters.

```
root # /usr/bin/monasca-setup -d apache -a 'url=http://192.168.245.3:9095/server-status?
auto'
```

If there is a change in the configuration it will restart the monasca-agent on the host so the configuration is loaded.

After the plugin is configured, you can verify that the configuration file has your changes (see the next **Verify that your check plugin is configured** section).

Use the monasca CLI to see if your metric exists (see the **Verify that metrics exist** section).

Using Ansible modules to configure plugins in HPE Helion OpenStack 8

The `monasca_agent_plugin` module is installed as part of the monasca-agent role.

The following Ansible example configures the process.py plugin for the Ceilometer detection plugin. The following example only passes in the name of the detection class.

```
- name: _CEI-CMN | monasca_configure |
    Run Monasca agent Cloud Lifecycle Manager specific ceilometer detection plugin
  become: yes
  monasca_agent_plugin:
    name: "ARDANACeilometer"
```

If a password or other sensitive data are passed to the detection plugin, the `no_log` option should be set to **True**. If the `no_log` option is not set to **True**, the data passed to the plugin will be logged to syslog.

The following Ansible example configures the Cassandra plugin and passes in related arguments.

```
 - name: Run Monasca Agent detection plugin for Cassandra
   monasca_agent_plugin:
     name: "Cassandra"
     args="directory_names={{ FND_CDB.vars.cassandra_data_dir }},
{{ FND_CDB.vars.cassandra_commit_log_dir }}
 process_username={{ FND_CDB.vars.cassandra_user }}"
   when: database_type == 'cassandra'
```

The following Ansible example configures the Keystone endpoint using the http_check.py detection plugin. The class name `httpcheck` of the http_check.py detection plugin is the name.

```
root # - name:  keystone-monitor | local_monitor |
    Setup active check on keystone internal endpoint locally
  become: yes
  monasca_agent_plugin:
    name: "httpcheck"
    args: "use_keystone=False \
           url=http://{{ keystone_internal_listen_ip }}:{{
               keystone_internal_port }}/v3 \
           dimensions=service:identity-service,\
                      component:keystone-api,\
                      api_endpoint:internal,\
                      monitored_host_type:instance"
  tags:
    - keystone
    - keystone_monitor
```

Verify that your check plugin is configured

All check configuration files are located in the following directory. You can see the plugins that are running by looking at the plugin configuration directory.

```
/etc/monasca/agent/conf.d/
```

When the monasca-agent starts up, all of the check plugins that have a matching configuration file in the `/etc/monasca/agent/conf.d/` directory will be loaded.

If there are errors running the check plugin they will be written to the following error log file.

```
/var/log/monasca/agent/collector.log
```

You can change the monasca-agent log level by modifying the `log_level` option in the `/etc/monasca/agent/agent.yaml` configuration file, and then restarting the monasca-agent, using the following command.

```
root # service openstack-monasca-agent restart
```

You can debug a check plugin by running `monasca-collector` with the check option. The following is an example of the `monasca-collector` command.

```
tux > sudo /usr/bin/monasca-collector check CHECK_NAME
```

Verify that metrics exist

Begin by logging in to your deployer or controller node.

Run the following set of commands, including the `monasca metric-list` command. If the metric exists, it will be displayed in the output.

```
ardana > source ~/service.osrc
ardana > monasca metric-list --name METRIC_NAME
```

3.4.5 Metric Performance Considerations

Collecting metrics on your virtual machines can greatly affect performance. HPE Helion OpenStack 8 supports 200 compute nodes, with up to 40 VMs each. If your environment is managing maximum number of VMs, adding a single metric for all VMs is the equivalent of adding 8000 metrics.

Because of the potential impact that new metrics have on system performance, consider adding only new metrics that are useful for alarm-definition, capacity planning, or debugging process failure.

3.4.6 Configuring Alarm Definitions

The monasca-api-spec, found here https://github.com/openstack/monasca-api/blob/master/docs/monasca-api-spec.md ↗ provides an explanation of Alarm Definitions and Alarms. You can find more information on alarm definition expressions at the following page: https://github.com/openstack/monasca-api/blob/master/docs/monasca-api-spec.md#alarm-definition-expressions ↗.

When an alarm definition is defined, the monasca-threshold engine will generate an alarm for each unique instance of the match_by metric dimensions found in the metric. This allows a single alarm definition that can dynamically handle the addition of new hosts.

There are default alarm definitions configured for all "process check" (process.py check) and "HTTP Status" (http_check.py check) metrics in the monasca-default-alarms role. The monasca-default-alarms role is installed as part of the Monasca deployment phase of your cloud's deployment. You do not need to create alarm definitions for these existing checks.

Third parties should create an alarm definition when they wish to alarm on a custom plugin metric. The alarm definition should only be defined once. Setting a notification method for the alarm definition is recommended but not required.

The following Ansible modules used for alarm definitions are installed as part of the monasca-alarm-definition role. This process takes place during the Monasca set up phase of your cloud's deployment.

- monasca_alarm_definition

- monasca_notification_method

The following examples, found in the `~/openstack/ardana/ansible/roles/monasca-default-alarms` directory, illustrate how Monasca sets up the default alarm definitions.

Monasca Notification Methods

The monasca-api-spec, found in the following link, provides details about creating a notification https://github.com/openstack/monasca-api/blob/master/docs/monasca-api-spec.md#create-notification-method ↗

The following are supported notification types.

- EMAIL

- WEBHOOK

- PAGERDUTY

The `keystone_admin_tenant` project is used so that the alarms will show up on the Operations Console UI.

The following file snippet shows variables from the `~/openstack/ardana/ansible/roles/monasca-default-alarms/defaults/main.yml` file.

```
---
notification_address: root@localhost
notification_name: 'Default Email'
notification_type: EMAIL

monasca_keystone_url: "{{ KEY_API.advertises.vips.private[0].url }}/v3"
monasca_api_url: "{{ MON_AGN.consumes_MON_API.vips.private[0].url }}/v2.0"
monasca_keystone_user: "{{ MON_API.consumes_KEY_API.vars.keystone_monasca_user }}"
monasca_keystone_password: "{{ MON_API.consumes_KEY_API.vars.keystone_monasca_password |
 quote }}"
monasca_keystone_project: "{{ KEY_API.vars.keystone_admin_tenant }}"
```

```
monasca_client_retries: 3
monasca_client_retry_delay: 2
```

You can specify a single default notification method in the `~/openstack/ardana/ansible/roles/monasca-default-alarms/tasks/main.yml` file. You can also add or modify the notification type and related details using the Operations Console UI or Monasca CLI.

The following is a code snippet from the `~/openstack/ardana/ansible/roles/monasca-default-alarms/tasks/main.yml` file.

```
---
- name: monasca-default-alarms | main | Setup default notification method
  monasca_notification_method:
    name: "{{ notification_name }}"
    type: "{{ notification_type }}"
    address: "{{ notification_address }}"
    keystone_url: "{{ monasca_keystone_url }}"
    keystone_user: "{{ monasca_keystone_user }}"
    keystone_password: "{{ monasca_keystone_password }}"
    keystone_project: "{{ monasca_keystone_project }}"
    monasca_api_url: "{{ monasca_api_url }}"
  no_log: True
  tags:
    - system_alarms
    - monasca_alarms
    - openstack_alarms
  register: default_notification_result
  until: not default_notification_result | failed
  retries: "{{ monasca_client_retries }}"
  delay: "{{ monasca_client_retry_delay }}"
```

Monasca Alarm Definition

In the alarm definition "expression" field, you can specify the metric name and threshold. The "match_by" field is used to create a new alarm for every unique combination of the match_by metric dimensions.

Find more details on alarm definitions at the Monasca API documentation: (https://github.com/stackforge/monasca-api/blob/master/docs/monasca-api-spec.md#alarm-definitions-and-alarms ↗).

The following is a code snippet from the `~/openstack/ardana/ansible/roles/monasca-default-alarms/tasks/main.yml` file.

```
- name: monasca-default-alarms | main | Create Alarm Definitions
  monasca_alarm_definition:
    name: "{{ item.name }}"
```

```
      description: "{{ item.description | default('') }}"
      expression: "{{ item.expression }}"
      keystone_token: "{{ default_notification_result.keystone_token }}"
      match_by: "{{ item.match_by | default(['hostname']) }}"
      monasca_api_url: "{{ default_notification_result.monasca_api_url }}"
      severity: "{{ item.severity | default('LOW') }}"
      alarm_actions:
        - "{{ default_notification_result.notification_method_id }}"
      ok_actions:
        - "{{ default_notification_result.notification_method_id }}"
      undetermined_actions:
        - "{{ default_notification_result.notification_method_id }}"
  register: monasca_system_alarms_result
  until: not monasca_system_alarms_result | failed
  retries: "{{ monasca_client_retries }}"
  delay: "{{ monasca_client_retry_delay }}"
  with_flattened:
    - monasca_alarm_definitions_system
    - monasca_alarm_definitions_monasca
    - monasca_alarm_definitions_openstack
    - monasca_alarm_definitions_misc_services
  when: monasca_create_definitions
```

In the following example `~/openstack/ardana/ansible/roles/monasca-default-alarms/vars/main.yml` Ansible variables file, the alarm definition named **Process Check** sets the **match_by** variable with the following parameters.

- process_name

- hostname

```
monasca_alarm_definitions_system:
  - name: "Host Status"
    description: "Alarms when the specified host is down or not reachable"
    severity: "HIGH"
    expression: "host_alive_status > 0"
    match_by:
      - "target_host"
      - "hostname"
  - name: "HTTP Status"
    description: >
      "Alarms when the specified HTTP endpoint is down or not reachable"
    severity: "HIGH"
    expression: "http_status > 0"
    match_by:
      - "service"
      - "component"
```

```
      - "hostname"
      - "url"
- name: "CPU Usage"
  description: "Alarms when CPU usage is high"
  expression: "avg(cpu.idle_perc) < 10 times 3"
- name: "High CPU IOWait"
  description: "Alarms when CPU IOWait is high, possible slow disk issue"
  expression: "avg(cpu.wait_perc) > 40 times 3"
  match_by:
    - "hostname"
- name: "Disk Inode Usage"
  description: "Alarms when disk inode usage is high"
  expression: "disk.inode_used_perc > 90"
  match_by:
    - "hostname"
    - "device"
  severity: "HIGH"
- name: "Disk Usage"
  description: "Alarms when disk usage is high"
  expression: "disk.space_used_perc > 90"
  match_by:
    - "hostname"
    - "device"
  severity: "HIGH"
- name: "Memory Usage"
  description: "Alarms when memory usage is high"
  severity: "HIGH"
  expression: "avg(mem.usable_perc) < 10 times 3"
- name: "Network Errors"
  description: >
    "Alarms when either incoming or outgoing network errors are high"
  severity: "MEDIUM"
  expression: "net.in_errors_sec > 5 or net.out_errors_sec > 5"
- name: "Process Check"
  description: "Alarms when the specified process is not running"
  severity: "HIGH"
  expression: "process.pid_count < 1"
  match_by:
    - "process_name"
    - "hostname"
- name: "Crash Dump Count"
  description: "Alarms when a crash directory is found"
  severity: "MEDIUM"
  expression: "crash.dump_count > 0"
  match_by:
    - "hostname"
```

The preceding configuration would result in the creation of an alarm for each unique metric that matched the following criteria.

```
process.pid_count + process_name + hostname
```

Check that the alarms exist

Begin by using the following commands, including `monasca alarm-definition-list`, to check that the alarm definition exists.

```
ardana > source ~/service.osrc
ardana > monasca alarm-definition-list --name ALARM_DEFINITION_NAME
```

Then use either of the following commands to check that the alarm has been generated. A status of "OK" indicates a healthy alarm.

```
ardana > monasca alarm-list --metric-name metric name
```

Or

```
ardana > monasca alarm-list --alarm-definition-id ID_FROM_ALARM-DEFINITION-LIST
```

☞ Note

To see CLI options use the `monasca help` command.

Alarm state upgrade considerations

If the name of a monitoring metric changes or is no longer being sent, existing alarms will show the alarm state as UNDETERMINED. You can update an alarm definition as long as you do not change the **metric name** or **dimension name** values in the **expression** or **match_by** fields. If you find that you need to alter either of these values, you must delete the old alarm definitions and create new definitions with the updated values.

If a metric is never sent, but had a related alarm definition, then no alarms would exist. If you find that no metrics are never sent, then you should remove the related alarm definition.

When removing an alarm definition, the Ansible module **monasca_alarm_definition** supports the state "absent".

The following file snippet shows an example of how to remove an alarm definition by setting the state to **absent**.

```
- name: monasca-pre-upgrade | Remove alarm definitions
```

```
  monasca_alarm_definition:
    name: "{{ item.name }}"
    state: "absent"
    keystone_url: "{{ monasca_keystone_url }}"
    keystone_user: "{{ monasca_keystone_user }}"
    keystone_password: "{{ monasca_keystone_password }}"
    keystone_project: "{{ monasca_keystone_project }}"
    monasca_api_url: "{{ monasca_api_url }}"
  with_items:
    - { name: "Kafka Consumer Lag" }
```

An alarm exists in the OK state when the monasca threshold engine has seen at least one metric associated with the alarm definition and has not exceeded the alarm definition threshold.

3.4.7   Openstack Integration of Custom Plugins into Monasca-Agent (if applicable)

Monasca-agent is an OpenStack open-source project. Monasca can also monitor non-openstack services. Third parties should install custom plugins into their HPE Helion OpenStack 8 system using the steps outlined in the *Section 3.4.3, "Writing Custom Plugins"*. If the OpenStack community determines that the custom plugins are of general benefit, the plugin may be added to the openstack/monasca-agent so that they are installed with the monasca-agent. During the review process for openstack/monasca-agent there are no guarantees that code will be approved or merged by a deadline. Open-source contributors are expected to help with codereviews in order to get their code accepted. Once changes are approved and integrated into the openstack/monasca-agent and that version of the monasca-agent is integrated with HPE Helion OpenStack 8, the third party can remove the custom plugin installation steps since they would be installed in the default monasca-agent venv.

Find the open source repository for the monaca-agent here: https://github.com/openstack/monasca-agent ↗

4 Managing Identity

The Identity service provides the structure for user authentication to your cloud.

4.1 The Identity Service

This topic explains the purpose and mechanisms of the identity service.

The HPE Helion OpenStack Identity service, based on the OpenStack Keystone API, is responsible for providing UserID authentication and access authorization to enable organizations to achieve their access security and compliance objectives and successfully deploy OpenStack. In short, the Identity Service is the gateway to the rest of the OpenStack services.

4.1.1 Which version of the Keystone Identity service should you use?

Use Identity API version 3.0. Identity API v2.0 is deprecated. Many features such as LDAP integration and fine-grained access control will not work with v2.0. Below are a few more questions you may have regarding versions.

Why does the Keystone identity catalog still show version 2.0?

Tempest tests still use the v2.0 API. They are in the process of migrating to v3.0. We will remove the v2.0 version once tempest has migrated the tests. The Identity catalog has v2.0 version just to support tempest migration.

Will the Keystone identity v3.0 API work if the identity catalog has only the v2.0 endpoint?

Identity v3.0 does not rely on the content of the catalog. It will continue to work regardless of the version of the API in the catalog.

Which CLI client should you use?

You should use the OpenStack CLI, not the Keystone CLI as it is deprecated. The Keystone CLI does not support v3.0 API, only the OpenStack CLI supports the v3.0 API.

4.1.2   Authentication

The authentication function provides the initial login function to OpenStack. Keystone supports multiple sources of authentication, including a native or built-in authentication system. The Keystone native system can be used for all user management functions for proof of concept deployments or small deployments not requiring integration with a corporate authentication system, but it lacks some of the advanced functions usually found in user management systems such as forcing password changes. The focus of the Keystone native authentication system is to be the source of authentication for OpenStack-specific users required for the operation of the various OpenStack services. These users are stored by Keystone in a default domain; the addition of these IDs to an external authentication system is not required.

Keystone is more commonly integrated with external authentication systems such as OpenLDAP or Microsoft Active Directory. These systems are usually centrally deployed by organizations to serve as the single source of user management and authentication for all in-house deployed applications and systems requiring user authentication. In addition to LDAP and Microsoft Active Directory, support for integration with Security Assertion Markup Language (SAML)-based identity providers from companies such as Ping, CA, IBM, Oracle, and others is also nearly "production-ready".

Keystone also provides architectural support via the underlying Apache deployment for other types of authentication systems such as Multi-Factor Authentication. These types of systems typically require driver support and integration from the respective provider vendors.

☞ Note

While support for Identity Providers and Multi-factor authentication is available in Keystone, it has not yet been certified by the HPE Helion OpenStack engineering team and is an experimental feature in HPE Helion OpenStack.

LDAP-compatible directories such as OpenLDAP and Microsoft Active Directory are recommended alternatives to using the Keystone local authentication. Both methods are widely used by organizations and are integrated with a variety of other enterprise applications. These directories act as the single source of user information within an organization. Keystone can be configured to authenticate against an LDAP-compatible directory on a per-domain basis.

Domains, as explained in *Section 4.3, "Understanding Domains, Projects, Users, Groups, and Roles"*, can be configured so that based on the user ID, a incoming user is automatically mapped to a specific domain. This domain can then be configured to authenticate against a specific LDAP directory.

The user credentials provided by the user to Keystone are passed along to the designated LDAP source for authentication. This communication can be optionally configured to be secure via SSL encryption. No special LDAP administrative access is required, and only read-only access is needed for this configuration. Keystone will not add any LDAP information. All user additions, deletions, and modifications are performed by the application's front end in the LDAP directories. After a user has been successfully authenticated, he is then assigned to the groups, roles, and projects defined by the Keystone domain or project administrators. This information is stored within the Keystone service database.

Another form of external authentication provided by the Keystone service is via integration with SAML-based Identity Providers (IdP) such as Ping Identity, IBM Tivoli, and Microsoft Active Directory Federation Server. A SAML-based identity provider provides authentication that is often called "single sign-on". The IdP server is configured to authenticate against identity sources such as Active Directory and provides a single authentication API against multiple types of downstream identity sources. This means that an organization could have multiple identity storage sources but a single authentication source. In addition, if a user has logged into one such source during a defined session time frame, they do not need to re-authenticate within the defined session. Instead, the IdP will automatically validate the user to requesting applications and services.

A SAML-based IdP authentication source is configured with Keystone on a per-domain basis similar to the manner in which native LDAP directories are configured. Extra mapping rules are required in the configuration that define which Keystone group an incoming UID is automatically assigned to. This means that groups need to be defined in Keystone first, but it also removes the requirement that a domain or project admin assign user roles and project membership on a per-user basis. Instead, groups are used to define project membership and roles and incoming users are automatically mapped to Keystone groups based on their upstream group membership. This provides a very consistent role-based access control (RBAC) model based on the upstream identity source. The configuration of this option is fairly straightforward. IdP vendors such as Ping and IBM are contributing to the maintenance of this function and have also produced their own integration documentation. Microsoft Active Directory Federation Services (ADFS) is used for functional testing and future documentation.

The third Keystone-supported authentication source is known as Multi-Factor Authentication (MFA). MFA typically requires an external source of authentication beyond a login name and password, and can include options such as SMS text, a temporal token generator, a fingerprint

scanner, etc. Each of these types of MFA are usually specific to a particular MFA vendor. The Keystone architecture supports an MFA-based authentication system, but this has not yet been certified or documented for HPE Helion OpenStack.

4.1.3 Authorization

The second major function provided by the Keystone service is access authorization that determines what resources and actions are available based on the UserID, the role of the user, and the projects that a user is provided access to. All of this information is created, managed, and stored by Keystone. These functions are applied via the Horizon web interface, the OpenStack command-line interface, or the direct Keystone API.

Keystone provides support for organizing users via three entities including:

Domains

> Domains provide the highest level of organization. Domains are intended to be used as high-level containers for multiple projects. A domain can represent different tenants, companies or organizations for an OpenStack cloud deployed for public cloud deployments or represent major business units, functions, or any other type of top-level organization unit in an OpenStack private cloud deployment. Each domain has at least one Domain Admin assigned to it. This Domain Admin can then create multiple projects within the domain and assign the project admin role to specific project owners. Each domain created in an OpenStack deployment is unique and the projects assigned to a domain cannot exist in another domain.

Projects

> Projects are entities within a domain that represent groups of users, each user role within that project, and how many underlying infrastructure resources can be consumed by members of the project.

Groups

> Groups are an optional function and provide the means of assigning project roles to multiple users at once.

Keystone also provides the means to create and assign roles to groups of users or individual users. The role names are created and user assignments are made within Keystone. The actual function of a role is defined currently per each OpenStack service via scripts. When a user requests

access to an OpenStack service, his access token contains information about his assigned project membership and role for that project. This role is then matched to the service-specific script and the user is allowed to perform functions within that service defined by the role mapping.

4.2 Supported Upstream Keystone Features

4.2.1 OpenStack upstream features that are enabled by default in HPE Helion OpenStack 8

The following supported Keystone features are enabled by default in the HPE Helion OpenStack 8 release.

Name	User/Admin	Note: API support only. No CLI/UI support
Implied Roles	Admin	https://blueprints.launchpad.net/keystone/+spec/implied-roles
Domain-Specific Roles	Admin	https://blueprints.launchpad.net/keystone/+spec/domain-specific-roles

Implied rules

To allow for the practice of hierarchical permissions in user roles, this feature enables roles to be linked in such a way that they function as a hierarchy with role inheritance.

When a user is assigned a superior role, the user will also be assigned all roles implied by any subordinate roles. The hierarchy of the assigned roles will be expanded when issuing the user a token.

Domain-specific roles

This feature extends the principle of **implied roles** to include a set of roles that are specific to a domain. At the time a token is issued, the domain-specific roles are not included in the token, however, the roles that they map to are.

4.2.2 OpenStack upstream features that are disabled by default in HPE Helion OpenStack 8

The following is a list of features which are fully supported in the HPE Helion OpenStack 8 release, but are disabled by default. Customers can run a playbook to enable the features.

Name	User/Admin	Reason Disabled
Support multiple LDAP backends via per-domain configuration	Admin	Needs explicit configuration.
WebSSO	User and Admin	Needs explicit configuration.
Keystone-to-Keystone (K2K) federation	User and Admin	Needs explicit configuration.
Fernet token provider	User and Admin	Needs explicit configuration.
Domain-specific config in SQL	Admin	Domain specific configuration options can be stored in SQL instead of configuration files, using the new REST APIs.

Multiple LDAP backends for each domain

This feature allows identity backends to be configured on a domain-by-domain basis. Domains will be capable of having their own exclusive LDAP service (or multiple services). A single LDAP service can also serve multiple domains, with each domain in a separate subtree.

To implement this feature, individual domains will require domain-specific configuration files. Domains that do not implement this feature will continue to share a common backend driver.

WebSSO

This feature enables the Keystone service to provide federated identity services through a token-based single sign-on page. This feature is disabled by default, as it requires explicit configuration.

Keystone-to-Keystone (K2K) federation

This feature enables separate Keystone instances to federate identities among the instances, offering inter-cloud authorization. This feature is disabled by default, as it requires explicit configuration.

Fernet token provider

Provides tokens in the fernet format. This is an experimental feature and is disabled by default.

Domain-specific config in SQL

Using the new REST APIs, domain-specific configuration options can be stored in a SQL database instead of in configuration files.

4.2.3   Stack upstream features that have been specifically disabled in HPE Helion OpenStack 8

The following is a list of extensions which are disabled by default in HPE Helion OpenStack 8, according to Keystone policy.

Target Release	Name	User/Admin	Reason Disabled
TBD	Endpoint Filtering	Admin	This extension was implemented to facilitate service activation. However, due to lack of enforcement at the service side, this feature is only half effective right now.
TBD	Endpoint Policy	Admin	This extension was intended to facilitate policy (policy.json) management and enforcement. This feature is useless right now due to lack of the needed middleware to utilize the policy files stored in Keystone.

Target Release	Name	User/Admin	Reason Disabled
TBD	OATH 1.0a	User and Admin	Complexity in workflow. Lack of adoption. Its alternative, Keystone Trust, is enabled by default. HEAT is using Keystone Trust.
TBD	Revocation Events	Admin	For PKI token only and PKI token is disabled by default due to usability concerns.
TBD	OS CERT	Admin	For PKI token only and PKI token is disabled by default due to usability concerns.
TBD	PKI Token	Admin	PKI token is disabled by default due to usability concerns.
TBD	Driver level caching	Admin	Driver level caching is disabled by default due to complexity in setup.
TBD	Tokenless Authz	Admin	Tokenless authorization with X.509 SSL client certificate.

Stack upstream features that have been specifically disabled in HPE Helion OpenStack

Target Release	Name	User/Admin	Reason Disabled
TBD	TOTP Authentication	User	Not fully baked. Has not been battle-tested.
TBD	is_admin_project	Admin	No integration with the services.

4.3 Understanding Domains, Projects, Users, Groups, and Roles

The identity service uses these concepts for authentication within your cloud and these are descriptions of each of them.

The HPE Helion OpenStack 8 identity service uses OpenStack Keystone and the concepts of domains, projects, users, groups, and roles to manage authentication. This page describes how these work together.

4.3.1 Domains, Projects, Users, Groups, and Roles

Most large business organizations use an identity system such as Microsoft Active Directory to store and manage their internal user information. A variety of applications such as HR systems are, in turn, used to manage the data inside of Active Directory. These same organizations often deploy a separate user management system for external users such as contractors, partners, and customers. Multiple authentication systems are then deployed to support multiple types of users.

An LDAP-compatible directory such as Active Directory provides a top-level organization or domain component. In this example, the organization is called Acme. The domain component (DC) is defined as acme.com. Underneath the top level domain component are entities referred to as organizational units (OU). Organizational units are typically designed to reflect the entity structure of the organization. For example, this particular schema has 3 different organizational units for the Marketing, IT, and Contractors units or departments of the Acme organization. Users (and other types of entities like printers) are then defined appropriately underneath each organizational entity. The Keystone domain entity can be used to match the LDAP OU entity; each LDAP OU can have a corresponding Keystone domain created. In this

example, both the Marketing and IT domains represent internal employees of Acme and use the same authentication source. The Contractors domain contains all external people associated with Acme. UserIDs associated with the Contractor domain are maintained in a separate user directory and thus have a different authentication source assigned to the corresponding Keystone-defined Contractors domain.

A public cloud deployment usually supports multiple, separate organizations. Keystone domains can be created to provide a domain per organization with each domain configured to the underlying organization's authentication source. For example, the ABC company would have a Keystone domain created called "abc". All users authenticating to the "abc" domain would be authenticated against the authentication system provided by the ABC organization; in this case ldap://ad.abc.com

4.3.2 Domains

A domain is a top-level container targeted at defining major organizational entities.

- Domains can be used in a multi-tenant OpenStack deployment to segregate projects and users from different companies in a public cloud deployment or different organizational units in a private cloud setting.

- Domains provide the means to identify multiple authentication sources.

- Each domain is unique within an OpenStack implementation.

- Multiple projects can be assigned to a domain but each project can only belong to a single domain.

- Each domain and project have an assigned admin.

- Domains are created by the "admin" service account and domain admins are assigned by the "admin" user.

- The "admin" UserID (UID) is created during the Keystone installation, has the "admin" role assigned to it, and is defined as the "Cloud Admin". This UID is created using the "magic" or "secret" admin token found in the default 'keystone.conf' file installed during HPE Helion OpenStack keystone installation after the Keystone service has been installed. This secret token should be removed after installation and the "admin" password changed.

- The "default" domain is created automatically during the HPE Helion OpenStack Keystone installation.

- The "default" domain contains all OpenStack service accounts that are installed during the HPE Helion OpenStack keystone installation process.

- No users but the OpenStack service accounts should be assigned to the "default" domain.

- Domain admins can be any UserID inside or outside of the domain.

4.3.3 Domain Administrator

A UUID is a domain administrator for a given domain if that UID has a domain-scoped token scoped for the given domain. This means that the UID has the "admin" role assigned to it for the selected domain.

- The Cloud Admin UID assigns the domain administrator role for a domain to a selected UID.

- A domain administrator can create and delete local users who have authenticated against Keystone. These users will be assigned to the domain belonging to the domain administrator who creates the UserID.

- A domain administrator can only create users and projects within her assigned domains.

- A domain administrator can assign the "admin" role of their domains to another UID or revoke it; each UID with the "admin" role for a specified domain will be a co-administrator for that domain.

- A UID can be assigned to be the domain admin of multiple domains.

- A domain administrator can assign non-admin roles to any users and groups within their assigned domain, including projects owned by their assigned domain.

- A domain admin UID can belong to projects within their administered domains.

- Each domain can have a different authentication source.

- The domain field is used during the initial login to define the source of authentication.

- The "List Users" function can only be executed by a UID with the domain admin role.

- A domain administrator can assign a UID from outside of their domain the "domain admin" role but it is assumed that the domain admin would know the specific UID and would not need to list users from an external domain.

- A domain administrator can assign a UID from outside of their domain the "project admin" role for a specific project within their domain but it is assumed that the domain admin would know the specific UID and would not need to list users from an external domain.

4.3.4   Projects

The domain administrator creates projects within his assigned domain and assigns the project admin role to each project to a selected UID. A UID is a project administrator for a given project if that UID has a project-scoped token scoped for the given project. There can be multiple projects per domain. The project admin sets the project quota settings, adds/deletes users and groups to and from the project, and defines the user/group roles for the assigned project. Users can be belong to multiple projects and have different roles on each project. Users are assigned to a specific domain and a default project. Roles are assigned per project.

4.3.5   Users and Groups

Each user belongs to one domain only. Domain assignments are defined either by the domain configuration files or by a domain administrator when creating a new, local (user authenticated against Keystone) user. There is no current method for "moving" a user from one domain to another. A user can belong to multiple projects within a domain with a different role assignment per project. A group is a collection of users. Users can be assigned to groups either by the project admin or automatically via mappings if an external authentication source is defined for the assigned domain. Groups can be assigned to multiple projects within a domain and have different roles assigned to the group per project. A group can be assigned the "admin" role for a domain or project. All members of the group will be an "admin" for the selected domain or project.

4.3.6   Roles

Service roles represent the functionality used to implement the OpenStack role based access control (RBAC), model used to manage access to each OpenStack service. Roles are named and assigned per user or group for each project by the identity service. Role definition and

policy enforcement are defined outside of the identity service independently by each OpenStack service. The token generated by the identity service for each user authentication contains the role assigned to that user for a particular project. When a user attempts to access a specific OpenStack service, the role is parsed by the service, compared to the service-specific policy file, and then granted the resource access defined for that role by the service policy file.

Each service has its own service policy file with the /etc/[SERVICE_CODENAME]/policy.json file name format where [SERVICE_CODENAME] represents a specific OpenStack service name. For example, the OpenStack Nova service would have a policy file called /etc/nova/policy.json. With Service policy files can be modified and deployed to control nodes from the Cloud Lifecycle Manager. Administrators are advised to validate policy changes before checking in the changes to the site branch of the local git repository before rolling the changes into production. Do not make changes to policy files without having a way to validate them.

The policy files are located at the following site branch locations on the Cloud Lifecycle Manager.

```
~/openstack/ardana/ansible/roles/GLA-API/templates/policy.json.j2
~/openstack/ardana/ansible/roles/ironic-common/files/policy.json
~/openstack/ardana/ansible/roles/KEYMGR-API/templates/policy.json
~/openstack/ardana/ansible/roles/heat-common/files/policy.json
~/openstack/ardana/ansible/roles/CND-API/templates/policy.json
~/openstack/ardana/ansible/roles/nova-common/files/policy.json
~/openstack/ardana/ansible/roles/CEI-API/templates/policy.json.j2
~/openstack/ardana/ansible/roles/neutron-common/templates/policy.json.j2
```

For test and validation, policy files can be modified in a non-production environment from the `~/scratch/` directory. For a specific policy file, run a search for policy.json. To deploy policy changes for a service, run the service specific reconfiguration playbook (for example, nova-reconfigure.yml). For a complete list of reconfiguration playbooks, change directories to `~/scratch/ansible/next/ardana/ansible` and run this command:

```
ardana > ls | grep reconfigure
```

A read-only role named `project_observer` is explicitly created in HPE Helion OpenStack 8. Any user who is granted this role can use `list_project`.

4.4 Identity Service Token Validation Example

The following diagram illustrates the flow of typical Identity Service (Keystone) requests/responses between HPE Helion OpenStack services and the Identity service. It shows how Keystone issues and validates tokens to ensure the identity of the caller of each service.
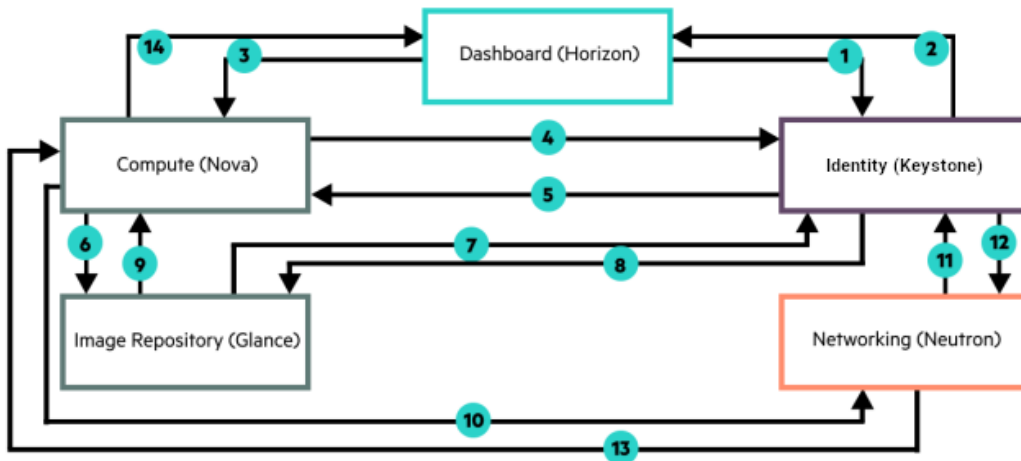
1. Horizon sends an HTTP authentication request to Keystone for user credentials.

2. Keystone validates the credentials and replies with token.

3. Horizon sends a POST request, with token to Nova to start provisioning a virtual machine.

4. Nova sends token to Keystone for validation.

5. Keystone validates the token.

6. Nova forwards a request for an image with the attached token.

7. Glance sends token to Keystone for validation.

8. Keystone validates the token.

9. Glance provides image-related information to Nova.

10. Nova sends request for networks to Neutron with token.

11. Neutron sends token to Keystone for validation.

12. Keystone validates the token.

13. Neutron provides network-related information to Nova.

14. Nova reports the status of the virtual machine provisioning request.

4.5 Configuring the Identity Service

4.5.1 What is the Identity service?

The HPE Helion OpenStack Identity service, based on the OpenStack Keystone API, provides UserID authentication and access authorization to help organizations achieve their access security and compliance objectives and successfully deploy OpenStack. In short, the Identity service is the gateway to the rest of the OpenStack services.

The identity service is installed automatically by the Cloud Lifecycle Manager (just after MySQL and RabbitMQ). When your cloud is up and running, you can customize Keystone in a number of ways, including integrating with LDAP servers. This topic describes the default configuration. See *Section 4.8, "Reconfiguring the Identity Service"* for changes you can implement. Also see *Section 4.9, "Integrating LDAP with the Identity Service"* for information on integrating with an LDAP provider.

4.5.2 Which version of the Keystone Identity service should you use?

Note that you should use identity API version 3.0. Identity API v2.0 was has been deprecated. Many features such as LDAP integration and fine-grained access control will not work with v2.0. The following are a few questions you may have regarding versions.

Why does the Keystone identity catalog still show version 2.0?

Tempest tests still use the v2.0 API. They are in the process of migrating to v3.0. We will remove the v2.0 version once tempest has migrated the tests. The Identity catalog has version 2.0 just to support tempest migration.

Will the Keystone identity v3.0 API work if the identity catalog has only the v2.0 endpoint?

Identity v3.0 does not rely on the content of the catalog. It will continue to work regardless of the version of the API in the catalog.

Which CLI client should you use?

You should use the OpenStack CLI, not the Keystone CLI, because it is deprecated. The Keystone CLI does not support the v3.0 API; only the OpenStack CLI supports the v3.0 API.

4.5.3 Authentication

The authentication function provides the initial login function to OpenStack. Keystone supports multiple sources of authentication, including a native or built-in authentication system. You can use the Keystone native system for all user management functions for proof-of-concept deployments or small deployments not requiring integration with a corporate authentication system, but it lacks some of the advanced functions usually found in user management systems such as forcing password changes. The focus of the Keystone native authentication system is to be the source of authentication for OpenStack-specific users required to operate various OpenStack services. These users are stored by Keystone in a default domain; the addition of these IDs to an external authentication system is not required.

Keystone is more commonly integrated with external authentication systems such as OpenLDAP or Microsoft Active Directory. These systems are usually centrally deployed by organizations to serve as the single source of user management and authentication for all in-house deployed applications and systems requiring user authentication. In addition to LDAP and Microsoft Active Directory, support for integration with Security Assertion Markup Language (SAML)-based identity providers from companies such as Ping, CA, IBM, Oracle, and others is also nearly "production-ready."

Keystone also provides architectural support through the underlying Apache deployment for other types of authentication systems, such as multi-factor authentication. These types of systems typically require driver support and integration from the respective providers.

☞ Note

> While support for Identity providers and multi-factor authentication is available in Keystone, it has not yet been certified by the HPE Helion OpenStack engineering team and is an experimental feature in HPE Helion OpenStack.

LDAP-compatible directories such as OpenLDAP and Microsoft Active Directory are recommended alternatives to using Keystone local authentication. Both methods are widely used by organizations and are integrated with a variety of other enterprise applications. These directories act as the single source of user information within an organization. You can configure Keystone to authenticate against an LDAP-compatible directory on a per-domain basis.

Domains, as explained in *Section 4.3, "Understanding Domains, Projects, Users, Groups, and Roles"*, can be configured so that, based on the user ID, an incoming user is automatically mapped to a specific domain. You can then configure this domain to authenticate against a specific LDAP

directory. User credentials provided by the user to Keystone are passed along to the designated LDAP source for authentication. You can optionally configure this communication to be secure through SSL encryption. No special LDAP administrative access is required, and only read-only access is needed for this configuration. Keystone will not add any LDAP information. All user additions, deletions, and modifications are performed by the application's front end in the LDAP directories. After a user has been successfully authenticated, that user is then assigned to the groups, roles, and projects defined by the Keystone domain or project administrators. This information is stored in the Keystone service database.

Another form of external authentication provided by the Keystone service is through integration with SAML-based identity providers (IdP) such as Ping Identity, IBM Tivoli, and Microsoft Active Directory Federation Server. A SAML-based identity provider provides authentication that is often called "single sign-on." The IdP server is configured to authenticate against identity sources such as Active Directory and provides a single authentication API against multiple types of downstream identity sources. This means that an organization could have multiple identity storage sources but a single authentication source. In addition, if a user has logged into one such source during a defined session time frame, that user does not need to reauthenticate within the defined session. Instead, the IdP automatically validates the user to requesting applications and services.

A SAML-based IdP authentication source is configured with Keystone on a per-domain basis similar to the manner in which native LDAP directories are configured. Extra mapping rules are required in the configuration that define which Keystone group an incoming UID is automatically assigned to. This means that groups need to be defined in Keystone first, but it also removes the requirement that a domain or project administrator assign user roles and project membership on a per-user basis. Instead, groups are used to define project membership and roles and incoming users are automatically mapped to Keystone groups based on their upstream group membership. This strategy provides a consistent role-based access control (RBAC) model based on the upstream identity source. The configuration of this option is fairly straightforward. IdP vendors such as Ping and IBM are contributing to the maintenance of this function and have also produced their own integration documentation. HPE is using the Microsoft Active Directory Federation Services (AD FS) for functional testing and future documentation.

The third Keystone-supported authentication source is known as multi-factor authentication (MFA). MFA typically requires an external source of authentication beyond a login name and password, and can include options such as SMS text, a temporal token generator, or a fingerprint

scanner. Each of these types of MFAs are usually specific to a particular MFA vendor. The Keystone architecture supports an MFA-based authentication system, but this has not yet been certified or documented for HPE Helion OpenStack.

4.5.4 Authorization

Another major function provided by the Keystone service is access authorization that determines which resources and actions are available based on the UserID, the role of the user, and the projects that a user is provided access to. All of this information is created, managed, and stored by Keystone. These functions are applied through the Horizon web interface, the OpenStack command-line interface, or the direct Keystone API.

Keystone provides support for organizing users by using three entities:

Domains

> Domains provide the highest level of organization. Domains are intended to be used as high-level containers for multiple projects. A domain can represent different tenants, companies, or organizations for an OpenStack cloud deployed for public cloud deployments or it can represent major business units, functions, or any other type of top-level organization unit in an OpenStack private cloud deployment. Each domain has at least one Domain Admin assigned to it. This Domain Admin can then create multiple projects within the domain and assign the project administrator role to specific project owners. Each domain created in an OpenStack deployment is unique and the projects assigned to a domain cannot exist in another domain.

Projects

> Projects are entities within a domain that represent groups of users, each user role within that project, and how many underlying infrastructure resources can be consumed by members of the project.

Groups

> Groups are an optional function and provide the means of assigning project roles to multiple users at once.

Keystone also makes it possible to create and assign roles to groups of users or individual users. Role names are created and user assignments are made within Keystone. The actual function of a role is defined currently for each OpenStack service via scripts. When users request access

to an OpenStack service, their access tokens contain information about their assigned project membership and role for that project. This role is then matched to the service-specific script and users are allowed to perform functions within that service defined by the role mapping.

4.5.5   Default settings

Identity service configuration settings

The identity service configuration options are described in the OpenStack documentation on the Keystone Configuration Options page (https://docs.openstack.org/keystone/pike/configuration/index.html) ↗ on the OpenStack site.

Default domain and service accounts

The "default" domain is automatically created during the installation to contain the various required OpenStack service accounts, including the following:

- neutron

- glance

- swift-monitor

- ceilometer

- swift

- monasca-agent

- glance-swift

- swift-demo

- nova

- monasca

- logging

- demo

- heat

- cinder

- admin

These are required accounts and are used by the underlying OpenStack services. These accounts should not be removed or reassigned to a different domain. These "default" domain should be used only for these service accounts.

For details on how to create additional users, see *Book "User Guide", Chapter 4 "Cloud Admin Actions with the Command Line"*.

4.5.6   Preinstalled roles

The following are the preinstalled roles. You can create additional roles by UIDs with the "admin" role. Roles are defined on a per-service basis (more information is available at Manage projects, users, and roles (http://docs.openstack.org/user-guide-admin/manage_projects_users_and_roles.html) ↗ on the OpenStack website).

Role	Description
admin	The "superuser" role. Provides full access to all HPE Helion OpenStack services across all domains and projects. This role should be given only to a cloud administrator.
member	A general role that enables a user to access resources within an assigned project including creating, modifying, and deleting compute, storage, and network resources.

You can find additional information on these roles in each service policy stored in the `/etc/PROJECT/policy.json` files where PROJECT is a placeholder for an OpenStack service. For example, the Compute (Nova) service roles are stored in the `/etc/nova/policy.json` file. Each service policy file defines the specific API functions available to a role label.

4.6   Retrieving the Admin Password

The admin password will be used to access the dashboard and Operations Console as well as allow you to authenticate to use the command-line tools and API.

In a default HPE Helion OpenStack 8 installation there is a randomly generated password for the Admin user created. These steps will show you how to retrieve this password.

4.6.1 Retrieving the Admin Password

You can retrieve the randomly generated Admin password by using this command on the Cloud Lifecycle Manager:

```
ardana > cat ~/service.osrc
```

In this example output, the value for `OS_PASSWORD` is the Admin password:

```
ardana > cat ~/service.osrc
unset OS_DOMAIN_NAME
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_VERSION=3
export OS_PROJECT_NAME=admin
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USERNAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PASSWORD=SlWSfwxuJY0
export OS_AUTH_URL=https://10.13.111.145:5000/v3
export OS_ENDPOINT_TYPE=internalURL
# OpenstackClient uses OS_INTERFACE instead of OS_ENDPOINT
export OS_INTERFACE=internal
export OS_CACERT=/etc/ssl/certs/ca-certificates.crt
export OS_COMPUTE_API_VERSION=2
```

4.7 Changing Service Passwords

HPE Helion OpenStack provides a process for changing the default service passwords, including your admin user password, which you may want to do for security or other purposes.

You can easily change the inter-service passwords used for authenticating communications between services in your HPE Helion OpenStack deployment, promoting better compliance with your organization's security policies. The inter-service passwords that can be changed include (but are not limited to) Keystone, MariaDB, RabbitMQ, Cloud Lifecycle Manager cluster, Monasca and Barbican.

The general process for changing the passwords is to:

- Indicate to the configuration processor which password(s) you want to change, and optionally include the value of that password

- Run the configuration processor to generate the new passwords (you do not need to run `git add` before this)

- Run ready-deployment

- Check your password name(s) against the tables included below to see which high-level credentials-change playbook(s) you need to run

- Run the appropriate high-level credentials-change playbook(s)

4.7.1   Password Strength

Encryption passwords supplied to the configuration processor for use with Ansible Vault and for encrypting the configuration processor's persistent state must have a minimum length of 12 characters and a maximum of 128 characters. Passwords must contain characters from each of the following three categories:

- Uppercase characters (A-Z)

- Lowercase characters (a-z)

- Base 10 digits (0-9)

Service Passwords that are automatically generated by the configuration processor are chosen from the 62 characters made up of the 26 uppercase, the 26 lowercase, and the 10 numeric characters, with no preference given to any character or set of characters, with the minimum and maximum lengths being determined by the specific requirements of individual services.

Important

It is possible to use special characters in passwords. However, the `$` character must be escaped by entering it twice. For example: the password `foo$bar` must be specified as `foo$$bar`.

4.7.2   Telling the configuration processor which password(s) you want to change

In HPE Helion OpenStack 8, the configuration processor will produce metadata about each of the passwords (and other variables) that it generates in the file `~/openstack/my_cloud/info/private_data_metadata_ccp.yml`. A snippet of this file follows. Expand the header to see the file:

4.7.3   private_data_metadata_ccp.yml

```
metadata_proxy_shared_secret:
  metadata:
  - clusters:
    - cluster1
    component: nova-metadata
    consuming-cp: ccp
    cp: ccp
  version: '2.0'
mysql_admin_password:
  metadata:
  - clusters:
    - cluster1
    component: ceilometer
    consumes: mysql
    consuming-cp: ccp
    cp: ccp
  - clusters:
    - cluster1
    component: heat
    consumes: mysql
    consuming-cp: ccp
    cp: ccp
  - clusters:
    - cluster1
    component: keystone
    consumes: mysql
    consuming-cp: ccp
    cp: ccp
  - clusters:
    - cluster1
    - compute
    component: nova
    consumes: mysql
    consuming-cp: ccp
    cp: ccp
  - clusters:
    - cluster1
    component: cinder
    consumes: mysql
    consuming-cp: ccp
    cp: ccp
  - clusters:
    - cluster1
    component: glance
    consumes: mysql
```

```
      consuming-cp: ccp
      cp: ccp
    - clusters:
      - cluster1
      - compute
      component: neutron
      consumes: mysql
      consuming-cp: ccp
      cp: ccp
    - clusters:
      - cluster1
      component: horizon
      consumes: mysql
      consuming-cp: ccp
      cp: ccp
    version: '2.0'
mysql_barbican_password:
  metadata:
    - clusters:
      - cluster1
      component: barbican
      consumes: mysql
      consuming-cp: ccp
      cp: ccp
    version: '2.0'
```

For each variable, there is a metadata entry for each pair of services that use the variable including a list of the clusters on which the service component that consumes the variable (defined as "component:" in `private_data_metadata_ccp.yml` above) runs.

Note above that the variable `mysql_admin_password` is used by a number of service components, and the service that is consumed in each case is `mysql`, which in this context refers to the MariaDB instance that is part of the product.

4.7.4 Steps to change a password

First, make sure that you have a copy of `private_data_metadata_ccp.yml`. If you do not, generate one to run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

Make a copy of the `private_data_metadata_ccp.yml` file and place it into the `~/openstack/change_credentials` directory:

```
ardana > cp ~/openstack/my_cloud/info/private_data_metadata_control-plane-1.yml \
 ~/openstack/change_credentials/
```

Edit the copied file in `~/openstack/change_credentials` leaving only those passwords you intend to change. All entries in this template file should be deleted *except for those passwords*.

🔔 Important

If you leave other passwords in that file that you do *not* want to change, they will be regenerated and no longer match those in use which could disrupt operations.

🎓 Note

It is required that you change passwords in batches of each category listed below.

For example, the snippet below would result in the configuration processor generating new random values for keystone_backup_password, keystone_ceilometer_password, and keystone_cinder_password:

```
keystone_backup_password:
  metadata:
  - clusters:
    - cluster0
    - cluster1
    - compute
    component: freezer-agent
    consumes: keystone-api
    consuming-cp: ccp
    cp: ccp
  version: '2.0'
keystone_ceilometer_password:
  metadata:
  - clusters:
    - cluster1
    component: ceilometer-common
    consumes: keystone-api
    consuming-cp: ccp
    cp: ccp
  version: '2.0'
```

```
keystone_cinder_password:
  metadata:
  - clusters:
    - cluster1
    component: cinder-api
    consumes: keystone-api
    consuming-cp: ccp
    cp: ccp
  version: '2.0'
```

4.7.5 Specifying password value

Optionally, you can specify a value for the password by including a "value:" key and value at the same level as metadata:

```
keystone_backup_password:
    value: 'new_password'
    metadata:
    - clusters:
        - cluster0
        - cluster1
        - compute
      component: freezer-agent
      consumes: keystone-api
      consuming-cp: ccp
      cp: ccp
    version: '2.0'
```

Note that you can have multiple files in openstack/change_credentials. The configuration processor will only read files that end in .yml or .yaml.

☞ Note

If you have specified a password value in your credential change file, you may want to encrypt it using ansible-vault. If you decide to encrypt with ansible-vault, make sure that you use the encryption key you have already used when running the configuration processor.

To encrypt a file using ansible-vault, execute:

```
ardana > cd ~/openstack/change_credentials
ardana > ansible-vault encrypt credential change file ending in .yml or .yaml
```

Be sure to provide the encryption key when prompted. Note that if you have specified the wrong ansible-vault password, the configuration-processor will error out with a message like the following:

```
################################################# Reading Persistent State
 ################################################

################################################################################
# The configuration processor failed.
# PersistentStateCreds: User-supplied creds file test1.yml was not parsed properly
################################################################################
```

4.7.6 Running the configuration processor to change passwords

The directory openstack/change_credentials is not managed by git, so to rerun the configuration processor to generate new passwords and prepare for the next deployment just enter the following commands:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

☞ Note

The files that you placed in `~/openstack/change_credentials` should be removed once you have run the configuration processor because the old password values and new password values will be stored in the configuration processor's persistent state.

Note that if you see output like the following after running the configuration processor:

```
################################################################################
# The configuration processor completed with warnings.
# PersistentStateCreds: User-supplied password name 'blah' is not valid
################################################################################
```

this tells you that the password name you have supplied, 'blah,' does not exist. A failure to correctly parse the credentials change file will result in the configuration processor erroring out with a message like the following:

```
################################################# Reading Persistent State
 ################################################
```

```
#############################################################################
# The configuration processor failed.
# PersistentStateCreds: User-supplied creds file test1.yml was not parsed properly
#############################################################################
```

Once you have run the configuration processor to change passwords, an information file `~/openstack/my_cloud/info/password_change.yml` similar to the `private_data_metadata_ccp.yml` is written to tell you which passwords have been changed, including metadata but not including the values.

4.7.7 Password change playbooks and tables

Once you have completed the steps above to change password(s) value(s) and then prepare for the deployment that will actually switch over to the new passwords, you will need to run some high-level playbooks. The passwords that can be changed are grouped into six categories. The tables below list the password names that belong in each category. The categories are:

Keystone

Playbook: ardana-keystone-credentials-change.yml

RabbitMQ

Playbook: ardana-rabbitmq-credentials-change.yml

MariaDB

Playbook: ardana-reconfigure.yml

Cluster:

Playbook: ardana-cluster-credentials-change.yml

Monasca:

Playbook: monasca-reconfigure-credentials-change.yml

Other:

Playbook: ardana-other-credentials-change.yml

It is recommended that you change passwords in batches; in other words, run through a complete password change process for each batch of passwords, preferably in the above order. Once you have followed the process indicated above to change password(s), check the names against the tables below to see which password change playbook(s) you should run.

Changing identity service credentials

The following table lists identity service credentials you can change.

Keystone credentials

```
Password name
barbican_admin_password
barbican_service_password
keystone_admin_pwd
keystone_admin_token
keystone_backup_password
keystone_ceilometer_password
keystone_cinder_password
keystone_cinderinternal_password
keystone_demo_pwd
keystone_designate_password
keystone_freezer_password
keystone_glance_password
keystone_glance_swift_password
keystone_heat_password
keystone_magnum_password
keystone_monasca_agent_password
keystone_monasca_password
keystone_neutron_password
keystone_nova_password
keystone_octavia_password
keystone_swift_dispersion_password
keystone_swift_monitor_password
keystone_swift_password
logging_keystone_password
nova_monasca_password
```

The playbook to run to change Keystone credentials is `ardana-keystone-credentials-change.yml`. Execute the following commands to make the changes:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts ardana-keystone-credentials-change.yml
```

Changing RabbitMQ credentials

The following table lists the RabbitMQ credentials you can change.

RabbitMQ credentials

```
Password name
ops_mon_rmq_password
rmq_barbican_password
```

<table>
<tr><td>RabbitMQ credentials</td></tr>
<tr><td>

```
rmq_ceilometer_password
rmq_cinder_password
rmq_designate_password
rmq_keystone_password
rmq_magnum_password
rmq_monasca_monitor_password
rmq_nova_password
rmq_octavia_password
rmq_service_password
```
</td></tr>
</table>

The playbook to run to change RabbitMQ credentials is `ardana-rabbitmq-credentials-change.yml`. Execute the following commands to make the changes:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts ardana-rabbitmq-credentials-change.yml
```

Changing MariaDB credentials

The following table lists the MariaDB credentials you can change.

<table>
<tr><td>MariaDB credentials</td></tr>
<tr><td>

```
Password name
mysql_admin_password
mysql_barbican_password
mysql_clustercheck_pwd
mysql_designate_password
mysql_magnum_password
mysql_monasca_api_password
mysql_monasca_notifier_password
mysql_monasca_thresh_password
mysql_octavia_password
mysql_powerdns_password
mysql_root_pwd
mysql_service_pwd
mysql_sst_password
ops_mon_mdb_password
mysql_monasca_transform_password
mysql_nova_api_password
password
```
</td></tr>
</table>

The playbook to run to change MariaDB credentials is `ardana-reconfigure.yml`. To make the changes, execute the following commands:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
```

```
ardana > ansible-playbook -i hosts/verb_hosts ardana-reconfigure.yml
```

Changing cluster credentials

The following table lists the cluster credentials you can change.

cluster credentials

Password name
haproxy_stats_password
keepalive_vrrp_password

The playbook to run to change cluster credentials is `ardana-cluster-credentials-change.yml`. To make changes, execute the following commands:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts ardana-cluster-credentials-change.yml
```

Changing Monasca credentials

The following table lists the Monasca credentials you can change.

Monasca credentials

Password name
mysql_monasca_api_password
mysql_monasca_persister_password
monitor_user_password
cassandra_monasca_api_password
cassandra_monasca_persister_password

The playbook to run to change Monasca credentials is `monasca-reconfigure-credentials-change.yml`. To make the changes, execute the following commands:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts monasca-reconfigure-credentials-change.yml
```

Changing other credentials

The following table lists the other credentials you can change.

Other credentials

Password name
logging_beaver_password
logging_api_password
logging_monitor_password
logging_kibana_password

The playbook to run to change these credentials is `ardana-other-credentials-change.yml`. To make the changes, execute the following commands:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts ardana-other-credentials-change.yml
```

4.7.8 Changing RADOS Gateway Credential

To change the keystone credentials of RADOS Gateway, follow the preceding steps documented in *Section 4.7, "Changing Service Passwords"* by modifying the `keystone_rgw_password` section in `private_data_metadata_ccp.yml` file in *Section 4.7.4, "Steps to change a password"* or *Section 4.7.5, "Specifying password value"*.

4.7.9 Immutable variables

The values of certain variables are immutable, which means that once they have been generated by the configuration processor they cannot be changed. These variables are:

- barbican_master_kek_db_plugin

- swift_hash_path_suffix

- swift_hash_path_prefix

- mysql_cluster_name

- heartbeat_key

- erlang_cookie

The configuration processor will not re-generate the values of the above passwords, nor will it allow you to specify a value for them. In addition to the above variables, the following are immutable in HPE Helion OpenStack 8:

- All ssh keys generated by the configuration processor

- All UUIDs generated by the configuration processor

- metadata_proxy_shared_secret

- horizon_secret_key

- ceilometer_metering_secret

4.8 Reconfiguring the Identity Service

4.8.1 Updating the Keystone Identity Service

This topic explains configuration options for the Identity service.

HPE Helion OpenStack lets you perform updates on the following parts of the Identity service configuration:

- Any content in the main keystone configuration file: `/etc/keystone/keystone.conf`. This lets you manipulate Keystone configuration parameters. Next, continue with *Section 4.8.2, "Updating the Main Identity Service Configuration File"*.

- Updating certain configuration options and enabling features, such as:

    - Verbosity of logs being written to Keystone log files.

    - Process counts for the Apache2 WSGI module, separately for admin and public Keystone interfaces.

    - Enabling/disabling auditing.

    - Enabling/disabling Fernet tokens.

    For more information, see *Section 4.8.3, "Enabling Identity Service Features"*.

- Creating and updating domain-specific configuration files: **/etc/keystone/domains/ keystone.<domain_name>.conf**. This lets you integrate Keystone with one or more external authentication sources, such as LDAP server. See the topic on *Section 4.9, "Integrating LDAP with the Identity Service"*.

4.8.2 Updating the Main Identity Service Configuration File

1. The main Keystone Identity service configuration file (**/etc/keystone/keystone.conf**), located on each control plane server, is generated from the following template file located on a Cloud Lifecycle Manager: `~/openstack/my_cloud/config/keystone/ keystone.conf.j2`

Modify this template file as appropriate. See Keystone Liberty documentation (http://docs.openstack.org/liberty/config-reference/content/keystone-configuration-file.html) ↗ for full descriptions of all settings. This is a Jinja2 template, which expects certain template variables to be set. Do not change values inside double curly braces: `{{ }}`.

☞ Note

HPE Helion OpenStack 8 has the following token expiration setting, which differs from the upstream value `3600`:

```
[token]
expiration = 14400
```

2. After you modify the template, commit the change to the local git repository, and rerun the configuration processor / deployment area preparation playbooks (as suggested in *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*):

```
ardana > cd ~/openstack
ardana > git checkout site
ardana > git add my_cloud/config/keystone/keystone.conf.j2
ardana > git commit -m "Adjusting some parameters in keystone.conf"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Run the reconfiguration playbook in the deployment area:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

4.8.3 Enabling Identity Service Features

To enable or disable Keystone features, do the following:

1. Adjust respective parameters in **~/openstack/my_cloud/config/keystone/keystone_deploy_config.yml**

2. Commit the change into local git repository, and rerun the configuration processor/deployment area preparation playbooks (as suggested in *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*):

```
ardana > cd ~/openstack
ardana > git checkout site
ardana > git add my_cloud/config/keystone/keystone_deploy_config.yml
ardana > git commit -m "Adjusting some WSGI or logging parameters for keystone"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Run the reconfiguration playbook in the deployment area:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

4.8.4 Fernet Tokens

HPE Helion OpenStack 8 supports Fernet tokens by default. The benefit of using Fernet tokens is that tokens are not persisted in a database, which is helpful if you want to deploy the Keystone Identity service as one master and multiple slaves; only roles, projects, and other details will need to be replicated from master to slaves, not the token table.

🎓 Note

Tempest does not work with Fernet tokens in HPE Helion OpenStack 8. If Fernet tokens are enabled, do not run token tests in Tempest.

🎓 Note

During reconfiguration when switching to a Fernet token provider or during Fernet key rotation, you may see a warning in `keystone.log` stating `[fernet_tokens]` `key_repository is world readable: /etc/keystone/fernet-keys/`. This is

expected. You can safely ignore this message. For other Keystone operations, you will not see this warning. Directory permissions are actually set to 600 (read/write by owner only), not world readable.

Fernet token-signing key rotation is being handled by a cron job, which is configured on one of the controllers. The controller with the Fernet token-signing key rotation cron job is also known as the Fernet Master node. By default, the Fernet token-signing key is being rotated once every 24 hours. The Fernet token-signing keys are distributed from the Fernet Master node to the rest of the controllers at each rotation. Therefore, the Fernet token-signing keys are consistent for all the controlers at all time.

When enabling Fernet token provider the first time, specific steps are needed to set up the necessary mechanisms for Fernet token-signing key distributions.

1. Set `keystone_configure_fernet` to `True` in `~/openstack/my_cloud/config/keystone/keystone_deploy_config.yml`.

2. Run the following commands to commit your change in Git and enable Fernet:

```
ardana > git add my_cloud/config/keystone/keystone_deploy_config.yml
ardana > git commit -m "enable Fernet token provider"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-deploy.yml
```

When the Fernet token provider is enabled, a Fernet Master alarm definition is also created on Monasca to monitor the Fernet Master node. If the Fernet Master node is offline or unreachable, a `CRITICAL` alarm will be raised for the Cloud Admin to take corrective actions. If the Fernet Master node is offline for a prolonged period of time, Fernet token-signing key rotation will not be performed. This may introduce security risks to the cloud. The Cloud Admin must take immediate actions to resurrect the Fernet Master node.

4.9 Integrating LDAP with the Identity Service

4.9.1 Integrating with an external LDAP server

The Keystone identity service provides two primary functions: user authentication and access authorization. The user authentication function validates a user's identity. Keystone has a very basic user management system that can be used to create and manage user login and password credentials but this system is intended only for proof of concept deployments due to the very limited password control functions. The internal identity service user management system is also commonly used to store and authenticate OpenStack-specific service account information.

The recommended source of authentication is external user management systems such as LDAP directory services. The identity service can be configured to connect to and use external systems as the source of user authentication. The identity service domain construct is used to define different authentication sources based on domain membership. For example, cloud deployment could consist of as few as two domains:

- The default domain that is pre-configured for the service account users that are authenticated directly against the identity service internal user management system

- A customer-defined domain that contains all user projects and membership definitions. This domain can then be configured to use an external LDAP directory such as Microsoft Active Directory as the authentication source.

HPE Helion OpenStack can support multiple domains for deployments that support multiple tenants. Multiple domains can be created with each domain configured to either the same or different external authentication sources. This deployment model is known as a "per-domain" model.

There are currently two ways to configure "per-domain" authentication sources:

- File store – each domain configuration is created and stored in separate text files. This is the older and current default method for defining domain configurations.

- Database store – each domain configuration can be created using either the identity service manager utility (recommended) or a Domain Admin API (http://developer.openstack.org/api-ref-identity-v3.html#domains-config-v3) ↗ (from OpenStack.org), and the results are stored in the identity service MariaDB database. This database store is a new method introduced in the OpenStack Kilo release and now available in HPE Helion OpenStack.

Instructions for initially creating per-domain configuration files and then migrating to the Database store method via the identity service manager utility are provided as follows.

4.9.2 Set up domain-specific driver configuration - file store

To update configuration to a specific LDAP domain:

1. Ensure that the following configuration options are in the main configuration file template: ~/openstack/my_cloud/config/keystone/keystone.conf.j2

```
[identity]
domain_specific_drivers_enabled = True
domain_configurations_from_database = False
```

2. Create a YAML file that contains the definition of the LDAP server connection. The sample file below is already provided as part of the Cloud Lifecycle Manager in the *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*. It is available on the Cloud Lifecycle Manager in the following file:

```
~/openstack/my_cloud/config/keystone/keystone_configure_ldap_sample.yml
```

Save a copy of this file with a new name, for example:

```
~/openstack/my_cloud/config/keystone/keystone_configure_ldap_my.yml
```

> **⫷ Note**
>
> Please refer to the LDAP section of the Keystone (https://github.com/openstack/keystone/blob/stable/pike/etc/keystone.conf.sample)↗ configuration example for OpenStack for the full option list and description.

Below are samples of YAML configurations for identity service LDAP certificate settings, optimized for Microsoft Active Directory server.
Sample YAML configuration keystone_configure_ldap_my.yml

```
---
keystone_domainldap_conf:

    # CA certificates file content.
    # Certificates are stored in Base64 PEM format. This may be entire LDAP server
```

```
    # certificate (in case of self-signed certificates), certificate of authority
    # which issued LDAP server certificate, or a full certificate chain (Root CA
    # certificate, intermediate CA certificate(s), issuer certificate).
    #
    cert_settings:
      cacert: |
        -----BEGIN CERTIFICATE-----

        certificate appears here

        -----END CERTIFICATE-----

    # A domain will be created in MariaDB with this name, and associated with ldap
 back end.
    # Installer will also generate a config file named /etc/keystone/domains/
keystone.<domain_name>.conf
    #
    domain_settings:
      name: ad
      description: Dedicated domain for ad users


    conf_settings:
      identity:
        driver: ldap



      # For a full list and description of ldap configuration options, please refer
 to
      # https://github.com/openstack/keystone/blob/master/etc/keystone.conf.sample
 or
      # http://docs.openstack.org/liberty/config-reference/content/keystone-
configuration-file.html.
      #
      # Please note:
      #  1. LDAP configuration is read-only. Configuration which performs write
 operations (i.e. creates users, groups, etc)
      #     is not supported at the moment.
      #  2. LDAP is only supported for identity operations (reading users and groups
 from LDAP). Assignment
      #     operations with LDAP (i.e. managing roles, projects) are not supported.
      #  3. LDAP is configured as non-default domain. Configuring LDAP as a default
 domain is not supported.
      #
      ldap:
        url: ldap://ad.hpe.net
        suffix: DC=hpe,DC=net
        query_scope: sub
```

```
        user_tree_dn: CN=Users,DC=hpe,DC=net
        user : CN=admin,CN=Users,DC=hpe,DC=net
        password: REDACTED
        user_objectclass: user
        user_id_attribute: cn
        user_name_attribute: cn
        group_tree_dn: CN=Users,DC=hpe,DC=net
        group_objectclass: group
        group_id_attribute: cn
        group_name_attribute: cn
        use_pool: True
        user_enabled_attribute: userAccountControl
        user_enabled_mask: 2
        user_enabled_default: 512
        use_tls: True
        tls_req_cert: demand
        # if you are configuring multiple LDAP domains, and LDAP server certificates
are issued
        # by different authorities, make sure that you place certs for all the LDAP
backend domains in the
        # cacert parameter as seen in this sample yml file so that all the certs are
combined in a single CA file
        # and every LDAP domain configuration points to the combined CA file.
        # Note:
        # 1. Please be advised that every time a new ldap domain is configured, the
single CA file gets overwritten
        # and hence ensure that you place certs for all the LDAP backend domains in
the cacert parameter.
        # 2. There is a known issue on one cert per CA file per domain when the
system processes
        # concurrent requests to multiple LDAP domains. Using the single CA file
with all certs combined
        # shall get the system working properly*.

        tls_cacertfile: /etc/keystone/ssl/certs/all_ldapdomains_ca.pem

        # The issue is in the underlying SSL library. Upstream is not investing in
python-ldap package anymore.
        # It is also not python3 compliant.
```

```
keystone_domain_MSAD_conf:

    # CA certificates file content.
    # Certificates are stored in Base64 PEM format. This may be entire LDAP server
    # certificate (in case of self-signed certificates), certificate of authority
    # which issued LDAP server certificate, or a full certificate chain (Root CA
    # certificate, intermediate CA certificate(s), issuer certificate).
```

```
    #
    cert_settings:
      cacert: |
        -----BEGIN CERTIFICATE-----

        certificate appears here

        -----END CERTIFICATE-----

    # A domain will be created in MariaDB with this name, and associated with ldap
 back end.
    # Installer will also generate a config file named /etc/keystone/domains/
keystone.<domain_name>.conf
    #
        domain_settings:
          name: msad
          description: Dedicated domain for msad users

        conf_settings:
          identity:
            driver: ldap


    # For a full list and description of ldap configuration options, please refer to
    # https://github.com/openstack/keystone/blob/master/etc/keystone.conf.sample or
    # http://docs.openstack.org/liberty/config-reference/content/keystone-
configuration-file.html.
    #
    # Please note:
    #  1. LDAP configuration is read-only. Configuration which performs write
 operations (i.e. creates users, groups, etc)
    #     is not supported at the moment.
    #  2. LDAP is only supported for identity operations (reading users and groups
 from LDAP). Assignment
    #     operations with LDAP (i.e. managing roles, projects) are not supported.
    #  3. LDAP is configured as non-default domain. Configuring LDAP as a default
 domain is not supported.
    #
    ldap:
      # If the url parameter is set to ldap then typically use_tls should be set to
 True. If
      # url is set to ldaps, then use_tls should be set to False
      url: ldaps://10.16.22.5
      use_tls: False
      query_scope: sub
      user_tree_dn: DC=l3,DC=local
      # this is the user and password for the account that has access to the AD
 server
```

```
      user: administrator@l3.local
      password: OpenStack123
      user_objectclass: user
      # For a default Active Directory schema this is where to find the user name,
openldap uses a different value
      user_id_attribute: userPrincipalName
      user_name_attribute: sAMAccountName
      group_tree_dn: DC=l3,DC=local
      group_objectclass: group
      group_id_attribute: cn
      group_name_attribute: cn
      # An upstream defect requires use_pool to be set false
      use_pool: False
      user_enabled_attribute: userAccountControl
      user_enabled_mask: 2
      user_enabled_default: 512
      tls_req_cert: allow
      # Referals may contain urls that can't be resolved and will cause timeouts,
ignore them
      chase_referrals: False
      # if you are configuring multiple LDAP domains, and LDAP server certificates
are issued
      # by different authorities, make sure that you place certs for all the LDAP
backend domains in the
      # cacert parameter as seen in this sample yml file so that all the certs are
combined in a single CA file
      # and every LDAP domain configuration points to the combined CA file.
      # Note:
      # 1. Please be advised that every time a new ldap domain is configured, the
single CA file gets overwritten
      # and hence ensure that you place certs for all the LDAP backend domains in
the cacert parameter.
      # 2. There is a known issue on one cert per CA file per domain when the system
processes
      # concurrent requests to multiple LDAP domains. Using the single CA file with
all certs combined
      # shall get the system working properly.

      tls_cacertfile: /etc/keystone/ssl/certs/all_ldapdomains_ca.pem
```

3. As suggested in *Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*, commit the new file to the local git repository, and rerun the configuration processor and ready deployment playbooks:

```
ardana > cd ~/openstack
ardana > git checkout site
```

```
ardana > git add my_cloud/config/keystone/keystone_configure_ldap_my.yml
ardana > git commit -m "Adding LDAP server integration config"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

4. Run the reconfiguration playbook in a deployment area, passing the YAML file created in the previous step as a command-line option:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml -e@~/
openstack/my_cloud/config/keystone/keystone_configure_ldap_my.yml
```

5. Follow these same steps for each LDAP domain with which you are integrating the identity service, creating a YAML file for each and running the reconfigure playbook once for each additional domain.

6. Ensure that a new domain was created for LDAP (Microsoft AD in this example) and set environment variables for admin level access

```
ardana > source keystone.osrc
```

Get a list of domains

```
ardana > openstack domain list
```

As output here:

```
+----------------------------------+---------+---------
+----------------------------------------------------------------------+
| ID                               | Name    | Enabled | Description
                                  |
+----------------------------------+---------+---------
+----------------------------------------------------------------------+
| 6740dbf7465a4108a36d6476fc967dbd | heat    | True    | Owns users and projects
 created by heat                              |
| default                          | Default | True    | Owns users and tenants
 (i.e. projects) available on Identity API v2. |
| b2aac984a52e49259a2bbf74b7c4108b | ad      | True    | Dedicated domain for users
 managed by Microsoft AD server          |
+----------------------------------+---------+---------
+----------------------------------------------------------------------+
```

> **☞ Note**
>
> LDAP domain is read-only. This means that you cannot create new user or group records in it.

7. Once the LDAP user is granted the appropriate role, he can authenticate within the specified domain. Set environment variables for admin-level access

```
ardana > source keystone.osrc
```

Get user record within the ad (Active Directory) domain

```
ardana > openstack user show testuser1 --domain ad
```

Note the output:

```
+-----------+------------------------------------------------------------------+
| Field     | Value                                                            |
+-----------+------------------------------------------------------------------+
| domain_id | 143af847018c4dc7bd35390402395886                                 |
| id        | e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 |
| name      | testuser1                                                        |
+-----------+------------------------------------------------------------------+
```

Now, get list of LDAP groups:

```
ardana > openstack group list --domain ad
```

Here you see testgroup1 and testgroup2:

```
+------------------------------------------------------------------+------------+
|  ID                                                              | Name       |
+------------------------------------------------------------------+------------+
|  03976b0ea6f54a8e4c0032e8f756ad581f26915c7e77500c8d4aaf0e83afcdc6| testgroup1 |
7ba52ee1c5829d9837d740c08dffa07ad118ea1db2d70e0dc7fa7853e0b79fcf   | testgroup2 |
+------------------------------------------------------------------+------------+
```

Create a new role. Note that the role is not bound to the domain.

```
ardana > openstack role create testrole1
```

Testrole1 has been created:

```
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 02251585319d459ab847409dea527dee |
| name  | testrole1                        |
+-------+----------------------------------+
```

Grant the user a role within the domain by executing the code below. Note that due to a current OpenStack CLI limitation, you must use the user ID rather than the user name when working with a non-default domain.

```
ardana > openstack role add testrole1 --user
 e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 --domain ad
```

Verify that the role was successfully granted, as shown here:

```
ardana > openstack role assignment list --user
 e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 --domain ad
+---------------------------------
+-----------------------------------------------------------------+-------
+---------+----------------------------------+
| Role                             | User
                | Group | Project | Domain                           |
+---------------------------------
+-----------------------------------------------------------------+-------
+---------+----------------------------------+
| 02251585319d459ab847409dea527dee |
 e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 |       |
 | 143af847018c4dc7bd35390402395886 |
+---------------------------------
+-----------------------------------------------------------------+-------
+---------+----------------------------------+
```

Authenticate (get a domain-scoped token) as a new user with a new role. The --os-* command-line parameters specified below override the respective OS_* environment variables set by the keystone.osrc script to provide admin access. To ensure that the command below is executed in a clean environment, you may want log out from the node and log in again.

```
ardana > openstack --os-identity-api-version 3 \
          --os-username testuser1 \
```

```
            --os-password testuser1_password \
            --os-auth-url http://10.0.0.6:35357/v3 \
            --os-domain-name ad \
            --os-user-domain-name ad \
            token issue
```

Here is the result:

```
+-----------+----------------------------------------------------------------+
| Field     | Value                                                          |
+-----------+----------------------------------------------------------------+
| domain_id | 143af847018c4dc7bd35390402395886                               |
| expires   | 2015-09-09T21:36:15.306561Z                                    |
| id        | 6f8f9f1a932a4d01b7ad9ab061eb0917                               |
| user_id   | e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 |
+-----------+----------------------------------------------------------------+
```

8. Users can also have a project within the domain and get a project-scoped token. To accomplish this, set environment variables for admin level access:

```
ardana > source keystone.osrc
```

Then create a new project within the domain:

```
ardana > openstack project create testproject1 --domain ad
```

The result shows that they have been created:

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | 143af847018c4dc7bd35390402395886 |
| enabled     | True                             |
| id          | d065394842d34abd87167ab12759f107 |
| name        | testproject1                     |
+-------------+----------------------------------+
```

Grant the user a role with a project, re-using the role created in the previous example. Note that due to a current OpenStack CLI limitation, you must use user ID rather than user name when working with a non-default domain.

```
ardana > openstack role add testrole1 --user
 e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 --project
 testproject1
```

Verify that the role was successfully granted by generating a list:

```
ardana > openstack role assignment list --user
 e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 --project
 testproject1
```

The output shows the result:

```
+--------------------------------
+-------------------------------------------------------------------+-------
+-------------------------------+--------+
| Role                           | User
             | Group | Project                                | Domain |
+--------------------------------
+-------------------------------------------------------------------+-------
+-------------------------------+--------+
| 02251585319d459ab847409dea527dee |
 e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 |        |
 d065394842d34abd87167ab12759f107 |        |
+--------------------------------
+-------------------------------------------------------------------+-------
+-------------------------------+--------+
```

Authenticate (get a project-scoped token) as the new user with a new role. The --os-*
command line parameters specified below override their respective OS_* environment
variables set by keystone.osrc to provide admin access. To ensure that the command below
is executed in a clean environment, you may want log out from the node and log in again.
Note that both the --os-project-domain-name and --os-project-user-name parameters are
needed to verify that both user and project are not in the default domain.

```
ardana > openstack --os-identity-api-version 3 \
          --os-username testuser1 \
          --os-password testuser1_password \
          --os-auth-url http://10.0.0.6:35357/v3 \
          --os-project-name testproject1 \
          --os-project-domain-name ad \
          --os-user-domain-name ad \
          token issue
```

Below is the result:

```
+------------+------------------------------------------------------------------+
| Field      | Value                                                            |
+------------+------------------------------------------------------------------+
```

```
| expires    | 2015-09-09T21:50:49.945893Z                                       |
| id         | 328e18486f69441fb13f4842423f52d1                                  |
| project_id | d065394842d34abd87167ab12759f107                                  |
| user_id    | e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 |
+------------+-------------------------------------------------------------------+
```

4.9.3 Set up or switch to domain-specific driver configuration using a database store

To make the switch, execute the steps below. Remember, you must have already set up the configuration for a file store as explained in *Section 4.9.2, "Set up domain-specific driver configuration - file store"*, and it must be working properly.

1. Ensure that the following configuration options are set in the main configuration file, ~/openstack/my_cloud/config/keystone/keystone.conf.j2:

   ```
   [identity]
   domain_specific_drivers_enabled = True
   domain_configurations_from_database = True

   [domain_config]
   driver = sql
   ```

2. Once the template is modified, commit the change to the local git repository, and rerun the configuration processor / deployment area preparation playbooks (as suggested at Using Git for Configuration Management):

   ```
   ardana > cd ~/openstack
   ardana > git checkout site
   ardana > git add -A
   ```

   Verify that the files have been added using git status:

   ```
   ardana > git status
   ```

   Then commit the changes:

   ```
   ardana > git commit -m "Use Domain-Specific Driver Configuration - Database Store:
    more description here..."
   ```

Next, run the configuration processor and ready deployment playbooks:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Run the reconfiguration playbook in a deployment area:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

4. Upload the domain-specific config files to the database if they have not been loaded. If they have already been loaded and you want to switch back to database store mode, then skip this upload step and move on to step 5.

   a. Go to one of the controller nodes where Keystone is deployed.

   b. Verify that domain-specific driver configuration files are located under the directory (default /etc/keystone/domains) with the format: keystone.<domain name>.conf Use the Keystone manager utility to load domain-specific config files to the database. There are two options for uploading the files:

      i. Option 1: Upload all configuration files to the SQL database:

      ```
      ardana > keystone-manage domain_config_upload --all
      ```

      ii. Option 2: Upload individual domain-specific configuration files by specifying the domain name one by one:

      ```
      ardana > keystone-manage domain_config_upload --domain-name domain name
      ```

      Here is an example:

      ```
      keystone-manage domain_config_upload --domain-name ad
      ```

Note that the Keystone manager utility does not upload the domain-specific driver configuration file the second time for the same domain. For the management of the domain-specific driver configuration in the database store, you may refer to OpenStack Identity API - Domain Configuration (http://developer.openstack.org/api-ref-identity-v3.html#domains-config-v3) ⌝.

5. Verify that the switched domain driver configuration for LDAP (Microsoft AD in this example) in the database store works properly. Then set the environment variables for admin level access:

```
ardana > source ~/keystone.osrc
```

Get a list of domain users:

```
ardana > openstack user list --domain ad
```

Note the three users returned:

```
+----------------------------------------------------------------+------------+
| ID                                                             | Name       |
+----------------------------------------------------------------+------------+
| e7dbec51ecaf07906bd743debcb49157a0e8af557b860a7c1dadd454bdab03fe | testuser1  |
| 8a09630fde3180c685e0cd663427e8638151b534a8a7ccebfcf244751d6f09bd | testuser2  |
| ea463d778dadcefdcfd5b532ee122a70dce7e790786678961420ae007560f35e | testuser3  |
+----------------------------------------------------------------+------------+
```

Get user records within the ad domain:

```
ardana > openstack user show testuser1 --domain ad
```

Here testuser1 is returned:

```
+-----------+------------------------------------------------------------------+
| Field     | Value                                                            |
+-----------+------------------------------------------------------------------+
| domain_id | 143af847018c4dc7bd35390402395886                                 |
| id        | e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 |
| name      | testuser1                                                        |
+-----------+------------------------------------------------------------------+
```

Get a list of LDAP groups:

```
ardana > openstack group list --domain ad
```

Note that testgroup1 and testgroup2 are returned:

```
+----------------------------------------------------------------+------------+
| ID                                                             | Name       |
+----------------------------------------------------------------+------------+
| 03976b0ea6f54a8e4c0032e8f756ad581f26915c7e77500c8d4aaf0e83afcdc6 | testgroup1 |
| 7ba52ee1c5829d9837d740c08dffa07ad118ea1db2d70e0dc7fa7853e0b79fcf | testgroup2 |
+----------------------------------------------------------------+------------+
```

☞ Note

LDAP domain is read-only. This means that you cannot create new user or group records in it.

4.9.4 Domain-specific driver configuration. Switching from a database to a file store

Following is the procedure to switch a domain-specific driver configuration from a database store to a file store. It is assumed that:

- The domain-specific driver configuration with a database store has been set up and is working properly.

- Domain-specific driver configuration files with the format: keystone.<domain name>.conf have already been located and verified in the specific directory (by default, /etc/keystone/domains/) on all of the controller nodes.

1. Ensure that the following configuration options are set in the main configuration file template in ~/openstack/my_cloud/config/keystone/keystone.conf.j2:

```
[identity]
 domain_specific_drivers_enabled = True
 domain_configurations_from_database = False


[domain_config]
```

Domain-specific driver configuration. Switching from a database to a file store   HPE Helion

99   OpenStack 8

```
# driver = sql
```

2. Once the template is modified, commit the change to the local git repository, and rerun the configuration processor / deployment area preparation playbooks (as suggested at Using Git for Configuration Management):

```
ardana > cd ~/openstack
ardana > git checkout site
ardana > git add -A
```

Verify that the files have been added using git status, then commit the changes:

```
ardana > git status
ardana > git commit -m "Domain-Specific Driver Configuration - Switch From Database
 Store to File Store: more description here..."
```

Then run the configuration processor and ready deployment playbooks:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Run reconfiguration playbook in a deployment area:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

4. Verify that the switched domain driver configuration for LDAP (Microsoft AD in this example) using file store works properly: Set environment variables for admin level access

```
ardana > source ~/keystone.osrc
```

Get list of domain users:

```
ardana > openstack user list --domain ad
```

Here you see the three users:

```
+------------------------------------------------------------------+------------+
| ID                                                               | Name       |
+------------------------------------------------------------------+------------+
| e7dbec51ecaf07906bd743debcb49157a0e8af557b860a7c1dadd454bdab03fe | testuser1  |
| 8a09630fde3180c685e0cd663427e8638151b534a8a7ccebfcf244751d6f09bd | testuser2  |
| ea463d778dadcefdcfd5b532ee122a70dce7e790786678961420ae007560f35e | testuser3  |
```

Domain-specific driver configuration. Switching from a database to a file store   HPE Helion

100   OpenStack 8

```
+-------------------------------------------------------------------+------------+
```

Get user records within the ad domain:

```
ardana > openstack user show testuser1 --domain ad
```

Here is the result:

```
+-----------+-------------------------------------------------------------------+
| Field     | Value                                                             |
+-----------+-------------------------------------------------------------------+
| domain_id | 143af847018c4dc7bd35390402395886                                  |
| id        | e6d8c90abdc4510621271b73cc4dda8bc6009f263e421d8735d5f850f002f607 |
| name      | testuser1                                                         |
+-----------+-------------------------------------------------------------------+
```

Get a list of LDAP groups:

```
ardana > openstack group list --domain ad
```

Here are the groups returned:

```
+------------------------------------------------------------------+------------+
| ID                                                               | Name       |
+------------------------------------------------------------------+------------+
| 03976b0ea6f54a8e4c0032e8f756ad581f26915c7e77500c8d4aaf0e83afcdc6 | testgroup1 |
| 7ba52ee1c5829d9837d740c08dffa07ad118ea1db2d70e0dc7fa7853e0b79fcf | testgroup2 |
+------------------------------------------------------------------+------------+
```

Note: Note: LDAP domain is read-only. This means that you can not create new user or group record in it.

4.9.5 Update LDAP CA certificates

There is a chance that LDAP CA certificates may expire or for some reason not work anymore. Below are steps to update the LDAP CA certificates on the identity service side. Follow the steps below to make the updates.

1. Locate the file keystone_configure_ldap_certs_sample.yml

```
~/openstack/my_cloud/config/keystone/keystone_configure_ldap_certs_sample.yml
```

2. Save a copy of this file with a new name, for example:

```
~/openstack/my_cloud/config/keystone/keystone_configure_ldap_certs_all.yml
```

3. Edit the file and specify the correct single file path name for the ldap certificates. This
   file path name has to be consistent with the one defined in tls_cacertfile of the domain-
   specific configuration. Edit the file and populate or update it with LDAP CA certificates
   for all LDAP domains.

4. As suggested in *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for
   Configuration Management"*, add the new file to the local git repository:

```
ardana > cd ~/openstack
ardana > git checkout site
ardana > git add -A
```

   Verify that the files have been added using git status and commit the file:

```
ardana > git status
ardana > git commit -m "Update LDAP CA certificates: more description here..."
```

   Then run the configuration processor and ready deployment playbooks:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

5. Run the reconfiguration playbook in the deployment area:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml -e@~/
openstack/my_cloud/config/keystone/keystone_configure_ldap_certs_all.yml
```

4.9.6　Limitations

HPE Helion OpenStack 8 domain-specific configuration:

- No Global User Listing: Once domain-specific driver configuration is enabled, listing all users and listing all groups are not supported operations. Those calls require a specific domain filter and a domain-scoped token for the target domain.

- You cannot have both a file store and a database store for domain-specific driver configuration in a single identity service instance. Once a database store is enabled within the identity service instance, any file store will be ignored, and vice versa.

- The identity service allows a list limit configuration to globally set the maximum number of entities that will be returned in an identity collection per request but it does not support per-domain list limit setting at this time.

- Each time a new domain is configured with LDAP integration the single CA file gets overwritten. Ensure that you place certs for all the LDAP back-end domains in the cacert parameter. Detailed CA file inclusion instructions are provided in the comments of the sample YAML configuration file `keystone_configure_ldap_my.yml` (*Section 4.9.2, "Set up domain-specific driver configuration - file store"*).

- LDAP is only supported for identity operations (reading users and groups from LDAP).

- Keystone assignment operations from LDAP records such as managing or assigning roles and projects, are not currently supported.

- The HPE Helion OpenStack 'default' domain is pre-configured to store service account users and is authenticated locally against the identity service. Domains configured for external LDAP integration are non-default domains.

- When using the current OpenStackClient CLI you must use the user ID rather than the user name when working with a non-default domain.

- Each LDAP connection with the identity service is for read-only operations. Configurations that require identity service write operations (to create users, groups, etc.) are not currently supported.

- LDAP is only supported for identity operations (reading users and groups from LDAP). Keystone assignment operations from LDAP records such as managing or assigning roles and projects, are not currently supported.

- When using the current OpenStackClient CLI you must use the user ID rather than the user name when working with a non-default domain.

HPE Helion OpenStack 8 API-based domain-specific configuration management

- No GUI dashboard for domain-specific driver configuration management

- API-based Domain specific config does not check for type of option.

- API-based Domain specific config does not check for option values supported.

- API-based Domain config method does not provide retrieval of default values of domain-specific configuration options.

- Status: Domain-specific driver configuration database store is a non-core feature for HPE Helion OpenStack 8.

Note

When integrating with an external identity provider, cloud security is dependent upon the security of that identify provider. You should examine the security of the identity provider, and in particular the SAML 2.0 token generation process and decide what security properties you need to ensure adequate security of your cloud deployment. More information about SAML can be found at https://www.owasp.org/index.php/SAML_Security_Cheat_Sheet.

4.10 Keystone-to-Keystone Federation

This topic explains how you can use one instance of Keystone as an identity provider and one as a service provider.

4.10.1 What Is Keystone-to-Keystone Federation?

Identity federation lets you configure HPE Helion OpenStack using existing identity management systems such as an LDAP directory as the source of user access authentication. The Keystone-to-Keystone federation (K2K) function extends this concept for accessing resources in multiple, separate HPE Helion OpenStack clouds. You can configure each cloud to trust the authentication credentials of other clouds to provide the ability for users to authenticate with their home cloud and to access authorized resources in another cloud without having to reauthenticate with the remote cloud. This function is sometimes referred to as "single sign-on" or SSO.

The HPE Helion OpenStack cloud that provides the initial user authentication is called the identity provider (IdP). The identity provider cloud can support domain-based authentication against external authentication sources including LDAP-based directories such as Microsoft Active Directory. The identity provider creates the user attributes, known as assertions, which are used to automatically authenticate users with other HPE Helion OpenStack clouds.

An HPE Helion OpenStack cloud that provides resources is called a service provider (SP). A service provider cloud accepts user authentication assertions from the identity provider and provides access to project resources based on the mapping file settings developed for each service provider cloud. The following are characteristics of a service provider:

- Each service provider cloud has a unique set of projects, groups, and group role assignments that are created and managed locally.

- The mapping file consists a set of rules that define user group membership.

- The mapping file enables the ability to auto-assign incoming users to a specific group. Project membership and access are defined by group membership.

- Project quotas are defined locally by each service provider cloud.

Keystone-to-Keystone federation is supported and enabled in HPE Helion OpenStack 8 using configuration parameters in specific Ansible files. Instructions are provided to define and enable the required configurations.

Support for Keystone-to-Keystone federation happens on the API level, and you must implement it using your own client code by calling the supported APIs. Python-keystoneclient has supported APIs to access the K2K APIs.

EXAMPLE 4.1: K2KCLIENT.PY

> The following k2kclient.py file is an example, and the request diagram *Figure 4.1, "Keystone Authentication Flow"* explains the flow of client requests.

```
import json
import os
import requests

import xml.dom.minidom

from keystoneclient.auth.identity import v3
from keystoneclient import session

class K2KClient(object):

    def __init__(self):
        # IdP auth URL
        self.auth_url = "http://192.168.245.9:35357/v3/"
        self.project_name = "admin"
        self.project_domain_name = "Default"
        self.username = "admin"
        self.password = "vvaQIZ1S"
        self.user_domain_name = "Default"
        self.session = requests.Session()
        self.verify = False
        # identity provider Id
        self.idp_id = "z420_idp"
        # service provider Id
        self.sp_id = "z620_sp"
        #self.sp_ecp_url = "https://16.103.149.44:8443/Shibboleth.sso/SAML2/ECP"
        #self.sp_auth_url = "https://16.103.149.44:8443/v3"

    def v3_authenticate(self):
        auth = v3.Password(auth_url=self.auth_url,
                           username=self.username,
                           password=self.password,
                           user_domain_name=self.user_domain_name,
                           project_name=self.project_name,
                           project_domain_name=self.project_domain_name)

        self.auth_session = session.Session(session=requests.session(),
                                            auth=auth, verify=self.verify)
        auth_ref = self.auth_session.auth.get_auth_ref(self.auth_session)
        self.token = self.auth_session.auth.get_token(self.auth_session)

    def _generate_token_json(self):
        return {
            "auth": {
                "identity": {
                    "methods": [
                        "token"
```

```
                    ],
                    "token": {
                        "id": self.token
                    }
                },
                "scope": {
                    "service_provider": {
                        "id": self.sp_id
                    }
                }
            }
        }

    def get_saml2_ecp_assertion(self):
        token = json.dumps(self._generate_token_json())
        url = self.auth_url + 'auth/OS-FEDERATION/saml2/ecp'
        r = self.session.post(url=url,
                              data=token,
                              verify=self.verify)
        if not r.ok:
            raise Exception("Something went wrong, %s" % r.__dict__)
        self.ecp_assertion = r.text

    def _get_sp_url(self):
        url = self.auth_url + 'OS-FEDERATION/service_providers/' + self.sp_id
        r = self.auth_session.get(
            url=url,
            verify=self.verify)
        if not r.ok:
            raise Exception("Something went wrong, %s" % r.__dict__)

        sp = json.loads(r.text)[u'service_provider']
        self.sp_ecp_url = sp[u'sp_url']
        self.sp_auth_url = sp[u'auth_url']

    def _handle_http_302_ecp_redirect(self, response, method, **kwargs):
        location = self.sp_auth_url + '/OS-FEDERATION/identity_providers/' +
self.idp_id + '/protocols/saml2/auth'
        return self.auth_session.request(location, method, authenticated=False,
**kwargs)

    def exchange_assertion(self):
        """Send assertion to a Keystone SP and get token."""
        self._get_sp_url()
        print("SP ECP Url:%s" % self.sp_ecp_url)
        print("SP Auth Url:%s" % self.sp_auth_url)
        #self.sp_ecp_url = 'https://16.103.149.44:8443/Shibboleth.sso/SAML2/ECP'
```

```
        r = self.auth_session.post(
            self.sp_ecp_url,
            headers={'Content-Type': 'application/vnd.paos+xml'},
            data=self.ecp_assertion,
            authenticated=False, redirect=False)
        r = self._handle_http_302_ecp_redirect(r, 'GET',
            headers={'Content-Type': 'application/vnd.paos+xml'})
        self.fed_token_id = r.headers['X-Subject-Token']
        self.fed_token = r.text


if __name__ == "__main__":
    client = K2KClient()
    client.v3_authenticate()
    client.get_saml2_ecp_assertion()
    client.exchange_assertion()
    print('Unscoped token_id: %s' % client.fed_token_id)
    print('Unscoped token body:
%s' % client.fed_token)
```

4.10.2   Setting Up a Keystone Provider

To set up Keystone as a service provider, follow these steps.

1. Create a config file called `k2k.yml` with the following parameters and place it in any directory on your Cloud Lifecycle Manager, such as /tmp.

```
keystone_trusted_idp: k2k
keystone_sp_conf:
  shib_sso_idp_entity_id: <protocol>://<idp_host>:<port>/v3/OS-FEDERATION/saml2/idp
  shib_sso_application_entity_id: http://service_provider_uri_entityId
  target_domain:
    name: domain1
    description: my domain
  target_project:
    name: project1
    description: my project
  target_group:
    name: group1
    description: my group
  role:
    name: service
  idp_metadata_file: /tmp/idp_metadata.xml
  identity_provider:
    id: my_idp_id
    description: This is the identity service provider.
```

```
mapping:
  id: mapping1
  rules_file: /tmp/k2k_sp_mapping.json
protocol:
  id: saml2
attribute_map:
  -
    name: name1
    id: id1
```

The following are descriptions of each of the attributes.

Attribute	Definition
keystone_trusted_idp	A flag to indicate if this configuration is used for Keystone-to-Keystone or WebSSO. The value can be either k2k or adfs.
keystone_sp_conf	
shib_sso_idp_entity_id	The identity provider URI used as an entity Id to identity the IdP. You shoud use the following value: <protocol>://<idp_host>:<port>/v3/OS-FEDERATION/saml2/idp.
shib_sso_application_entity_id	The service provider URI used as an entity Id. It can be any URI here for Keystone-to-Keystone.
target_domain	A domain where the group will be created.
name	Any domain name. If it does not exist, it will be created or updated.
description	Any description.
target_project	A project scope of the group.

Attribute	Definition
name	Any project name. If it does not exist, it will be created or updated.
description	Any description.
target_group	A group will be created from target_domain.
name	Any group name. If it does not exist, it will be created or updated.
description	Any description.
role	A role will be assigned on target_project. This role impacts the IdP user scoped token permission on the service provider side.
name	Must be an existing role.
idp_metadata_file	A reference to the IdP metadata file that validates the SAML2 assertion.
identity_provider	A supported IdP.
id	Any Id. If it does not exist, it will be created or updated. This Id needs to be shared with the client so that the right mapping will be selected.
description	Any description.
mapping	A mapping in JSON format that maps a federated user to a corresponding group.
id	Any Id. If it does not exist, it will be created or updated.

Attribute	Definition
rules_file	A reference to the file that has the mapping in JSON.
protocol	The supported federation protocol.
id	Security Assertion Markup Language 2.0 (SAML2) is the only supported protocol for K2K.
attribute_map	A shibboleth mapping that defines additional attributes to map the attributes from the SAML2 assertion to the K2K mapping that the service provider understands. K2K does not require any additional attribute mapping.
name	An attribute name from the SAML2 assertion.
id	An Id that the preceding name will be mapped to.

2. Create a metadata file that is referenced from `k2k.yml`, such as `/tmp/idp_metadata.xml`. The content of the metadata file comes from the identity provider and can be found in `/etc/keystone/idp_metadata.xml`.

   a. Create a mapping file that is referenced in k2k.yml, shown previously. An example is `/tmp/k2k_sp_mapping.json`. You can see the reference in bold in the preceding k2k.yml example. The following is an example of the mapping file.

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
```

```
            "group": {
               "name": "group1",
               "domain":{
                  "name": "domain1"
               }
            }
         }
      ],
      "remote":[{
         "type": "openstack_user"
      },
      {
         "type": "Shib-Identity-Provider",
         "any_one_of":[
            "https://idp_host:5000/v3/OS-FEDERATION/saml2/idp"
         ]
       }
      ]
   }
]
```

You can find more information on how the K2K mapping works at http://docs.openstack.org ↗.

3. Go to `~/stack/scratch/ansible/next/ardana/ansible` and run the following playbook to enable the service provider:

```
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml -e@/tmp/
k2k.yml
```

Setting Up an Identity Provider

To set up Keystone as an identity provider, follow these steps:

1. Create a config file `k2k.yml` with the following parameters and place it in any directory on your Cloud Lifecycle Manager, such as `/tmp`. Note that the certificate and key here are excerpted for space.

```
keystone_k2k_idp_conf:
    service_provider:
        -
            id: my_sp_id
            description: This is service provider.
            sp_url: https://sp_host:5000
            auth_url: https://sp_host:5000/v3
    signer_cert: -----BEGIN CERTIFICATE-----
```

```
MIIDmDCCAoACCQDS+ZDoUfr
    cIzANBgkqhkiG9w0BAQsFADCBjDELMAkGA1UEBhMC\ nVVMxEzARBgNVB
    AgMCkNhbGlmb3JuaWExEjAQBgNVBAcMCVN1bm55dmFsZTEMMAoG\


         ...
    nOpKEvhlMsl5I/tle
-----END CERTIFICATE-----
    signer_key: -----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA1gRiHiwSO6L5PrtroHi/f17DQBOpJ1KMnS9FOHS


         ...
```

The following are descriptions of each of the attributes under keystone_k2k_idp_conf

service_provider

> One or more service providers can be defined. If it does not exist, it will be created or updated.

id

> Any Id. If it does not exist, it will be created or updated. This Id needs to be shared with the client so that it knows where the service provider is.

description

> Any description.

sp_url

> Service provider base URL.

auth_url

> Service provider auth URL.

signer_cert

> Content of self-signed certificate that is embedded in the metadata file. We recommend setting the validity for a longer period of time, such as 3650 days (10 years).

signer_key

> A private key that has a key size of 2048 bits.

2. Create a private key and a self-signed certificate. The command-line tool, openssl, is required to generate the keys and certificates. If the system does not have it, you must install it.

a. Create a private key of size 2048.

```
ardana > openssl genrsa -out myidp.key 2048
```

b. Generate a certificate request named myidp.csr. When prompted, choose CommonName for the server's hostname.

```
ardana > openssl req -new -key myidp.key -out myidp.csr
```

c. Generate a self-signed certificate named myidp.cer.

```
ardana > openssl x509 -req -days 3650 -in myidp.csr -signkey myidp.key -out
 myidp.cer
```

3. Go to `~/scratch/ansible/next/ardana/ansible` and run the following playbook to enable the service provider in Keystone:

```
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml -e@/tmp/
k2k.yml
```

4.10.3   Test It Out

You can use the script listed earlier, `k2kclient.py` (*Example 4.1, "k2kclient.py"*), as an example for the end-to-end flows. To run `k2kclient.py`, follow these steps:

1. A few parameters must be changed in the beginning of `k2kclient.py`. For example, enter your specific URL, project name, and user name, as follows:

```
# IdP auth URL
self.auth_url = "http://idp_host:5000/v3/"
self.project_name = "my_project_name"
self.project_domain_name = "my_project_domain_name"
self.username = "test"
self.password = "mypass"
self.user_domain_name = "my_domain"
# identity provider Id that is defined in the SP config
self.idp_id = "my_idp_id"
# service provider Id that is defined in the IdP config
```

```
self.sp_id = "my_sp_id"
```

2. Install python-keystoneclient along with its dependencies.

3. Run the `k2kclient.py` script. An unscoped token will be returned from the service provider.

At this point, the domain or project scope of the unscoped taken can be discovered by sending the following URLs:

```
ardana > curl -k -X GET -H "X-Auth-Token: unscoped token" \
 https://<sp_public_endpoint>:5000/v3/OS-FEDERATION/domains
ardana > curl -k -X GET -H "X-Auth-Token: unscoped token" \
 https://<sp_public_endpoint:5000/v3/OS-FEDERATION/projects
```

4.10.4 Inside Keystone-to-Keystone Federation

K2K federation places a lot of responsibility with the user. The complexity is apparent from the following diagram.

1. Users must first authenticate to their home or local cloud, or local identity provider Keystone instance to obtain a scoped token.

2. Users must discover which service providers (or remote clouds) are available to them by querying their local cloud.

3. For a given remote cloud, users must discover which resources are available to them by querying the remote cloud for the projects they can scope to.

4. To talk to the remote cloud, users must first exchange, with the local cloud, their locally scoped token for a SAML2 assertion to present to the remote cloud.

5. Users then present the SAML2 assertion to the remote cloud. The remote cloud applies its mapping for the incoming SAML2 assertion to map each user to a local ephemeral persona (such as groups) and issues an unscoped token.

6. Users query the remote cloud for the list of projects they have access to.

7. Users then rescope their token to a given project.

8. Users now have access to the resources owned by the project.

The following diagram illustrates the flow of authentication requests.
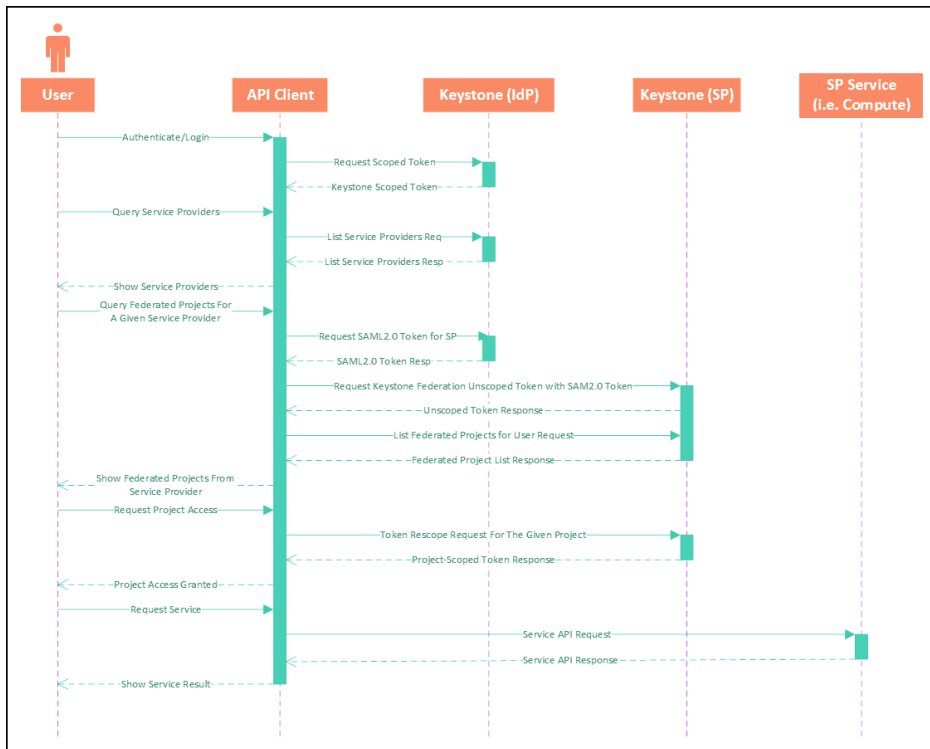
FIGURE 4.1: KEYSTONE AUTHENTICATION FLOW

4.10.5 Additional Testing Scenarios

The following tests assume one identity provider and one service provider.

Test Case 1: Any federated user in the identity provider maps to a single designated group in the service provider

1. On the identity provider side:

   ```
   hostname=myidp.com
   username=user1
   ```

2. On the service provider side:

   ```
   group=group1
   group_domain_name=domain1
   'group1' scopes to 'project1'
   ```

3. Mapping used:

   ```
   testcase1_1.json
   ```

testcase1_1.json

```json
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group1",
          "domain":{
            "name": "domain1"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
        "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
]
```

4. Expected result: The federated user will scope to project1.

Test Case 2: A federated user in a specific domain in the identity provider maps to two different groups in the service provider

1. On the identity provider side:

```
hostname=myidp.com
username=user1
user_domain_name=Default
```

2. On the service provider side:

```
group=group1
```

```
group_domain_name=domain1
'group1' scopes to 'project1' group=group2
group_domain_name=domain2
'group2' scopes to 'project2'
```

3. Mapping used:

```
testcase1_2.json
```

testcase1_2.json

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group1",
          "domain":{
            "name": "domain1"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
        "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
```

```
        "group": {
          "name": "group2",
          "domain":{
            "name": "domain2"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "openstack_user_domain",
      "any_one_of": [
          "Default"
      ]
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
          "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
]
```

4. Expected result: The federated user will scope to both project1 and project2.

Test Case 3: A federated user with a specific project in the identity provider maps to a specific group in the service provider

1. On the identity provider side:

```
hostname=myidp.com
username=user4
user_project_name=test1
```

2. On the service provider side:

```
group=group4
group_domain_name=domain4
'group4' scopes to 'project4'
```

3. Mapping used:

```
testcase1_3.json
```

testcase1_3.json

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
            "name": "group4",
            "domain":{
              "name": "domain4"
            }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "openstack_project",
      "any_one_of": [
          "test1"
      ]
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
          "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  },
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
```

```
      {
        "group": {
          "name": "group5",
          "domain":{
            "name": "domain5"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "openstack_roles",
      "not_any_of": [
          "_member_"
      ]
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
        "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
]
```

4. Expected result: The federated user will scope to project4.

Test Case 4: A federated user with a specific role in the identity provider maps to a specific group in the service provider

1. On the identity provider side:

```
hostname=myidp.com, username=user5, role_name=_member_
```

2. On the service provider side:

```
group=group5, group_domain_name=domain5, 'group5' scopes to 'project5'
```

3. Mapping used:

```
testcase1_3.json
```

testcase1_3.json

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group4",
          "domain":{
            "name": "domain4"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "openstack_project",
      "any_one_of": [
          "test1"
      ]
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
          "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
     }
    ]
   },
   {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group5",
          "domain":{
            "name": "domain5"
```

```
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "openstack_roles",
      "not_any_of": [
          "_member_"
      ]
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
          "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
]
```

4. Expected result: The federated user will scope to project5.

Test Case 5: Retain the previous scope for a federated user

1. On the identity provider side:

```
hostname=myidp.com, username=user1, user_domain_name=Default
```

2. On the service provider side:

```
group=group1, group_domain_name=domain1, 'group1' scopes to 'project1'
```

3. Mapping used:

```
testcase1_1.json
```

testcase1_1.json

```
[
  {
    "local": [
      {
        "user": {
```

```
        "name": "{0}"
      }
    },
    {
      "group": {
         "name": "group1",
         "domain":{
            "name": "domain1"
         }
      }
    }
  ],
  "remote":[{
    "type": "openstack_user"
  },
  {
    "type": "Shib-Identity-Provider",
    "any_one_of":[
       "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
    ]
   }
  ]
 }
]
```

4. Expected result: The federated user will scope to project1. Later, we would like to scope federated users who have the default domain in the identity provider to project2 in addition to project1.

5. On the identity provider side:

```
hostname=myidp.com, username=user1, user_domain_name=Default
```

6. On the service provider side:

```
group=group1
group_domain_name=domain1
'group1' scopes to 'project1' group=group2
group_domain_name=domain2
'group2' scopes to 'project2'
```

7. Mapping used:

```
testcase1_2.json
```

testcase1_2.json

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group1",
          "domain":{
            "name": "domain1"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
        "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group2",
          "domain":{
            "name": "domain2"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
```

```
    },
    {
      "type": "openstack_user_domain",
      "any_one_of": [
          "Default"
      ]
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
          "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
]
```

8. Expected result: The federated user will scope to project1 and project2.

Test Case 6: Scope a federated user to a domain

1. On the identity provider side:

```
hostname=myidp.com, username=user1
```

2. On the service provider side:

```
group=group1, group_domain_name=domain1, 'group1' scopes to 'project1'
```

3. Mapping used:

```
testcase1_1.json
```

testcase1_1.json

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
            "name": "group1",
```

```
          "domain":{
            "name": "domain1"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
          "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
     }
    ]
   }
]
```

4. Expected result:

   - The federated user will scope to project1.

   - User uses CLI/Curl to assign any existing role to group1 on domain1.

   - User uses CLI/Curl to remove project1 scope from group1.

5. Final result: The federated user will scope to domain1.

Test Case 7: Test five remote attributes for mapping

1. Test all five different remote attributes, as follows, with similar test cases as noted previously.

   - openstack_user

   - openstack_user_domain

   - openstack_roles

   - openstack_project

   - openstack_project_domain

The attribute openstack_user does not make much sense for testing because it is mapped only to a specific username. The preceding test cases have already covered the attributes openstack_user_domain, openstack_roles, and openstack_project.

Note that similar tests have also been run for two identity providers with one service provider, and for one identity provider with two service providers.

4.10.6   Known Issues and Limitations

Keep the following points in mind:

- When a user is disabled in the identity provider, the issued federated token from the service provider still remains valid until the token is expired based on the Keystone expiration setting.

- An already issued federated token will retain its scope until its expiration. Any changes in the mapping on the service provider will not impact the scope of an already issued federated token. For example, if an already issued federated token was mapped to group1 that has scope on project1, and mapping is changed to group2 that has scope on project2, the prevously issued federated token still has scope on project1.

- Access to service provider resources is provided only through the python-keystone CLI client or the Keystone API. No Horizon web interface support is currently available.

- Domains, projects, groups, roles, and quotas are created per the service provider cloud. Support for federated projects, groups, roles, and quotas is currently not available.

- Keystone-to-Keystone federation and WebSSO cannot be configured by putting both sets of configuration attributes in the same config file; they will overwrite each other. Consequently, they need to be configured individually.

- Scoping the federated user to a domain is not supported by default in the playbook. Please follow the steps at *Section 4.10.7, "Scope Federated User to Domain"*.

4.10.7 Scope Federated User to Domain

Use the following steps to scope a federated user to a domain:

1. On the IdP side, set `hostname=myidp.com` and `username=user1`.

2. On the service provider side, set: `group=group1`, `group_domain_name=domain1`, group1 scopes to project1.

3. Mapping used: testcase1_1.json.
   testcase1_1.json

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group1",
          "domain":{
            "name": "domain1"
          }
        }
      }
    ],
    "remote":[{
      "type": "openstack_user"
    },
    {
      "type": "Shib-Identity-Provider",
      "any_one_of":[
        "https://myidp.com:5000/v3/OS-FEDERATION/saml2/idp"
      ]
    }
    ]
  }
]
```

4. Expected result: The federated user will scope to project1. Use CLI/Curl to assign any existing role to group1 on domain1. Use CLI/Curl to remove project1 scope from group1.

5. Result: The federated user will scope to domain1.

4.11 Configuring Web Single Sign-On

This topic explains how to implement web single sign-on.

4.11.1 What is WebSSO?

WebSSO, or web single sign-on, is a method for web browsers to receive current authentication information from an identity provider system without requiring a user to log in again to the application displayed by the browser. Users initially access the identity provider web page and supply their credentials. If the user successfully authenticates with the identity provider, the authentication credentials are then stored in the user's web browser and automatically provided to all web-based applications, such as the Horizon dashboard in HPE Helion OpenStack 8. If users have not yet authenticated with an identity provider or their credentials have timed out, they are automatically redirected to the identity provider to renew their credentials.

4.11.2 Limitations

- The WebSSO function supports only Horizon web authentication. It is not supported for direct API or CLI access.

- WebSSO works only with Fernet token provider. See *Section 4.8.4, "Fernet Tokens"*.

- The HPE Helion OpenStack WebSSO function was tested with Microsoft Active Directory Federation Services (AD FS). The instructions provided are pertinent to AD FS and are intended to provide a sample configuration for deploying WebSSO with an external identity provider. If you have a different identity provider such as Ping Identity or IBM Tivoli, consult with those vendors for specific instructions for those products.

- Only WebSSO federation using the SAML method is supported in HPE Helion OpenStack 8 . OpenID-based federation is not currently supported.

- **WebSSO has a change password option in User Settings, but note that this function is not accessible for users authenticating with external systems such as LDAP or SAML Identity Providers.**

4.11.3 Enabling WebSSO

HPE Helion OpenStack 8 provides WebSSO support for the Horizon web interface. This support requires several configuration steps including editing the Horizon configuration file as well as ensuring that the correct Keystone authentication configuration is enabled to receive the authentication assertions provided by the identity provider.

The following is the workflow that depicts how Horizon and Keystone supports WebSSO if no current authentication assertion is available.

1. Horizon redirects the web browser to the Keystone endpoint.

2. Keystone automatically redirects the web browser to the correct identity provider authentication web page based on the Keystone configuration file.

3. The user authenticates with the identity provider.

4. The identity provider automatically redirects the web browser back to the Keystone endpoint.

5. Keystone generates the required Javascript code to POST a token back to Horizon.

6. Keystone automatically redirects the web browser back to Horizon and the user can then access projects and resources assigned to the user.

The following diagram provides more details on the WebSSO authentication workflow.
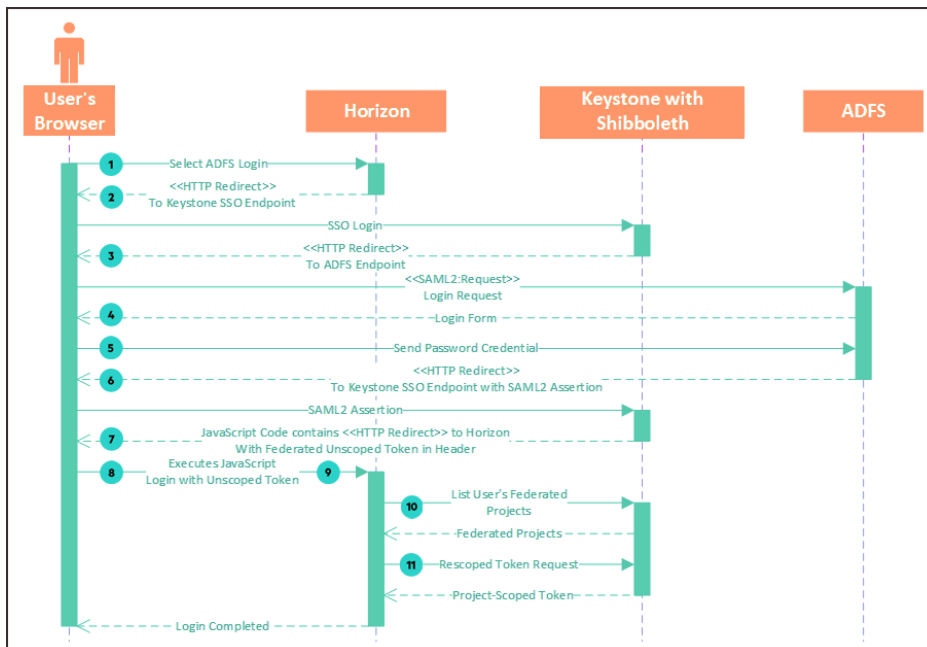
Note that the Horizon dashboard service never talks directly to the Keystone identity service until the end of the sequence, after the federated unscoped token negotiation has completed. The browser interacts with the Horizon dashboard service, the Keystone identity service, and AD FS on their respective public endpoints.

The following sequence of events is depicted in the diagram.

1. The user's browser reaches the Horizon dashboard service's login page. The user selects AD FS login from the drop-down menu.

2. The Horizon dashboard service issues an HTTP Redirect (301) to redirect the browser to the Keystone identity service's (public) SAML2 Web SSO endpoint (/auth/OS-FEDERATION/websso/saml2). The endpoint is protected by Apache mod_shib (shibboleth).

3. The browser talks to the Keystone identity service. Because the user's browser does not have an active session with AD FS, the Keystone identity service issues an HTTP Redirect (301) to the browser, along with the required SAML2 request, to the AD FS endpoint.

4. The browser talks to AD FS. AD FS returns a login form. The browser presents it to the user.

5. The user enters credentials (such as username and password) and submits the form to AD FS.

6. Upon successful validation of the user's credentials, AD FS issues an HTTP Redirect (301) to the browser, along with the SAML2 assertion, to the Keystone identity service's (public) SAML2 endpoint (/auth/OS-FEDERATION/websso/saml2).

7. The browser talks to the Keystone identity service. the Keystone identity service validates the SAML2 assertion and issues a federated unscoped token. the Keystone identity service returns JavaScript code to be executed by the browser, along with the federated unscoped token in the headers.

8. Upon execution of the JavaScript code, the browser is redirected to the Horizon dashboard service with the federated unscoped token in the header.

9. The browser talks to the Horizon dashboard service with the federated unscoped token.

10. With the unscoped token, the Horizon dashboard service talks to the Keystone identity service's (internal) endpoint to get a list of projects the user has access to.

11. The Horizon dashboard service rescopes the token to the first project in the list. At this point, the user is successfully logged in.

4.11.4 Prerequisites

4.11.4.1 Creating AD FS metadata

For information about creating Active Directory Federation Services metadata, see the section *To create edited AD FS 2.0 metadata with an added scope element* of https://technet.microsoft.com/en-us/library/gg317734 ↗.

1. On the AD FS computer, use a browser such as Internet Explorer to view `https://<adfs_server_hostname>/FederationMetadata/2007-06/FederationMetadata.xml.`

2. On the File menu, click Save as, and then navigate to the Windows desktop and save the file with the name adfs_metadata.xml. Make sure to change the Save as type drop-down box to All Files (*.*).

3. Use Windows Explorer to navigate to the Windows desktop, right-click adfs_metadata.xml, and then click Edit.

4. In Notepad, insert the following XML in the first element. Before editing, the EntityDescriptor appears as follows:

```
<EntityDescriptor ID="abc123" entityID=http://WIN-CAICP35LF2I.vlan44.domain/adfs/
services/trust xmlns="urn:oasis:names:tc:SAML:2.0:metadata" >
```

After editing, it should look like this:

```
<EntityDescriptor ID="abc123" entityID="http://WIN-CAICP35LF2I.vlan44.domain/
adfs/services/trust" xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
 xmlns:shibmd="urn:mace:shibboleth:metadata:1.0">
```

5. In Notepad, on the Edit menu, click Find. In Find what, type IDPSSO, and then click Find Next.

6. Insert the following XML in this section: Before editing, the IDPSSODescriptor appears as follows:

```
<IDPSSODescriptor
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"><KeyDescriptor
 use="encryption">
```

After editing, it should look like this:

```
<IDPSSODescriptor
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"><Extensions><shibmd:Scope
 regexp="false">vlan44.domain</shibmd:Scope></Extensions><KeyDescriptor
 use="encryption">
```

7. Delete the metadata document signature section of the file (the bold text shown in the following code). Because you have edited the document, the signature will now be invalid. Before editing the signature appears as follows:

```
<EntityDescriptor ID="abc123" entityID="http://FSWEB.contoso.com/
adfs/services/trust" xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
 xmlns:shibmd="urn:mace:shibboleth:metadata:1.0">
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    SIGNATURE DATA
</ds:Signature>
<RoleDescriptor xsi:type=…>
```

After editing it should look like this:

```
<EntityDescriptor ID="abc123" entityID="http://FSWEB.contoso.com/
adfs/services/trust" xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
 xmlns:shibmd="urn:mace:shibboleth:metadata:1.0">
<RoleDescriptor xsi:type=…>
```

8. Save and close adfs_metadata.xml.

9. Copy adfs_metadata.xml to the Cloud Lifecycle Manager node in your preferred location. Here it is /tmp.

4.11.4.2   Setting Up WebSSO

Start by creating a config file `adfs_config.yml` with the following parameters and place it in any directory on your Cloud Lifecycle Manager, such as `/tmp`.

```
keystone_trusted_idp: adfs
keystone_sp_conf:
    idp_metadata_file: /tmp/adfs_metadata.xml
    shib_sso_application_entity_id: http://sp_uri_entityId
    shib_sso_idp_entity_id: http://default_idp_uri_entityId
    target_domain:
        name: domain1
```

```
            description: my domain
    target_project:
        name: project1
        description: my project
    target_group:
        name: group1
        description: my group
    role:
        name: service
    identity_provider:
        id: adfs_idp1
        description: This is the AD FS identity provider.
    mapping:
        id: mapping1
        rules_file: adfs_mapping.json
    protocol:
        id: saml2
    attribute_map:
        -
          name: http://schemas.xmlsoap.org/claims/Group
          id: ADFS_GROUP
        -
          name: urn:oid:1.3.6.1.4.1.5923.1.1.1.6
          id: ADFS_LOGIN
```

A sample config file like this exists in roles/KEY-API/files/samples/websso/ keystone_configure_adfs_sample.yml. Here are some detailed descriptions for each of the config options:

```
keystone_trusted_idp: A flag to indicate if this configuration is used for WebSSO or K2K.
 The value can be either 'adfs' or 'k2k'.
keystone_sp_conf:
    shib_sso_idp_entity_id: The AD FS URI used as an entity Id to identity the IdP.
    shib_sso_application_entity_id: The Service Provider URI used as a entity Id. It can
 be any URI here for Websso as long as it is unique to the SP.
    target_domain: A domain where the group will be created from.
        name: Any domain name. If it does not exist, it will be created or be updated.
        description: Any description.
    target_project: A project scope that the group has.
        name: Any project name. If it does not exist, it will be created or be updated.
        description: Any description.
    target_group: A group will be created from 'target_domain'.
        name: Any group name. If it does not exist, it will be created or be updated.
        description: Any description.
    role: A role will be assigned on 'target_project'. This role impacts the idp user
 scoped token permission at sp side.
        name: It has to be an existing role.
```

```
    idp_metadata_file: A reference to the AD FS metadata file that validates the SAML2
assertion.
    identity_provider: An AD FS IdP
        id: Any Id. If it does not exist, it will be created or be updated. This Id needs
to be shared with the client so that the right mapping will be selected.
        description: Any description.
    mapping: A mapping in json format that maps a federated user to a corresponding
group.
        id: Any Id. If it does not exist, it will be created or be updated.
        rules_file: A reference to the file that has the mapping in json.
    protocol: The supported federation protocol.
        id: 'saml2' is the only supported protocol for Websso.
    attribute_map: A shibboleth mapping defined additional attributes to map the
attributes from the SAML2 assertion to the Websso mapping that SP understands.
        -
          name: An attribute name from the SAML2 assertion.
          id: An Id that the above name will be mapped to.
```

1. In the preceding config file, /tmp/adfs_config.yml, make sure the idp_metadata_file
   references the previously generated AD FS metadata file. In this case:

   ```
   idp_metadata_file: /tmp/adfs_metadata.xml
   ```

2. Create a mapping file that is referenced from the preceding config file, such
   as /tmp/adfs_sp_mapping.json. rules_file: /tmp/adfs_sp_mapping.json. The following is
   an example of the mapping file, existing in roles/KEY-API/files/samples/websso/
   adfs_sp_mapping.json:

   ```
   [
           {
             "local": [{
                   "user": {
                       "name": "{0}"
                   }
               }],
               "remote": [{
                   "type": "ADFS_LOGIN"
               }]
           },
           {
             "local": [{
                   "group": {
                       "id": "GROUP_ID"
                   }
               }],
   ```

```
                "remote": [{
                    "type": "ADFS_GROUP",
                "any_one_of": [
                    "Domain Users"
                    ]
                }]
            }
    ]
```

You can find more details about how the WebSSO mapping works at http://docs.openstack.org ↗. Also see *Section 4.11.4.3, "Mapping rules"* for more information.

3. Go to ~/scratch/ansible/next/ardana/ansible and run the following playbook to enable WebSSO in the Keystone identity service:

```
ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml -e@/tmp/
adfs_config.yml
```

4. Enable WebSSO in the Horizon dashboard service by setting horizon_websso_enabled flag to True in roles/HZN-WEB/defaults/main.yml and then run the horizon-reconfigure playbook:

```
ardana > ansible-playbook -i hosts/verb_hosts horizon-reconfigure.yml
```

4.11.4.3   Mapping rules

One IdP-SP has only one mapping. The last mapping that the customer configures will be the one used and will overwrite the old mapping setting. Therefore, if the example mapping adfs_sp_mapping.json is used, the following behavior is expected because it maps the federated user only to the one group configured in `keystone_configure_adfs_sample.yml`.

- Configure domain1/project1/group1, mapping1; websso login horizon, see project1;

- Then reconfigure: domain1/project2/group1. mapping1, websso login horizon, see project1 and project2;

- Reconfigure: domain3/project3/group3; mapping1, websso login horizon, only see project3; because now the IDP mapping maps the federated user to group3, which only has priviliges on project3.

If you need a more complex mapping, you can use a custom mapping file, which needs to be specified in keystone_configure_adfs_sample.yml -> rules_file.

You can use different attributes of the AD FS user in order to map to different or multiple groups.
An example of a more complex mapping file is adfs_sp_mapping_multiple_groups.json, as follows.

adfs_sp_mapping_multiple_groups.json

```
[
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group1",
          "domain":{
            "name": "domain1"
          }
        }
      }
    ],
    "remote":[{
      "type": "ADFS_LOGIN"
    },
    {
      "type": "ADFS_GROUP",
      "any_one_of":[
        "Domain Users"
      ]
    }
    ]
  },
  {
    "local": [
      {
        "user": {
          "name": "{0}"
        }
      },
      {
        "group": {
          "name": "group2",
          "domain":{
            "name": "domain2"
          }
        }
      }
```

```
      }
    ],
    "remote":[{
      "type": "ADFS_LOGIN"
    },
    {
      "type": "ADFS_SCOPED_AFFILIATION",
      "any_one_of": [
          "member@contoso.com"
      ]
    },
    ]
  }
]
```

The adfs_sp_mapping_multiple_groups.json must be run together with keystone_configure_mutiple_groups_sample.yml, which adds a new attribute for the shibboleth mapping. That file is as follows:

keystone_configure_mutiple_groups_sample.yml

```
#
# Licensed under the Apache License, Version 2.0 (the "License"); you may
# not use this file except in compliance with the License. You may obtain
# a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
# License for the specific language governing permissions and limitations
# under the License.
#
---

keystone_trusted_idp: adfs
keystone_sp_conf:
    identity_provider:
        id: adfs_idp1
        description: This is the AD FS identity provider.
    idp_metadata_file: /opt/stack/adfs_metadata.xml

    shib_sso_application_entity_id: http://blabla
    shib_sso_idp_entity_id: http://WIN-CAICP35LF2I.vlan44.domain/adfs/services/trust

    target_domain:
```

```
        name: domain2
        description: my domain

    target_project:
        name: project6
        description: my project

    target_group:
        name: group2
        description: my group

    role:
        name: admin

    mapping:
        id: mapping1
        rules_file: /opt/stack/adfs_sp_mapping_multiple_groups.json

    protocol:
        id: saml2

    attribute_map:
        -
          name: http://schemas.xmlsoap.org/claims/Group
          id: ADFS_GROUP
        -
          name: urn:oid:1.3.6.1.4.1.5923.1.1.1.6
          id: ADFS_LOGIN
        -
          name: urn:oid:1.3.6.1.4.1.5923.1.1.1.9
          id: ADFS_SCOPED_AFFILIATION
```

4.11.5   Setting up the AD FS server as the identity provider

For AD FS to be able to communicate with the Keystone identity service, you need to add the Keystone identity service as a trusted relying party for AD FS and also specify the user attributes that you want to send to the Keystone identity service when users authenticate via WebSSO.

For more information, see the Microsoft AD FS wiki (https://technet.microsoft.com/en-us/library/gg317734)↗, section "Step 2: Configure AD FS 2.0 as the identity provider and shibboleth as the Relying Party".

Log in to the AD FS server.

Add a relying party using metadata

1. From Server Manager Dashboard, click Tools on the upper right, then ADFS Management.

2. Right-click ADFS, and then select Add Relying Party Trust.

3. Click Start, leave the already selected option `Import data about the relying party published online or on a local network`.

4. In the Federation metadata address field, type `<keystone_publicEndpoint>/Shibboleth.sso/Metadata` (your Keystone identity service Metadata endpoint), and then click Next. You can also import metadata from a file. Create a file with the content of the result of the following curl command

   ```
   curl <keystone_publicEndpoint>/Shibboleth.sso/Metadata
   ```

   and then choose this file for importing the metadata for the relying party.

5. In the Specify Display Name page, choose a proper name to identify this trust relationship, and then click Next.

6. On the Choose Issuance Authorization Rules page, leave the default Permit all users to access the relying party selected, and then click Next.

7. Click Next, and then click Close.

Edit claim rules for relying party trust

1. The Edit Claim Rules dialog box should already be open. If not, In the ADFS center pane, under Relying Party Trusts, right-click your newly created trust, and then click Edit Claim Rules.

2. On the Issuance Transform Rules tab, click Add Rule.

3. On the Select Rule Template page, select Send LDAP Attributes as Claims, and then click Next.

4. On the Configure Rule page, in the Claim rule name box, type Get Data.

5. In the Attribute Store list, select Active Directory.

6. In the Mapping of LDAP attributes section, create the following mappings.

LDAP Attribute	Outgoing Claim Type
Token-Groups – Unqualified Names	Group

LDAP Attribute	Outgoing Claim Type
User-Principal-Name	UPN

7. Click Finish.

8. On the Issuance Transform Rules tab, click Add Rule.

9. On the Select Rule Template page, select Send Claims Using a Custom Rule, and then click Next.

10. In the Configure Rule page, in the Claim rule name box, type Transform UPN to epPN.

11. In the Custom Rule window, type or copy and paste the following:

```
c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"]
=> issue(Type = "urn:oid:1.3.6.1.4.1.5923.1.1.1.6", Value = c.Value,
 Properties["http://schemas.xmlsoap.org/ws/2005/05/identity/claimproperties/
attributename"] = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri");
```

12. Click Finish.

13. On the Issuance Transform Rules tab, click Add Rule.

14. On the Select Rule Template page, select Send Claims Using a Custom Rule, and then click Next.

15. On the Configure Rule page, in the Claim rule name box, type Transform Group to epSA.

16. In the Custom Rule window, type or copy and paste the following:

```
c:[Type == "http://schemas.xmlsoap.org/claims/Group", Value == "Domain Users"]
=> issue(Type = "urn:oid:1.3.6.1.4.1.5923.1.1.1.9", Value = "member@contoso.com",
 Properties["http://schemas.xmlsoap.org/ws/2005/05/identity/claimproperties/
attributename"] = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri");
```

17. Click Finish, and then click OK.

This list of Claim Rules is just an example and can be modified or enhanced based on the customer's necessities and AD FS setup specifics.

Create a sample user on the AD FS server

1. From the Server Manager Dashboard, click Tools on the upper right, then Active Directory Users and Computer.

2. Right click User, then New, and then User.

3. Follow the on-screen instructions.

You can test the Horizon dashboard service "Login with ADFS" by opening a browser at the Horizon dashboard service URL and choose `Authenticate using: ADFS Credentials`. You should be redirected to the ADFS login page and be able to log into the Horizon dashboard service with your ADFS credentials.

4.12 Identity Service Notes and Limitations

4.12.1 Notes

This topic describes limitations of and important notes pertaining to the identity service. **Domains**

- Domains can be created and managed by the Horizon web interface, Keystone API and OpenStackClient CLI.

- The configuration of external authentication systems requires the creation and usage of Domains.

- All configurations are managed by creating and editing specific configuration files.

- End users can authenticate to a particular project and domain via the Horizon web interface, Keystone API and OpenStackClient CLI.

- A new Horizon login page that requires a Domain entry is now installed by default.

Keystone-to-Keystone Federation

- Keystone-to-Keystone (K2K) Federation provides the ability to authenticate once with one cloud and then use these credentials to access resources on other federated clouds.

- All configurations are managed by creating and editing specific configuration files.

Multi-Factor Authentication (MFA)

- The Keystone architecture provides support for MFA deployments.

- MFA provides the ability to deploy non-password based authentication; for example: token providing hardware and text messages.

Hierarchical Multitenancy

- Provides the ability to create sub-projects within a Domain-Project hierarchy.

4.12.2   Limitations

Authentication with external authentication systems (LDAP, Active Directory (AD) or Identity Providers)

- No Horizon web portal support currently exists for the creation and management of external authentication system configurations.

Integration with LDAP services HPE Helion OpenStack 8 domain-specific configuration:

- No Global User Listing: Once domain-specific driver configuration is enabled, listing all users and listing all groups are not supported operations. Those calls require a specific domain filter and a domain-scoped token for the target domain.

- You cannot have both a file store and a database store for domain-specific driver configuration in a single identity service instance. Once a database store is enabled within the identity service instance, any file store will be ignored, and vice versa.

- The identity service allows a list limit configuration to globally set the maximum number of entities that will be returned in an identity collection per request but it does not support per-domain list limit setting at this time.

- Each time a new domain is configured with LDAP integration the single CA file gets overwritten. Ensure that you place certs for all the LDAP back-end domains in the cacert parameter. Detailed CA file inclusion instructions are provided in the comments of the sample YAML configuration file `keystone_configure_ldap_my.yml` (see *Section 4.9.2, "Set up domain-specific driver configuration - file store"*).

- LDAP is only supported for identity operations (reading users and groups from LDAP).

- Keystone assignment operations from LDAP records such as managing or assigning roles and projects, are not currently supported.

- The HPE Helion OpenStack 'default' domain is pre-configured to store service account users and is authenticated locally against the identity service. Domains configured for external LDAP integration are non-default domains.

- When using the current OpenStackClient CLI you must use the user ID rather than the user name when working with a non-default domain.

- Each LDAP connection with the identity service is for read-only operations. Configurations that require identity service write operations (to create users, groups, etc.) are not currently supported.

- LDAP is only supported for identity operations (reading users and groups from LDAP). Keystone assignment operations from LDAP records such as managing or assigning roles and projects, are not currently supported.

- When using the current OpenStackClient CLI you must use the user ID rather than the user name when working with a non-default domain.

HPE Helion OpenStack 8 API-based domain-specific configuration management

- No GUI dashboard for domain-specific driver configuration management

- API-based Domain specific config does not check for type of option.

- API-based Domain specific config does not check for option values supported.

- API-based Domain config method does not provide retrieval of default values of domain-specific configuration options.

- Status: Domain-specific driver configuration database store is a non-core feature for HPE Helion OpenStack 8.

4.12.3 Keystone-to-Keystone federation

- When a user is disabled in the identity provider, the issued federated token from the service provider still remains valid until the token is expired based on the Keystone expiration setting.

- An already issued federated token will retain its scope until its expiration. Any changes in the mapping on the service provider will not impact the scope of an already issued federated token. For example, if an already issued federated token was mapped to group1 that has scope on project1, and mapping is changed to group2 that has scope on project2, the prevously issued federated token still has scope on project1.

- Access to service provider resources is provided only through the python-keystone CLI client or the Keystone API. No Horizon web interface support is currently available.

- Domains, projects, groups, roles, and quotas are created per the service provider cloud. Support for federated projects, groups, roles, and quotas is currently not available.

- Keystone-to-Keystone federation and WebSSO cannot be configured by putting both sets of configuration attributes in the same config file; they will overwrite each other. Consequently, they need to be configured individually.

- Scoping the federated user to a domain is not supported by default in the playbook. To enable it, see the steps in *Section 4.10.7, "Scope Federated User to Domain"*.

- No Horizon web portal support currently exists for the creation and management of federation configurations.

- All end user authentication is available only via the Keystone API and OpenStackClient CLI.

- Additional information can be found at http://docs.openstack.org ↗ .

WebSSO

- The WebSSO function supports only Horizon web authentication. It is not supported for direct API or CLI access.

- WebSSO works only with Fernet token provider. See *Section 4.8.4, "Fernet Tokens"*.

- The HPE Helion OpenStack WebSSO function was tested with Microsoft Active Directory Federation Services (ADFS). The instructions provided are pertinent to ADFS and are intended to provide a sample configuration for deploying WebSSO with an external identity provider. If you have a different identity provider such as Ping Identity or IBM Tivoli, consult with those vendors for specific instructions for those products.

- Only WebSSO federation using the SAML method is supported in HPE Helion OpenStack 8 . OpenID-based federation is not currently supported.

- **WebSSO has a change password option in User Settings, but note that this function is not accessible for users authenticating with external systems such as LDAP or SAML Identity Providers.**

Multi-factor authentication (MFA)

- HPE Helion OpenStack MFA support is a custom configuration requiring Professional Services support.

- MFA drivers are not included with HPE Helion OpenStack and need to be provided by a specific MFA vendor.

- Additional information can be found at http://docs.openstack.org/security-guide/content/identity-authentication-methods.html#identity-authentication-methods-external-authentication-methods ↗ .

Hierarchical multitenancy

- This function requires additional support from various OpenStack services to be functional. It is a non-core function in HPE Helion OpenStack and is not ready for either proof of concept or production deployments.

- Additional information can be found at http://specs.openstack.org/openstack/keystone-specs/specs/juno/hierarchical_multitenancy.html ↗ .

Missing quota information for compute resources

> ☞ Note
>
> An error message that will appear in the default Horizon page if you are running a Swift-only deployment (no Compute service). In this configuration, you will not see any quota information for Compute resources and will see the following error message:

The Compute service is not installed or is not configured properly. No information is available for Compute resources. This error message is expected as no Compute service is configured for this deployment. Please ignore the error message.

The following is the benchmark of the performance that is based on 150 concurrent requests and run for 10 minute periods of stable load time.

Operation	In HPE Helion OpenStack 8 (secs/request)	In HPE Helion OpenStack 8 3.0 (secs/request)
Token Creation	0.86	0.42
Token Validation	0.47	0.41

Considering that token creation operations do not happen as frequently as token validation operations, you are likely to experience less of a performance problem regardless of the extended time for token creation.

4.12.4   System cron jobs need setup

Keystone relies on two cron jobs to periodically clean up expired tokens and for token revocation. The following is how the cron jobs appear on the system:

```
1 1 * * * /opt/stack/service/keystone/venv/bin/keystone-manage token_flush
1 1,5,10,15,20 * * * /opt/stack/service/keystone/venv/bin/revocation_cleanup.sh
```

By default, the two cron jobs are enabled on controller node 1 only, not on the other two nodes. When controller node 1 is down or has failed for any reason, these two cron jobs must be manually set up on one of the other two nodes.

5 Managing Compute

Information about managing and configuring the Compute service.

5.1 Managing Compute Hosts using Aggregates and Scheduler Filters

OpenStack Nova has the concepts of availability zones and host aggregates that enable you to segregate your compute hosts. Availability zones are used to specify logical separation within your cloud based on the physical isolation or redundancy you have set up. Host aggregates are used to group compute hosts together based upon common features, such as operation system. For more information, read this topic.

OpenStack Nova has the concepts of availability zones and host aggregates that enable you to segregate your Compute hosts. Availability zones are used to specify logical separation within your cloud based on the physical isolation or redundancy you have set up. Host aggregates are used to group compute hosts together based upon common features, such as operation system. For more information, see Scaling and Segregating your Cloud (http://docs.openstack.org/openstack-ops/content/scaling.html) ↗ .

The Nova scheduler also has a filter scheduler, which supports both filtering and weighting to make decisions on where new compute instances should be created. For more information, see Filter Scheduler (http://docs.openstack.org/developer/nova/filter_scheduler.html) ↗ and Scheduling (http://docs.openstack.org/mitaka/config-reference/compute/scheduler.html) ↗ .

This document is going to show you how to set up both a Nova host aggregate and configure the filter scheduler to further segregate your compute hosts.

5.1.1 Creating a Nova Aggregate

These steps will show you how to create a Nova aggregate and how to add a compute host to it. You can run these steps on any machine that contains the NovaClient that also has network access to your cloud environment. These requirements are met by the Cloud Lifecycle Manager.

1. Log in to the Cloud Lifecycle Manager.

2. Source the administrative creds:

   ```
   ardana > source ~/service.osrc
   ```

3. List your current Nova aggregates:

   ```
   ardana > nova aggregate-list
   ```

4. Create a new Nova aggregate with this syntax:

   ```
   ardana > nova aggregate-create AGGREGATE-NAME
   ```

   If you wish to have the aggregate appear as an availability zone, then specify an availability zone with this syntax:

   ```
   ardana > nova aggregate-create AGGREGATE-NAME AVAILABILITY-ZONE-NAME
   ```

   So, for example, if you wish to create a new aggregate for your SUSE Linux Enterprise compute hosts and you wanted that to show up as the SLE availability zone, you could use this command:

   ```
   ardana > nova aggregate-create SLE SLE
   ```

   This would produce an output similar to this:

   ```
   +----+------+------------------+-------+-----------------+
   | Id | Name | Availability Zone | Hosts | Metadata
   +----+------+------------------+-------+------------------------+
   | 12 | SLE  | SLE              |       | 'availability_zone=SLE'
   +----+------+------------------+-------+------------------------+
   ```

5. Next, you need to add compute hosts to this aggregate so you can start by listing your current hosts. You will want to limit the output of this command to only the hosts running the compute service, like this:

   ```
   ardana > nova host-list | grep compute
   ```

6. You can then add host(s) to your aggregate with this syntax:

```
ardana > nova aggregate-add-host AGGREGATE-NAME HOST
```

7. Then you can confirm that this has been completed by listing the details of your aggregate:

```
nova aggregate-details AGGREGATE-NAME
```

You can also list out your availability zones using this command:

```
ardana > nova availability-zone-list
```

5.1.2 Using Nova Scheduler Filters

The Nova scheduler has two filters that can help with differentiating between different compute hosts that we'll describe here.

Filter	Description
AggregateImagePropertiesIsolation	Isolates compute hosts based on image properties and aggregate metadata. You can use commas to specify multiple values for the same property. The filter will then ensure at least one value matches.
AggregateInstanceExtraSpecsFilter	Checks that the aggregate metadata satisfies any extra specifications associated with the instance type. This uses `aggregate_instance_extra_specs`

☞ Note

For details about other available filters, see Filter Scheduler (http://docs.openstack.org/developer/nova/filter_scheduler.html)↗.

Using the AggregateImagePropertiesIsolation Filter

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/config/nova/nova.conf.j2` file and add `AggregateImagePropertiesIsolation` to the scheduler_filters section. Example below, in bold:

```
# Scheduler
...
scheduler_available_filters = nova.scheduler.filters.all_filters
scheduler_default_filters = AvailabilityZoneFilter,RetryFilter,ComputeFilter,
 DiskFilter,RamFilter,ImagePropertiesFilter,ServerGroupAffinityFilter,
 ServerGroupAntiAffinityFilter,ComputeCapabilitiesFilter,NUMATopologyFilter,
 AggregateImagePropertiesIsolation
...
```

Optionally, you can also add these lines:

```
aggregate_image_properties_isolation_namespace = <a prefix string>
```

```
aggregate_image_properties_isolation_separator = <a separator character>
```

(defaults to `.`)

If these are added, the filter will only match image properties starting with the name space and separator - for example, setting to `my_name_space` and `:` would mean the image property `my_name_space:image_type=SLE` matches metadata `image_type=SLE`, but `an_other=SLE` would not be inspected for a match at all.

If these are not added all image properties will be matched against any similarly named aggregate metadata.

3. Add image properties to images that should be scheduled using the above filter

4. Commit the changes to git:

```
ardana > git add -A
ardana > git commit -a -m "editing nova schedule filters"
```

5. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Run the ready deployment playbook:

```
ardana > cd ~/openstack/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Nova reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

Using the AggregateInstanceExtraSpecsFilter Filter

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/config/nova/nova.conf.j2` file and add
   `AggregateInstanceExtraSpecsFilter` to the scheduler_filters section. Example below,
   in bold:

```
# Scheduler
...
scheduler_available_filters = nova.scheduler.filters.all_filters
 scheduler_default_filters = AvailabilityZoneFilter,RetryFilter,ComputeFilter,
 DiskFilter,RamFilter,ImagePropertiesFilter,ServerGroupAffinityFilter,
 ServerGroupAntiAffinityFilter,ComputeCapabilitiesFilter,NUMATopologyFilter,
 AggregateInstanceExtraSpecsFilter
...
```

3. There is no additional configuration needed because the following is true:

   a. The filter assumes `:` is a separator

   b. The filter will match all simple keys in extra_specs plus all keys with a separator if
      the prefix is `aggregate_instance_extra_specs` - for example, `image_type=SLE`
      and `aggregate_instance_extra_specs:image_type=SLE` will both be matched
      against aggregate metadata `image_type=SLE`

4. Add `extra_specs` to flavors that should be scheduled according to the above.

5. Commit the changes to git:

```
ardana > git add -A
ardana > git commit -a -m "Editing nova scheduler filters"
```

6. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. Run the ready deployment playbook:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. Run the Nova reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

5.2   Using Flavor Metadata to Specify CPU Model

`Libvirt` is a collection of software used in OpenStack to manage virtualization. It has the ability to emulate a host CPU model in a guest VM. In HPE Helion OpenStack Nova, the ComputeCapabilitiesFilter limits this ability by checking the exact CPU model of the compute host against the requested compute instance model. It will only pick compute hosts that have the `cpu_model` requested by the instance model, and if the selected compute host does not have that `cpu_model`, the ComputeCapabilitiesFilter moves on to find another compute host that matches, if possible. Selecting an unavailable vCPU model may cause Nova to fail with `no valid host found`.

To assist, there is a Nova scheduler filter that captures `cpu_models` as a subset of a particular CPU family. The filter determines if the host CPU model is capable of emulating the guest CPU model by maintaining the mapping of the vCPU models and comparing it with the host CPU model.

There is a limitation when a particular `cpu_model` is specified with `hw:cpu_model` via a compute flavor: the `cpu_mode` will be set to `custom`. This mode ensures that a persistent guest virtual machine will see the same hardware no matter what host physical machine the guest virtual machine is booted on. This allows easier live migration of virtual machines. Because of this limitation, only some of the features of a CPU are exposed to the guest. Requesting particular CPU features is not supported.

5.2.1 Editing the flavor metadata in the Horizon dashboard

These steps can be used to edit a flavor's metadata in the Horizon dashboard to add the `extra_specs` for a `cpu_model`:

1. Access the Horizon dashboard and log in with admin credentials.

2. Access the Flavors menu by (A) clicking on the menu button, (B) navigating to the Admin section, and then (C) clicking on Flavors:
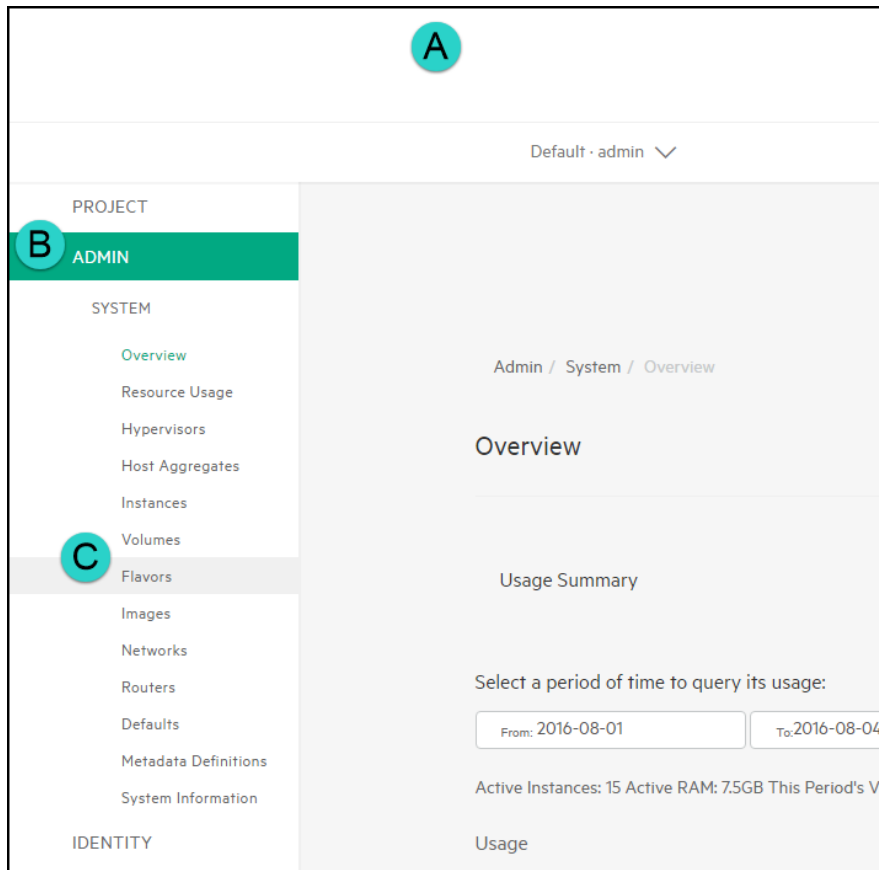
3. In the list of flavors, choose the flavor you wish to edit and click on the entry under the Metadata column:

> 🎓 **Note**
>
> You can also create a new flavor and then choose that one to edit.

4. In the Custom field, enter `hw:cpu_model` and then click on the `+` (plus) sign to continue:

5.3 Forcing CPU and RAM Overcommit Settings

HPE Helion OpenStack supports overcommitting of CPU and RAM resources on compute nodes. Overcommitting is a technique of allocating more virtualized CPUs and/or memory than there are physical resources.

The default settings for this are:

Setting	Default Value	Description
cpu_allocation_ratio	16	Virtual CPU to physical CPU allocation ratio which affects all CPU filters. This configuration specifies a global ratio for CoreFilter. For AggregateCoreFilter, it will fall back to this configuration value if no per-aggregate setting found. ☞ **Note** This can be set per-compute, or if set to `0.0`, the value set on the scheduler node(s) will be used and defaulted to `16.0`.
ram_allocation_ratio	1.0	Virtual RAM to physical RAM allocation ratio which affects all RAM filters. This configuration specifies a global ratio for RamFilter. For AggregateRamFilter, it will fall back to this configuration value if no per-aggregate setting found.

Setting	Default Value	Description
		☞ Note This can be set per-compute, or if set to `0.0`, the value set on the scheduler node(s) will be used and defaulted to `1.5`.
disk_allocation_ratio	1.0	This is the virtual disk to physical disk allocation ratio used by the disk_filter.py script to determine if a host has sufficient disk space to fit a requested instance. A ratio greater than 1.0 will result in over-subscription of the available physical disk, which can be useful for more efficiently packing instances created with images that do not use the entire virtual disk,such as sparse or compressed images. It can be set to a value between 0.0 and 1.0 in order to preserve a percentage of the disk for uses other than instances.

Setting	Default Value	Description
		🎓 Note This can be set per-compute, or if set to `0.0`, the value set on the scheduler node(s) will be used and defaulted to `1.0`.

5.3.1 Changing the overcommit ratios for your entire environment

If you wish to change the CPU and/or RAM overcommit ratio settings for your entire environment then you can do so via your Cloud Lifecycle Manager with these steps.

1. Log in to the Cloud Lifecycle Manager.

2. Edit the Nova configuration settings located in this file:

   ```
   ~/openstack/my_cloud/config/nova/nova.conf.j2
   ```

3. Add or edit the following lines to specify the ratios you wish to use:

   ```
   cpu_allocation_ratio = 16
   ram_allocation_ratio = 1.0
   ```

4. Commit your configuration to the Git repository (*Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "setting Nova overcommit settings"
   ```

5. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

6. Update your deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ```

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Nova reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

5.4 Enabling the Nova Resize and Migrate Features

The Nova resize and migrate features are disabled by default. If you wish to utilize these options, these steps will show you how to enable it in your cloud.

The two features below are disabled by default:

- **Resize** - this feature allows you to change the size of a Compute instance by changing its flavor. See the OpenStack User Guide (http://docs.openstack.org/user-guide/cli_change_the_size_of_your_server.html) ↗ for more details on its use.

- **Migrate** - read about the differences between "live" migration (enabled by default) and regular migration (disabled by default) in *Section 13.1.3.3, "Live Migration of Instances"*.

These two features are disabled by default because they require passwordless SSH access between Compute hosts with the user having access to the file systems to perform the copy.

5.4.1 Enabling Nova Resize and Migrate

If you wish to enable these features, use these steps on your lifecycle manager. This will deploy a set of public and private SSH keys to the Compute hosts, allowing the `nova` user SSH access between each of your Compute hosts.

1. Log in to the Cloud Lifecycle Manager.

2. Run the Nova reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml --extra-vars
 nova_migrate_enabled=true
```

3. To ensure that the resize and migration options show up in the Horizon dashboard, run
   the Horizon reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts horizon-reconfigure.yml
```

5.4.2 Disabling Nova Resize and Migrate

This feature is disabled by default. However, if you have previously enabled it and wish to re-disable it, you can use these steps on your lifecycle manager. This will remove the set of public and private SSH keys that were previously added to the Compute hosts, removing the `nova` users SSH access between each of your Compute hosts.

1. Log in to the Cloud Lifecycle Manager.

2. Run the Nova reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml --extra-vars
 nova_migrate_enabled=false
```

3. To ensure that the resize and migrate options are removed from the Horizon dashboard,
   run the Horizon reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts horizon-reconfigure.yml
```

5.5 Enabling ESX Compute Instance(s) Resize Feature

The resize of ESX compute instance is disabled by default. If you want to utilize this option, these steps will show you how to configure and enable it in your cloud.

The following feature is disabled by default:

- **Resize** - this feature allows you to change the size of a Compute instance by changing its flavor. See the OpenStack User Guide (http://docs.openstack.org/user-guide/cli_change_the_size_of_your_server.html)↗ for more details on its use.

5.5.1 Procedure

If you want to configure and re-size ESX compute instance(s), perform the following steps:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the ~ /openstack/my_cloud/config/nova/nova.conf.j2 to add the following parameter under **Policy**:

```
# Policy
allow_resize_to_same_host=True
```

3. Commit your configuration:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "<commit message>"
```

4. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

By default the nova resize feature is disabled. To enable nova resize, refer to *Section 5.4, "Enabling the Nova Resize and Migrate Features"*.

By default an ESX console log is not set up. For more details about its setup, refer to VMware vSphere (https://docs.openstack.org/nova/pike/admin/configuration/hypervisor-vmware.html)↗.

5.6 Configuring the Image Service

The image service, based on OpenStack Glance, works out of the box and does not need any special configuration. However, we show you how to enable Glance image caching as well as how to configure your environment to allow the Glance copy-from feature if you choose to do so. A few features detailed below will require some additional configuration if you choose to use them.

🛡 Warning

Glance images are assigned IDs upon creation, either automatically or specified by the user. The ID of an image should be unique, so if a user assigns an ID which already exists, a conflict (409) will occur.

This only becomes a problem if users can publicize or share images with others. If users can share images AND cannot publicize images then your system is not vulnerable. If the system has also been purged (via `glance-manage db purge`) then it is possible for deleted image IDs to be reused.

If deleted image IDs can be reused then recycling of public and shared images becomes a possibility. This means that a new (or modified) image can replace an old image, which could be malicious.

If this is a problem for you, please contact Professional Services.

5.6.1 How to enable Glance image caching

In HPE Helion OpenStack 8, by default, the Glance image caching option is not enabled. You have the option to have image caching enabled and these steps will show you how to do that.

The main benefits to using image caching is that it will allow the Glance service to return the images faster and it will cause less load on other services to supply the image.

In order to use the image caching option you will need to supply a logical volume for the service to use for the caching.

If you wish to use the Glance image caching option, you will see the section below in your `~/openstack/my_cloud/definition/data/disks_controller.yml` file. You will specify the mount point for the logical volume you wish to use for this.

1. Log in to the Cloud Lifecycle Manager.

2. Edit your `~/openstack/my_cloud/definition/data/disks_controller.yml` file and specify the volume and mount point for your `glance-cache`. Here is an example:

```
# Glance cache: if a logical volume with consumer usage glance-cache
# is defined Glance caching will be enabled. The logical volume can be
# part of an existing volume group or a dedicated volume group.
 - name: glance-vg
   physical-volumes:
     - /dev/sdx
   logical-volumes:
     - name: glance-cache
       size: 95%
       mount: /var/lib/glance/cache
       fstype: ext4
       mkfs-opts: -O large_file
       consumer:
         name: glance-api
         usage: glance-cache
```

If you are enabling image caching during your initial installation, prior to running `site.yml` the first time, then continue with the installation steps. However, if you are making this change post-installation then you will need to commit your changes with the steps below.

3. Commit your configuration to the Git repository (*Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "My config or other commit message"
```

4. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the Glance reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/verb_hosts glance-reconfigure.yml
```

An existing volume image cache is not properly deleted when Cinder detects the source image has changed. After updating any source image, delete the cache volume so that the cache is refreshed.

The volume image cache must be deleted before trying to use the associated source image in any other volume operations. This includes creating bootable volumes or booting an instance with `create volume` enabled and the updated image as the source image.

5.6.2 Allowing the Glance copy-from option in your environment

When creating images, one of the options you have is to copy the image from a remote location to your local Glance store. You do this by specifying the `--copy-from` option when creating the image. To use this feature though you need to ensure the following conditions are met:

- The server hosting the Glance service must have network access to the remote location that is hosting the image.

- There cannot be a proxy between Glance and the remote location.

- The Glance v1 API must be enabled, as v2 does not currently support the `copy-from` function.

- The http Glance store must be enabled in the environment, following the steps below.

Enabling the HTTP Glance Store

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/config/glance/glance-api.conf.j2` file and add `http` to the list of Glance stores in the `[glance_store]` section as seen below in bold:

   ```
   [glance_store]
   stores = {{ glance_stores }}, http
   ```

3. Commit your configuration to the Git repository (*Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*), as follows:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

4. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the Glance reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts glance-reconfigure.yml
```

7. Run the Horizon reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts horizon-reconfigure.yml
```

6 Managing ESX

Information about managing and configuring the ESX service.

6.1 Networking for ESXi Hypervisor (OVSvApp)

To provide the network as a service for tenant VM's hosted on ESXi Hypervisor, a service VM named OVSvApp VM is deployed on each ESXi Hypervisor within a cluster managed by OpenStack Nova, as shown in the following figure.
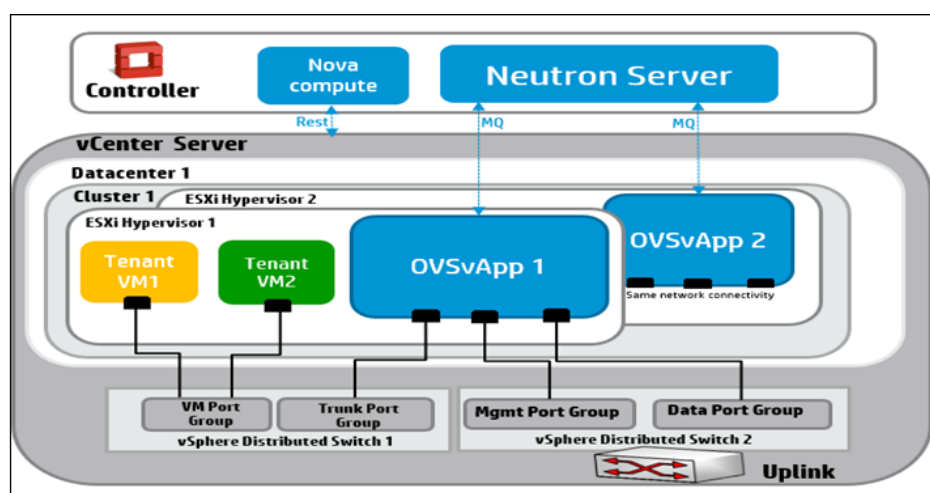
The OVSvApp VM runs SLES as a guest operating system, and has Open vSwitch 2.1.0 or above installed. It also runs an agent called OVSvApp agent, which is responsible for dynamically creating the port groups for the tenant VMs and manages OVS bridges, which contain the flows related to security groups and L2 networking.

To facilitate fault tolerance and mitigation of data path loss for tenant VMs, run the **neutron-ovsapp-agent-monitor** process as part of the **neutron-ovsvapp-agent service**, responsible for monitoring the Open vSwitch module within the OVSvApp VM. It also uses a `nginx` server to provide the health status of the Open vSwitch module to the Neutron server for mitigation actions. There is a mechanism to keep the **neutron-ovsvapp-agent service** alive through a `systemd` script.

When a OVSvApp Service VM crashes, an agent monitoring mechanism starts a cluster mitigation process. You can mitigate data path traffic loss for VMs on the failed ESX host in that cluster by putting the failed ESX host in the maintenance mode. This, in turn, triggers the vCenter DRS migrates tenant VMs to other ESX hosts within the same cluster. This ensures data path continuity of tenant VMs traffic.

To View Cluster Mitigation

An administrator can view the cluster mitigation status using the following commands.

1. `neutron ovsvapp-mitigated-cluster-list`
   Lists all the clusters where at least one round of host mitigation has happened.
   Example:

   ```
   neutron ovsvapp-mitigated-cluster-list
   +---------------+-------------+---------------------+--------------------------
   +
   | vcenter_id    | cluster_id  | being_mitigated     | threshold_reached
    |
   +---------------+-------------+---------------------+--------------------------
   +
   | vcenter1      | cluster1    | True                | False
    |
   | vcenter2      | cluster2    | False               | True
    |
   +---------------+-----------+----------------+------------------------------------
   +
   ```

2. `neutron ovsvapp-mitigated-cluster-show --vcenter-id <VCENTER_ID> --cluster-id <CLUSTER_ID>`
   Shows the status of a particular cluster.
   Example :

   ```
   neutron ovsvapp-mitigated-cluster-show --vcenter-id vcenter1 --cluster-id cluster1
   +--------------------------+-------------+
   | Field                    | Value       |
   +--------------------------+-------------+
   | being_mitigated          | True        |
   | cluster_id               | cluster1    |
   | threshold_reached        | False       |
   | vcenter_id               | vcenter1    |
   +--------------------------+-------------+
   ```

There can be instances where a triggered mitigation may not succeed and the neutron server is not informed of such failure (for example, if the selected agent which had to mitigate the host, goes down before finishing the task). In this case, the cluster will be locked. To unlock the cluster for further mitigations, use the update command.

3. `neutron ovsvapp-mitigated-cluster-update --vcenter-id <VCENTER_ID> --cluster-id <CLUSTER_ID>`

- Update the status of a mitigated cluster:

  Modify the values of **being-mitigated** from **True** to **False** to unlock the cluster.

  Example:

  ```
  neutron ovsvapp-mitigated-cluster-update --vcenter-id vcenter1 --cluster-id
    cluster1 --being-mitigated False
  ```

- Update the threshold value:

  Update the threshold-reached value to **True**, if no further migration is required in the selected cluster.

  Example :

  ```
  neutron ovsvapp-mitigated-cluster-update --vcenter-id vcenter1 --cluster-id
    cluster1 --being-mitigated False --threshold-reached True
  ```

Rest API

- ```
  curl -i -X GET http://<ip>:9696/v2.0/ovsvapp_mitigated_clusters \
    -H "User-Agent: python-neutronclient" -H "Accept: application/json" -H \
    "X-Auth-Token: <token_id>"
  ```

6.1.1   More Information

For more information on the Networking for ESXi Hypervisor (OVSvApp), see the following references:

- VBrownBag session in Vancouver OpenStack Liberty Summit:

  https://www.youtube.com/watch?v=icYA_ixhwsM&feature=youtu.be ↗

- Wiki Link:

  https://wiki.openstack.org/wiki/Neutron/Networking-vSphere ↗

- Codebase:

  https://github.com/openstack/networking-vsphere/ ↗

- Whitepaper:

  https://github.com/hp-networking/ovsvapp/blob/master/OVSvApp_Solution.pdf ↗

6.2 Validating the Neutron Installation

You can validate that the ESX compute cluster is added to the cloud successfully using the following command:

```
# neutron agent-list

+-----------------+---------------------+----------------------+------------------+------
+---------------+-------------------------+
| id              | agent_type          | host                 | availability_zone | alive |
 admin_state_up | binary                  |
+-----------------+---------------------+----------------------+------------------+------
+---------------+-------------------------+
| 05ca6ef...999c09 | L3 agent            | doc-cp1-comp0001-mgmt | nova             | :-)   | True
     | neutron-l3-agent        |
| 3b9179a...28e2ef | Metadata agent      | doc-cp1-comp0001-mgmt |                  | :-)   | True
     | neutron-metadata-agent  |
| 3d756d7...a719a2 | Loadbalancerv2 agent | doc-cp1-comp0001-mgmt |                  | :-)   | True
     | neutron-lbaasv2-agent   |
| 4e8f84f...c9c58f | Metadata agent      | doc-cp1-comp0002-mgmt |                  | :-)   | True
     | neutron-metadata-agent  |
| 55a5791...c17451 | L3 agent            | doc-cp1-c1-m1-mgmt    | nova             | :-)   | True
     | neutron-vpn-agent       |
| 5e3db8f...87f9be | Open vSwitch agent  | doc-cp1-c1-m1-mgmt    |                  | :-)   | True
     | neutron-openvswitch-agent |
| 6968d9a...b7b4e9 | L3 agent            | doc-cp1-c1-m2-mgmt    | nova             | :-)   | True
     | neutron-vpn-agent       |
| 7b02b20...53a187 | Metadata agent      | doc-cp1-c1-m2-mgmt    |                  | :-)   | True
     | neutron-metadata-agent  |
| 8ece188...5c3703 | Open vSwitch agent  | doc-cp1-comp0002-mgmt |                  | :-)   | True
     | neutron-openvswitch-agent |
| 8fcb3c7...65119a | Metadata agent      | doc-cp1-c1-m1-mgmt    |                  | :-)   | True
     | neutron-metadata-agent  |
| 9f48967...36effe | OVSvApp agent       | doc-cp1-comp0002-mgmt |                  | :-)   | True
     | ovsvapp-agent           |
| a2a0b78...026da9 | Open vSwitch agent  | doc-cp1-comp0001-mgmt |                  | :-)   | True
     | neutron-openvswitch-agent |
| a2fbd4a...28a1ac | DHCP agent          | doc-cp1-c1-m2-mgmt    | nova             | :-)   | True
     | neutron-dhcp-agent      |
| b2428d5...ee60b2 | DHCP agent          | doc-cp1-c1-m1-mgmt    | nova             | :-)   | True
     | neutron-dhcp-agent      |
| c0983a6...411524 | Open vSwitch agent  | doc-cp1-c1-m2-mgmt    |                  | :-)   | True
     | neutron-openvswitch-agent |
```

```
| c32778b...a0fc75 | L3 agent             | doc-cp1-comp0002-mgmt | nova               | :-)   | True
    | neutron-l3-agent       |
+-----------------+--------------------+----------------------+------------------+------
+---------------+--------------------------+
```

6.3 Removing a Cluster from the Compute Resource Pool

6.3.1 Prerequisites

Write down the Hostname and ESXi configuration IP addresses of OVSvAPP VMs of that ESX cluster before deleting the VMs. These IP address and Hostname will be used to cleanup Monasca alarm definitions.

Perform the following steps:

1. Login to vSphere client.

2. Select the ovsvapp node running on each ESXi host and click **Summary** tab as shown in the following example.
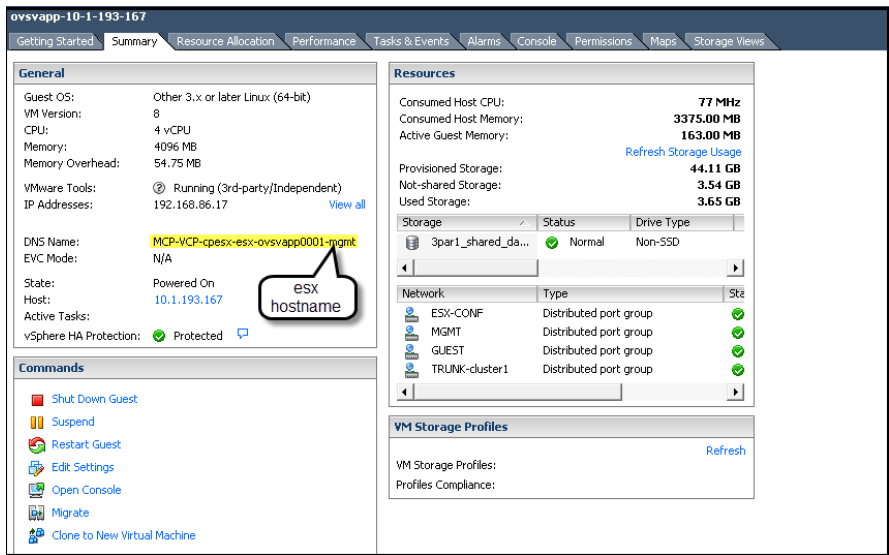
Similarly you can retrieve the compute-proxy node information.
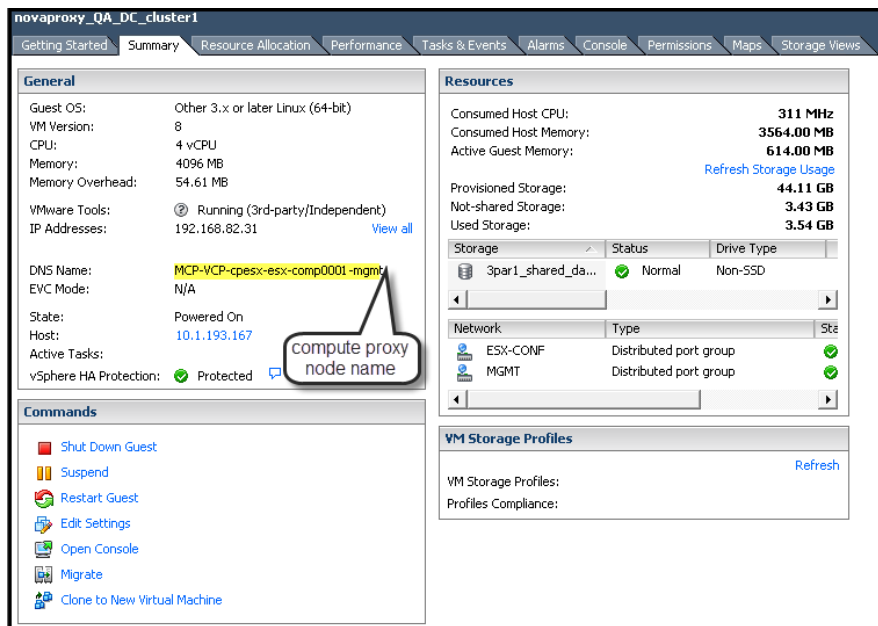
6.3.2   Removing an existing cluster from the compute resource pool

Perform the following steps to remove an existing cluster from the compute resource pool.

1. Run the following command to check for the instances launched in that cluster:

```
# nova list --host <hostname>
+--------------------------------------+------+--------+------------+-------------
+-----------------+
| ID                                   | Name | Status | Task State | Power State |
 Networks        |
+--------------------------------------+------+--------+------------+-------------
+-----------------+
| 80e54965-758b-425e-901b-9ea756576331 | VM1  | ACTIVE | -          | Running     |
 private=10.0.0.2 |
+--------------------------------------+------+--------+------------+-------------
+-----------------+
```

where:

- **hostname**: Specifies hostname of the compute proxy present in that cluster.

2. Delete all instances spawned in that cluster:

```
# nova delete <server> [<server ...>]
```

where:

- **server**: Specifies the name or ID of server (s)

OR
Migrate all instances spawned in that cluster.

```
# nova migrate <server>
```

3. Run the following playbooks for stop the Compute (Nova) and Networking (Neutron) services:

```
ansible-playbook -i hosts/verb_hosts nova-stop --limit <hostname>;
ansible-playbook -i hosts/verb_hosts neutron-stop --limit <hostname>;
```

where:

- **hostname**: Specifies hostname of the compute proxy present in that cluster.

6.3.3 Cleanup Monasca Agent for OVSvAPP Service

Perform the following procedure to cleanup Monasca agents for ovsvapp-agent service.

1. If Monasca-API is installed on different node, copy the `service.orsc` from Cloud Lifecycle Manager to Monasca API server.

```
scp service.orsc $USER@ardana-cp1-mtrmon-m1-mgmt:
```

2. SSH to Monasca API server. You must SSH to each Monasca API server for cleanup. For example:

```
ssh ardana-cp1-mtrmon-m1-mgmt
```

3. Edit `/etc/monasca/agent/conf.d/host_alive.yaml` file to remove the reference to the OVSvAPP you removed. This requires **sudo** access.

```
sudo vi /etc/monasca/agent/conf.d/host_alive.yaml
```

A sample of `host_alive.yaml`:

```
- alive_test: ping
  built_by: HostAlive
  host_name: esx-cp1-esx-ovsvapp0001-mgmt
  name: esx-cp1-esx-ovsvapp0001-mgmt ping
  target_hostname: esx-cp1-esx-ovsvapp0001-mgmt
```

where *HOST_NAME* and *TARGET_HOSTNAME* is mentioned at the DNS name field at the vSphere client. (Refer to *Section 6.3.1, "Prerequisites"*).

4. After removing the reference on each of the Monasca API servers, restart the monasca-agent on each of those servers by executing the following command.

```
tux > sudo service openstack-monasca-agent restart
```

5. With the OVSvAPP references removed and the monasca-agent restarted, you can delete the corresponding alarm to complete the cleanup process. We recommend using the Monasca CLI which is installed on each of your Monasca API servers by default. Execute the following command from the Monasca API server (for example: `ardana-cp1-mtrmon-mX-mgmt`).

```
monasca alarm-list --metric-name host_alive_status --metric-dimensions
 hostname=<ovsvapp deleted>
```

For example: You can execute the following command to get the alarm ID, if the OVSvAPP appears as a preceding example.

```
monasca alarm-list --metric-name host_alive_status --metric-dimensions hostname=MCP-
VCP-cpesx-esx-ovsvapp0001-mgmt
+--------------------------------------+------------------------------------
+----------------------+------------------
+--------------------------------------------+----------+-------
+----------------+------+------------------------+------------------------
+------------------------+
| id                                   | alarm_definition_id                  |
  alarm_definition_name | metric_name        | metric_dimensions
        | severity | state | lifecycle_state | link | state_updated_timestamp |
  updated_timestamp        | created_timestamp        |
```

```
+--------------------------------------+--------------------------------------
+----------------------+-------------------
+----------------------------------------------+---------+-------
+----------------+------+------------------------+------------------------
+-------------------------+
| cfc6bfa4-2485-4319-b1e5-0107886f4270 | cca96c53-a927-4b0a-9bf3-cb21d28216f3 |
 Host Status             | host_alive_status | service: system
        | HIGH    | OK   | None                  | None | 2016-10-27T06:33:04.256Z |
 2016-10-27T06:33:04.256Z | 2016-10-23T13:41:57.258Z |
|                                      |                                      |
                 |                                      | cloud_name: entry-scale-kvm-esx-mml     |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | test_type: ping                         |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | hostname: ardana-cp1-esx-ovsvapp0001-mgmt |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | control_plane: control-plane-1          |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | cluster: mtrmon                         |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | observer_host: ardana-cp1-mtrmon-m1-mgmt  |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 | host_alive_status | service: system                         |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | cloud_name: entry-scale-kvm-esx-mml     |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | test_type: ping                         |
        |         |      |                       |      |                          |
          |                          |
|                                      |                                      |
                 |                                      | hostname: ardana-cp1-esx-ovsvapp0001-mgmt |
```

```
|       |     |          |      |                     |
   |          |
|                        |                              |
      |                                 | control_plane: control-plane-1       |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | cluster: mtrmon                      |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | observer_host: ardana-cp1-mtrmon-m3-mgmt |
   |     |          |      |                     |
      |                        |
|                        |                              |
              | host_alive_status | service: system                 |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | cloud_name: entry-scale-kvm-esx-mml    |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | test_type: ping                       |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | hostname: ardana-cp1-esx-ovsvapp0001-mgmt |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | control_plane: control-plane-1       |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | cluster: mtrmon                      |
   |     |          |      |                     |
      |                        |
|                        |                              |
                         | observer_host: ardana-cp1-mtrmon-m2-mgmt |
   |     |          |      |                     |
      |                        |
+----------------------------------------+----------------------------------
+-----------------------+-------------------
+--------------------------------------------+----------+-------
```

```
+-----------------+------+-------------------------+-------------------------
+------------------------+
```

6. Delete the Monasca alaram.

```
monasca alarm-delete <alarm ID>
```

For example:

```
monasca alarm-delete cfc6bfa4-2485-4319-b1e5-0107886f4270Successfully deleted alarm
```

After deleting the alarms and updating the monasca-agent configuration, those alarms will be removed from the Operations Console UI. You can login to Operations Console and view the status.

6.3.4   Removing the Compute Proxy from Monitoring

Once you have removed the Compute proxy, the alarms against them will still trigger. Therefore to resolve this, you must perform the following steps.

1. SSH to Monasca API server. You must SSH to each Monasca API server for cleanup.
   For example:

   ```
   ssh ardana-cp1-mtrmon-m1-mgmt
   ```

2. Edit /etc/monasca/agent/conf.d/host_alive.yaml file to remove the reference to the Compute proxy you removed. This requires **sudo** access.

   ```
   sudo vi /etc/monasca/agent/conf.d/host_alive.yaml
   ```

   A sample of host_alive.yaml file.

   ```
   - alive_test: ping
     built_by: HostAlive
     host_name: MCP-VCP-cpesx-esx-comp0001-mgmt
     name: MCP-VCP-cpesx-esx-comp0001-mgmt ping
   ```

3. Once you have removed the references on each of your Monasca API servers, execute the following command to restart the monasca-agent on each of those servers.

   ```
   tux > sudo service openstack-monasca-agent restart
   ```

4. With the Compute proxy references removed and the monasca-agent restarted, delete the corresponding alarm to complete this process. complete the cleanup process. We recommend using the Monasca CLI which is installed on each of your Monasca API servers by default.

```
monasca alarm-list --metric-dimensions hostname= <compute node deleted>
```

For example: You can execute the following command to get the alarm ID, if the Compute proxy appears as a preceding example.

```
monasca alarm-list --metric-dimensions hostname=ardana-cp1-comp0001-mgmt
```

5. Delete the Monasca alarm

```
monasca alarm-delete <alarm ID>
```

6.3.5 Cleaning the Monasca Alarms Related to ESX Proxy and vCenter Cluster

Perform the following procedure:

1. Using the ESX proxy hostname, execute the following command to list all alarms.

```
monasca alarm-list --metric-dimensions hostname=COMPUTE_NODE_DELETED
```

where `COMPUTE_NODE_DELETED` - hostname is taken from the vSphere client (refer to *Section 6.3.1, "Prerequisites"*).

🎓 Note

Ensure to make a note of all the alarm IDs that is displayed after executing the preceding command.

For example, the compute proxy hostname is `MCP-VCP-cpesx-esx-comp0001-mgmt`.

```
monasca alarm-list --metric-dimensions hostname=MCP-VCP-cpesx-esx-comp0001-mgmt
ardana@R28N6340-701-cp1-c1-m1-mgmt:~$ monasca alarm-list --metric-dimensions
 hostname=R28N6340-701-cp1-esx-comp0001-mgmt
+------------------------------------+------------------------------------
+----------------------+----------------------
```

```
+------------------------------------------------------+----------+-------
+----------------+------+-----------------------+------------------------
+------------------------+
| id                                   | alarm_definition_id
   | alarm_definition_name | metric_name             | metric_dimensions
                        | severity | state | lifecycle_state | link |
 state_updated_timestamp | updated_timestamp      | created_timestamp       |
+------------------------------------+------------------------------------
+----------------------+-----------------------
+------------------------------------------------------+----------+-------
+----------------+------+-----------------------+------------------------
+------------------------+
| 02342bcb-da81-40db-a262-09539523c482 | 3e302297-0a36-4f0e-a1bd-03402b937a4e | HTTP
 Status               | http_status             | service: compute
         | HIGH     | OK    | None          | None | 2016-11-11T06:58:11.717Z |
 2016-11-11T06:58:11.717Z | 2016-11-10T08:55:45.136Z |
|                                      |                          |
                  |                          | cloud_name: entry-scale-esx-kvm
         |          |       |                 |      |                      |
                 |                          |
|                                      |                          |
                  |                          | url: https://10.244.209.9:8774
         |          |       |                 |      |                      |
                 |                          |
|                                      |                          |
                  |                          | hostname: R28N6340-701-cp1-esx-
comp0001-mgmt    |          |       |                 |      |
     |                          |                          |
|                                      |                          |
                  |                          | component: nova-api
         |          |       |                 |      |                      |
                 |                          |
|                                      |                          |
                  |                          | control_plane: control-plane-1
         |          |       |                 |      |                      |
                 |                          |
|                                      |                          |
                  |                          | cluster: esx-compute
         |          |       |                 |      |                      |
                 |                          |
| 04cb36ce-0c7c-4b4c-9ebc-c4011e2f6c0a | 15c593de-fa54-4803-bd71-afab95b980a4 | Disk
 Usage                | disk.space_used_perc   | mount_point: /proc/sys/fs/binfmt_misc
         | HIGH     | OK    | None          | None | 2016-11-10T08:52:52.886Z |
 2016-11-10T08:52:52.886Z | 2016-11-10T08:51:29.197Z |
|                                      |                          |
                  |                          | service: system
```

```
                     |            |       |                 |          |                      |
                            |                                    |
      |                                       |                                       |
                     |                                | cloud_name: entry-scale-esx-kvm
             |            |       |                 |          |                      |
                            |                                    |
      |                                       |                                       |
                     |                                | hostname: R28N6340-701-cp1-esx-
comp0001-mgmt        |            |       |                 |          |
          |                            |                                 |
      |                                       |                                       |
                     |                                | control_plane: control-plane-1
             |            |       |                 |          |                      |
                            |                                    |
      |                                       |                                       |
                     |                                | cluster: esx-compute
             |            |       |                 |          |                      |
                            |                                    |
      |                                       |                                       |
                     |                                | device: systemd-1
             |            |       |                 |          |                      |
                            |                                    |
      +--------------------------------------+-------------------------------------
      +--------------------+----------------------
      +-------------------------------------------------+----------+-------
      +---------------+------+------------------------+------------------------
      +----------------------+
```

2. Delete the alarm using the alarm IDs.

```
monasca alarm-delete <alarm ID>
```

This step has to be performed for all alarm IDs listed from the preceding step (*Step 1*).
For Example:

```
monasca alarm-delete 1cc219b1-ce4d-476b-80c2-0cafa53e1a12
```

6.4 Removing an ESXi Host from a Cluster

This topic describes how to remove an existing ESXi host from a cluster and clean up of services for OVSvAPP VM.

☞ Note

Before performing this procedure, wait until VCenter migrates all the tenant VMs to other active hosts in that same cluster.

6.4.1 Prerequisite

Write down the Hostname and ESXi configuration IP addresses of OVSvAPP VMs of that ESX cluster before deleting the VMs. These IP address and Hostname will be used to clean up Monasca alarm definitions.

1. Login to vSphere client.

2. Select the ovsvapp node running on the ESXi host and click **Summary** tab.

6.4.2 Procedure

1. Right-click and put the host in the maintenance mode. This will automatically migrate all the tenant VMs except OVSvApp.
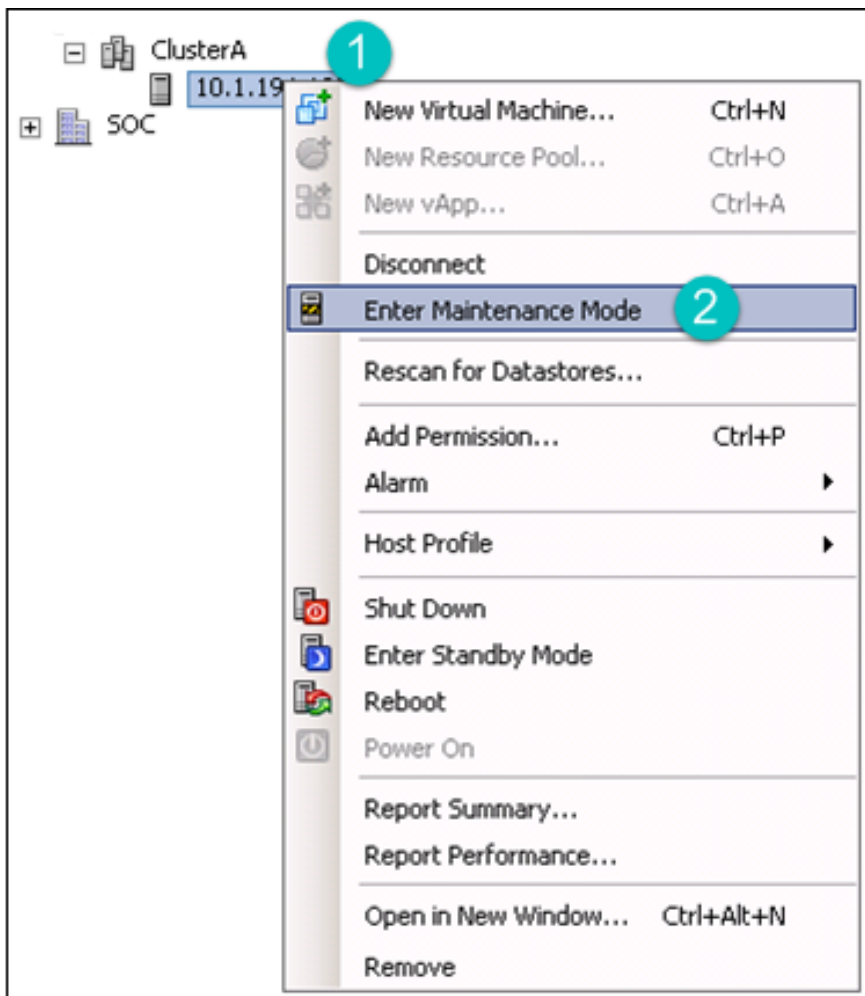
2. Cancel the maintenance mode task.

3. Right-click the **ovsvapp VM (IP Address)** node, select **Power**, and then click **Power Off**.
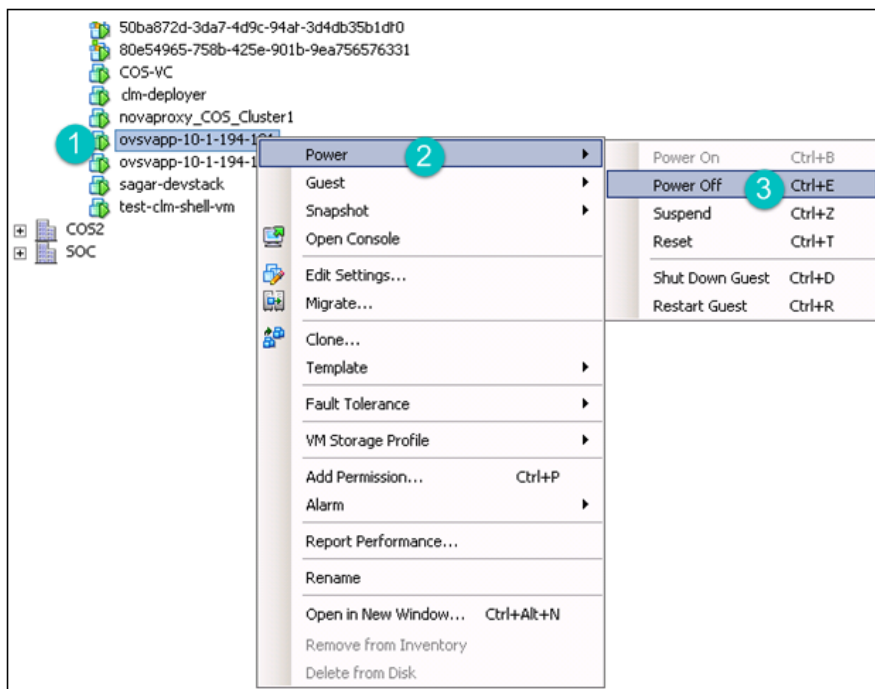
4. Right-click the node and then click **Delete from Disk**.
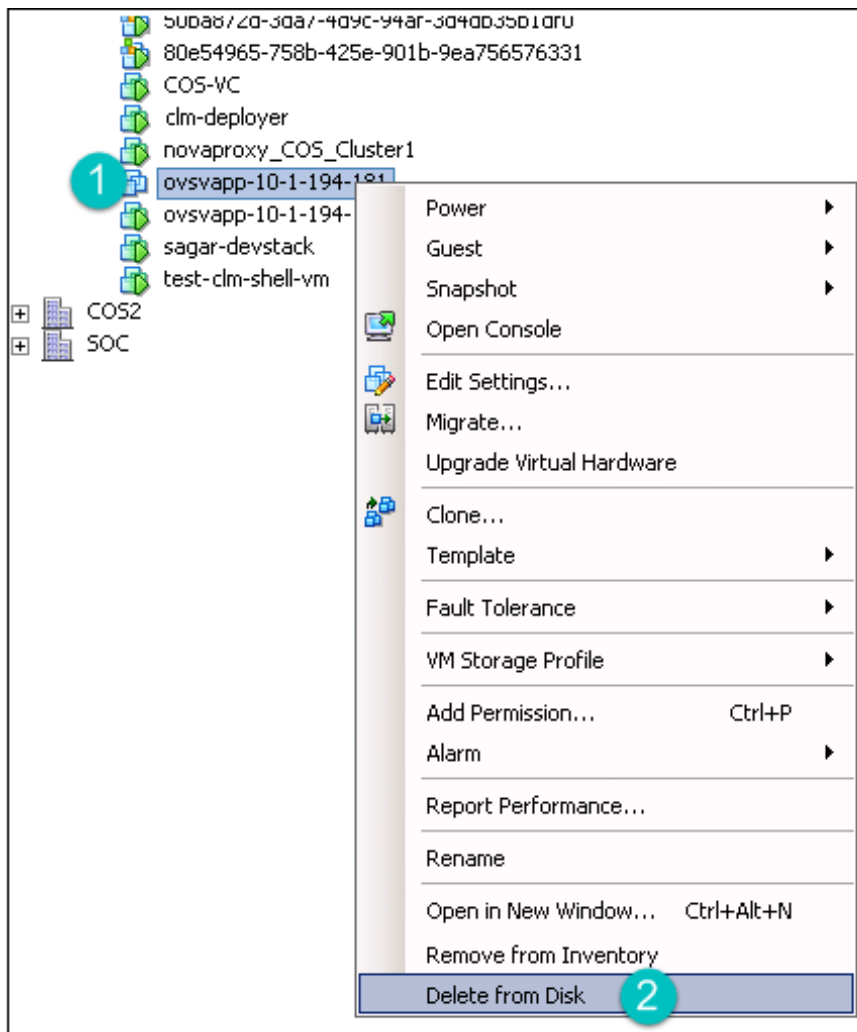
5. Right-click the **Host**, and then click **Enter Maintenance Mode**.

6. Disconnect the VM. Right-click the VM, and then click **Disconnect**.
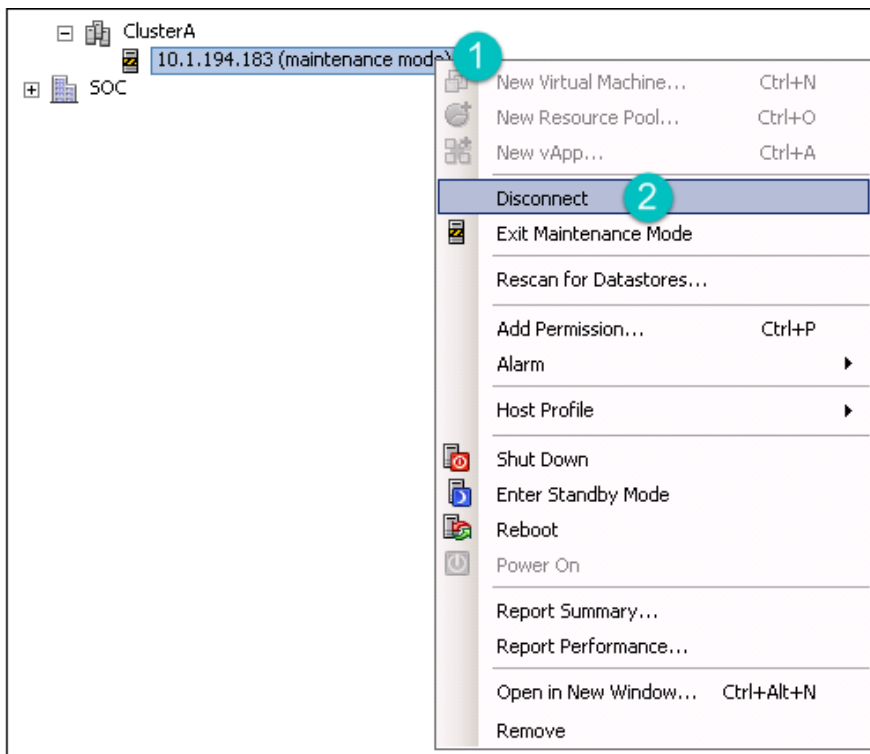
The ESXi node is removed from the vCenter.

6.4.3 Clean up Neutron Agent for OVSvAPP Service

After removing ESXi node from a vCenter, perform the following procedure to clean up neutron agents for ovsvapp-agent service.

1. Login to Cloud Lifecycle Manager.

2. Source the credentials.

   ```
   source service.osrc
   ```

3. Execute the following command.

   ```
   neutron agent-list | grep <OVSvapp hostname>
   ```

   For example:

   ```
   neutron agent-list | grep MCP-VCP-cpesx-esx-ovsvapp0001-mgmt
   ```

```
| 92ca8ada-d89b-43f9-b941-3e0cd2b51e49 | OVSvApp Agent      | MCP-VCP-cpesx-esx-
ovsvapp0001-mgmt |                    | :-)   | True          | ovsvapp-agent
     |
```

4. Delete the OVSvAPP agent.

```
neutron agent-delete <Agent -ID>
```

For example:

```
neutron agent-delete 92ca8ada-d89b-43f9-b941-3e0cd2b51e49
```

If you have more than one host, perform the preceding procedure for all the hosts.

6.4.4   Clean up Monasca Agent for OVSvAPP Service

Perform the following procedure to clean up Monasca agents for ovsvapp-agent service.

1. If Monasca-API is installed on different node, copy the `service.orsc` from Cloud Lifecycle Manager to Monasca API server.

```
scp service.orsc $USER@ardana-cp1-mtrmon-m1-mgmt:
```

2. SSH to Monasca API server. You must SSH to each Monasca API server for cleanup. For example:

```
ssh ardana-cp1-mtrmon-m1-mgmt
```

3. Edit `/etc/monasca/agent/conf.d/host_alive.yaml` file to remove the reference to the OVSvAPP you removed. This requires **sudo** access.

```
sudo vi /etc/monasca/agent/conf.d/host_alive.yaml
```

A sample of `host_alive.yaml`:

```
- alive_test: ping
  built_by: HostAlive
  host_name: MCP-VCP-cpesx-esx-ovsvapp0001-mgmt
  name: MCP-VCP-cpesx-esx-ovsvapp0001-mgmt ping
  target_hostname: MCP-VCP-cpesx-esx-ovsvapp0001-mgmt
```

where `host_name` and `target_hostname` are mentioned at the DNS name field at the vSphere client. (Refer to *Section 6.4.1, "Prerequisite"*).

4. After removing the reference on each of the Monasca API servers, restart the monasca-agent on each of those servers by executing the following command.

```
tux > sudo service openstack-monasca-agent restart
```

5. With the OVSvAPP references removed and the monasca-agent restarted, you can delete the corresponding alarm to complete the cleanup process. We recommend using the Monasca CLI which is installed on each of your Monasca API servers by default. Execute the following command from the Monasca API server (for example: `ardana-cp1-mtrmon-mX-mgmt`).

```
monasca alarm-list --metric-name host_alive_status --metric-dimensions
 hostname=<ovsvapp deleted>
```

For example: You can execute the following command to get the alarm ID, if the OVSvAPP appears as a preceding example.

```
monasca alarm-list --metric-name host_alive_status --metric-dimensions hostname=MCP-
VCP-cpesx-esx-ovsvapp0001-mgmt
+---------------------------------------+-------------------------------------
+----------------------+------------------
+--------------------------------------------+----------+-------
+----------------+------+------------------------+------------------------
+------------------------+
| id                                    | alarm_definition_id                |
 alarm_definition_name | metric_name      | metric_dimensions
       | severity | state | lifecycle_state | link | state_updated_timestamp  |
 updated_timestamp       | created_timestamp      |
+---------------------------------------+-------------------------------------
+----------------------+------------------
+--------------------------------------------+----------+-------
+----------------+------+------------------------+------------------------
+------------------------+
| cfc6bfa4-2485-4319-b1e5-0107886f4270 | cca96c53-a927-4b0a-9bf3-cb21d28216f3 |
 Host Status          | host_alive_status | service: system
       | HIGH     | OK    | None            | None | 2016-10-27T06:33:04.256Z |
 2016-10-27T06:33:04.256Z | 2016-10-23T13:41:57.258Z |
|                                       |                                     |
                   |                     | cloud_name: entry-scale-kvm-esx-mml        |
         |       |                 |      |                             |
           |                        |
|                                       |                                     |
                   |                     | test_type: ping                            |
         |       |                 |      |                             |
           |                        |
```

```
|                     |                   |                                        |
|                     |                   | hostname: ardana-cp1-esx-ovsvapp0001-mgmt |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | control_plane: control-plane-1         |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | cluster: mtrmon                        |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | observer_host: ardana-cp1-mtrmon-m1-mgmt |
|                     |                   |                                        |
|                     |                   |                                        |
|                     | host_alive_status | service: system                        |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | cloud_name: entry-scale-kvm-esx-mml    |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | test_type: ping                        |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | hostname: ardana-cp1-esx-ovsvapp0001-mgmt |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | control_plane: control-plane-1         |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | cluster: mtrmon                        |
|                     |                   |                                        |
|                     |                   |                                        |
|                     |                   | observer_host: ardana-cp1-mtrmon-m3-mgmt |
|                     |                   |                                        |
|                     |                   |                                        |
|                     | host_alive_status | service: system                        |
```

```
          |            |                    |          |                              |
            |               |                              |
  |                             |                                           |              |
                |                     | cloud_name: entry-scale-kvm-esx-mml       |
          |        |             |          |                              |
            |                              |
  |                             |                                           |              |
                |                     | test_type: ping                           |
          |        |             |          |                              |
            |                              |
  |                             |                                           |              |
                |                     | hostname: ardana-cp1-esx-ovsvapp0001-mgmt |
          |        |             |          |                              |
            |                              |
  |                             |                                           |              |
                |                     | control_plane: control-plane-1            |
          |        |             |          |                              |
            |                              |
  |                             |                                           |              |
                |                     | cluster: mtrmon                           |
          |        |             |          |                              |
            |                              |
  |                             |                                           |              |
                |                     | observer_host: ardana-cp1-mtrmon-m2-mgmt  |
          |        |             |          |                              |
            |                              |
+------------------------------------+------------------------------------
+----------------------+------------------
+-----------------------------------------+---------+-------
+----------------+------+-------------------------+-------------------------
+------------------------+
```

6. Delete the Monasca alaram.

```
monasca alarm-delete <alarm ID>
```

For example:

```
monasca alarm-delete cfc6bfa4-2485-4319-b1e5-0107886f4270Successfully deleted alarm
```

After deleting the alarms and updating the monasca-agent configuration, those alarms will be removed from the Operations Console UI. You can login to Operations Console and view the status.

6.4.5　Clean up the entries of OVSvAPP VM from /etc/host

Perform the following procedure to clean up the entries of OVSvAPP VM from `/etc/host`.

1. Login to Cloud Lifecycle Manager.

2. Edit `/etc/host`.

   ```
   vi /etc/host
   ```

   For example: `MCP-VCP-cpesx-esx-ovsvapp0001-mgmt` VM is present in the `/etc/host`.

   ```
   192.168.86.17     MCP-VCP-cpesx-esx-ovsvapp0001-mgmt
   ```

3. Delete the OVSvAPP entries from `/etc/host`.

6.4.6　Remove the OVSVAPP VM from the servers.yml and pass_through.yml files and run the Configuration Processor

Complete these steps from the Cloud Lifecycle Manager to remove the OVSvAPP VM:

1. Log in to the Cloud Lifecycle Manager

2. Edit `servers.yml` file to remove references to the OVSvAPP VM(s) you want to remove:

   ```
   ~/openstack/my_cloud/definition/data/servers.yml
   ```

   For example:

   ```
   - ip-addr:192.168.86.17
     server-group: AZ1     role:
     OVSVAPP-ROLE    id:
     6afaa903398c8fc6425e4d066edf4da1a0f04388
   ```

3. Edit `~/openstack/my_cloud/definition/data/pass_through.yml` file to remove the OVSvAPP VM references using the server-id above section to find the references.

   ```
   - data:
     vmware:
     vcenter_cluster: Clust1
     cluster_dvs_mapping: 'DC1/host/Clust1:TRUNK-DVS-Clust1'
     esx_hostname: MCP-VCP-cpesx-esx-ovsvapp0001-mgmt
   ```

```
   vcenter_id: 0997E2ED9-5E4F-49EA-97E6-E2706345BAB2
id: 6afaa903398c8fc6425e4d066edf4da1a0f04388
```

4. Commit the changes to git:

```
git commit -a -m "Remove ESXi host <name>"
```

5. Run the configuration processor. You may want to use the `remove_deleted_servers` and `free_unused_addresses` switches to free up the resources when running the configuration processor. See *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 7 "Other Topics", Section 7.3 "Persisted Data"* for more details.

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml -e
 remove_deleted_servers="y" -e free_unused_addresses="y"
```

6. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6.4.7 Remove Distributed Resource Scheduler (DRS) Rules

Perform the following procedure to remove DRS rules, which is added by OVSvAPP installer to ensure that OVSvAPP does not get migrated to other hosts.

1. Login to vCenter.

2. Right click on cluster and select **Edit settings**.
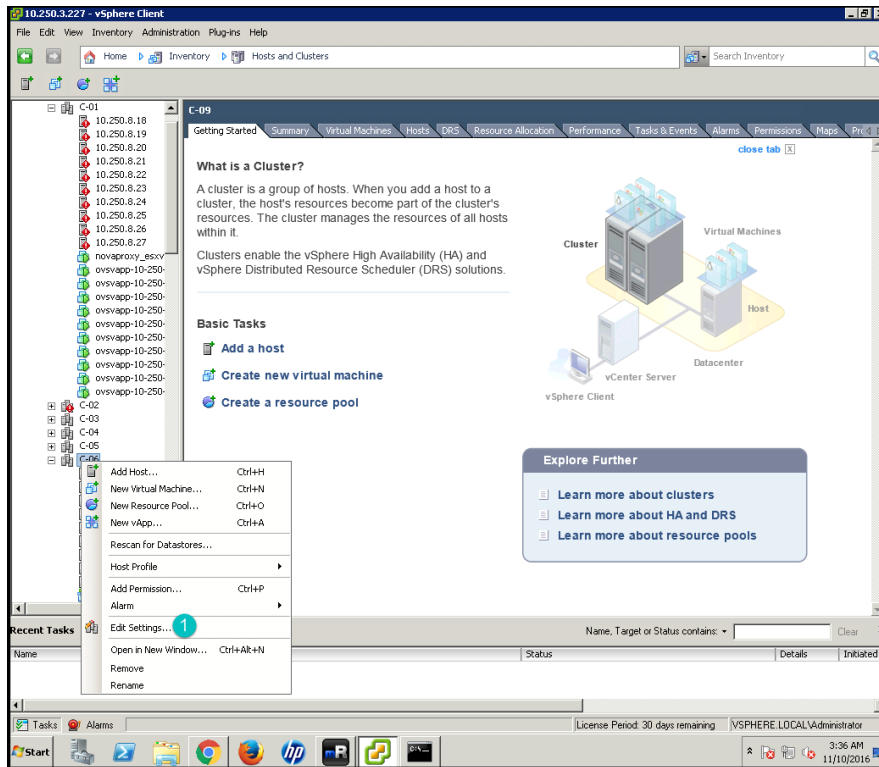
A cluster settings page appears.

3. Click **DRS Groups Manager** on the left hand side of the pop-up box. Select the group which is created for deleted OVSvAPP and click Remove.
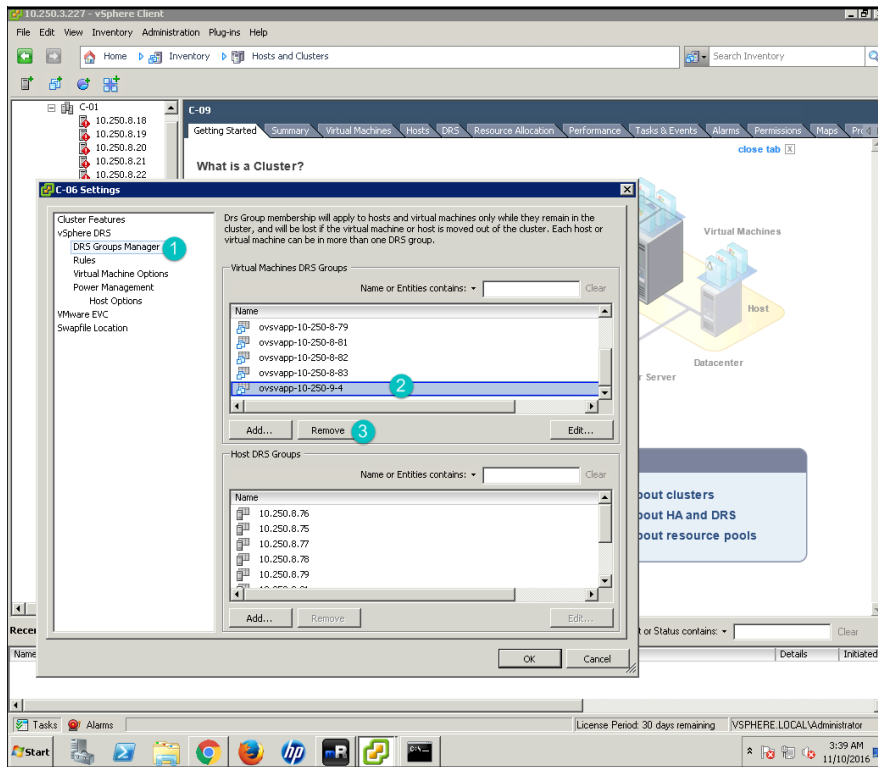
4. Click **Rules** on the left hand side of the pop-up box and select the checkbox for deleted OVSvAPP and click **Remove**.
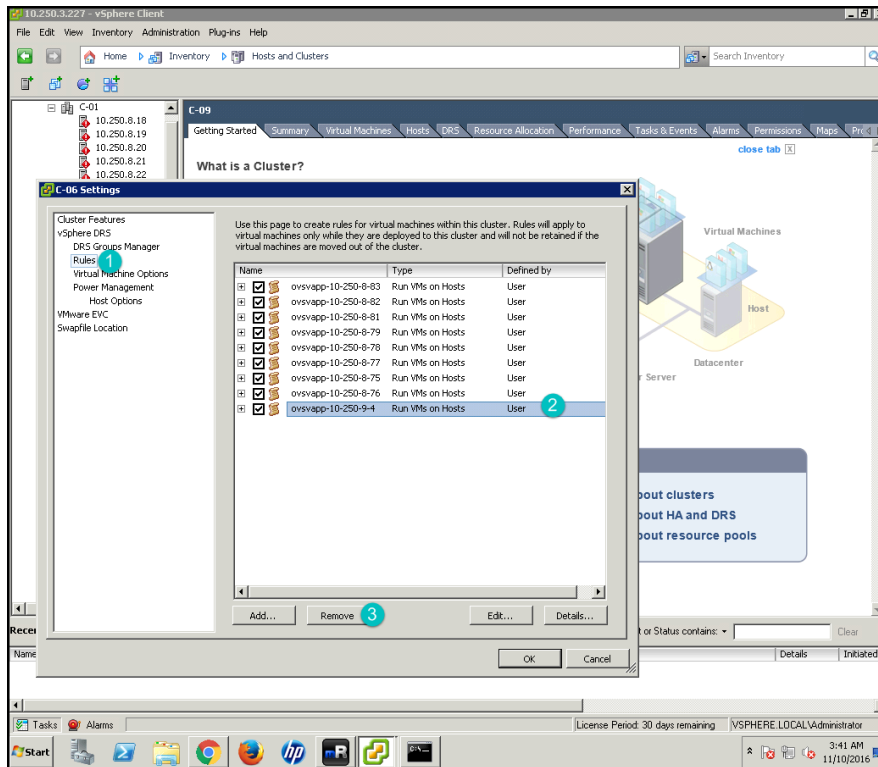
5. Click **OK**.

6.5 Configuring Debug Logging

6.5.1 To Modify the OVSVAPP VM Log Level

To change the OVSVAPP log level to DEBUG, do the following:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the file below:

   ```
   ~/openstack/ardana/ansible/roles/neutron-common/templates/ovsvapp-agent-
   logging.conf.j2
   ```

3. Set the logging level value of the `logger_root` section to `DEBUG`, like this:

   ```
   [logger_root]
   qualname: root
   ```

```
handlers: watchedfile, logstash
level: DEBUG
```

4. Commit your configuration to the Git repository (*Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

5. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Deploy your changes:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
```

6.5.2 To Enable OVSVAPP Service for Centralized Logging

To enable OVSVAPP Service for centralized logging:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the file below:

```
~/openstack/my_cloud/config/logging/vars/neutron-ovsvapp-clr.yml
```

3. Set the value of `centralized_logging` to **true** as shown in the following sample:

```
logr_services:
  neutron-ovsvapp:
    logging_options:
    - centralized_logging:
        enabled: true
```

```
        format: json
        ...
```

4. Commit your configuration to the Git repository (*Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

5. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Deploy your changes, specifying the hostname for your OVSAPP host:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml --limit <hostname>
```

The hostname of the node can be found in the list generated from the output of the following command:

```
grep hostname ~/openstack/my_cloud/info/server_info.yml
```

6.6 Making Scale Configuration Changes

This procedure describes how to make the recommended configuration changes to achieve 8,000 virtual machine instances.

Note

In a scale environment for ESX computes, the configuration of vCenter Proxy VM has to be increased to 8 vCPUs and 16 GB RAM. By default it is 4 vCPUs and 4 GB RAM.

1. Change the directory. The `nova.conf.j2` file is present in following directories:

```
cd ~/openstack/ardana/ansible/roles/nova-common/templates
```

2. Edit the DEFAULT section in the `nova.conf.j2` file as below:

```
[DEFAULT]
rpc_responce_timeout = 180
server_down_time = 300
report_interval = 30
```

3. Commit your configuration:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "<commit message>"
```

4. Prepare your environment for deployment:

```
ansible-playbook -i hosts/localhost ready-deployment.yml;
cd ~/scratch/ansible/next/ardana/ansible;
```

5. Execute the `nova-reconfigure` playbook:

```
ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

6.7 Monitoring vCenter Clusters

Remote monitoring of activated ESX cluster is enabled through vCenter Plugin of Monasca. The Monasca-agent running in each ESX Compute proxy node is configured with the vcenter plugin, to monitor the cluster.

Alarm definitions are created with the default threshold values and whenever the threshold limit breaches respective alarms (OK/ALARM/UNDETERMINED) are generated.

The configuration file details is given below:

```
init_config: {}
instances:
  - vcenter_ip: <vcenter-ip>
      username: <vcenter-username>
      password: <center-password>
      clusters: <[cluster list]>
```

Metrics List of metrics posted to monasca by vCenter Plugin are listed below:

- vcenter.cpu.total_mhz

- vcenter.cpu.used_mhz

- vcenter.cpu.used_perc

- vcenter.cpu.total_logical_cores

- vcenter.mem.total_mb

- vcenter.mem.used_mb

- vcenter.mem.used_perc

- vcenter.disk.total_space_mb

- vcenter.disk.total_used_space_mb

- vcenter.disk.total_used_space_perc

monasca measurement-list --dimensions esx_cluster_id=domain-c7.D99502A9-63A8-41A2-B3C3-D8E31B591224 vcenter.disk.total_used_space_mb 2016-08-30T11:20:08

```
+---------------------------------------------
+-------------------------------------------------------------------------------------
+---------------------------------+-----------------+-----------------+
| name                                         | dimensions
                                          | timestamp
 | value             | value_meta      |
+---------------------------------------------
+-------------------------------------------------------------------------------------
+---------------------------------+-----------------+-----------------+
| vcenter.disk.total_used_space_mb             | vcenter_ip: 10.1.200.91
                                          | 2016-08-30T11:20:20.703Z
 | 100371.000        |                 |
|                                              | esx_cluster_id: domain-
c7.D99502A9-63A8-41A2-B3C3-D8E31B591224                                       |
 2016-08-30T11:20:50.727Z             | 100371.000      |                 |
|                                              | hostname: MCP-VCP-cpesx-esx-comp0001-
mgmt                                          | 2016-08-30T11:21:20.707Z
   | 100371.000        |                 |
|                                              |
                                          | 2016-08-30T11:21:50.700Z
 | 100371.000        |                 |
```

```
|                                                   |
                                                   | 2016-08-30T11:22:20.700Z
 | 100371.000      |                |
|                                                   |
                                                   | 2016-08-30T11:22:50.700Z
 | 100371.000      |                |
|                                                   |
                                                   | 2016-08-30T11:23:20.620Z
 | 100371.000      |                |
+---------------------------------------
 +----------------------------------------------------------------------------------
 +--------------------------------+----------------+----------------+
```

Dimensions

Each metric will have the dimension as below

vcenter_ip

> FQDN/IP Address of the registered vCenter

server esx_cluster_id

> clusterName.vCenter-id, as seen in the nova hypervisor-list

hostname

> ESX compute proxy name

Alarms

Alarms are created for monitoring cpu, memory and disk usages for each activated clusters. The alarm definitions details are

Name	Expression	Severity	Match_by
ESX cluster CPU Usage	avg(vcenter.cpu.used_perc) > 90 times 3	High	esx_cluster_id
ESX cluster Memory Usage	avg(vcenter.mem.used_perc) > 90 times 3	High	esx_cluster_id
ESX cluster Disk Usage	vcenter.disk.total_used_space_perc > 90	High	esx_cluster_id

6.8 Monitoring Integration with OVSvApp Appliance

6.8.1 Processes Monitored with Monasca Agent

Using the Monasca agent, the following services are monitored on the OVSvApp appliance:

- **Neutron_ovsvapp_agent service** - This is the Neutron agent which runs in the appliance which will help enable networking for the tenant virtual machines.

- **Openvswitch** - This service is used by the neutron_ovsvapp_agent service for enabling the datapath and security for the tenant virtual machines.

- **Ovsdb-server** - This service is used by the neutron_ovsvapp_agent service.

If any of the above three processes fail to run on the OVSvApp appliance it will lead to network disruption for the tenant virtual machines. This is why they are monitored.

The monasca-agent periodically reports the status of these processes and metrics data ('load' - cpu.load_avg_1min, 'process' - process.pid_count, 'memory' - mem.usable_perc, 'disk' - disk.space_used_perc, 'cpu' - cpu.idle_perc for examples) to the Monasca server.

6.8.2 How It Works

Once the vApp is configured and up, the monasca-agent will attempt to register with the Monasca server. After successful registration, the monitoring begins on the processes listed above and you will be able to see status updates on the server side.

The monasca-agent monitors the processes at the system level so, in the case of failures of any of the configured processes, updates should be seen immediately from Monasca.

To check the events from the server side, log into the Operations Console. For more details on how to use the Operations Console, see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.1 "Operations Console Overview"*.

7 Managing Block Storage

Information about managing and configuring the Block Storage service.

7.1 Managing Block Storage using Cinder

HPE Helion OpenStack Block Storage volume operations use the OpenStack Cinder service to manage storage volumes, which includes creating volumes, attaching/detaching volumes to Nova instances, creating volume snapshots, and configuring volumes.

HPE Helion OpenStack supports the following storage back ends for block storage volumes and backup datastore configuration:

- Volumes

  - SUSE Enterprise Storage; for more information, see *Book "Installing with Cloud Lifecycle Manager", Chapter 23 "Integrations", Section 23.3 "SUSE Enterprise Storage Integration"*.

  - 3PAR FC or iSCSI; for more information, see *Book "Installing with Cloud Lifecycle Manager", Chapter 23 "Integrations", Section 23.1 "Configuring for 3PAR Block Storage Backend"*.

- Backup

  - Swift

7.1.1 Setting Up Multiple Block Storage Back-ends

HPE Helion OpenStack supports setting up multiple block storage backends and multiple volume types.

Whether you have a single or multiple block storage back-ends defined in your `cinder.conf.j2` file, you can create one or more volume types using the specific attributes associated with the back-end. You can find details on how to do that for each of the supported back-end types here:

- *Book "Installing with Cloud Lifecycle Manager", Chapter 23 "Integrations", Section 23.3 "SUSE Enterprise Storage Integration"*

- *Book "Installing with Cloud Lifecycle Manager", Chapter 23 "Integrations", Section 23.1 "Configuring for 3PAR Block Storage Backend"*

7.1.2 Creating a Volume Type for your Volumes

Creating volume types allows you to create standard specifications for your volumes.

Volume types are used to specify a standard Block Storage back-end and collection of extra specifications for your volumes. This allows an administrator to give its users a variety of options while simplifying the process of creating volumes.

The tasks involved in this process are:

7.1.2.1 Create a Volume Type for your Volumes

The default volume type will be thin provisioned and will have no fault tolerance (RAID 0). You should configure Cinder to fully provision volumes, and you may want to configure fault tolerance. Follow the instructions below to create a new volume type that is fully provisioned and fault tolerant:

Perform the following steps to create a volume type using the Horizon GUI:

1. Log in to the Horizon dashboard. See *Book "User Guide", Chapter 3 "Cloud Admin Actions with the Dashboard"* for details on how to do this.

2. Ensure that you are scoped to your `admin` Project. Then under the *Admin* menu in the navigation pane, click on *Volumes* under the *System* subheading.

3. Select the *Volume Types* tab and then click the *Create Volume Type* button to display a dialog box.

4. Enter a unique name for the volume type and then click the *Create Volume Type* button to complete the action.

The newly created volume type will be displayed in the `Volume Types` list confirming its creation.

🔔 Important

If you do not specify a default type then your volumes will default unpredictably. We recommend that you create a volume type that meets the needs of your environment and specify it here.

7.1.2.2 Associate the Volume Type to the Back-end

After the volume type(s) have been created, you can assign extra specification attributes to the volume types. Each Block Storage back-end option has unique attributes that can be used.

- *Section 7.1.2.3, "Extra Specification Options for 3PAR"*

To map a volume type to a back-end, do the following:

1. Log into the Horizon dashboard. See *Book "User Guide", Chapter 3 "Cloud Admin Actions with the Dashboard"* for details on how to do this.

2. Ensure that you are scoped to your *admin* Project (for more information, see *Section 4.10.7, "Scope Federated User to Domain"*. Then under the *Admin* menu in the navigation pane, click on *Volumes* under the *System* subheading.

3. Click the *Volume Type* tab to list the volume types.

4. In the *Actions* column of the Volume Type you created earlier, click the drop-down option and select *View Extra Specs* which will bring up the *Volume Type Extra Specs* options.

5. Click the *Create* button on the `Volume Type Extra Specs` screen.

6. In the `Key` field, enter one of the key values in the table in the next section. In the `Value` box, enter its corresponding value. Once you have completed that, click the *Create* button to create the extra volume type specs.

Once the volume type is mapped to a back-end, you can create volumes with this volume type.

7.1.2.3 Extra Specification Options for 3PAR

3PAR supports volumes creation with additional attributes. These attributes can be specified using the extra specs options for your volume type. The administrator is expected to define appropriate extra spec for 3PAR volume type as per the guidelines provided at http:// docs.openstack.org/liberty/config-reference/content/hp-3par-supported-ops.html ↗ .

The following Cinder Volume Type extra-specs options enable control over the 3PAR storage provisioning type:

Key	Value	Description
volume_backend_name	*volume backend name*	The name of the back-end to which you want to associate the volume type, which you also specified earlier in the `cinder.conf.j2` file.
hp3par:provisioning (optional)	thin, full, or dedup	

See OpenStack HPE 3PAR StoreServ Block Storage Driver Configuration Best Practices (https://h20195.www2.hpe.com/v2/GetPDF.aspx/4AA5-1930ENW.pdf) ↗ for more details.

7.1.3　Managing Cinder Volume and Backup Services

🔔 Important: Use Only When Needed

If the host running the `cinder-volume` service fails for any reason, it should be restarted as quickly as possible. Often, the host running Cinder services also runs high availability (HA) services such as MariaDB and RabbitMQ. These HA services are at risk while one of the nodes in the cluster is down. If it will take a significant amount of time to recover the failed node, then you may migrate the `cinder-volume` service and its backup service to one of the other controller nodes. When the node has been recovered, you should migrate the `cinder-volume` service and its backup service to the original (default) node.

The `cinder-volume` service and its backup service migrate as a pair. If you migrate the `cinder-volume` service, its backup service will also be migrated.

7.1.3.1　Migrating the cinder-volume service

The following steps will migrate the cinder-volume service and its backup service.

1. Log in to the Cloud Lifecycle Manager node.

2. Determine the host index numbers for each of your control plane nodes. This host index number will be used in a later step. They can be obtained by running this playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts cinder-show-volume-hosts.yml
```

Here is an example snippet showing the output of a single three node control plane, with the host index numbers in bold:

```
TASK: [_CND-CMN | show_volume_hosts | Show Cinder Volume hosts index and hostname]
 ***
ok: [ardana-cp1-c1-m1] => (item=(0, 'ardana-cp1-c1-m1')) => {
    "item": [
        0,
        "ardana-cp1-c1-m1"
    ],
    "msg": "Index 0 Hostname ardana-cp1-c1-m1"
}
ok: [ardana-cp1-c1-m1] => (item=(1, 'ardana-cp1-c1-m2')) => {
    "item": [
        1,
        "ardana-cp1-c1-m2"
    ],
    "msg": "Index 1 Hostname ardana-cp1-c1-m2"
}
ok: [ardana-cp1-c1-m1] => (item=(2, 'ardana-cp1-c1-m3')) => {
    "item": [
        2,
        "ardana-cp1-c1-m3"
    ],
    "msg": "Index 2 Hostname ardana-cp1-c1-m3"
}
```

3. Locate the control plane fact file for the control plane you need to migrate the service from. It will be located in the following directory:

```
/etc/ansible/facts.d/
```

These fact files use the following naming convention:

```
cinder_volume_run_location_<control_plane_name>.fact
```

4. Edit the fact file to include the host index number of the control plane node you wish to migrate the `cinder-volume` services to. For example, if they currently reside on your first controller node, host index 0, and you wish to migrate them to your second controller, you would change the value in the fact file to `1`.

5. If you are using data encryption on your Cloud Lifecycle Manager, ensure you have included the encryption key in your environment variables. For more information see *Book "Security Guide", Chapter 9 "Encryption of Passwords and Sensitive Data"*.

```
export HOS_USER_PASSWORD_ENCRYPT_KEY=<encryption key>
```

6. After you have edited the control plane fact file, run the Cinder volume migration playbook for the control plane nodes involved in the migration. At minimum this includes the one to start cinder-volume manager on and the one on which to stop it:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts cinder-migrate-volume.yml --
limit=<limit_pattern1,limit_pattern2>
```

> **⚲ Note**
>
> *< limit_pattern >* is the pattern used to limit the hosts that are selected to those within a specific control plane. For example, with the nodes in the snippet shown above, `--limit=>ardana-cp1-c1-m1,ardana-cp1-c1-m2<`

7. Even though the playbook summary reports no errors, you may disregard informational messages such as:

```
msg: Marking ardana_notify_cinder_restart_required to be cleared from the fact cache
```

8. Ensure that once your maintenance or other tasks are completed that you migrate the `cinder-volume` services back to their original node using these same steps.

8 Managing Object Storage

Information about managing and configuring the Object Storage service.

Managing your object storage environment includes tasks related to ensuring your Swift rings stay balanced and we discuss that and other topics in more detail in this section.

You can verify the Swift object storage operational status using commands and utilities. This section covers the following topics:

8.1 Running the Swift Dispersion Report

Swift contains a tool called `swift-dispersion-report` that can be used to determine whether your containers and objects have three replicas like they are supposed to. This tool works by populating a percentage of partitions in the system with containers and objects (using `swift-dispersion-populate`) and then running the report to see if all the replicas of these containers and objects are in the correct place. For a more detailed explanation of this tool in Openstack Swift, please see OpenStack Swift - Administrator's Guide (http://docs.openstack.org/developer/swift/admin_guide.html#cluster-health) ↗.

8.1.1 Configuring the Swift dispersion populate

Once a Swift system has been fully deployed in HPE Helion OpenStack 8, you can setup the swift-dispersion-report using the default parameters found in `~/openstack/ardana/ansible/roles/swift-dispersion/templates/dispersion.conf.j2`. This populates 1% of the partitions on the system and if you are happy with this figure, please proceed to step 2 below. Otherwise, follow step 1 to edit the configuration file.

1. If you wish to change the dispersion coverage percentage then edit the value of `dispersion_coverage` in the `~/openstack/ardana/ansible/roles/swift-dispersion/templates/dispersion.conf.j2` file to the value you wish to use. In the example below we have altered the file to create 5% dispersion:

   ```
   ...
   [dispersion]
   auth_url = {{ keystone_identity_uri }}/v3
   auth_user = {{ swift_dispersion_tenant }}:{{ swift_dispersion_user }}
   auth_key = {{ swift_dispersion_password  }}
   endpoint_type = {{ endpoint_type }}
   ```

```
auth_version = {{ disp_auth_version }}
# Set this to the percentage coverage. We recommend a value
# of 1%. You can increase this to get more coverage. However, if you
# decrease the value, the dispersion containers and objects are
# not deleted.
dispersion_coverage = 5.0
```

2. Commit your configuration to the Git repository (*Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
ardana > git add -A
ardana > git commit -m "My config or other commit message"
```

3. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

4. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

5. Reconfigure the Swift servers:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
```

6. Run this playbook to populate your Swift system for the health check:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-dispersion-populate.yml
```

8.1.2 Running the Swift dispersion report

Check the status of the Swift system by running the Swift dispersion report with this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-dispersion-report.yml
```

The output of the report will look similar to this:

```
TASK: [swift-dispersion | report | Display dispersion report results] *********
ok: [padawan-ccp-c1-m1-mgmt] => {
```

```
    "var": {
        "dispersion_report_result.stdout_lines": [
            "Using storage policy: General ",
            "",
            "[KQueried 40 containers for dispersion reporting, 0s, 0 retries",
            "100.00% of container copies found (120 of 120)",
            "Sample represents 0.98% of the container partition space",
            "",
            "[KQueried 40 objects for dispersion reporting, 0s, 0 retries",
            "There were 40 partitions missing 0 copies.",
            "100.00% of object copies found (120 of 120)",
            "Sample represents 0.98% of the object partition space"
        ]
    }
}
...
```

In addition to being able to run the report above, there will be a cron-job running every 2 hours on the first proxy node of your system that will run `dispersion-report` and save the results to the following file:

```
/var/cache/swift/dispersion-report
```

When interpreting the results you get from this report, we recommend using Swift Administrator's Guide - Cluster Health (http://docs.openstack.org/developer/swift/admin_guide.html#cluster-health) ↗

8.2   Gathering Swift Data

The `swift-recon` command retrieves data from Swift servers and displays the results. To use this command, log on as a root user to any node which is running the swift-proxy service.

8.2.1   Notes

For help with the `swift-recon` command you can use this:

```
tux > sudo swift-recon --help
```

⛨ Warning

The `--driveaudit` option is not supported.

⛨ Warning

HPE Helion OpenStack does not support ec_type `isa_l_rs_vand` and `ec_num_parity_fragments` greater than or equal to **5** in the storage-policy configuration. This particular policy is known to harm data durability.

8.2.2　Using the swift-recon Command

The following command retrieves and displays disk usage information:

```
tux > sudo swift-recon --diskusage
```

For example:

```
tux > sudo swift-recon --diskusage
===============================================================================
--> Starting reconnaissance on 3 hosts
===============================================================================
[2015-09-14 16:01:40] Checking disk usage now
Distribution Graph:
 10%    3 *********************************************************************
 11%    1 **********************
 12%    2 ********************************************
Disk usage: space used: 13745373184 of 119927734272
Disk usage: space free: 106182361088 of 119927734272
Disk usage: lowest: 10.39%, highest: 12.96%, avg: 11.4613798613%
===============================================================================
```

In the above example, the results for several nodes are combined together. You can also view the results from individual nodes by adding the **-v** option as shown in the following example:

```
tux > sudo swift-recon --diskusage -v
===============================================================================
--> Starting reconnaissance on 3 hosts
===============================================================================
[2015-09-14 16:12:30] Checking disk usage now
-> http://192.168.245.3:6000/recon/diskusage: [{'device': 'disk1', 'avail': 17398411264,
 'mounted': True, 'used': 2589544448, 'size': 19987955712}, {'device': 'disk0', 'avail':
 17904222208, 'mounted': True, 'used': 2083733504, 'size': 19987955712}]
-> http://192.168.245.2:6000/recon/diskusage: [{'device': 'disk1', 'avail': 17769721856,
 'mounted': True, 'used': 2218233856, 'size': 19987955712}, {'device': 'disk0', 'avail':
 17793581056, 'mounted': True, 'used': 2194374656, 'size': 19987955712}]
-> http://192.168.245.4:6000/recon/diskusage: [{'device': 'disk1', 'avail': 17912147968,
 'mounted': True, 'used': 2075807744, 'size': 19987955712}, {'device': 'disk0', 'avail':
 17404235776, 'mounted': True, 'used': 2583719936, 'size': 19987955712}]
```

```
Distribution Graph:
  10%     3 ************************************************************
  11%     1 **********************
  12%     2 *********************************************
Disk usage: space used: 13745414144 of 119927734272
Disk usage: space free: 106182320128 of 119927734272
Disk usage: lowest: 10.39%, highest: 12.96%, avg: 11.4614140152%
  ================================================================================
```

By default, `swift-recon` uses the object-0 ring for information about nodes and drives. For some commands, it is appropriate to specify **account**, **container**, or **object** to indicate the type of ring. For example, to check the checksum of the account ring, use the following:

```
tux > sudo swift-recon --md5 account
  ================================================================================
--> Starting reconnaissance on 3 hosts
  ================================================================================
[2015-09-14 16:17:28] Checking ring md5sums
3/3 hosts matched, 0 error[s] while checking hosts.
  ================================================================================
[2015-09-14 16:17:28] Checking swift.conf md5sum
3/3 hosts matched, 0 error[s] while checking hosts.
  ================================================================================
```

8.3 Gathering Swift Monitoring Metrics

The `swiftlm-scan` command is the mechanism used to gather metrics for the Monasca system. These metrics are used to derive alarms. For a list of alarms that can be generated from this data, see *Section 15.1.1, "Alarm Resolution Procedures"*.

To view the metrics, use the `swiftlm-scan` command directly. Log on to the Swift node as the root user. The following example shows the command and a snippet of the output:

```
tux > sudo swiftlm-scan --pretty
. . .
  {
    "dimensions": {
      "device": "sdc",
      "hostname": "padawan-ccp-c1-m2-mgmt",
      "service": "object-storage"
    },
    "metric": "swiftlm.swift.drive_audit",
    "timestamp": 1442248083,
```

```
    "value": 0,
    "value_meta": {
      "msg": "No errors found on device: sdc"
    }
  },
. . .
```

📖 Note

To make the JSON file easier to read, use the `--pretty` option.

The fields are as follows:

`metric`	Specifies the name of the metric.
`dimensions`	Provides information about the source or location of the metric. The dimensions differ depending on the metric in question. The following dimensions are used by `swiftlm-scan`: • **service**: This is always **object-storage**. • **component**: This identifies the component. For example, **swift-object-server** indicates that the metric is about the swift-object-server process. • **hostname**: This is the name of the node the metric relates to. This is not necessarily the name of the current node. • **url**: If the metric is associated with a URL, this is the URL.

	• **port**: If the metric relates to connectivity to a node, this is the port used.
	• **device**: This is the block device a metric relates to.
`value`	The value of the metric. For many metrics, this is simply the value of the metric. However, if the value indicates a status. If `value_meta` contains a **msg** field, the value is a status. The following status values are used: • 0 - no error • 1 - warning • 2 - failure
`value_meta`	Additional information. The **msg** field is the most useful of this information.

8.3.1   Optional Parameters

You can focus on specific sets of metrics by using one of the following optional parameters:

`--replication`	Checks replication and health status.
`--file-ownership`	Checks that Swift owns its relevant files and directories.
`--drive-audit`	Checks for logged events about corrupted sectors (unrecoverable read errors) on drives.

`--connectivity`	Checks connectivity to various servers used by the Swift system, including: • Checks this node can connect to all memcachd servers • Checks that this node can connect to the Keystone service (only applicable if this is a proxy server node)
`--swift-services`	Check that the relevant Swift processes are running.
`--network-interface`	Checks NIC speed and reports statistics for each interface.
`--check-mounts`	Checks that the node has correctly mounted drives used by Swift.
`--hpssacli`	If this server uses a Smart Array Controller, this checks the operation of the controller and disk drives.

8.4   Using the Swift Command-line Client (CLI)

The `swift` utility (or Swift CLI) is installed on the Cloud Lifecycle Manager node and also on all other nodes running the Swift proxy service. To use this utility on the Cloud Lifecycle Manager, you can use the `~/service.osrc` file as a basis and then edit it with the credentials of another user if you need to.

```
ardana > cp ~/service.osrc ~/swiftuser.osrc
```

Then you can use your preferred editor to edit swiftuser.osrc so you can authenticate using the `OS_USERNAME`, `OS_PASSWORD`, and `OS_PROJECT_NAME` you wish to use. For example, if you would like to use the `demo` user that is created automatically for you, then it might look like this:

```
unset OS_DOMAIN_NAME
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_VERSION=3
```

```
export OS_PROJECT_NAME=demo
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USERNAME=demo
export OS_USER_DOMAIN_NAME=Default
export OS_PASSWORD=<password>
export OS_AUTH_URL=<auth_URL>
export OS_ENDPOINT_TYPE=internalURL
# OpenstackClient uses OS_INTERFACE instead of OS_ENDPOINT
export OS_INTERFACE=internal
export OS_CACERT=/etc/ssl/certs/ca-certificates.crt
export OS_COMPUTE_API_VERSION=2
```

You must use the appropriate password for the demo user and select the correct endpoint for the **OS_AUTH_URL** value, which should be in the `~/service.osrc` file you copied.

You can then examine the following account data using this command:

```
ardana > swift stat
```

Example showing an environment with no containers or objects:

```
ardana > swift stat
        Account: AUTH_205804d000a242d385b8124188284998
     Containers: 0
        Objects: 0
          Bytes: 0
X-Put-Timestamp: 1442249536.31989
     Connection: keep-alive
    X-Timestamp: 1442249536.31989
     X-Trans-Id: tx5493faa15be44efeac2e6-0055f6fb3f
   Content-Type: text/plain; charset=utf-8
```

Use the following command and create a container:

```
ardana > swift post CONTAINER_NAME
```

Example, creating a container named `documents`:

```
ardana > swift post documents
```

The newly created container appears. But there are no objects:

```
ardana > swift stat documents
          Account: AUTH_205804d000a242d385b8124188284998
        Container: documents
          Objects: 0
            Bytes: 0
         Read ACL:
```

```
     Write ACL:
       Sync To:
      Sync Key:
  Accept-Ranges: bytes
X-Storage-Policy: General
     Connection: keep-alive
    X-Timestamp: 1442249637.69486
     X-Trans-Id: tx1f59d5f7750f4ae8a3929-0055f6fbcc
   Content-Type: text/plain; charset=utf-8
```

Upload a document:

```
ardana > swift upload CONTAINER_NAME FILENAME
```

Example:

```
ardana > swift upload documents mydocument
mydocument
```

List objects in the container:

```
ardana > swift list CONTAINER_NAME
```

Example:

```
ardana > swift list documents
mydocument
```

🎓 Note

This is a brief introduction to the **swift** CLI. Use the **swift --help** command for more information. You can also use the OpenStack CLI, see **openstack -h** for more information.

8.5 Managing Swift Rings

Swift rings are a machine-readable description of which disk drives are used by the Object Storage service (for example, a drive is used to store account or object data). Rings also specify the policy for data storage (for example, defining the number of replicas). The rings are automatically built during the initial deployment of your cloud, with the configuration provided during setup of the HPE Helion OpenStack Input Model. For more information, see *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 5 "Input Model"*.

After successful deployment of your cloud, you may want to change or modify the configuration for Swift. For example, you may want to add or remove Swift nodes, add additional storage policies, or upgrade the size of the disk drives. For instructions, see *Section 8.5.5, "Applying Input Model Changes to Existing Rings"* and *Section 8.5.6, "Adding a New Swift Storage Policy"*.

> 🎓 Note
>
> The process of modifying or adding a configuration is similar to other configuration or topology changes in the cloud. Generally, you make the changes to the input model files at `~/openstack/my_cloud/definition/` on the Cloud Lifecycle Manager and then run Ansible playbooks to reconfigure the system.

Changes to the rings require several phases to complete, therefore, you may need to run the playbooks several times over several days.

The following topics cover ring management.

8.5.1   Rebalancing Swift Rings

The Swift ring building process tries to distribute data evenly among the available disk drives. The data is stored in partitions. (For more information, see *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 11 "Modifying Example Configurations for Object Storage using Swift", Section 11.10 "Understanding Swift Ring Specifications".*) If you, for example, double the number of disk drives in a ring, you need to move 50% of the partitions to the new drives so that all drives contain the same number of partitions (and hence same amount of data). However, it is not possible to move the partitions in a single step. It can take minutes to hours to move partitions from the original drives to their new drives (this process is called the replication process).

If you move all partitions at once, there would be a period where Swift would expect to find partitions on the new drives, but the data has not yet replicated there so that Swift could not return the data to the user. Therefore, Swift will not be able to find all of the data in the middle of replication because some data has finished replication while other bits of data are still in the old locations and have not yet been moved. So it is considered best practice to move only one replica at a time. If the replica count is 3, you could first move 16.6% of the partitions and then wait until all data has replicated. Then move another 16.6% of partitions. Wait again and then finally move the remaining 16.6% of partitions. For any given object, only one of the replicas is moved at a time.

8.5.1.1 Reasons to Move Partitions Gradually

Due to the following factors, you must move the partitions gradually:

- Not all devices are of the same size. HPE Helion OpenStack 8 automatically assigns different weights to drives so that smaller drives store fewer partitions than larger drives.

- The process attempts to keep replicas of the same partition in different servers.

- Making a large change in one step (for example, doubling the number of drives in the ring), would result in a lot of network traffic due to the replication process and the system performance suffers. There are two ways to mitigate this:

    - Add servers in smaller groups

    - Set the weight-step attribute in the ring specification. For more information, see *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"*.

8.5.2 Using the Weight-Step Attributes to Prepare for Ring Changes

Swift rings are built during a deployment and this process sets the weights of disk drives such that smaller disk drives have a smaller weight than larger disk drives. When making changes in the ring, you should limit the amount of change that occurs. HPE Helion OpenStack 8 does this by limiting the weights of the new drives to a smaller value and then building new rings. Once the replication process has finished, HPE Helion OpenStack 8 will increase the weight and rebuild rings to trigger another round of replication. (For more information, see *Section 8.5.1, "Rebalancing Swift Rings"*.)

In addition, you should become familiar with how the replication process behaves on your system during normal operation. Before making ring changes, use the `swift-recon` command to determine the typical oldest replication times for your system. For instructions, see *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"*.

In HPE Helion OpenStack, the weight-step attribute is set in the ring specification of the input model. The weight-step value specifies a maximum value for the change of the weight of a drive in any single rebalance. For example, if you add a drive of 4TB, you would normally assign a weight of 4096. However, if the weight-step attribute is set to 1024 instead then when you add that drive the weight is initially set to 1024. The next time you rebalance the ring, the weight is set to 2048. The subsequent rebalance would then set the weight to the final value of 4096.

The value of the weight-step attribute is dependent on the size of the drives, number of the servers being added, and how experienced you are with the replication process. A common starting value is to use 20% of the size of an individual drive. For example, when adding X number of 4TB drives a value of 820 would be appropriate. As you gain more experience with your system, you may increase or reduce this value.

8.5.2.1 Setting the weight-step attribute

Perform the following steps to set the weight-step attribute:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/definition/data/swift/rings.yml` file containing the ring-specifications for the account, container, and object rings.
   Add the weight-step attribute to the ring in this format:

   ```
   - name: account
     weight-step: WEIGHT_STEP_VALUE
     display-name: Account Ring
     min-part-hours: 16
     ...
   ```

   For example, to set weight-step to 820, add the attribute like this:

   ```
   - name: account
     weight-step: 820
     display-name: Account Ring
     min-part-hours: 16
     ...
   ```

3. Repeat step 2 for the other rings, if necessary (container, object-0, etc).

4. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

5. Use the playbook to create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ```

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. To complete the configuration, use the ansible playbooks documented in *Section 8.5.3, "Managing Rings Using Swift Playbooks"*.

8.5.3   Managing Rings Using Swift Playbooks

The following table describes how playbooks relate to ring management.

All of these playbooks will be run from the Cloud Lifecycle Manager from the `~/scratch/ansible/next/ardana/ansible` directory.

Playbook	Description	Notes
`swift-update-from-model-rebalance-rings.yml`	There are two steps in this playbook: • Make delta It processes the input model and compares it against the existing rings. After comparison, it produces a list of differences between the input model and the existing rings. This is called the ring delta. The ring delta covers drives being added, drives being removed, weight changes, and replica count changes. • Rebalance The ring delta is then converted into a series of commands (such as `add`) to the swift-	This playbook performs its actions on the first node running the swift-proxy service. (For more information, see *Section 15.6.2.4, "Identifying the Swift Ring Building Server"*.) However, it also scans all Swift nodes to find the size of disk drives. If there are no changes in the ring delta, the `rebalance` command is still executed to rebalance the rings. If `min-part-hours` has not yet elapsed or if no partitions need to be moved, new rings are not written.

Playbook	Description	Notes
	ring-builder program. Finally, the `rebalance` command is issued to the swift-ring-builder program.	
`swift-compare-model-rings.yml`	There are two steps in this playbook: • Make delta This is the same as described for `swift-update-from-model-rebalance-rings.yml`. • Report This prints a summary of the proposed changes that will be made to the rings (that is, what would happen if you rebalanced). The playbook reports any issues or problems it finds with the input model. This playbook can be useful to confirm that there are no errors in the input model. It also allows you to check that when you change the input model, that the proposed ring changes are as expected. For example, if you have added a server to the input model, but this playbook	There is troubleshooting information related to the information that you receive in this report that you can view on this page: *Section 15.6.2.3, "Interpreting Swift Input Model Validation Errors".*

Playbook	Description	Notes
	reports that no drives are being added, you should determine the cause.	
`swift-deploy.yml`	`swift-deploy.yml` is responsible for installing software and configuring Swift on nodes. As part of installing and configuring, it runs the `swift-update-from-model-rebalance-rings.yml` and `swift-reconfigure.yml` playbooks.	This playbook is included in the `ardana-deploy.yml` and `site.yml` playbooks, so if you run either of those playbooks, the `swift-deploy.yml` playbook is also run.
`swift-reconfigure.yml`	`swift-reconfigure.yml` takes rings that the `swift-update-from-model-rebalance-rings.yml` playbook has changed and copies those rings to all Swift nodes.	Every time that you directly use the `swift-update-from-model-rebalance-rings.yml` playbook, you must copy these rings to the system using the `swift-reconfigure.yml` playbook. If you forget and run `swift-update-from-model-rebalance-rings.yml` twice, the process may move two replicates of some partitions at the same time.

8.5.3.1 Optional Ansible variables related to ring management

The following optional variables may be specified when running the playbooks outlined above. They are specified using the `--extra-vars` option.

Variable	Description and Use
`limit_ring`	Limit changes to the named ring. Other rings will not be examined or updated. This option may be used with any of the Swift playbooks. For example, to only update the `object-1` ring, use the following command: ``` ardana > ansible-playbook -i hosts/ verb_hosts swift-update-from-model- rebalance-rings.yml --extra-vars "limit- ring=object-1" ```
drive_detail	Used only with the swift-compare-model-rings.yml playbook. The playbook will include details of changes to every drive where the model and existing rings differ. If you omit the drive_detail variable, only summary information is provided. The following shows how to use the drive_detail variable: ``` ardana > ansible-playbook -i hosts/ verb_hosts swift-compare-model-rings.yml --extra-vars "drive_detail=yes" ```

8.5.3.2 Interpreting the report from the swift-compare-model-rings.yml playbook

The `swift-compare-model-rings.yml` playbook compares the existing Swift rings with the input model and prints a report telling you how the rings and the model differ. Specifically, it will tell you what actions will take place when you next run the `swift-update-from-model-rebalance-rings.yml` playbook (or a playbook such as `ardana-deploy.yml` that runs `swift-update-from-model-rebalance-rings.yml`).

The `swift-compare-model-rings.yml` playbook will make no changes, but is just an advisory report.

Here is an example output from the playbook. The report is between "report.stdout_lines" and "PLAY RECAP":

```
TASK: [swiftlm-ring-supervisor | validate-input-model | Print report] *********
ok: [ardana-cp1-c1-m1-mgmt] => {
    "var": {
        "report.stdout_lines": [
            "Rings:",
            "  ACCOUNT:",
            "    ring exists (minimum time to next rebalance: 8:07:33)",
            "    will remove 1 devices (18.00GB)",
            "    ring will be rebalanced",
            "  CONTAINER:",
            "    ring exists (minimum time to next rebalance: 8:07:35)",
            "    no device changes",
            "    ring will be rebalanced",
            "  OBJECT-0:",
            "    ring exists (minimum time to next rebalance: 8:07:34)",
            "    no device changes",
            "    ring will be rebalanced"
        ]
    }
}
```

The following describes the report in more detail:

Message	Description
ring exists	The ring already exists on the system.
ring will be created	The ring does not yet exist on the system.
no device changes	The devices in the ring exactly match the input model. There are no servers being added or removed and the weights are appropriate for the size of the drives.
minimum time to next rebalance	If this time is `0:00:00`, if you run one of the Swift playbooks that update rings, the ring will be rebalanced.
	If the time is non-zero, it means that not enough time has elapsed since the ring was last rebalanced. Even if you run a Swift

Message	Description
	playbook that attempts to change the ring, the ring will not actually rebalance. This time is determined by the `min-part-hours` attribute.
set-weight ardana-ccp-c1-m1-mgmt:disk0:/dev/sdc 8.00 > 12.00 > 18.63	The weight of disk0 (mounted on /dev/sdc) on server `ardana-ccp-c1-m1-mgmt` is currently set to 8.0 but should be 18.83 given the size of the drive. However, in this example, we cannot go directly from 8.0 to 18.63 because of the weight-step attribute. Hence, the proposed weight change is from 8.0 to 12.0. This information is only shown when you the `drive_detail=yes` argument when running the playbook.
will change weight on 12 devices (6.00TB)	The weight of 12 devices will be increased. This might happen for example, if a server had been added in a prior ring update. However, with use of the `weight-step` attribute, the system gradually increases the weight of these new devices. In this example, the change in weight represents 6TB of total available storage. For example, if your system currently has 100TB of available storage, when the weight of these devices is changed, there will be 106TB of available storage. If your system is 50% utilized, this means that when the ring is rebalanced, up to 3TB of data may be moved by the replication process. This is an estimate - in practice, because only one copy of a given

Message	Description
	replica is moved in any given rebalance, it may not be possible to move this amount of data in a single ring rebalance.
add: ardana-ccp-c1-m1-mgmt:disk0:/dev/sdc	The disk0 device will be added to the ardana-ccp-c1-m1-mgmt server. This happens when a server is added to the input model or if a disk model is changed to add additional devices. This information is only shown when you the `drive_detail=yes` argument when running the playbook.
remove: ardana-ccp-c1-m1-mgmt:disk0:/dev/sdc	The device is no longer in the input model and will be removed from the ring. This happens if a server is removed from the model, a disk drive is removed from a disk model or the server is marked for removal using the pass-through feature. This information is only shown when you the `drive_detail=yes` argument when running the playbook.
will add 12 devices (6TB)	There are 12 devices in the input model that have not yet been added to the ring. Usually this is because one or more servers have been added. In this example, this could be one server with 12 drives or two servers, each with 6 drives. The size in the report is the change in total available capacity. When the weight-step attribute is used, this may be a fraction of the total size of the disk drives. In this example, 6TB of capacity is being added. For example, if your system currently has 100TB of available storage,

Message	Description
	when these devices are added, there will be 106TB of available storage. If your system is 50% utilized, this means that when the ring is rebalanced, up to 3TB of data may be moved by the replication process. This is an estimate - in practice, because only one copy of a given replica is moved in any given rebalance, it may not be possible to move this amount of data in a single ring rebalance.
will remove 12 devices (6TB)	There are 12 devices in rings that no longer appear in the input model. Usually this is because one or more servers have been removed. In this example, this could be one server with 12 drives or two servers, each with 6 drives. The size in the report is the change in total removed capacity. In this example, 6TB of capacity is being removed. For example, if your system currently has 100TB of available storage, when these devices are removed, there will be 94TB of available storage. If your system is 50% utilized, this means that when the ring is rebalanced, approximately 3TB of data must be moved by the replication process.
min-part-hours will be changed	The `min-part-hours` attribute has been changed in the ring specification in the input model.
replica-count will be changed	The `replica-count` attribute has been changed in the ring specification in the input model.

Message	Description
ring will be rebalanced	This is always reported. Every time the `swift-update-from-model-rebalance-rings.yml` playbook is run, it will execute the swift-ring-builder rebalance command. This happens even if there were no input model changes. If the ring is already well balanced, the swift-ring-builder will not rewrite the ring.

8.5.4   Determining When to Rebalance and Deploy a New Ring

Before deploying a new ring, you must be sure the change that has been applied to the last ring is complete (that is, all the partitions are in their correct location). There are three aspects to this:

- Is the replication system busy?

  You might want to postpone a ring change until after replication has finished. If the replication system is busy repairing a failed drive, a ring change will place additional load on the system. To check that replication has finished, use the `swift-recon` command with the **--replication** argument. (For more information, see *Section 8.2, "Gathering Swift Data"*.) The oldest completion time can indicate that the replication process is very busy. If it is more than 15 or 20 minutes then the object replication process are probably still very busy. The following example indicates that the oldest completion is 120 seconds, so that the replication process is probably not busy:

```
root # swift-recon --replication
===========================================================================
--> Starting reconnaissance on 3 hosts
===========================================================================
[2015-10-02 15:31:45] Checking on replication
[replication_time] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0,
 reported: 3
Oldest completion was 2015-10-02 15:31:32 (120 seconds ago) by 192.168.245.4:6000.
Most recent completion was 2015-10-02 15:31:43 (10 seconds ago) by
 192.168.245.3:6000.
===========================================================================
```

- Are there drive or server failures?

A drive failure does not preclude deploying a new ring. In principle, there should be two copies elsewhere. However, another drive failure in the middle of replication might make data temporary unavailable. If possible, postpone ring changes until all servers and drives are operating normally.

- Has `min-part-hours` elapsed?

  The `swift-ring-builder` will refuse to build a new ring until the `min-part-hours` has elapsed since the last time it built rings. You must postpone changes until this time has elapsed.

  You can determine how long you must wait by running the `swift-compare-model-rings.yml` playbook, which will tell you how long you until the `min-part-hours` has elapsed. For more details, see *Section 8.5.3, "Managing Rings Using Swift Playbooks"*.

  You can change the value of `min-part-hours`. (For instructions, see *Section 8.5.7, "Changing min-part-hours in Swift"*).

- Is the Swift dispersion report clean?

  Run the `swift-dispersion-report.yml` playbook (as described in *Section 8.1, "Running the Swift Dispersion Report"*) and examine the results. If the replication process has not yet replicated partitions that were moved to new drives in the last ring rebalance, the dispersion report will indicate that some containers or objects are missing a copy.

  For example:

  ```
  There were 462 partitions missing one copy.
  ```

  Assuming all servers and disk drives are operational, the reason for the missing partitions is that the replication process has not yet managed to copy a replica into the partitions. You should wait an hour and rerun the dispersion report process and examine the report. The number of partitions missing one copy should have reduced. Continue to wait until this reaches zero before making any further ring rebalances.

> ☞ Note
>
> It is normal to see partitions missing one copy if disk drives or servers are down. If all servers and disk drives are mounted, and you did not recently perform a ring rebalance, you should investigate whether there are problems with the replication process. You can use the Operations Console to investigate replication issues.

⌂ Important

If there are any partitions missing two copies, you must reboot or repair any failed servers and disk drives as soon as possible. Do not shutdown any Swift nodes in this situation. Assuming a replica count of 3, if you are missing two copies you are in danger of losing the only remaining copy.

8.5.5 Applying Input Model Changes to Existing Rings

This page describes a general approach for making changes to your existing Swift rings. This approach applies to actions such as adding and removing a server and replacing and upgrading disk drives, and must be performed as a series of phases, as shown below:

8.5.5.1 Changing the Input Model Configuration Files

The first step to apply new changes to the Swift environment is to update the configuration files. Follow these steps:

1. Log in to the Cloud Lifecycle Manager.

2. Set the weight-step attribute, as needed, for the nodes you are altering. (For instructions, see *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"*).

3. Edit the configuration files as part of the Input Model as appropriate. (For general information about the Input Model, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 6 "Configuration Objects", Section 6.14 "Networks"*. For more specific information about the Swift parts of the configuration files, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 11 "Modifying Example Configurations for Object Storage using Swift"*)

4. Once you have completed all of the changes, commit your configuration to the local git repository. (For more information, see *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management".*) :

```
ardana > git add -A
root # git commit -m "commit message"
```

5. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Swift playbook that will validate your configuration files and give you a report as an output:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
root # ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml
```

8. Use the report to validate that the number of drives proposed to be added or deleted, or the weight change, is correct. Fix any errors in your input model. At this stage, no changes have been made to rings.

8.5.5.2 First phase of Ring Rebalance

To begin the rebalancing of the Swift rings, follow these steps:

1. After going through the steps in the section above, deploy your changes to all of the Swift nodes in your environment by running this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

2. Wait until replication has finished or `min-part-hours` has elapsed (whichever is longer). For more information, see *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"*

8.5.5.3 Weight Change Phase of Ring Rebalance

At this stage, no changes have been made to the input model. However, when you set the `weight-step` attribute, the rings that were rebuilt in the previous rebalance phase have weights that are different than their target/final value. You gradually move to the target/final weight by rebalancing a number of times as described on this page. For more information about the weight-step attribute, see *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"*.

To begin the re-balancing of the rings, follow these steps:

1. Rebalance the rings by running the playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-update-from-model-rebalance-
rings.yml
```

2. Run the reconfiguration:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
```

3. Wait until replication has finished or `min-part-hours` has elapsed (whichever is longer). For more information, see *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"*

4. Run the following command and review the report:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml --limit
 SWF*
```

The following is an example of the output after executing the above command. In the example **no** weight changes are proposed:

```
TASK: [swiftlm-ring-supervisor | validate-input-model | Print report] *********
ok: [padawan-ccp-c1-m1-mgmt] => {
    "var": {
        "report.stdout_lines": [
            "Need to add 0 devices",
            "Need to remove 0 devices",
            "Need to set weight on 0 devices"
        ]
    }
}
```

5. When there are no proposed weight changes, you proceed to the final phase.

6. If there are proposed weight changes repeat this phase again.

8.5.5.4 Final Rebalance Phase

The final rebalance phase moves all replicas to their final destination.

1. Rebalance the rings by running the playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-update-from-model-rebalance-
rings.yml | tee /tmp/rebalance.log
```

🎓 Note

The output is saved for later reference.

2. Review the output from the previous step. If the output for all rings is similar to the following, the rebalance had no effect. That is, the rings are balanced and no further changes are needed. In addition, the ring files were not changed so you do not need to deploy them to the Swift nodes:

```
"Running: swift-ring-builder /etc/swiftlm/cloud1/cp1/builder_dir/account.builder
 rebalance 999",
      "NOTE: No partitions could be reassigned.",
      "Either none need to be or none can be due to min_part_hours [16]."
```

The text **No partitions could be reassigned** indicates that no further rebalances are necessary. If this is true for all the rings, you have completed the final phase.

🎓 Note

You must have allowed enough time to elapse since the last rebalance. As mentioned in the above example, `min_part_hours [16]` means that you must wait at least 16 hours since the last rebalance. If not, you should wait until enough time has elapsed and repeat this phase.

3. Run the `swift-reconfigure.yml` playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
```

4. Wait until replication has finished or `min-part-hours` has elapsed (whichever is longer). For more information see *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"*

5. Repeat the above steps until the ring is rebalanced.

8.5.5.5 System Changes that Change Existing Rings

There are many system changes ranging from adding servers to replacing drives, which might require you to rebuild and rebalance your rings.

Actions	Process
Adding Servers(s)	- If not already done, set the weight change attribute. For instructions, see *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"* - Add servers in phases: - This reduces the impact of the changes on your system. - If your rings use Swift zones, ensure that you remove the same number of servers to each zone at each phase. - To add a server, see *Section 13.1.5.1.1, "Adding a Swift Object Node"*. - Incrementally change the weights and perform the final rebalance. For instructions, see *Section 8.5.5.4, "Final Rebalance Phase"*.
Removing Server(s)	In HPE Helion OpenStack, when you remove servers from the input model, the disk drives are removed from the ring - the weight is not gradually reduced using the `weight-step` attribute.

Actions	Process
	- Remove servers in phases: - This reduces the impact of the changes on your system. - If your rings use Swift zones, ensure you remove the same number of servers for each zone at each phase. - To remove a server, see *Section 13.1.5.1.4, "Removing a Swift Node"*. - To perform the final rebalance, see *Section 8.5.5.4, "Final Rebalance Phase"*.
Replacing Disk Drive(s)	When a drive fails, replace it as soon as possible. Do not attempt to remove it from the ring - this creates operator overhead. Swift will continue to store the correct number of replicas by handing off objects to other drives instead of the failed drive. If the disk drives are of the same size as the original when the drive is replaced, no ring changes are required. You can confirm this by running the `swift-update-from-model-rebalance-rings.yml` playbook. It should report that no weight changes are needed. For a single drive replacement, even if the drive is significantly larger than the original drives, you do not need to rebalance the ring (however, the extra space on the drive will not be used).

Actions	Process

Upgrading Disk Drives	If the drives are different size (for example, you are upgrading your system), you can proceed as follows:
	
	• If not already done, set the weight-step attribute
	
	• Replace drives in phases:
	
	    • Avoid replacing too many drives at once.
	
	    • If your rings use swift zones, upgrade a number of drives in the same zone at the same time - not drives in several zones.
	
	    • It is also safer to upgrade one server instead of drives in several servers at the same time.
	
	    • Remember that the final size of all Swift zones must be the same, so you may need to replace a small number of drives in one zone, then a small number in second zone, then return to the first zone and replace more drives, etc.

8.5.6 Adding a New Swift Storage Policy

This page describes how to add an additional storage policy to an existing system. For an overview of storage policies, see *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 11 "Modifying Example Configurations for Object Storage using Swift", Section 11.11 "Designing Storage Policies"*.

To Add a Storage Policy

Perform the following steps to add the storage policy to an existing system.

1. Log in to the Cloud Lifecycle Manager.

2. Select a storage policy index and ring name.
   For example, if you already have object-0 and object-1 rings in your ring-specifications (usually in the `~/openstack/my_cloud/definition/data/swift/rings.yml` file), the next index is 2 and the ring name is object-2.

3. Select a user-visible name so that you can see when you examine container metadata or when you want to specify the storage policy used when you create a container. The name should be a single word (hyphen and dashes are allowed).

4. Decide if this new policy will be the default for all new containers.

5. Decide on other attributes such as `partition-power` and `replica-count` if you are using a standard replication ring. However, if you are using an erasure coded ring, you also need to decide on other attributes: `ec-type`, `ec-num-data-fragments`, `ec-num-parity-fragments`, and `ec-object-segment-size`. For more details on the required attributes, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 11 "Modifying Example Configurations for Object Storage using Swift", Section 11.10 "Understanding Swift Ring Specifications"*.

6. Edit the `ring-specifications` attribute (usually in the `~/openstack/my_cloud/definition/data/swift/rings.yml` file) and add the new ring specification. If this policy is to be the default storage policy for new containers, set the `default` attribute to **yes**.

🎓 Note

   a. Ensure that only one object ring has the `default` attribute set to `yes`. If you set two rings as default, Swift processes will not start.

   b. Do not specify the `weight-step` attribute for the new object ring. Since this is a new ring there is no need to gradually increase device weights.

7. Update the appropriate disk model to use the new storage policy (for example, the `data/disks_swobj.yml` file). The following sample shows that the **object-2** has been added to the list of existing rings that use the drives:

```
disk-models:
- name: SWOBJ-DISKS
  ...
  device-groups:
  - name: swobj
    devices:
       ...
    consumer:
        name: swift
        attrs:
            rings:
            - object-0
            - object-1
            - object-2
  ...
```

> # ☞ Note
>
> You must use the new object ring on at least one node that runs the `swift-object` service. If you skip this step and continue to run the `swift-compare-model-rings.yml` or `swift-deploy.yml` playbooks, they will fail with an error *There are no devices in this ring, or all devices have been deleted,* as shown below:
>
> ```
> TASK: [swiftlm-ring-supervisor | build-rings | Build ring (make-delta,
>  rebalance)] ***
> failed: [padawan-ccp-c1-m1-mgmt] => {"changed": true, "cmd": ["swiftlm-ring-
> supervisor", "--make-delta", "--rebalance"], "delta": "0:00:03.511929", "end":
>  "2015-10-07 14:02:03.610226", "rc": 2, "start": "2015-10-07 14:02:00.098297",
>  "warnings": []}
> ...
> Running: swift-ring-builder /etc/swiftlm/cloud1/cp1/builder_dir/
> object-2.builder rebalance 999
> ERROR:
>  -----------------------------------------------------------------------------
> An error has occurred during ring validation. Common
> causes of failure are rings that are empty or do not
> have enough devices to accommodate the replica count.
> Original exception message:
> There are no devices in this ring, or all devices have been deleted
>  -----------------------------------------------------------------------------
> ```

8. Commit your configuration:

   ```
   ardana > git add -A
   ardana > git commit -m "commit message"
   ```

9. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

10. Create a deployment directory:

    ```
    ardana > cd ~/openstack/ardana/ansible
    ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
    ```

11. Validate the changes by running the `swift-compare-model-rings.yml` playbook:

    ```
    ardana > cd ~/scratch/ansible/next/ardana/ansible
    ```

```
ardana > ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml
```

If any errors occur, correct them. For instructions, see *Section 15.6.2.3, "Interpreting Swift Input Model Validation Errors"*. Then, re-run steps **5 - 10**.

12. Create the new ring (for example, object-2). Then verify the Swift service status and reconfigure the Swift node to use a new storage policy, by running these playbooks:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-status.yml
ardana > ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

After adding a storage policy, there is no need to rebalance the ring.

8.5.7 Changing min-part-hours in Swift

The `min-part-hours` parameter specifies the number of hours you must wait before Swift will allow a given partition to be moved. In other words, it constrains how often you perform ring rebalance operations. Before changing this value, you should get some experience with how long it takes your system to perform replication after you make ring changes (for example, when you add servers).

See *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"* for more information about determining when replication has completed.

8.5.7.1 Changing the min-part-hours Value

To change the `min-part-hours` value, following these steps:

1. Log in to the Cloud Lifecycle Manager.

2. Edit your `~/openstack/my_cloud/definition/data/swift/rings.yml` file and change the value(s) of `min-part-hours` for the rings you desire. The value is expressed in hours and a value of zero is not allowed.

3. Commit your configuration to the local Git repository (*Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "My config or other commit message"
```

4. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Apply the changes by running this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

8.5.8   Changing Swift Zone Layout

Before changing the number of Swift zones or the assignment of servers to specific zones, you must ensure that your system has sufficient storage available to perform the operation. Specifically, if you are adding a new zone, you may need additional storage. There are two reasons for this:

- You cannot simply change the Swift zone number of disk drives in the ring. Instead, you need to remove the server(s) from the ring and then re-add the server(s) with a new Swift zone number to the ring. At the point where the servers are removed from the ring, there must be sufficient spare capacity on the remaining servers to hold the data that was originally hosted on the removed servers.

- The total amount of storage in each Swift zone must be the same. This is because new data is added to each zone at the same rate. If one zone has a lower capacity than the other zones, once that zone becomes full, you cannot add more data to the system – even if there is unused space in the other zones.

As mentioned above, you cannot simply change the Swift zone number of disk drives in an existing ring. Instead, you must remove and then re-add servers. This is a summary of the process:

1. Identify appropriate server groups that correspond to the desired Swift zone layout.

2. Remove the servers in a server group from the rings. This process may be protracted, either by removing servers in small batches or by using the weight-step attribute so that you limit the amount of replication traffic that happens at once.

3. Once all the targeted servers are removed, edit the `swift-zones` attribute in the ring specifications to add or remove a Swift zone.

4. Re-add the servers you had temporarily removed to the rings. Again you may need to do this in batches or rely on the weight-step attribute.

5. Continue removing and re-adding servers until you reach your final configuration.

8.5.8.1 Process for Changing Swift Zones

This section describes the detailed process or reorganizing Swift zones. As a concrete example, we assume we start with a single Swift zone and the target is three Swift zones. The same general process would apply if you were reducing the number of zones as well.

The process is as follows:

1. Identify the appropriate server groups that represent the desired final state. In this example, we are going to change the Swift zone layout as follows:

Original Layout	Target Layout
``swift-zones:
- 1d: 1
server-groups:
- AZ1
- AZ2
- AZ3
``	```swift-zones:
- 1d: 1
server-groups:
- AZ1
- id: 2
- AZ2
- id: 3
- AZ3
``

   The plan is to move servers from server groups `AZ2` and `AZ3` to a new Swift zone number. The servers in `AZ1` will remain in Swift zone 1.

2. If you have not already done so, consider setting the weight-step attribute as described in *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"*.

3. Identify the servers in the `AZ2` server group. You may remove all servers at once or remove them in batches. If this is the first time you have performed a major ring change, we suggest you remove one or two servers only in the first batch. When you see how long this takes and the impact replication has on your system you can then use that experience to decide whether you can remove a larger batch of servers, or increase or decrease the weight-step

attribute for the next server-removal cycle. To remove a server, use steps 2-9 as described in *Section 13.1.5.1.4, "Removing a Swift Node"* ensuring that you do not remove the servers from the input model.

4. This process may take a number of ring rebalance cycles until the disk drives are removed from the ring files. Once this happens, you can edit the ring specifications and add Swift zone 2 as shown in this example:

```
swift-zones:
  - id: 1
    server-groups:
      - AZ1
      - AZ3
  - id: 2
        - AZ2
```

5. The server removal process in step #3 set the "remove" attribute in the `pass-through` attribute of the servers in server group `AZ2`. Edit the input model files and remove this `pass-through` attribute. This signals to the system that the servers should be used the next time we rebalance the rings (that is, the server should be added to the rings).

6. Commit your configuration to the local Git repository (*Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "My config or other commit message"
```

7. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

8. Use the playbook to create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

9. Rebuild and deploy the Swift rings containing the re-added servers by running this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

10. Wait until replication has finished. For more details, see *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"*.

11. You may need to continue to rebalance the rings. For instructions, see the "Final Rebalance Stage" steps at *Section 8.5.5, "Applying Input Model Changes to Existing Rings"*.

12. At this stage, the servers in server group `AZ2` are responsible for Swift zone 2. Repeat the process in steps #3-9 to remove the servers in server group `AZ3` from the rings and then re-add them to Swift zone 3. The ring specifications for zones (step 4) should be as follows:

```
swift-zones:
  - ld: 1
    server-groups:
      - AZ1
  - id: 2
      - AZ2
  - id: 3
      - AZ3
```

13. Once complete, all data should be dispersed (that is, each replica is located) in the Swift zones as specified in the input model.

8.6 Configuring your Swift System to Allow Container Sync

Swift has a feature where all the contents of a container can be mirrored to another container through background synchronization. Swift operators configure their system to allow/accept sync requests to/from other systems, and the user specifies where to sync their container to along with a secret synchronization key. For an overview of this feature, refer to OpenStack Swift - Container to Container Synchronization (http://docs.openstack.org/developer/swift/overview_container_sync.html) ↗.

8.6.1 Notes and limitations

The container synchronization is done as a background action. When you put an object into the source container, it will take some time before it becomes visible in the destination container. Storage services will not necessarily copy objects in any particular order, meaning they may be transferred in a different order to which they were created.

Container sync may not be able to keep up with a moderate upload rate to a container. For example, if the average object upload rate to a container is greater than one object per second, then container sync may not be able to keep the objects synced.

If container sync is enabled on a container that already has a large number of objects then container sync may take a long time to sync the data. For example, a container with one million 1KB objects could take more than 11 days to complete a sync.

You may operate on the destination container just like any other container -- adding or deleting objects -- including the objects that are in the destination container because they were copied from the source container. To decide how to handle object creation, replacement or deletion, the system uses timestamps to determine what to do. In general, the latest timestamp "wins". That is, if you create an object, replace it, delete it and the re-create it, the destination container will eventually contain the most recently created object. However, if you also create and delete objects in the destination container, you get some subtle behaviours as follows:

- If an object is copied to the destination container and then deleted, it remains deleted in the destination even though there is still a copy in the source container. If you modify the object (replace or change its metadata) in the source container, it will reappear in the destination again.

- The same applies to a replacement or metadata modification of an object in the destination container -- the object will remain as-is unless there is a replacement or modification in the source container.

- If you replace or modify metadata of an object in the destination container and then delete it in the source container, it is **not** deleted from the destination. This is because your modified object has a later timestamp than the object you deleted in the source.

- If you create an object in the source container and before the system has a chance to copy it to the destination, you also create an object of the same name in the destination, then the object in the destination is **not** overwritten by the source container's object.

Segmented objects

Segmented objects (objects larger than 5GB) will not work seamlessly with container synchronization. If the manifest object is copied to the destination container before the object segments, when you perform a GET operation on the manifest object, the system may fail to find some or all of the object segments. If your manifest and object segments are in different containers, do not forget that both containers must be synchonized and that the container name of the object segments must be the same on both source and destination.

8.6.2 Prerequisites

Container to container synchronization requires that SSL certificates are configured on both the source and destination systems. For more information on how to implement SSL, see *Book "Installing with Cloud Lifecycle Manager", Chapter 30 "Configuring Transport Layer Security (TLS)"*.

8.6.3 Configuring container sync

Container to container synchronization requires that both the source and destination Swift systems involved be configured to allow/accept this. In the context of container to container synchronization, Swift uses the term *cluster* to denote a Swift system. Swift *clusters* correspond to *Control Planes* in OpenStack terminology.

Gather the public API endpoints for both Swift systems

Gather information about the external/public URL used by each system, as follows:

1. On the Cloud Lifecycle Manager of one system, get the public API endpoint of the system by running the following commands:

```
ardana > source ~/service.osrc
ardana > openstack endpoint list | grep swift
```

The output of the command will look similar to this:

```
ardana > openstack endpoint list | grep swift
| 063a84b205c44887bc606c3ba84fa608 | region0 | swift        | object-store   |
 True    | admin    | https://10.13.111.176:8080/v1/AUTH_%(tenant_id)s |
| 3c46a9b2a5f94163bb5703a1a0d4d37b | region0 | swift        | object-store   |
 True    | public   | https://10.13.120.105:8080/v1/AUTH_%(tenant_id)s |
| a7b2f4ab5ad14330a7748c950962b188 | region0 | swift        | object-store   |
 True    | internal | https://10.13.111.176:8080/v1/AUTH_%(tenant_id)s |
```

The portion that you want is the endpoint up to, but not including, the `AUTH` part. It is bolded in the above example, `https://10.13.120.105:8080/v1`.

2. Repeat these steps on the other Swift system so you have both of the public API endpoints for them.

Validate connectivity between both systems

The Swift nodes running the `swift-container` service must be able to connect to the public API endpoints of each other for the container sync to work. You can validate connectivity on each system using these steps.

For the sake of the examples, we will use the terms *source* and *destination* to notate the nodes doing the synchronization.

1. Log in to a Swift node running the `swift-container` service on the source system. You can determine this by looking at the service list in your `~/openstack/my_cloud/info/service_info.yml` file for a list of the servers containing this service.

2. Verify the SSL certificates by running this command against the destination Swift server:

   ```
   echo | openssl s_client -connect PUBLIC_API_ENDPOINT:8080 -CAfile /etc/ssl/certs/ca-certificates.crt
   ```

   If the connection was successful you should see a return code of `0 (ok)` similar to this:

   ```
   ...
   Timeout   : 300 (sec)
   Verify return code: 0 (ok)
   ```

3. Also verify that the source node can connect to the destination Swift system using this command:

   ```
   ardana > curl -k DESTINATION_IP OR HOSTNAME:8080/healthcheck
   ```

   If the connection was successful, you should see a response of `OK`.

4. Repeat these verification steps on any system involved in your container synchronization setup.

Configure container to container synchronization

Both the source and destination Swift systems must be configured the same way, using sync realms. For more details on how sync realms work, see OpenStack Swift - Configuring Container Sync (http://docs.openstack.org/developer/swift/overview_container_sync.html#configuring-container-sync) ↗.

To configure one of the systems, follow these steps:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/config/swift/container-sync-realms.conf.j2` file and uncomment the sync realm section.
   Here is a sample showing this section in the file:

   ```
   #Add sync realms here, for example:
   # [realm1]
   # key = realm1key
   # key2 = realm1key2
   # cluster_name1 = https://host1/v1/
   # cluster_name2 = https://host2/v1/
   ```

3. Add in the details for your source and destination systems. Each realm you define is a set of clusters that have agreed to allow container syncing between them. These values are case sensitive.
   Only one `key` is required. The second key is optional and can be provided to allow an operator to rotate keys if desired. The values for the clusters must contain the prefix `cluster_` and will be populated with the public API endpoints for the systems.

4. Commit the changes to git:

   ```
   ardana > git add -A
   ardana > git commit -a -m "Add node <name>"
   ```

5. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

6. Update the deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

7. Run the Swift reconfigure playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible/
   ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
   ```

8. Run this command to validate that your container synchronization is configured:

```
ardana > source ~/service.osrc
ardana > swift capabilities
```

Here is a snippet of the output showing the container sync information. This should be populated with your cluster names:

```
...
Additional middleware: container_sync
 Options:
   realms: {u'INTRACLUSTER': {u'clusters': {u'THISCLUSTER': {}}}}
```

9. Repeat these steps on any other Swift systems that will be involved in your sync realms.

8.6.4 Configuring Intra Cluster Container Sync

It is possible to use the swift container sync functionality to sync objects between containers within the same swift system. Swift is automatically configured to allow intra cluster container sync. Each swift PAC server will have an intracluster container sync realm defined in `/etc/swift/container-sync-realms.conf`.

For example:

```
# The intracluster realm facilitates syncing containers on this system
[intracluster]
key = lQ8JjuZfO
# key2 =
cluster_thiscluster = http://SWIFT-PROXY-VIP:8080/v1/
```

The keys defined in `/etc/swift/container-sync-realms.conf` are used by the container-sync daemon to determine trust. On top of this the containers that will be in sync will need a seperate shared key they both define in container metadata to establish their trust between each other.

1. Create two containers, for example container-src and container-dst. In this example we will sync one way from container-src to container-dst.

```
ardana > swift post container-src
ardana > swift post container-dst
```

2. Determine your swift account. In the following example it is AUTH_1234

```
ardana > swift stat
                                      Account: AUTH_1234
                                   Containers: 3
                                      Objects: 42
                                        Bytes: 21692421
Containers in policy "erasure-code-ring": 3
   Objects in policy "erasure-code-ring": 42
     Bytes in policy "erasure-code-ring": 21692421
                                 Content-Type: text/plain; charset=utf-8
             X-Account-Project-Domain-Id: default
                                  X-Timestamp: 1472651418.17025
                                  X-Trans-Id: tx81122c56032548aeae8cd-0057cee40c
                               Accept-Ranges: bytes
```

3. Configure container-src to sync to container-dst using a key specified by both containers. Replace *KEY* with your key.

```
ardana > swift post -t '//intracluster/thiscluster/AUTH_1234/container-dst' -k 'KEY'
 container-src
```

4. Configure container-dst to accept synced objects with this key

```
ardana > swift post -k 'KEY' container-dst
```

5. Upload objects to container-src. Within a number of minutes the objects should be automatically synced to container-dst.

Changing the intracluster realm key

The intracluster realm key used by container sync to sync objects between containers in the same swift system is automatically generated. The process for changing passwords is described in *Section 4.7, "Changing Service Passwords"*.

The steps to change the intracluster realm key are as follows.

1. On the Cloud Lifecycle Manager create a file called `~/openstack/change_credentials/swift_data_metadata.yml` with the contents included below. The `consuming-cp` and `cp` are the control plane name specified in `~/openstack/my_cloud/definition/data/control_plane.yml` where the swift-container service is running.

```
swift_intracluster_sync_key:
 metadata:
 - clusters:
    - swpac
```

```
    component: swift-container
    consuming-cp: control-plane-1
    cp: control-plane-1
 version: '2.0'
```

2. Run the following commands

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Reconfigure the swift credentials

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure-credentials-
change.yml
```

4. Delete ~/openstack/change_credentials/swift_data_metadata.yml

```
ardana > rm ~/openstack/change_credentials/swift_data_metadata.yml
```

5. On a swift PAC server check that the intracluster realm key has been updated in /etc/
   swift/container-sync-realms.conf

```
# The intracluster realm facilitates syncing containers on this system
[intracluster]
key = aNlDn3kWK
```

6. Update any containers using the intracluster container sync to use the new intracluster
   realm key

```
ardana > swift post -k 'aNlDn3kWK' container-src
ardana > swift post -k 'aNlDn3kWK' container-dst
```

9 Managing Networking

Information about managing and configuring the Networking service.

9.1 Configuring the HPE Helion OpenStack Firewall

The following instructions provide information about how to identify and modify the overall HPE Helion OpenStack firewall that is configured in front of the control services. This firewall is administered only by a cloud admin and is not available for tenant use for private network firewall services.

During the installation process, the configuration processor will automatically generate "allow" firewall rules for each server based on the services deployed and block all other ports. These are populated in `~/openstack/my_cloud/info/firewall_info.yml`, which includes a list of all the ports by network, including the addresses on which the ports will be opened. This is described in more detail in *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 5 "Input Model", Section 5.2 "Concepts", Section 5.2.10 "Networking", Section 5.2.10.5 "Firewall Configuration"*.

The `firewall_rules.yml` file in the input model allows you to define additional rules for each network group. You can read more about this in *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 6 "Configuration Objects", Section 6.15 "Firewall Rules"*.

The purpose of this document is to show you how to make post-installation changes to the firewall rules if the need arises.

Important

This process is not to be confused with Firewall-as-a-Service (see *Book "User Guide", Chapter 14 "Using Firewall as a Service (FWaaS)"*), which is a separate service that enables the ability for HPE Helion OpenStack tenants to create north-south, network-level firewalls to provide stateful protection to all instances in a private, tenant network. This service is optional and is tenant-configured.

9.1.1 Making Changes to the Firewall Rules

1. Log in to your Cloud Lifecycle Manager.

2. Edit your `~/openstack/my_cloud/definition/data/firewall_rules.yml` file and add the lines necessary to allow the port(s) needed through the firewall.
   In this example we are going to open up port range 5900-5905 to allow VNC traffic through the firewall:

```
- name: VNC
  network-groups:
- MANAGEMENT
  rules:
   - type: allow
     remote-ip-prefix:  0.0.0.0/0
     port-range-min: 5900
     port-range-max: 5905
     protocol: tcp
```

> **☞ Note**
>
> The example above shows a `remote-ip-prefix` of `0.0.0.0/0` which opens the ports up to all IP ranges. To be more secure you can specify your local IP address CIDR you will be running the VNC connect from.

3. Commit those changes to your local git:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "firewall rule update"
```

4. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Create the deployment directory structure:

```
ardana > cd ~/openstack/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Change to the deployment directory and run the `osconfig-iptables-deploy.yml` playbook to update your iptable rules to allow VNC:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts osconfig-iptables-deploy.yml
```

You can repeat these steps as needed to add, remove, or edit any of these firewall rules.

9.2 DNS Service Overview

HPE Helion OpenStack DNS service provides multi-tenant Domain Name Service with REST API management for domain and records.

⛨ Warning

The DNS Service is not intended to be used as an *internal* or *private* DNS service. The name records in DNSaaS should be treated as public information that anyone could query. There are controls to prevent tenants from creating records for domains they do not own. TSIG provides a **T**ransaction **SIG** nature to ensure integrity during zone transfer to other DNS servers.

9.2.1 For More Information

- For more information about Designate REST APIs, see the OpenStack REST API Documentation at http://docs.openstack.org/developer/designate/rest.html ↗.

- For a glossary of terms for Designate, see the OpenStack glossary at http://docs.openstack.org/developer/designate/glossary.html ↗.

9.2.2 Designate Initial Configuration

After the HPE Helion OpenStack installation has been completed, Designate requires initial configuration to operate.

9.2.2.1 Identifying Name Server Public IPs

Depending on the back-end, the method used to identify the name servers' public IPs will differ.

9.2.2.1.1 InfoBlox

InfoBlox will act as your public name servers, consult the InfoBlox management UI to identify the IPs.

9.2.2.1.2 PowerDNS or BIND Back-end

You can find the name server IPs in `/etc/hosts` by looking for the `ext-api` addresses, which are the addresses of the controllers. For example:

```
192.168.10.1 example-cp1-c1-m1-extapi
192.168.10.2 example-cp1-c1-m2-extapi
192.168.10.3 example-cp1-c1-m3-extapi
```

9.2.2.1.3 Creating Name Server A Records

Each name server requires a public name, for example `ns1.example.com.`, to which Designate-managed domains will be delegated. There are two common locations where these may be registered, either within a zone hosted on Designate itself, or within a zone hosted on a external DNS service.

If you are using an externally managed zone for these names:

- For each name server public IP, create the necessary A records in the external system.

If you are using a Designate-managed zone for these names:

1. Create the zone in Designate which will contain the records:

```
ardana > openstack zone create --email hostmaster@example.com example.com.
+---------------+-----------------------------------+
| Field         | Value                             |
+---------------+-----------------------------------+
| action        | CREATE                            |
| created_at    | 2016-03-09T13:16:41.000000        |
| description   | None                              |
| email         | hostmaster@example.com            |
| id            | 23501581-7e34-4b88-94f4-ad8cec1f4387 |
```

```
| masters        |                                       |
| name           | example.com.                          |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842  |
| project_id     | a194d740818942a8bea6f3674e0a3d71      |
| serial         | 1457529400                            |
| status         | PENDING                               |
| transferred_at | None                                  |
| ttl            | 3600                                  |
| type           | PRIMARY                               |
| updated_at     | None                                  |
| version        | 1                                     |
+----------------+---------------------------------------+
```

2. For each name server public IP, create an A record. For example:

```
ardana > openstack recordset create --records 192.168.10.1 --type A example.com.
 ns1.example.com.
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2016-03-09T13:18:36.000000           |
| description | None                                 |
| id          | 09e962ed-6915-441a-a5a1-e8d93c3239b6 |
| name        | ns1.example.com.                     |
| records     | 192.168.10.1                         |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | 23501581-7e34-4b88-94f4-ad8cec1f4387 |
+-------------+--------------------------------------+
```

3. When records have been added, list the record sets in the zone to validate:

```
ardana > openstack recordset list example.com.
+-------------+-----------------+------
+--------------------------------------------------+
| id          | name            | type | records
  |
+-------------+-----------------+------
+--------------------------------------------------+
| 2d6cf...655b | example.com.    | SOA  | ns1.example.com. hostmaster.example.com
 145...600 |
| 33466...bd9c | example.com.    | NS   | ns1.example.com.
  |
| da98c...bc2f | example.com.    | NS   | ns2.example.com.
  |
```

```
| 672ee...74dd | example.com.     | NS   | ns3.example.com.
  |
| 09e96...39b6 | ns1.example.com. | A    | 192.168.10.1
  |
| bca4f...a752 | ns2.example.com. | A    | 192.168.10.2
  |
| 0f123...2117 | ns3.example.com. | A    | 192.168.10.3
  |
+-------------+----------------+------
+---------------------------------------------+
```

4. Contact your domain registrar requesting *Glue Records* to be registered in the `com.` zone for the nameserver and public IP address pairs above. If you are using a sub-zone of an existing company zone (for example, `ns1.cloud.mycompany.com.`), the Glue must be placed in the `mycompany.com.` zone.

9.2.2.1.4 For More Information

For additional DNS integration and configuration information, see the OpenStack Designate documentation at https://docs.openstack.org/designate/pike/index.html↗.

For more information on creating servers, domains and examples, see the OpenStack REST API documentation at https://developer.openstack.org/api-ref/dns/↗.

9.2.3 DNS Service Monitoring Support

9.2.3.1 DNS Service Monitoring Support

Additional monitoring support for the DNS Service (Designate) has been added to HPE Helion OpenStack.

In the Networking section of the Operations Console, you can see alarms for all of the DNS Services (Designate), such as designate-zone-manager, designate-api, designate-pool-manager, designate-mdns, and designate-central after running `designate-stop.yml`.

You can run `designate-start.yml` to start the DNS Services back up and the alarms will change from a red status to green and be removed from the **New Alarms** panel of the Operations Console.

An example of the generated alarms from the Operations Console is provided below after running `designate-stop.yml`:

```
ALARM:  STATE:  ALARM ID:  LAST CHECK:  DIMENSION:
Process Check
0f221056-1b0e-4507-9a28-2e42561fac3e 2016-10-03T10:06:32.106Z hostname=ardana-cp1-c1-m1-
mgmt,
service=dns,
cluster=cluster1,
process_name=designate-zone-manager,
component=designate-zone-manager,
control_plane=control-plane-1,
cloud_name=entry-scale-kvm

Process Check
50dc4c7b-6fae-416c-9388-6194d2cfc837 2016-10-03T10:04:32.086Z hostname=ardana-cp1-c1-m1-
mgmt,
service=dns,
cluster=cluster1,
process_name=designate-api,
component=designate-api,
control_plane=control-plane-1,
cloud_name=entry-scale-kvm

Process Check
55cf49cd-1189-4d07-aaf4-09ed08463044 2016-10-03T10:05:32.109Z hostname=ardana-cp1-c1-m1-
mgmt,
service=dns,
cluster=cluster1,
process_name=designate-pool-manager,
component=designate-pool-manager,
control_plane=control-plane-1,
cloud_name=entry-scale-kvm

Process Check
c4ab7a2e-19d7-4eb2-a9e9-26d3b14465ea 2016-10-03T10:06:32.105Z hostname=ardana-cp1-c1-m1-
mgmt,
service=dns,
cluster=cluster1,
process_name=designate-mdns,
component=designate-mdns,
control_plane=control-plane-1,
cloud_name=entry-scale-kvm
HTTP Status
c6349bbf-4fd1-461a-9932-434169b86ce5 2016-10-03T10:05:01.731Z service=dns,
cluster=cluster1,
```

```
url=http://100.60.90.3:9001/,
hostname=ardana-cp1-c1-m3-mgmt,
component=designate-api,
control_plane=control-plane-1,
api_endpoint=internal,
cloud_name=entry-scale-kvm,
monitored_host_type=instance


Process Check
ec2c32c8-3b91-4656-be70-27ff0c271c89 2016-10-03T10:04:32.082Z hostname=ardana-cp1-c1-m1-
mgmt,
service=dns,
cluster=cluster1,
process_name=designate-central,
component=designate-central,
control_plane=control-plane-1,
cloud_name=entry-scale-kvm
```

9.3 Networking Service Overview

HPE Helion OpenStack Networking is a virtual networking service that leverages the OpenStack Neutron service to provide network connectivity and addressing to HPE Helion OpenStack Compute service devices.

The Networking service also provides an API to configure and manage a variety of network services.

You can use the Networking service to connect guest servers or you can define and configure your own virtual network topology.

9.3.1 Installing the Networking service

HPE Helion OpenStack Network Administrators are responsible for planning for the Neutron networking service, and once installed, to configure the service to meet the needs of their cloud network users.

9.3.2 Working with the Networking service

To perform tasks using the Networking service, you can use the dashboard, API or CLI.

9.3.3 Reconfiguring the Networking service

If you change any of the network configuration after installation, it is recommended that you reconfigure the Networking service by running the neutron-reconfigure playbook.

On the Cloud Lifecycle Manager:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
```

9.3.4 For more information

For information on how to operate your cloud we suggest you read the OpenStack Operations Guide (http://docs.openstack.org/ops/) ↗. The *Architecture* section contains useful information about how an OpenStack Cloud is put together. However, HPE Helion OpenStack takes care of these details for you. The *Operations* section contains information on how to manage the system.

9.3.5 Neutron External Networks

9.3.5.1 External networks overview

This topic explains how to create a Neutron external network.

External networks provide access to the internet.

The typical use is to provide an IP address that can be used to reach a VM from an external network which can be a public network like the internet or a network that is private to an organization.

9.3.5.2 Using the Ansible Playbook

This playbook will query the Networking service for an existing external network, and then create a new one if you do not already have one. The resulting external network will have the name `ext-net` with a subnet matching the CIDR you specify in the command below.

If you need to specify more granularity, for example specifying an allocation pool for the subnet, use the *Section 9.3.5.3, "Using the NeutronClient CLI"*.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/verb_hosts neutron-cloud-configure.yml -e
 EXT_NET_CIDR=<CIDR>
```

The table below shows the optional switch that you can use as part of this playbook to specify environment-specific information:

Switch	Description
`-e EXT_NET_CIDR=<CIDR>`	Optional. You can use this switch to specify the external network CIDR. If you choose not to use this switch, or use a wrong value, the VMs will not be accessible over the network. This CIDR will be from the `EXTERNAL VM` network.

9.3.5.3 Using the NeutronClient CLI

For more granularity you can utilize the Neutron command line tool to create your external network.

1. Log in to the Cloud Lifecycle Manager.

2. Source the Admin creds:

   ```
   ardana > source ~/service.osrc
   ```

3. Create the external network and then the subnet using these commands below.
   Creating the network:

   ```
   ardana > neutron net-create --router:external <external-network-name>
   ```

   Creating the subnet:

   ```
   ardana > neutron subnet-create EXTERNAL-NETWORK-NAME CIDR --gateway GATEWAY --
   allocation-pool start=IP_START,end=IP_END [--disable-dhcp]
   ```

Where:

Value	Description
external-network-name	This is the name given to your external network. This is a unique value that you will choose. The value `ext-net` is usually used.
CIDR	You can use this switch to specify the external network CIDR. If you choose not to use this switch, or use a wrong value, the VMs will not be accessible over the network. This CIDR will be from the EXTERNAL VM network.
--gateway	Optional switch to specify the gateway IP for your subnet. If this is not included then it will choose the first available IP.
--allocation-pool start end	Optional switch to specify a start and end IP address to use as the allocation pool for this subnet.
--disable-dhcp	Optional switch if you want to disable DHCP on this subnet. If this is not specified then DHCP will be enabled.

9.3.5.4   Multiple External Networks

HPE Helion OpenStack provides the ability to have multiple external networks, by using the Network Service (Neutron) provider networks for external networks. You can configure HPE Helion OpenStack to allow the use of provider VLANs as external networks by following these steps.

1. Do NOT include the `neutron.l3_agent.external_network_bridge` tag in the network_groups definition for your cloud. This results in the `l3_agent.ini` `external_network_bridge` being set to an empty value (rather than the traditional br-ex).

2. Configure your cloud to use provider VLANs, by specifying the `provider_physical_network` tag on one of the network_groups defined for your cloud. For example, to run provider VLANS over the EXAMPLE network group: (some attributes omitted for brevity)

```
network-groups:

  - name: EXAMPLE
    tags:
      - neutron.networks.vlan:
          provider-physical-network: physnet1
```

3. After the cloud has been deployed, you can create external networks using provider VLANs. For example, using the Network Service CLI:

   a. Create external network 1 on vlan101

   ```
   neutron net-create --provider:network_type vlan --provider:physical_network
     physnet1 --provider:segmentation_id 101 ext-net1 --router:external true
   ```

   b. Create external network 2 on vlan102

   ```
   neutron net-create --provider:network_type vlan --provider:physical_network
     physnet1 --provider:segmentation_id 102 ext-net2 --router:external true
   ```

9.3.6 Neutron Provider Networks

This topic explains how to create a Neutron provider network.

A provider network is a virtual network created in the HPE Helion OpenStack cloud that is consumed by HPE Helion OpenStack services. The distinctive element of a provider network is that it does not create a virtual router; rather, it depends on L3 routing that is provided by the infrastructure.

A provider network is created by adding the specification to the HPE Helion OpenStack input model. It consists of at least one network and one or more subnets.

9.3.6.1 HPE Helion OpenStack input model

The input model is the primary mechanism a cloud admin uses in defining a HPE Helion OpenStack installation. It exists as a directory with a data subdirectory that contains YAML files. By convention, any service that creates a Neutron provider network will create a subdirectory under the data directory and the name of the subdirectory shall be the project name. For example, the Octavia project will use Neutron provider networks so it will have a subdirectory named 'octavia' and the config file that specifies the neutron network will exist in that subdirectory.

```
├── cloudConfig.yml
    ├── data
    │   ├── control_plane.yml
    │   ├── disks_compute.yml
    │   ├── disks_controller_1TB.yml
    │   ├── disks_controller.yml
    │   ├── firewall_rules.yml
    │   ├── net_interfaces.yml
    │   ├── network_groups.yml
    │   ├── networks.yml
    │   ├── neutron
    │   │   └── neutron_config.yml
    │   ├── nic_mappings.yml
    │   ├── server_groups.yml
    │   ├── server_roles.yml
    │   ├── servers.yml
    │   ├── swift
    │   │   └── rings.yml
    │   └── octavia
    │       └── octavia_config.yml
    ├── README.html
    └── README.md
```

9.3.6.2 Network/Subnet specification

The elements required in the input model for you to define a network are:

- name

- network_type

- physical_network

Elements that are optional when defining a network are:

- segmentation_id
- shared

Required elements for the subnet definition are:

- cidr

Optional elements for the subnet definition are:

- allocation_pools which will require start and end addresses
- host_routes which will require a destination and nexthop
- gateway_ip
- no_gateway
- enable-dhcp

NOTE: Only IPv4 is supported at the present time.

9.3.6.3   Network details

The following table outlines the network values to be set, and what they represent.

Attribute	Required/ optional	Allowed Values	Usage
name	Required		
network_type	Required	flat, vlan, vxlan	The type of desired network
physical_network	Required	Valid	Name of physical network that is overlayed with the virtual network
segmentation_id	Optional	vlan or vxlan ranges	VLAN id for vlan or tunnel id for vxlan
shared	Optional	True	Shared by all projects or private to a single project

9.3.6.4 Subnet details

The following table outlines the subnet values to be set, and what they represent.

Attribute	Req/Opt	Allowed Values	Usage
cidr	Required	Valid CIDR range	for example, 172.30.0.0/24
allocation_pools	Optional	See allocation_pools table below	
host_routes	Optional	See host_routes table below	
gateway_ip	Optional	Valid IP addr	Subnet gateway to other nets
no_gateway	Optional	True	No distribution of gateway
enable-dhcp	Optional	True	Enable dhcp for this subnet

9.3.6.5 ALLOCATION_POOLS details

The following table explains allocation pool settings.

Attribute	Req/Opt	Allowed Values	Usage
start	Required	Valid IP addr	First ip address in pool
end	Required	Valid IP addr	Last ip address in pool

9.3.6.6 HOST_ROUTES details

The following table explains host route settings.

Attribute	Req/Opt	Allowed Values	Usage
destination	Required	Valid CIDR	Destination subnet

Attribute	Req/Opt	Allowed Values	Usage
nexthop	Required	Valid IP addr	Hop to take to destination subnet

☞ Note

Multiple destination/nexthop values can be used.

9.3.6.7 Examples

The following examples show the configuration file settings for Neutron and Octavia.

Octavia configuration

This file defines the mapping. It does not need to be edited unless you want to change the name of your VLAN.

Path: `~/openstack/my_cloud/definition/data/octavia/octavia_config.yml`

```
---
  product:
    version: 2

  configuration-data:
    - name: OCTAVIA-CONFIG-CP1
      services:
        - octavia
      data:
        amp_network_name: OCTAVIA-MGMT-NET
```

Neutron configuration

Input your network configuration information for your provider VLANs in `neutron_config.yml` found here:

`~/openstack/my_cloud/definition/data/neutron/`.

```
---
  product:
    version: 2

  configuration-data:
    - name:  NEUTRON-CONFIG-CP1
      services:
```

```
      - neutron
  data:
    neutron_provider_networks:
    - name: OCTAVIA-MGMT-NET
      provider:
         - network_type: vlan
           physical_network: physnet1
           segmentation_id: 2754
      cidr: 10.13.189.0/24
      no_gateway:  True
      enable_dhcp: True
      allocation_pools:
         - start: 10.13.189.4
           end: 10.13.189.252
      host_routes:
        # route to MANAGEMENT-NET
        - destination: 10.13.111.128/26
          nexthop:  10.13.189.5
```

9.3.6.8 Implementing your changes

1. Commit the changes to git:

   ```
   ardana > git add -A
   ardana > git commit -a -m "configuring provider network"
   ```

2. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

3. Update your deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

4. Then continue with your clean cloud installation.

5. If you are only adding a Neutron Provider network to an existing model, then run the neutron-deploy.yml playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts neutron-deploy.yml
   ```

9.3.6.9 Multiple Provider Networks

The physical network infrastructure must be configured to convey the provider VLAN traffic as tagged VLANs to the cloud compute nodes and network service network nodes. Configuration of the physical network infrastructure is outside the scope of the HPE Helion OpenStack 8 software.

HPE Helion OpenStack 8 automates the server networking configuration and the Network Service configuration based on information in the cloud definition. To configure the system for provider VLANs, specify the *neutron.networks.vlan* tag with a *provider-physical-network* attribute on one or more network groups. For example (some attributes omitted for brevity):

```
network-groups:

      - name: NET_GROUP_A
      tags:
      - neutron.networks.vlan:
      provider-physical-network: physnet1

      - name: NET_GROUP_B
      tags:
      - neutron.networks.vlan:
      provider-physical-network: physnet2
```

A network group is associated with a server network interface via an interface model. For example (some attributes omitted for brevity):

```
interface-models:
      - name: INTERFACE_SET_X
      network-interfaces:
      - device:
      name: bond0
      network-groups:
      - NET_GROUP_A
      - device:
      name: eth3
      network-groups:
      - NET_GROUP_B
```

A network group used for provider VLANs may contain only a single HPE Helion OpenStack network, because that VLAN must span all compute nodes and any Network Service network nodes/controllers (that is, it is a single L2 segment). The HPE Helion OpenStack network must be defined with tagged-vlan false, otherwise a Linux VLAN network interface will be created. For example:

```
networks:
```

```
        - name: NET_A
        tagged-vlan: false
        network-group: NET_GROUP_A

        - name: NET_B
        tagged-vlan: false
        network-group: NET_GROUP_B
```

When the cloud is deployed, HPE Helion OpenStack 8 will create the appropriate bridges on the servers, and set the appropriate attributes in the Neutron configuration files (for example, bridge_mappings).

After the cloud has been deployed, create Network Service network objects for each provider VLAN. For example, using the Network Service CLI:

```
ardana > neutron net-create --provider:network_type vlan --provider:physical_network
 physnet1 --provider:segmentation_id 101 mynet101
ardana > neutron net-create --provider:network_type vlan --provider:physical_network
 physnet2 --provider:segmentation_id 234 mynet234
```

9.3.6.10   More Information

For more information on the Network Service command-line interface (CLI), see the OpenStack networking command-line client reference: http://docs.openstack.org/cli-reference/content/neutronclient_commands.html ↗

9.3.7   Using IPAM Drivers in the Networking Service

This topic describes how to choose and implement an IPAM driver.

9.3.7.1   Selecting and implementing an IPAM driver

Beginning with the Liberty release, OpenStack networking includes a pluggable interface for the IP Address Management (IPAM) function. This interface creates a driver framework for the allocation and de-allocation of subnets and IP addresses, enabling the integration of alternate IPAM implementations or third-party IP Address Management systems.

There are three possible IPAM driver options:

- Non-pluggable driver. This option is the default when the ipam_driver parameter is not specified in neutron.conf.

- Pluggable reference IPAM driver. The pluggable IPAM driver interface was introduced in HPE Helion OpenStack 8 (OpenStack Liberty). It is a refactoring of the Kilo non-pluggable driver to use the new pluggable interface. The setting in neutron.conf to specify this driver is `ipam_driver = internal`.

- Pluggable Infoblox IPAM driver. The pluggable Infoblox IPAM driver is a third-party implementation of the pluggable IPAM interface. the corresponding setting in neutron.conf to specify this driver is `ipam_driver = networking_infoblox.ipam.driver.InfobloxPool`.

> 🎓 **Note**
>
> You can use either the non-pluggable IPAM driver or a pluggable one. However, you cannot use both.

9.3.7.2 Using the Pluggable reference IPAM driver

To indicate that you want to use the Pluggable reference IPAM driver, the only parameter needed is "ipam_driver." You can set it by looking for the following commented line in the `neutron.conf.j2` template (ipam_driver = internal) uncommenting it, and committing the file. After following the standard steps to deploy Neutron, Neutron will be configured to run using the Pluggable reference IPAM driver.

As stated, the file you must edit is `neutron.conf.j2` on the Cloud Lifecycle Manager in the directory `~/openstack/my_cloud/config/neutron`. Here is the relevant section where you can see the `ipam_driver` parameter commented out:

```
[DEFAULT]
  ...
  l3_ha_net_cidr = 169.254.192.0/18

  # Uncomment the line below if the Reference Pluggable IPAM driver is to be used
  # ipam_driver = internal
  ...
```

After uncommenting the line `ipam_driver = internal`, commit the file using git commit from the `openstack/my_cloud` directory:

```
ardana > git commit -a -m 'My config for enabling the internal IPAM Driver'
```

Then follow the steps to deploy HPE Helion OpenStack in the *Book "Installing with Cloud Lifecycle Manager", Preface "Installation Overview"* appropriate to your cloud configuration.

🎓 Note

Currently there is no migration path from the non-pluggable driver to a pluggable IPAM driver because changes are needed to database tables and Neutron currently cannot make those changes.

9.3.7.3 Using the Infoblox IPAM driver

As suggested above, using the Infoblox IPAM driver requires changes to existing parameters in `nova.conf` and `neutron.conf`. If you want to use the infoblox appliance, you will need to add the "infoblox service-component" to the service-role containing the neutron API server. To use the infoblox appliance for IPAM, both the agent *and* the Infoblox IPAM driver are required. The `infoblox-ipam-agent` should be deployed on the same node where the neutron-server component is running. Usually this is a Controller node.

1. Have the Infoblox appliance running on the management network (the Infoblox appliance admin or the datacenter administrator should know how to perform this step).

2. Change the control plane definition to add `infoblox-ipam-agent` as a service in the controller node cluster (see change in bold). Make the changes in `control_plane.yml` found here: `~/openstack/my_cloud/definition/data/control_plane.yml`

```
---
  product:
    version: 2

  control-planes:
    - name: ccp
      control-plane-prefix: ccp
...
      clusters:
        - name: cluster0
```

```
          cluster-prefix: c0
          server-role: ARDANA-ROLE
          member-count: 1
          allocation-policy: strict
          service-components:
            - lifecycle-manager
        - name: cluster1
          cluster-prefix: c1
          server-role: CONTROLLER-ROLE
          member-count: 3
          allocation-policy: strict
          service-components:
            - ntp-server
...
            - neutron-server
            - infoblox-ipam-agent
...
            - designate-client
            - powerdns
      resources:
        - name: compute
          resource-prefix: comp
          server-role: COMPUTE-ROLE
          allocation-policy: any
```

3. Modify the `~/openstack/my_cloud/config/neutron/neutron.conf.j2` file on the controller node to comment and uncomment the lines noted below to enable use with the Infoblox appliance:

```
[DEFAULT]
          ...
          l3_ha_net_cidr = 169.254.192.0/18


          # Uncomment the line below if the Reference Pluggable IPAM driver is to
 be used
          # ipam_driver = internal


          # Comment out the line below if the Infoblox IPAM Driver is to be used
          # notification_driver = messaging

          # Uncomment the lines below if the Infoblox IPAM driver is to be used
          ipam_driver = networking_infoblox.ipam.driver.InfobloxPool
          notification_driver = messagingv2
```

```
            # Modify the infoblox sections below to suit your cloud environment

            [infoblox]
            cloud_data_center_id = 1
            # This name of this section is formed by "infoblox-
dc:<infoblox.cloud_data_center_id>"
            # If cloud_data_center_id is 1, then the section name is "infoblox-dc:1"

            [infoblox-dc:0]
            http_request_timeout = 120
            http_pool_maxsize = 100
            http_pool_connections = 100
            ssl_verify = False
            wapi_version = 2.2
            admin_user_name = admin
            admin_password = infoblox
            grid_master_name = infoblox.localdomain
            grid_master_host = 1.2.3.4


            [QUOTAS]
            ...
```

4. Change `nova.conf.j2` to replace the notification driver "messaging" to "messagingv2"

```
...

# Oslo messaging
notification_driver = log

#  Note:
#  If the infoblox-ipam-agent is to be deployed in the cloud, change the
#  notification_driver setting from "messaging" to "messagingv2".
notification_driver = messagingv2
notification_topics = notifications

# Policy
...
```

5. Commit the changes:

```
ardana > cd ~/openstack/my_cloud
```

```
ardana > git commit —a —m 'My config for enabling the Infoblox IPAM driver'
```

6. Deploy the cloud with the changes. Due to changes to the control_plane.yml, you will need to rerun the config-processor-run.yml playbook if you have run it already during the install process.

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts site.yml
```

9.3.7.4 Configuration parameters for using the Infoblox IPAM driver

Changes required in the notification parameters in nova.conf:

Parameter Name	Section in nova.conf	Default Value	Current Value	Description
notify_on_state_change	DEFAULT	None	vm_and_task_state	Send compute.instance.update notifications on instance state changes. Vm_and_task_state means notify on vm and task state changes. Infoblox requires the value to be vm_state (notify on vm state change). **Thus NO CHANGE is needed for infoblox**

Parameter Name	Section in nova.conf	Default Value	Current Value	Description
notification_topics	DEFAULT	empty list	notifications	**NO CHANGE is needed for infoblox.** The infoblox installation guide requires the notifications to be "notifications"
notification_driver	DEFAULT	None	messaging	**Change needed.** The infoblox installation guide requires the notification driver to be "messagingv2".

Changes to existing parameters in neutron.conf

Parameter Name	Section in neutron.conf	Default Value	Current Value	Description
ipam_driver	DEFAULT	None	None (param is undeclared in neutron.conf)	Pluggable IPAM driver to be used by Neutron API server. For infoblox, the value is "networking_infoblox.ipam.driv
notification_driver	DEFAULT	empty list	messaging	The driver used to send notifications

Parameter Name	Section in neutron.conf	Default Value	Current Value	Description
				from the Neutron API server to the Neutron agents. The installation guide for networking-infoblox calls for the notification_driver to be "messagingv2"
notification_topics	DEFAULT	None	notifications	**No change needed**. The row is here show the changes in the Neutron parameters described in the installation guide for networking-infoblox

Parameters specific to the Networking Infoblox Driver. All the parameters for the Infoblox IPAM driver must be defined in neutron.conf.

Parameter Name	Section in neutron.conf	Default Value	Description
cloud_data_center_id	infoblox	0	ID for selecting a particular grid from one or more grids to

Parameter Name	Section in neutron.conf	Default Value	Description
			serve networks in the Infoblox back end
ipam_agent_workers	infoblox	1	Number of Infoblox IPAM agent works to run
grid_master_host	infoblox-dc.<cloud_data_center_id>	empty string	IP address of the grid master. WAPI requests are sent to the grid_master_host
ssl_verify	infoblox-dc.<cloud_data_center_id>	False	Ensure whether WAPI requests sent over HTTPS require SSL verification
WAPI Version	infoblox-dc.<cloud_data_center_id>	1.4	The WAPI version. Value should be 2.2.
admin_user_name	infoblox-dc.<cloud_data_center_id>	empty string	Admin user name to access the grid master or cloud platform appliance
admin_password	infoblox-dc.<cloud_data_center_id>	empty string	Admin user password
http_pool_connections	infoblox-dc.<cloud_data_center_id>	100	
http_pool_maxsize	infoblox-dc.<cloud_data_center_id>	100	
http_request_timeout	infoblox-dc.<cloud_data_center_id>	120	

The diagram below shows Nova compute sending notification to the infoblox-ipam-agent
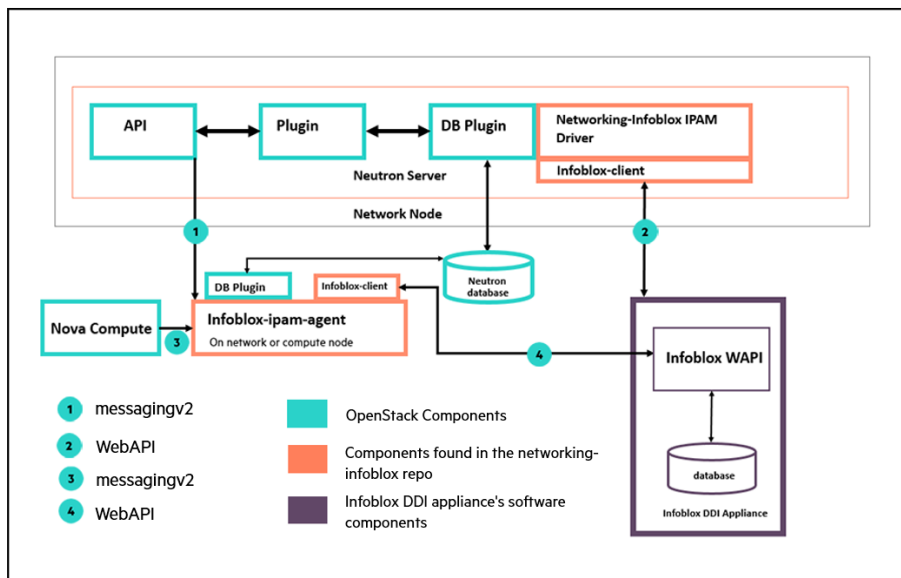
9.3.7.5 Limitations

- There is no IPAM migration path from non-pluggable to pluggable IPAM driver (https://bugs.launchpad.net/neutron/+bug/1516156 ↗). This means there is no way to reconfigure the Neutron database if you wanted to change Neutron to use a pluggable IPAM driver. Unless you change the default of non-pluggable IPAM configuration to a pluggable driver at install time, you will have no other opportunity to make that change because reconfiguration of HPE Helion OpenStack 8from using the default non-pluggable IPAM configuration to HPE Helion OpenStack 8 using a pluggable IPAM driver is not supported.

- Upgrade from previous versions of HPE Helion OpenStack to HPE Helion OpenStack 8 to use a pluggable IPAM driver is not supported.

- The Infoblox appliance does not allow for overlapping IPs. For example, only one tenant can have a CIDR of 10.0.0.0/24.

- The infoblox IPAM driver fails the creation of a subnet when a there is no gateway-ip supplied. For example, the command "neutron subnet-create ... --no-gateway ..." will fail.

9.3.8 Configuring Load Balancing as a Service (LBaaS)

HPE Helion OpenStack 8 LBaaS Configuration

Load Balancing as a Service (LBaaS) is an advanced networking service that allows load balancing of multi-node environments. It provides the ability to spread requests across multiple servers thereby reducing the load on any single server. This document describes the installation steps for LBaaS v1 (see prerequisites) and the configuration for LBaaS v1 and v2.

HPE Helion OpenStack 8 can support either LBaaS v1 or LBaaS v2 to allow for wide ranging customer requirements. If the decision is made to utilize LBaaS v1 it is highly unlikely that you will be able to perform an on-line upgrade of the service to v2 after the fact as the internal data structures are significantly different. Should you wish to attempt an upgrade, support will be needed from Professional Services and your chosen load balancer partner.

⛨ Warning

The LBaaS architecture is based on a driver model to support different load balancers. LBaaS-compatible drivers are provided by load balancer vendors including F5 and Citrix. A new software load balancer driver was introduced in the OpenStack Liberty release called "*Octavia*". The Octavia driver deploys a software load balancer called HAProxy. Octavia is the default load balancing provider in HPE Helion OpenStack 8 for LBaaS V2. Until Octavia is configured the creation of load balancers will fail with an error. Please refer to *Book "*Installing with Cloud Lifecycle Manager*", Chapter 32 "Configuring Load Balancer as a Service"* document for information on installing Octavia.

⛨ Warning

Before upgrading to HPE Helion OpenStack 8, contact F5 and HPE to determine which F5 drivers have been certified for use with HPE Helion OpenStack. Loading drivers not certified by HPE may result in failure of your cloud deployment.

LBaaS V2 offers with *Book "*Installing with Cloud Lifecycle Manager*", Chapter 32 "Configuring Load Balancer as a Service"* a software load balancing solution that supports both a highly available control plane and data plane. However, should an external hardware load balancer be selected the cloud operation can achieve additional performance and availability.

LBaaS v1

Reasons to select this version.

1. You must be able to configure LBaaS via Horizon.

2. Your hardware load balancer vendor does not currently support LBaaS v2.

Reasons not to select this version.

1. No active development is being performed on this API in the OpenStack community. (Security fixes are still being worked upon).

2. It does not allow for multiple ports on the same VIP (for example, to support both port 80 and 443 on a single VIP).

3. It will never be able to support TLS termination/re-encryption at the load balancer.

4. It will never be able to support L7 rules for load balancing.

5. LBaaS v1 will likely become officially deprecated by the OpenStack community at the Tokyo (October 2015) summit.

LBaaS v2

Reasons to select this version.

1. Your vendor already has a driver that supports LBaaS v2. Many hardware load balancer vendors already support LBaaS v2 and this list is growing all the time.

2. You intend to script your load balancer creation and management so a UI is not important right now (Horizon support will be added in a future release).

3. You intend to support TLS termination at the load balancer.

4. You intend to use the Octavia software load balancer (adding HA and scalability).

5. You do not want to take your load balancers offline to perform subsequent LBaaS upgrades.

6. You intend in future releases to need L7 load balancing.

Reasons not to select this version.

1. Your LBaaS vendor does not have a v2 driver.

2. You must be able to manage your load balancers from Horizon.

3. You have legacy software which utilizes the LBaaS v1 API.

LBaaS v1 requires configuration changes prior to installation and is not recommended. LBaaS v2 is installed by default with HPE Helion OpenStack and requires minimal configuration to start the service.

Note

Only LBaaS V2 API currently supports load balancer failover with Octavia. However, in LBaaS V1 and if Octavia is not deployed when a load balancer is deleted it will need to be manually recreated. LBaaS v2 API includes automatic failover of a deployed load balancer with Octavia. More information about this driver can be found in *Book "Installing with Cloud Lifecycle Manager", Chapter 32 "Configuring Load Balancer as a Service"*.

9.3.8.1 Prerequisites

HPE Helion OpenStack LBaaS v1

Installing LBaaS v1

Important

It is not recommended that LBaaS v1 is used in a production environment. It is recommended you use LBaaS v2. If you do deploy LBaaS v1, the upgrade to LBaaS v2 is non-trivial and may require the use of professional services.

Note

If you need to run LBaaS v1 instead of the default LBaaS v2, you should make appropriate installation preparations during HPE Helion OpenStack installation since LBaaS v2 is the default. If you have selected to install and use LBaaS v1 you will replace the `control_plane.yml` directories and `neutron.conf.j2` file to use version 1.

Before you modify the control_plane.yml file, it is recommended that you back up the original version of this file. Once you have backed them up, modify the control_plane.yml file.

1. Edit ~/openstack/my_cloud/definition/data/control_plane.yml - depending on your installation the control_plane.yml file might be in a different location.

2. In the section specifying the compute nodes (resources/compute) replace neutron-lbaasv2-agent with neutron-lbaas-agent - there will only be one occurrence in that file.

3. Save the modified file.

4. Follow the steps in *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"* to commit and apply the changes.

5. To test the installation follow the steps outlined in *Book "Installing with Cloud Lifecycle Manager", Chapter 32 "Configuring Load Balancer as a Service"* after you have created a suitable subnet, see: *Book "Installing with Cloud Lifecycle Manager", Chapter 28 "UI Verification", Section 28.4 "Creating an External Network"*.

HPE Helion OpenStack LBaaS v2

1. HPE Helion OpenStack must be installed for LBaaS v2.

2. Follow the instructions to install *Book "Installing with Cloud Lifecycle Manager", Chapter 32 "Configuring Load Balancer as a Service"*

9.3.9   Load Balancer: Octavia Driver Administration

This document provides the instructions on how to enable and manage various components of the Load Balancer Octavia driver if that driver is enabled.

- *Section 9.3.9.1, "Monasca Alerts"*

- *Section 9.3.9.2, "Tuning Octavia Installation"*

  - Homogeneous Compute Configuration

  - Octavia and Floating IP's

  - Configuration Files

  - Spare Pools

- *Section 9.3.9.3, "Managing Amphora"*

  - Updating the Cryptographic Certificates

  - Accessing VM information in Nova

  - Initiating Failover of an Amphora VM

9.3.9.1 Monasca Alerts

The Monasca-agent has the following Octavia-related plugins:

- Process checks – checks if octavia processes are running. When it starts, it detects which processes are running and then monitors them.

- http_connect check – checks if it can connect to octavia api servers.

Alerts are displayed in the Operations Console. For more information see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.1 "Operations Console Overview"*.

9.3.9.2 Tuning Octavia Installation

Homogeneous Compute Configuration

Octavia works only with homogeneous compute node configurations. Currently, Octavia does not support multiple nova flavors. If Octavia needs to be supported on multiple compute nodes, then all the compute nodes should carry same set of physnets (which will be used for Octavia).

Octavia and Floating IPs

Due to a Neutron limitation Octavia will only work with CVR routers. Another option is to use VLAN provider networks which do not require a router.

You cannot currently assign a floating IP address as the VIP (user facing) address for a load balancer created by the Octavia driver if the underlying Neutron network is configured to support Distributed Virtual Router (DVR). The Octavia driver uses a Neutron function known as *allowed address pairs* to support load balancer fail over.

There is currently a Neutron bug that does not support this function in a DVR configuration

Octavia Configuration Files

The system comes pre-tuned and should not need any adjustments for most customers. If in rare instances manual tuning is needed, follow these steps:

⛊ Warning

Changes might be lost during HPE Helion OpenStack upgrades.

Edit the Octavia configuration files in `my_cloud/config/octavia`. It is recommended that any changes be made in all of the Octavia configuration files.

- octavia-api.conf.j2

- octavia-health-manager.conf.j2

- octavia-housekeeping.conf.j2

- octavia-worker.conf.j2

After the changes are made to the configuration files, redeploy the service.

1. Commit changes to git.

   ```
   ardana > cd ~/openstack
   ardana > git add -A
   ardana > git commit -m "My Octavia Config"
   ```

2. Run the configuration processor and ready deployment.

   ```
   ardana > cd ~/openstack/ardana/ansible/
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

3. Run the Octavia reconfigure.

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts octavia-reconfigure.yml
   ```

Spare Pools

The Octavia driver provides support for creating spare pools of the HAProxy software installed in VMs. This means instead of creating a new load balancer when loads increase, create new load balancer calls will pull a load balancer from the spare pool. The spare pools feature consumes resources, therefore the load balancers in the spares pool has been set to 0, which is the default and also disables the feature.

Reasons to enable a load balancing spare pool in HPE Helion OpenStack

1. You expect a large number of load balancers to be provisioned all at once (puppet scripts, or ansible scripts) and you want them to come up quickly.

2. You want to reduce the wait time a customer has while requesting a new load balancer.

To increase the number of load balancers in your spares pool, edit the Octavia configuration files by uncommenting the `spare_amphora_pool_size` and adding the number of load balancers you would like to include in your spares pool.

```
# Pool size for the spare pool
# spare_amphora_pool_size = 0
```

⏲ Important

In HPE Helion OpenStack the spare pool cannot be used to speed up fail overs. If a load balancer fails in HPE Helion OpenStack, Octavia will always provision a new VM to replace that failed load balancer.

9.3.9.3   Managing Amphora

Octavia starts a separate VM for each load balancing function. These VMs are called amphora.

Updating the Cryptographic Certificates

Octavia uses two-way SSL encryption for communication between amphora and the control plane. Octavia keeps track of the certificates on the amphora and will automatically recycle them. The certificates on the control plane are valid for one year after installation of HPE Helion OpenStack.

You can check on the status of the certificate by logging into the controller node as root and running:

```
ardana > cd /opt/stack/service/octavia-SOME UUID/etc/certs/
openssl x509 -in client.pem  -text —noout
```

This prints the certificate out where you can check on the expiration dates.

To renew the certificates, reconfigure Octavia. Reconfiguring causes Octavia to automatically generate new certificates and deploy them to the controller hosts.

On the Cloud Lifecycle Manager execute octavia-reconfigure:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts octavia-reconfigure.yml
```

Accessing VM information in Nova

You can use `openstack project list` as an administrative user to obtain information about the tenant or project-id of the Octavia project. In the example below, the Octavia project has a project-id of `37fd6e4feac14741b6e75aba14aea833`.

```
ardana > openstack project list
```

```
+---------------------------------+------------------+
| ID                              | Name             |
+---------------------------------+------------------+
| 055071d8f25d450ea0b981ca67f7ccee | glance-swift     |
| 37fd6e4feac14741b6e75aba14aea833 | octavia          |
| 4b431ae087ef4bd285bc887da6405b12 | swift-monitor    |
| 8ecf2bb5754646ae97989ba6cba08607 | swift-dispersion |
| b6bd581f8d9a48e18c86008301d40b26 | services         |
| bfcada17189e4bc7b22a9072d663b52d | cinderinternal   |
| c410223059354dd19964063ef7d63eca | monitor          |
| d43bc229f513494189422d88709b7b73 | admin            |
| d5a80541ba324c54aeae58ac3de95f77 | demo             |
| ea6e039d973e4a58bbe42ee08eaf6a7a | backup           |
+---------------------------------+------------------+
```

You can then use `nova list --tenant <project-id>` to list the VMs for the Octavia tenant. Take particular note of the IP address on the OCTAVIA-MGMT-NET; in the example below it is `172.30.1.11`. For additional nova command-line options see *Section 9.3.9.5, "For More Information"*.

```
ardana > nova list --tenant 37fd6e4feac14741b6e75aba14aea833
+--------------------------------------+--------------------------------------------
+---------------------------------+--------+------------+-------------
+---------------------------------------------+
| ID                                   | Name                                        |
 Tenant ID                        | Status | Task State | Power State | Networks
                                  |
+--------------------------------------+--------------------------------------------
+---------------------------------+--------+------------+-------------
+---------------------------------------------+
| 1ed8f651-de31-4208-81c5-817363818596 | amphora-1c3a4598-5489-48ea-8b9c-60c821269e4c |
 37fd6e4feac14741b6e75aba14aea833 | ACTIVE | -          | Running     | private=10.0.0.4;
 OCTAVIA-MGMT-NET=172.30.1.11 |
+--------------------------------------+--------------------------------------------
+---------------------------------+--------+------------+-------------
+---------------------------------------------+
```

🔔 Important

The Amphora VMs do not have SSH or any other access. In the rare case that there is a problem with the underlying load balancer the whole amphora will need to be replaced.

Initiating Failover of an Amphora VM

Under normal operations Octavia will monitor the health of the amphora constantly and automatically fail them over if there are any issues. This helps to minimize any potential downtime for load balancer users. There are, however, a few cases a failover needs to be initiated manually:

1. The Loadbalancer has become unresponsive and Octavia has not detected an error.

2. A new image has become available and existing load balancers need to start using the new image.

3. The cryptographic certificates to control and/or the HMAC password to verify Health information of the amphora have been compromised.

To minimize the impact for end users we will keep the existing load balancer working until shortly before the new one has been provisioned. There will be a short interruption for the load balancing service so keep that in mind when scheduling the failovers. To achieve that follow these steps (assuming the management ip from the previous step):

1. Assign the IP to a SHELL variable for better readability.

```
ardana > export MGM_IP=172.30.1.11
```

2. Identify the port of the vm on the management network.

```
ardana > neutron port-list | grep $MGM_IP
| 0b0301b9-4ee8-4fb6-a47c-2690594173f4 |
        | fa:16:3e:d7:50:92 |
{"subnet_id": "3e0de487-e255-4fc3-84b8-60e08564c5b7", "ip_address": "172.30.1.11"} |
```

3. Disable the port to initiate a failover. Note the load balancer will still function but cannot be controlled any longer by Octavia.

> 🎓 Note
>
> Changes after disabling the port will result in errors.

```
ardana > neutron port-update --admin-state-up False 0b0301b9-4ee8-4fb6-
a47c-2690594173f4
```

```
Updated port: 0b0301b9-4ee8-4fb6-a47c-2690594173f4
```

4. You can check to see if the amphora failed over with `nova list --tenant <project-id>`. This may take some time and in some cases may need to be repeated several times. You can tell that the failover has been successful by the changed IP on the management network.

```
ardana > nova list --tenant 37fd6e4feac14741b6e75aba14aea833
+------------------------------------
+----------------------------------------------
+----------------------------------+--------+-----------+-------------
+----------------------------------------------+
| ID                                  | Name
   | Tenant ID                        | Status | Task State | Power State | Networks
                                      |
+------------------------------------
+----------------------------------------------
+----------------------------------+--------+-----------+-------------
+----------------------------------------------+
| 1ed8f651-de31-4208-81c5-817363818596 |
 amphora-1c3a4598-5489-48ea-8b9c-60c821269e4c | 37fd6e4feac14741b6e75aba14aea833 |
 ACTIVE | -            | Running     | private=10.0.0.4; OCTAVIA-MGMT-NET=172.30.1.12
  |
+------------------------------------
+----------------------------------------------
+----------------------------------+--------+-----------+-------------
+----------------------------------------------+
```

⛨ Warning

Do not issue too many failovers at once. In a big installation you might be tempted to initiate several failovers in parallel for instance to speed up an update of amphora images. This will put a strain on the nova service and depending on the size of your installation you might need to throttle the failover rate.

9.3.9.4 Removing load balancers

The following procedures demonstrate how to delete a load balancer that is in the `ERROR`, `PENDING_CREATE`, or `PENDING_DELETE` state.

PROCEDURE 9.1: MANUALLY DELETING LOAD BALANCERS CREATED WITH NEUTRON LBAASV2 (IN AN UPGRADE/ MIGRATION SCENARIO)

1. Query the Neutron service for the loadbalancer ID:

```
ardana > neutron lbaas-loadbalancer-list
neutron CLI is deprecated and will be removed in the future. Use openstack CLI
 instead.
+-----------------------------------+---------+--------------------------------
+--------------+---------------------+----------+
| id                                | name    | tenant_id
 | vip_address  | provisioning_status | provider |
+-----------------------------------+---------+--------------------------------
+--------------+---------------------+----------+
| 7be4e4ab-e9c6-4a57-b767-da9af5ba7405 | test-lb | d62a1510b0f54b5693566fb8afeb5e33
 | 192.168.1.10 | ERROR               | haproxy  |
+-----------------------------------+---------+--------------------------------
+--------------+---------------------+----------+
```

2. Connect to the neutron database:

```
mysql> use ovs_neutron
```

3. Get the pools and healthmonitors associated with the loadbalancer:

```
mysql> select id, healthmonitor_id, loadbalancer_id from lbaas_pools where
 loadbalancer_id = '7be4e4ab-e9c6-4a57-b767-da9af5ba7405';
+-----------------------------------+----------------------------------
+-----------------------------------+
| id                                | healthmonitor_id                  |
 loadbalancer_id                   |
+-----------------------------------+----------------------------------
+-----------------------------------+
| 26c0384b-fc76-4943-83e5-9de40dd1c78c | 323a3c4b-8083-41e1-b1d9-04e1fef1a331 |
 7be4e4ab-e9c6-4a57-b767-da9af5ba7405 |
+-----------------------------------+----------------------------------
+-----------------------------------+
```

4. Get the members associated with the pool:

```
mysql> select id, pool_id from lbaas_members where pool_id = '26c0384b-
fc76-4943-83e5-9de40dd1c78c';
+-----------------------------------+------------------------------------+
| id                                | pool_id                            |
+-----------------------------------+------------------------------------+
| 6730f6c1-634c-4371-9df5-1a880662acc9 | 26c0384b-fc76-4943-83e5-9de40dd1c78c |
| 06f0cfc9-379a-4e3d-ab31-cdba1580afc2 | 26c0384b-fc76-4943-83e5-9de40dd1c78c |
```

```
+-------------------------------------+-------------------------------------+
```

5. Delete the pool members:

```
mysql> delete from lbaas_members where id = '6730f6c1-634c-4371-9df5-1a880662acc9';
mysql> delete from lbaas_members where id = '06f0cfc9-379a-4e3d-ab31-cdba1580afc2';
```

6. Find and delete the listener associated with the loadbalancer:

```
mysql> select id, loadbalancer_id, default_pool_id from lbaas_listeners where
 loadbalancer_id = '7be4e4ab-e9c6-4a57-b767-da9af5ba7405';
+-------------------------------------+-------------------------------------
+-----------------------------------+
| id                                  | loadbalancer_id                     |
 default_pool_id                     |
+-------------------------------------+-------------------------------------
+-----------------------------------+
| 3283f589-8464-43b3-96e0-399377642e0a | 7be4e4ab-e9c6-4a57-b767-da9af5ba7405 |
 26c0384b-fc76-4943-83e5-9de40dd1c78c |
+-------------------------------------+-------------------------------------
+-----------------------------------+
mysql> delete from lbaas_listeners where id =
 '3283f589-8464-43b3-96e0-399377642e0a';
```

7. Delete the pool associated with the loadbalancer:

```
mysql> delete from lbaas_pools where id = '26c0384b-fc76-4943-83e5-9de40dd1c78c';
```

8. Delete the healthmonitor associated with the pool:

```
mysql> delete from lbaas_healthmonitors where id = '323a3c4b-8083-41e1-
b1d9-04e1fef1a331';
```

9. Delete the loadbalancer:

```
mysql> delete from lbaas_loadbalancer_statistics where loadbalancer_id = '7be4e4ab-
e9c6-4a57-b767-da9af5ba7405';
mysql> delete from lbaas_loadbalancers where id = '7be4e4ab-e9c6-4a57-b767-
da9af5ba7405';
```

PROCEDURE 9.2: MANUALLY DELETING LOAD BALANCERS CREATED WITH OCTAVIA

1. Query the Octavia service for the loadbalancer ID:

```
ardana > openstack loadbalancer list --column id --column name --column
 provisioning_status
```

```
+----------------------------------------+----------+---------------------+
| id                                     | name     | provisioning_status |
+----------------------------------------+----------+---------------------+
| d8ac085d-e077-4af2-b47a-bdec0c162928   | test-lb  | ERROR               |
+----------------------------------------+----------+---------------------+
```

2. Query the Octavia service for the amphora IDs (in this example we use `ACTIVE/STANDBY` topology with 1 spare Amphora):

```
ardana > openstack loadbalancer amphora list
+----------------------------------------+-----------------------------------------
+-----------+--------+---------------+-------------+
| id                                     | loadbalancer_id                         |
 status     | role   | lb_network_ip | ha_ip       |
+----------------------------------------+-----------------------------------------
+-----------+--------+---------------+-------------+
| 6dc66d41-e4b6-4c33-945d-563f8b26e675   | d8ac085d-e077-4af2-b47a-bdec0c162928 |
 ALLOCATED | BACKUP | 172.30.1.7    | 192.168.1.8 |
| 1b195602-3b14-4352-b355-5c4a70e200cf   | d8ac085d-e077-4af2-b47a-bdec0c162928 |
 ALLOCATED | MASTER | 172.30.1.6    | 192.168.1.8 |
| b2ee14df-8ac6-4bb0-a8d3-3f378dbc2509   | None                                    |
 READY     | None   | 172.30.1.20   | None        |
+----------------------------------------+-----------------------------------------
+-----------+--------+---------------+-------------+
```

3. Query the Octavia service for the loadbalancer pools:

```
ardana > openstack loadbalancer pool list
+----------------------------------------+-----------
+------------------------------+--------------------+----------+--------------
+----------------+
| id                                     | name      | project_id
   | provisioning_status | protocol | lb_algorithm | admin_state_up |
+----------------------------------------+-----------
+------------------------------+--------------------+----------+--------------
+----------------+
| 39c4c791-6e66-4dd5-9b80-14ea11152bb5   | test-pool |
 86fba765e67f430b83437f2f25225b65 | ACTIVE              | TCP      | ROUND_ROBIN  |
 True            |
+----------------------------------------+-----------
+------------------------------+--------------------+----------+--------------
+----------------+
```

4. Connect to the octavia database:

```
mysql> use octavia
```

5. Delete any listeners, pools, health monitors, and members from the load balancer:

```
mysql> delete from listener where load_balancer_id = 'd8ac085d-e077-4af2-b47a-
bdec0c162928';
mysql> delete from health_monitor where pool_id =
  '39c4c791-6e66-4dd5-9b80-14ea11152bb5';
mysql> delete from member where pool_id = '39c4c791-6e66-4dd5-9b80-14ea11152bb5';
mysql> delete from pool where load_balancer_id = 'd8ac085d-e077-4af2-b47a-
bdec0c162928';
```

6. Delete the amphora entries in the database:

```
mysql> delete from amphora_health where amphora_id = '6dc66d41-
e4b6-4c33-945d-563f8b26e675';
mysql> update amphora set status = 'DELETED' where id = '6dc66d41-
e4b6-4c33-945d-563f8b26e675';
mysql> delete from amphora_health where amphora_id = '1b195602-3b14-4352-
b355-5c4a70e200cf';
mysql> update amphora set status = 'DELETED' where id = '1b195602-3b14-4352-
b355-5c4a70e200cf';
```

7. Delete the load balancer instance:

```
mysql> update load_balancer set provisioning_status = 'DELETED' where id =
  'd8ac085d-e077-4af2-b47a-bdec0c162928';
```

8. The following script automates the above steps:

```
#!/bin/bash

if (( $# != 1 )); then
  echo "Please specify a loadbalancer ID"
  exit 1
fi

LB_ID=$1

set -u -e -x

readarray -t AMPHORAE < <(openstack loadbalancer amphora list \
  --format value \
  --column id \
  --column loadbalancer_id \
  | grep ${LB_ID} \
  | cut -d ' ' -f 1)
```

```
readarray -t POOLS < <(openstack loadbalancer show ${LB_ID} \
  --format value \
  --column pools)

mysql octavia --execute "delete from listener where load_balancer_id = '${LB_ID}';"
for p in "${POOLS[@]}"; do
  mysql octavia --execute "delete from health_monitor where pool_id = '${p}';"
  mysql octavia --execute "delete from member where pool_id = '${p}';"
done
mysql octavia --execute "delete from pool where load_balancer_id = '${LB_ID}';"
for a in "${AMPHORAE[@]}"; do
  mysql octavia --execute "delete from amphora_health where amphora_id = '${a}';"
  mysql octavia --execute "update amphora set status = 'DELETED' where id = '${a}';"
done
mysql octavia --execute "update load_balancer set provisioning_status = 'DELETED'
 where id = '${LB_ID}';"
```

9.3.9.5 For More Information

For more information on the Nova command-line client, see the OpenStack Compute command-line client (http://docs.openstack.org/cli-reference/nova.html) ↗ guide.

For more information on Octavia terminology, see the OpenStack Octavia Glossary (http://docs.octavia.io/review/master/main/glossary.html) ↗

9.3.10 Role-based Access Control in Neutron

This topic explains how to achieve more granular access control for your Neutron networks.

Previously in HPE Helion OpenStack, a network object was either private to a project or could be used by all projects. If the network's shared attribute was True, then the network could be used by every project in the cloud. If false, only the members of the owning project could use it. There was no way for the network to be shared by only a subset of the projects.

Neutron Role Based Access Control (RBAC) solves this problem for networks. Now the network owner can create RBAC policies that give network access to target projects. Members of a targeted project can use the network named in the RBAC policy the same way as if the network was owned by the project. Constraints are described in the section *Section 9.3.10.10, "Limitations"*.

With RBAC you are able to let another tenant use a network that you created, but as the owner of the network, you need to create the subnet and the router for the network.

9.3.10.1 Creating a Network

```
ardana > openstack network create demo-net
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2018-07-25T17:43:59Z                 |
| description               |                                      |
| dns_domain                |                                      |
| id                        | 9c801954-ec7f-4a65-82f8-e313120aabc4 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| is_vlan_transparent       | None                                 |
| mtu                       | 1450                                 |
| name                      | demo-net                             |
| port_security_enabled     | False                                |
| project_id                | cb67c79e25a84e328326d186bf703e1b     |
| provider:network_type     | vxlan                                |
| provider:physical_network | None                                 |
| provider:segmentation_id  | 1009                                 |
| qos_policy_id             | None                                 |
| revision_number           | 2                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| updated_at                | 2018-07-25T17:43:59Z                 |
+---------------------------+--------------------------------------+
```

9.3.10.2 Creating an RBAC Policy

Here we will create an RBAC policy where a member of the project called 'demo' will share the network with members of project 'demo2'

To create the RBAC policy, run:

```
ardana > openstack network rbac create  --target-project DEMO2-PROJECT-ID --type network
 --action access_as_shared demo-net
```

Here is an example where the DEMO2-PROJECT-ID is 5a582af8b44b422fafcd4545bd2b7eb5

```
ardana > openstack network rbac create --target-tenant 5a582af8b44b422fafcd4545bd2b7eb5 \
  --type network --action access_as_shared demo-net
```

9.3.10.3   Listing RBACs

To list all the RBAC rules/policies, execute:

```
ardana > openstack network rbac list
+-------------------------------------+-------------
+-------------------------------------+
| ID                                  | Object Type | Object ID
   |
+-------------------------------------+-------------
+-------------------------------------+
| 0fdec7f0-9b94-42b4-a4cd-b291d04282c1 | network     | 7cd94877-4276-488d-
b682-7328fc85d721 |
+-------------------------------------+-------------
+-------------------------------------+
```

9.3.10.4   Listing the Attributes of an RBAC

To see the attributes of a specific RBAC policy, run

```
ardana > openstack network rbac show POLICY-ID
```

For example:

```
ardana > openstack network rbac show 0fd89dcb-9809-4a5e-adc1-39dd676cb386
```

Here is the output:

```
+--------------+-------------------------------------+
| Field        | Value                               |
+--------------+-------------------------------------+
| action       | access_as_shared                    |
| id           | 0fd89dcb-9809-4a5e-adc1-39dd676cb386 |
| object_id    | c3d55c21-d8c9-4ee5-944b-560b7e0ea33b |
| object_type  | network                             |
| target_tenant | 5a582af8b44b422fafcd4545bd2b7eb5    |
| tenant_id    | 75eb5efae5764682bca2fede6f4d8c6f    |
+--------------+-------------------------------------+
```

9.3.10.5   Deleting an RBAC Policy

To delete an RBAC policy, run `openstack network rbac delete` passing the policy id:

```
ardana > openstack network rbac delete POLICY-ID
```

For example:

```
ardana > openstack network rbac delete 0fd89dcb-9809-4a5e-adc1-39dd676cb386
```

Here is the output:

```
Deleted rbac_policy: 0fd89dcb-9809-4a5e-adc1-39dd676cb386
```

9.3.10.6   Sharing a Network with All Tenants

Either the administrator or the network owner can make a network shareable by all tenants.

The administrator can make a tenant's network shareable by all tenants. To make the network `demo-shareall-net` accessible by all tenants in the cloud:

To share a network with all tenants:

1. Get a list of all projects

   ```
   ardana > ~/service.osrc
   ardana > openstack project list
   ```

   which produces the list:

   ```
   +----------------------------------+-----------------+
   | ID                               | Name            |
   +----------------------------------+-----------------+
   | 1be57778b61645a7a1c07ca0ac488f9e | demo            |
   | 5346676226274cd2b3e3862c2d5ceadd | admin           |
   | 749a557b2b9c482ca047e8f4abf348cd | swift-monitor   |
   | 8284a83df4df429fb04996c59f9a314b | swift-dispersion |
   | c7a74026ed8d4345a48a3860048dcb39 | demo-sharee     |
   | e771266d937440828372090c4f99a995 | glance-swift    |
   | f43fb69f107b4b109d22431766b85f20 | services        |
   +----------------------------------+-----------------+
   ```

2. Get a list of networks:

   ```
   ardana > openstack network list
   ```

This produces the following list:

```
+---------------------------------------+-------------------
+---------------------------------------------------+
| id                                    | name            | subnets
         |
+---------------------------------------+-------------------
+---------------------------------------------------+
| f50f9a63-c048-444d-939d-370cb0af1387  | ext-net         | ef3873db-fc7a-4085-8454-5566fb5578ea
 172.31.0.0/16 |
| 9fb676f5-137e-4646-ac6e-db675a885fd3  | demo-net        | 18fb0b77-fc8b-4f8d-9172-ee47869f92cc
 10.0.1.0/24   |
| 8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e  | demo-shareall-net | 2bbc85a9-3ffe-464c-944b-2476c7804877
 10.0.250.0/24 |
| 73f946ee-bd2b-42e9-87e4-87f19edd0682  | demo-share-subset | c088b0ef-f541-42a7-b4b9-6ef3c9921e44
 10.0.2.0/24   |
+---------------------------------------+-------------------
+---------------------------------------------------+
```

3. Set the network you want to share to a shared value of True:

```
ardana > openstack network set --share 8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e
```

You should see the following output:

```
Updated network: 8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e
```

4. Check the attributes of that network by running the following command using the ID of the network in question:

```
ardana > openstack network show 8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e
```

The output will look like this:

```
+--------------------------+------------------------------------+
| Field                    | Value                              |
+--------------------------+------------------------------------+
| admin_state_up           | UP                                 |
| availability_zone_hints  |                                    |
| availability_zones       |                                    |
| created_at               | 2018-07-25T17:43:59Z               |
| description              |                                    |
| dns_domain               |                                    |
| id                       | 8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e |
| ipv4_address_scope       | None                               |
| ipv6_address_scope       | None                               |
| is_default               | None                               |
| is_vlan_transparent      | None                               |
| mtu                      | 1450                               |
```

```
| name                      | demo-net                             |
| port_security_enabled     | False                                |
| project_id                | cb67c79e25a84e328326d186bf703e1b     |
| provider:network_type     | vxlan                                |
| provider:physical_network | None                                 |
| provider:segmentation_id  | 1009                                 |
| qos_policy_id             | None                                 |
| revision_number           | 2                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| updated_at                | 2018-07-25T17:43:59Z                 |
+---------------------------+--------------------------------------+
```

5. As the owner of the `demo-shareall-net` network, view the RBAC attributes for
   `demo-shareall-net` (`id=8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e`) by first getting
   an RBAC list:

```
ardana > echo $OS_USERNAME ; echo $OS_PROJECT_NAME
demo
demo
ardana > openstack network rbac list
```

This produces the list:

```
+--------------------------------------+--------------------------------------+
| id                                   | object_id                            |
+--------------------------------------+--------------------------------------+
| ...                                                                         |
| 3e078293-f55d-461c-9a0b-67b5dae321e8 | 8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e |
+--------------------------------------+--------------------------------------+
```

6. View the RBAC information:

```
ardana > openstack network rbac show 3e078293-f55d-461c-9a0b-67b5dae321e8

+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| action        | access_as_shared                     |
| id            | 3e078293-f55d-461c-9a0b-67b5dae321e8 |
| object_id     | 8eada4f7-83cf-40ba-aa8c-5bf7d87cca8e |
| object_type   | network                              |
| target_tenant | *                                    |
```

```
| tenant_id      | 1be57778b61645a7a1c07ca0ac488f9e      |
+---------------+-----------------------------------+
```

7. With network RBAC, the owner of the network can also make the network shareable by all tenants. First create the network:

```
ardana > echo $OS_PROJECT_NAME ; echo $OS_USERNAME
demo
demo
ardana > openstack network create test-net
```

The network is created:

```
+--------------------------+--------------------------------------+
| Field                    | Value                                |
+--------------------------+--------------------------------------+
| admin_state_up           | UP                                   |
| availability_zone_hints  |                                      |
| availability_zones       |                                      |
| created_at               | 2018-07-25T18:04:25Z                 |
| description              |                                      |
| dns_domain               |                                      |
| id                       | a4bd7c3a-818f-4431-8cdb-fedf7ff40f73 |
| ipv4_address_scope       | None                                 |
| ipv6_address_scope       | None                                 |
| is_default               | False                                |
| is_vlan_transparent      | None                                 |
| mtu                      | 1450                                 |
| name                     | test-net                             |
| port_security_enabled    | False                                |
| project_id               | cb67c79e25a84e328326d186bf703e1b     |
| provider:network_type    | vxlan                                |
| provider:physical_network | None                                |
| provider:segmentation_id | 1073                                 |
| qos_policy_id            | None                                 |
| revision_number          | 2                                    |
| router:external          | Internal                             |
| segments                 | None                                 |
| shared                   | False                                |
| status                   | ACTIVE                               |
| subnets                  |                                      |
| tags                     |                                      |
| updated_at               | 2018-07-25T18:04:25Z                 |
+--------------------------+--------------------------------------+
```

8. Create the RBAC. It is important that the asterisk is surrounded by single-quotes to prevent the shell from expanding it to all files in the current directory.

```
ardana > openstack network rbac create --type network \
  --action access_as_shared --target-project '*' test-net
```

Here are the resulting RBAC attributes:

```
+---------------+-------------------------------------+
| Field         | Value                               |
+---------------+-------------------------------------+
| action        | access_as_shared                    |
| id            | 0b797cc6-debc-48a1-bf9d-d294b077d0d9 |
| object_id     | a4bd7c3a-818f-4431-8cdb-fedf7ff40f73 |
| object_type   | network                             |
| target_tenant | *                                   |
| tenant_id     | 1be57778b61645a7a1c07ca0ac488f9e    |
+---------------+-------------------------------------+
```

9.3.10.7   Target Project (demo2) View of Networks and Subnets

Note that the owner of the network and subnet is not the tenant named `demo2`. Both the network and subnet are owned by tenant `demo`. `Demo2` members cannot create subnets of the network. They also cannot modify or delete subnets owned by `demo`.

As the tenant `demo2`, you can get a list of neutron networks:

```
ardana > openstack network list
```

```
+--------------------------------------+-----------+
+-----------------------------------------------+
| id                                   | name      | subnets
     |
+--------------------------------------+-----------+
+-----------------------------------------------+
| f60f3896-2854-4f20-b03f-584a0dcce7a6 | ext-net   | 50e39973-b2e3-466b-81c9-31f4d83d990b
     |
| c3d55c21-d8c9-4ee5-944b-560b7e0ea33b | demo-net  | d9b765da-45eb-4543-be96-1b69a00a2556
 10.0.1.0/24 |
   ...
+--------------------------------------+-----------+
+-----------------------------------------------+
```

And get a list of subnets:

```
ardana > openstack subnet list --network c3d55c21-d8c9-4ee5-944b-560b7e0ea33b
```

```
+--------------------------------------+--------+--------------------------------------+--------------+
| ID                                   | Name   | Network                              | Subnet       |
+--------------------------------------+--------+--------------------------------------+--------------+
| a806f28b-ad66-47f1-b280-a1caa9beb832 | ext-net | c3d55c21-d8c9-4ee5-944b-560b7e0ea33b | 10.0.1.0/24 |
+--------------------------------------+--------+--------------------------------------+--------------+
```

To show details of the subnet:

```
ardana > openstack subnet show d9b765da-45eb-4543-be96-1b69a00a2556
```

```
+-------------------+---------------------------------------------+
| Field             | Value                                       |
+-------------------+---------------------------------------------+
| allocation_pools  | {"start": "10.0.1.2", "end": "10.0.1.254"}  |
| cidr              | 10.0.1.0/24                                  |
| dns_nameservers   |                                             |
| enable_dhcp       | True                                        |
| gateway_ip        | 10.0.1.1                                    |
| host_routes       |                                             |
| id                | d9b765da-45eb-4543-be96-1b69a00a2556        |
| ip_version        | 4                                           |
| ipv6_address_mode |                                             |
| ipv6_ra_mode      |                                             |
| name              | sb-demo-net                                 |
| network_id        | c3d55c21-d8c9-4ee5-944b-560b7e0ea33b        |
| subnetpool_id     |                                             |
| tenant_id         | 75eb5efae5764682bca2fede6f4d8c6f            |
+-------------------+---------------------------------------------+
```

9.3.10.8 Target Project: Creating a Port Using demo-net

The owner of the port is demo2. Members of the network owner project (demo) will not see this port.

Running the following command:

```
ardana > openstack port create c3d55c21-d8c9-4ee5-944b-560b7e0ea33b
```

Creates a new port:

```
+----------------------
+------------------------------------------------------------------------------------------------+
| Field                | Value
                 |
+----------------------
+------------------------------------------------------------------------------------------------+
| admin_state_up       | True
                 |
```

```
| allowed_address_pairs |
                       |
| binding:vnic_type    | normal
                       |
| device_id            |
                       |
| device_owner         |
                       |
| dns_assignment       | {"hostname": "host-10-0-1-10", "ip_address": "10.0.1.10", "fqdn":
 "host-10-0-1-10.openstacklocal."} |
| dns_name             |
                       |
| fixed_ips            | {"subnet_id": "d9b765da-45eb-4543-be96-1b69a00a2556", "ip_address": "10.0.1.10"}
                       |
| id                   | 03ef2dce-20dc-47e5-9160-942320b4e503
                       |
| mac_address          | fa:16:3e:27:8d:ca
                       |
| name                 |
                       |
| network_id           | c3d55c21-d8c9-4ee5-944b-560b7e0ea33b
                       |
| security_groups      | 275802d0-33cb-4796-9e57-03d8ddd29b94
                       |
| status               | DOWN
                       |
| tenant_id            | 5a582af8b44b422fafcd4545bd2b7eb5
                       |
+----------------------
+-----------------------------------------------------------------------------------------------------+
```

9.3.10.9   Target Project Booting a VM Using Demo-Net

Here the tenant `demo2` boots a VM that uses the `demo-net` shared network:

```
ardana > openstack server create --flavor 1 --image $OS_IMAGE --nic net-id=c3d55c21-
d8c9-4ee5-944b-560b7e0ea33b demo2-vm-using-demo-net-nic
```

```
+------------------------------------+----------------------------------------------+
| Property                           | Value                                        |
+------------------------------------+----------------------------------------------+
| OS-EXT-AZ:availability_zone        |                                              |
| OS-EXT-STS:power_state             | 0                                            |
| OS-EXT-STS:task_state              | scheduling                                   |
| OS-EXT-STS:vm_state                | building                                     |
| OS-SRV-USG:launched_at             | -                                            |
| OS-SRV-USG:terminated_at           | -                                            |
| accessIPv4                         |                                              |
| accessIPv6                         |                                              |
| adminPass                          | sS9uSv9PT79F                                 |
| config_drive                       |                                              |
| created                            | 2016-01-04T19:23:24Z                         |
| flavor                             | m1.tiny (1)                                  |
| hostId                             |                                              |
```

```
| id                                | 3a4dc44a-027b-45e9-acf8-054a7c2dca2a      |
| image                             | cirros-0.3.3-x86_64 (6ae23432-8636-4e...1efc5) |
| key_name                          | -                                         |
| metadata                          | {}                                        |
| name                              | demo2-vm-using-demo-net-nic               |
| os-extended-volumes:volumes_attached | []                                     |
| progress                          | 0                                         |
| security_groups                   | default                                   |
| status                            | BUILD                                     |
| tenant_id                         | 5a582af8b44b422fafcd4545bd2b7eb5          |
| updated                           | 2016-01-04T19:23:24Z                      |
| user_id                           | a0e6427b036344fdb47162987cb0cee5          |
+-----------------------------------+-------------------------------------------+
```

Run openstack server list:

```
ardana > openstack server list
```

See the VM running:

```
+------------------+----------------------------+--------+------------+-------------
+-------------------+
| ID               | Name                       | Status | Task State | Power State | Networks
        |
+------------------+----------------------------+--------+------------+-------------
+-------------------+
| 3a4dc...a7c2dca2a | demo2-vm-using-demo-net-nic | ACTIVE | -          | Running     | demo-
net=10.0.1.11 |
+------------------+----------------------------+--------+------------+-------------
+-------------------+
```

Run openstack port list:

```
ardana > neutron port-list --device-id 3a4dc44a-027b-45e9-acf8-054a7c2dca2a
```

View the subnet:

```
+-------------------+------+------------------
+-------------------------------------------------------------------+
| id                | name | mac_address      | fixed_ips
                  |
+-------------------+------+------------------
+-------------------------------------------------------------------+
| 7d14ef8b-9...80348f |      | fa:16:3e:75:32:8e | {"subnet_id": "d9b765da-45...00a2556",
 "ip_address": "10.0.1.11"} |
+-------------------+------+------------------
+-------------------------------------------------------------------+
```

Run neutron port-show:

```
ardana > openstack port show 7d14ef8b-9d48-4310-8c02-00c74d80348f
```

```
+---------------------
+----------------------------------------------------------------------------------------------+
```

```
| Field                 | Value
                |
+----------------------
+-----------------------------------------------------------------------------------------------------+
| admin_state_up        | True
                |
| allowed_address_pairs |
                |
| binding:vnic_type     | normal
                |
| device_id             | 3a4dc44a-027b-45e9-acf8-054a7c2dca2a
                |
| device_owner          | compute:None
                |
| dns_assignment        | {"hostname": "host-10-0-1-11", "ip_address": "10.0.1.11", "fqdn":
  "host-10-0-1-11.openstacklocal."} |
| dns_name              |
                |
| extra_dhcp_opts       |
                |
| fixed_ips             | {"subnet_id": "d9b765da-45eb-4543-be96-1b69a00a2556", "ip_address": "10.0.1.11"}
                |
| id                    | 7d14ef8b-9d48-4310-8c02-00c74d80348f
                |
| mac_address           | fa:16:3e:75:32:8e
                |
| name                  |
                |
| network_id            | c3d55c21-d8c9-4ee5-944b-560b7e0ea33b
                |
| security_groups       | 275802d0-33cb-4796-9e57-03d8ddd29b94
                |
| status                | ACTIVE
                |
| tenant_id             | 5a582af8b44b422fafcd4545bd2b7eb5
                |
+----------------------
+-----------------------------------------------------------------------------------------------------+
```

9.3.10.10 Limitations

Note the following limitations of RBAC in Neutron.

- Neutron network is the only supported RBAC Neutron object type.

- The "access_as_external" action is not supported – even though it is listed as a valid action by python-neutronclient.

- The neutron-api server will not accept action value of 'access_as_external'. The `access_as_external` definition is not found in the specs.

- The target project users cannot create, modify, or delete subnets on networks that have RBAC policies.

- The subnet of a network that has an RBAC policy cannot be added as an interface of a target tenant's router. For example, the command `neutron router-interface-add tgt-tenant-router <sb-demo-net uuid>` will error out.

- The security group rules on the network owner do not apply to other projects that can use the network.

- A user in target project can boot up VMs using a VNIC using the shared network. The user of the target project can assign a floating IP (FIP) to the VM. The target project must have SG rules that allows SSH and/or ICMP for VM connectivity.

- Neutron RBAC creation and management are currently not supported in Horizon. For now, the Neutron CLI has to be used to manage RBAC rules.

- A RBAC rule tells Neutron whether a tenant can access a network (Allow). Currently there is no DENY action.

- Port creation on a shared network fails if `--fixed-ip` is specified in the `neutron port-create` command.

9.3.11 Configuring Maximum Transmission Units in Neutron

This topic explains how you can configure MTUs, what to look out for, and the results and implications of changing the default MTU settings. It is important to note that every network within a network group will have the same MTU.

⬡ Warning

An MTU change will not affect existing networks that have had VMs created on them. It will only take effect on new networks created after the reconfiguration process.

9.3.11.1 Overview

A Maximum Transmission Unit, or MTU is the maximum packet size (in bytes) that a network device can or is configured to handle. There are a number of places in your cloud where MTU configuration is relevant: the physical interfaces managed and configured by HPE Helion OpenStack, the virtual interfaces created by Neutron and Nova for Neutron networking, and the interfaces inside the VMs.

HPE Helion OpenStack-managed physical interfaces

HPE Helion OpenStack-managed physical interfaces include the physical interfaces and the bonds, bridges, and VLANs created on top of them. The MTU for these interfaces is configured via the 'mtu' property of a network group. Because multiple network groups can be mapped to one physical interface, there may have to be some resolution of differing MTUs between the untagged and tagged VLANs on the same physical interface. For instance, if one untagged VLAN, vlan101 (with an MTU of 1500) and a tagged VLAN vlan201 (with an MTU of 9000) are both on one interface (eth0), this means that eth0 can handle 1500, but the VLAN interface which is created on top of eth0 (that is, `vlan201@eth0`) wants 9000. However, vlan201 cannot have a higher MTU than eth0, so vlan201 will be limited to 1500 when it is brought up, and fragmentation will result.

In general, a VLAN interface MTU must be lower than or equal to the base device MTU. If they are different, as in the case above, the MTU of eth0 can be overridden and raised to 9000, but in any case the discrepancy will have to be reconciled.

Neutron/Nova interfaces

Neutron/Nova interfaces include the virtual devices created by Neutron and Nova during the normal process of realizing a Neutron network/router and booting a VM on it (qr-*, qg-*, tap-*, qvo-*, qvb-*, etc.). There is currently no support in Neutron/Nova for per-network MTUs in which every interface along the path for a particular Neutron network has the correct MTU for that network. There is, however, support for globally changing the MTU of devices created by Neutron/Nova (see network_device_mtu below). This means that if you want to enable jumbo frames for any set of VMs, you will have to enable it for all your VMs. You cannot just enable them for a particular Neutron network.

VM interfaces

VMs typically get their MTU via DHCP advertisement, which means that the dnsmasq processes spawned by the neutron-dhcp-agent actually advertise a particular MTU to the VMs. In HPE Helion OpenStack 8, the DHCP server advertises to all VMS a 1400 MTU via a forced setting in dnsmasq-neutron.conf. This is suboptimal for every network type (vxlan, flat, vlan, etc) but it does prevent fragmentation of a VM's packets due to encapsulation.

For instance, if you set the new *-mtu configuration options to a default of 1500 and create a VXLAN network, it will be given an MTU of 1450 (with the remaining 50 bytes used by the VXLAN encapsulation header) and will advertise a 1450 MTU to any VM booted on that network. If you create a provider VLAN network, it will have an MTU of 1500 and will advertise 1500 to booted VMs on the network. It should be noted that this default starting point for MTU calculation and advertisement is also global, meaning you cannot have an MTU of 8950 on one VXLAN network and 1450 on another. However, you can have provider physical networks with different MTUs by using the physical_network_mtus config option, but Nova still requires a global MTU option for the interfaces it creates, thus you cannot really take advantage of that configuration option.

9.3.11.2 Network settings in the input model

MTU can be set as an attribute of a network group in network_groups.yml. Note that this applies only to KVM. That setting means that every network in the network group will be assigned the specified MTU. The MTU value must be set individually for each network group. For example:

```
network-groups:
      - name: GUEST
      mtu: 9000
      ...

      - name: EXTERNAL-API
      mtu: 9000
      ...

      - name: EXTERNAL-VM
      mtu: 9000
      ...
```

9.3.11.3 Infrastructure support for jumbo frames

If you want to use jumbo frames, or frames with an MTU of 9000 or more, the physical switches and routers that make up the infrastructure of the HPE Helion OpenStack installation must be configured to support them. To realize the advantages, generally all devices in the same broadcast domain must have the same MTU.

If you want to configure jumbo frames on compute and controller nodes, then all switches joining the compute and controller nodes must have jumbo frames enabled. Similarly, the "infrastructure gateway" through which the external VM network flows, commonly known as the default route for the external VM VLAN, must also have the same MTU configured.

You can also consider anything in the same broadcast domain to be anything in the same VLAN or anything in the same IP subnet.

9.3.11.4 Enabling end-to-end jumbo frames for a VM

1. Add an 'mtu' attribute to all the network groups in your model. Note that adding the MTU for the network groups will only affect the configuration for physical network interfaces. To add the mtu attribute, find the YAML file that contains your network-groups entry. We will assume it is network_groups.yml, unless you have changed it. Whatever the file is named, it will be found in ~/openstack/my_cloud/definition/data/.
   To edit these files, begin by checking out the **site** branch on the Cloud Lifecycle Manager node. You may already be on that branch. If so, you will remain there.

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git checkout site
   ```

   Then begin editing the files. In network_groups.yml, add mtu: 9000

   ```
   network-groups:
           - name: GUEST
           hostname-suffix: guest
           mtu: 9000
           tags:
           - neutron.networks.vxlan
   ```

   This will set the physical interface managed by HPE Helion OpenStack 8 that has the GUEST network group tag assigned to it. This can be found in the interfaces_set.yml file under the interface-models section.

2. Next, edit the layer 2 agent config file, ml2_conf.ini.j2, found in ~/openstack/my_cloud/ config/neutron/ to set the path_mtu to 0, this ensures that global_physnet_mtu is used.

```
[ml2]
...
path_mtu = 0
```

3. Next, edit neutron.conf.j2 found in ~/openstack/my_cloud/config/neutron/ to set advertise_mtu (to true) and global_physnet_mtu to 9000 under [DEFAULT]:

```
[DEFAULT]
...
advertise_mtu = True
global_physnet_mtu = 9000
```

This allows Neutron to advertise the optimal MTU to instances (based upon global_physnet_mtu minus the encapsulation size).

4. Next, remove the "dhcp-option-force=26,1400" line from `~/openstack/my_cloud/ config/neutron/dnsmasq-neutron.conf.j2`.

5. OvS will set `br-int` to the value of the lowest physical interface. If you are using Jumbo frames on some of your networks, `br-int` on the controllers may be set to 1500 instead of 9000. Work around this condition by running:

```
ovs-vsctl set int br-int mtu_request=9000
```

6. Commit your changes

```
ardana > git add -A
ardana > git commit -m "your commit message goes here in quotes"
```

7. If HPE Helion OpenStack has not been deployed yet, do normal deployment and skip to step 8.

8. Assuming it has been deployed already, continue here:
   Run the configuration processor:

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

and ready the deployment:

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

Then run the network_interface-reconfigure.yml playbook, changing directories first:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts network_interface-reconfigure.yml
```

Then run neutron-reconfigure.yml:

```
ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
```

Then nova-reconfigure.yml:

```
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

Note: adding/changing network-group mtu settings will likely require a network restart during network_interface-reconfigure.yml.

9. Follow the normal process for creating a Neutron network and booting a VM or two. In this example, if a VXLAN network is created and a VM is booted on it, the VM will have an MTU of 8950, with the remaining 50 bytes used by the VXLAN encapsulation header.

10. Test and verify that the VM can send and receive jumbo frames without fragmentation. You can use ping. For example, to test an MTU of 9000 using VXLAN:

```
ardana > ping —M do —s 8950 YOUR_VM_FLOATING_IP
```

Just substitute your actual floating IP address for the `YOUR_VM_FLOATING_IP`.

9.3.11.5   Enabling Optimal MTU Advertisement Feature

To enable the optimal MTU feature, follow these steps:

1. Edit `~/openstack/my_cloud/config/neutron/neutron.conf.j2` to **remove** advertise_mtu variable under [DEFAULT]

```
[DEFAULT]
...
advertise_mtu = False #remove this
```

2. Remove the `dhcp-option-force=26,1400` line from `~/openstack/my_cloud/config/neutron/dnsmasq-neutron.conf.j2`

3. If HPE Helion OpenStack has already been deployed, follow the remaining steps, otherwise follow the normal deployment procedures.

4. Commit your changes

```
ardana > git add -A
ardana > git commit -m "your commit message goes here in quotes"
```

5. Run the configuration processor:

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Run ready deployment:

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the `network_interface-reconfigure.yml` playbook, changing directories first:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts network_interface-reconfigure.yml
```

8. Run neutron-reconfigure.yml:

```
ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
```

🔔 Important

If you are upgrading an existing deployment, attention must be paid to avoid creating MTU mismatch between network interfaces in preexisting VMs and that of VMs created after upgrade. If you do have an MTU mismatch, then the new VMs (having interface with 1500 minus the underlay protocol overhead) will not be able to have L2 connectivity with preexisting VMs (with 1400 MTU due to dhcp-option-force).

9.3.12 Improve Network Peformance with Isolated Metadata Settings

In HPE Helion OpenStack, Neutron currently sets `enable_isolated_metadata = True` by default in `dhcp_agent.ini` because several services require isolated networks (Neutron networks without a router). This has the effect of spawning a neutron-ns-metadata-proxy process on one of the controller nodes for every active Neutron network.

In environments that create many Neutron networks, these extra `neutron-ns-metadata-proxy` processes can quickly eat up a lot of memory on the controllers, which does not scale well.

For deployments that do not require isolated metadata (that is, they do not require the Platform Services and will always create networks with an attached router), you can set `enable_isolated_metadata = False` in dhcp_agent.ini to reduce Neutron memory usage on controllers, allowing a greater number of active Neutron networks.

Note that the `dhcp_agent.ini.j2` template is found in `~/openstack/my_cloud/config/neutron` on the Cloud Lifecycle Manager node. The edit can be made there and the standard deployment can be run if this is install time. In a deployed cloud, run the Neutron reconfiguration procedure outlined here:

1. First check out the site branch:

   ```
   ardana > cd ~/openstack/my_cloud/config/neutron
   ardana > git checkout site
   ```

2. Edit the `dhcp_agent.ini.j2` file to change the `enable_isolated_metadata = {{ neutron_enable_isolated_metadata }}` line in the `[DEFAULT]` section to read:

   ```
   enable_isolated_metadata = False
   ```

3. Commit the file:

   ```
   ardana > git add -A
   ardana > git commit -m "your commit message goes here in quotes"
   ```

4. Run the `ready-deployment.yml` playbook from `~/openstack/ardana/ansible`:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

5. Then run the `neutron-reconfigure.yml` playbook, changing directories first:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
   ```

9.3.13 Moving from DVR deployments to non_DVR

If you have an older deployment of HPE Helion OpenStack which is using DVR as a default and you are attempting to move to non_DVR, make sure you follow these steps:

1. Remove all your existing DVR routers and their workloads. Make sure to remove interfaces, floating ips and gateways, if applicable.

   ```
   neutron router-interface-delete ROUTER-NAME SUBNET-NAME/SUBNET-ID
   neutron floatingip-disassociate FLOATINGIP-ID PRIVATE-PORT-ID
   neutron router-gateway-clear ROUTER-NAME -NET-NAME/EXT-NET-ID
   ```

2. Then delete the router.

   ```
   neutron router-delete ROUTER-NAME
   ```

3. Before you create any non_DVR router make sure that l3-agents and metadata-agents are not running in any compute host. You can run the command `neutron agent-list` to see if there are any neutron-l3-agent running in any compute-host in your deployment.
   You must disable neutron-l3-agent and neutron-metadata-agent on every compute host by running the following commands:

   ```
   ardana > neutron agent-list
   +--------------------------------------+----------------------
   +------------------------+------------------+-------+----------------
   +--------------------------+
   | id                                   | agent_type           | host
       | availability_zone | alive | admin_state_up | binary               |
   +--------------------------------------+----------------------
   +------------------------+------------------+-------+----------------
   +--------------------------+
   | 208f6aea-3d45-4b89-bf42-f45a51b05f29 | Loadbalancerv2 agent | ardana-cp1-comp0001-
   mgmt |                   | :-)   | True           | neutron-lbaasv2-agent    |
   | 810f0ae7-63aa-4ee3-952d-69837b4b2fe4 | L3 agent             | ardana-cp1-comp0001-
   mgmt | nova              | :-)   | True           | neutron-l3-agent         |
   | 89ac17ba-2f43-428a-98fa-b3698646543d | Metadata agent       | ardana-cp1-comp0001-
   mgmt |                   | :-)   | True           | neutron-metadata-agent   |
   | f602edce-1d2a-4c8a-ba56-fa41103d4e17 | Open vSwitch agent   | ardana-cp1-comp0001-
   mgmt |                   | :-)   | True           | neutron-openvswitch-agent |
   ...
   +--------------------------------------+----------------------
   +------------------------+------------------+-------+----------------
   +--------------------------+
   ```

```
$ neutron agent-update 810f0ae7-63aa-4ee3-952d-69837b4b2fe4 --admin-state-down
Updated agent: 810f0ae7-63aa-4ee3-952d-69837b4b2fe4

$ neutron agent-update 89ac17ba-2f43-428a-98fa-b3698646543d --admin-state-down
Updated agent: 89ac17ba-2f43-428a-98fa-b3698646543d
```

> ⚲ **Note**
>
> Only L3 and Metadata agents were disabled.

4. Once L3 and metadata neutron agents are stopped, follow steps 1 through 7 in the document *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 12 "Alternative Configurations", Section 12.2 "Configuring HPE Helion OpenStack without DVR"* and then run the `neutron-reconfigure.yml` playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
```

9.3.14 OVS-DPDK Support

HPE Helion OpenStack uses a version of Open vSwitch (OVS) that is built with the Data Plane Development Kit (DPDK) and includes a QEMU hypervisor which supports vhost-user.

The OVS-DPDK package modifes the OVS fast path, which is normally performed in kernel space, and allows it to run in userspace so there is no context switch to the kernel for processing network packets.

The EAL component of DPDK supports mapping the Network Interface Card (NIC) registers directly into userspace. The DPDK provides a Poll Mode Driver (PMD) that can access the NIC hardware from userspace and uses polling instead of interrupts to avoid the user to kernel transition.

The PMD maps the shared address space of the VM that is provided by the vhost-user capability of QEMU. The vhost-user mode causes Neutron to create a Unix domain socket that allows communication between the PMD and QEMU. The PMD uses this in order to acquire the file descriptors to the pre-allocated VM memory. This allows the PMD to directly access the VM memory space and perform a fast zero-copy of network packets directly into and out of the VMs virtio_net vring.

This yields performance improvements in the time it takes to process network packets.

9.3.14.1　Usage considerations

The target for a DPDK Open vSwitch is VM performance and VMs only run on compute nodes so the following considerations are compute node specific.

1. You are required to *Section 9.3.14.3, "Configuring Hugepages for DPDK in Neutron Networks"* in order to use DPDK with VMs. The memory to be used must be allocated at boot time so you must know beforehand how many VMs will be scheduled on a node. Also, for NUMA considerations, you want those hugepages on the same NUMA node as the NIC. A VM maps its entire address space into a hugepage.

2. For maximum performance you must reserve logical cores for DPDK poll mode driver (PMD) usage and for hypervisor (QEMU) usage. This keeps the Linux kernel from scheduling processes on those cores. The PMD threads will go to 100% cpu utilization since it uses polling of the hardware instead of interrupts. There will be at least 2 cores dedicated to PMD threads. Each VM will have a core dedicated to it although for less performance VMs can share cores.

3. VMs can use the virtio_net or the virtio_pmd drivers. There is also a PMD for an emulated e1000.

4. Only VMs that use hugepages can be sucessfully launched on a DPDK enabled NIC. If there is a need to support both DPDK and non-DPDK based VMs an additional port managed by the Linux kernel must exist.

5. OVS/DPDK does not support jumbo frames. Please review https://github.com/openvswitch/ovs/blob/branch-2.5/INSTALL.DPDK.md#restrictions ↗ for restrictions.

6. The Open vSwitch firewall driver in networking-ovs-dpdk repo is stateless and not a stateful one that would use iptables and conntrack. In the past, Neutron core has declined to pull in stateless type FW. https://bugs.launchpad.net/neutron/+bug/1531205 ↗ The native firewall driver is stateful, which is why conntrack was added to Open vSwitch. But this is not supported on DPDK and will not be until OVS 2.6.

9.3.14.2　For more information

See the following topics for more information:

9.3.14.3 Configuring Hugepages for DPDK in Neutron Networks

To take advantage of DPDK and its network performance enhancements, enable hugepages first.

With hugepages, physical RAM is reserved at boot time and dedicated to a virtual machine. Only that virtual machine and Open vSwitch can use this specifically allocated RAM. The host OS cannot access it. This memory is contiguous, and because of its larger size, reduces the number of entries in the memory map and number of times it must be read.

The hugepage reservation is made in `/etc/default/grub`, but this is handled by the Cloud Lifecycle Manager.

In addition to hugepages, to use DPDK, CPU isolation is required. This is achieved with the 'isolcups' command in `/etc/default/grub`, but this is also managed by the Cloud Lifecycle Manager using a new input model file.

The two new input model files introduced with this release to help you configure the necessary settings and persist them are:

- memory_models.yml (for hugepages)

- cpu_models.yml (for CPU isolation)

9.3.14.3.1 memory_models.yml

In this file you set your huge page size along with the number of such huge-page allocations.

```
---
product:
  version: 2

memory-models:
  - name: COMPUTE-MEMORY-NUMA
    default-huge-page-size: 1G
    huge-pages:
      - size: 1G
        count: 24
        numa-node: 0
      - size: 1G
        count: 24
        numa-node: 1
      - size: 1G
        count: 48
```

9.3.14.3.2   cpu_models.yml

```
---
  product:
    version: 2

  cpu-models:

    - name: COMPUTE-CPU
      assignments:
       - components:
           - nova-compute-kvm
         cpu:
           - processor-ids: 3-5,12-17
             role: vm

       - components:
           - openvswitch
         cpu:
           - processor-ids: 0
             role: eal
           - processor-ids: 1-2
             role: pmd
```

9.3.14.3.3   NUMA memory allocation

As mentioned above, the memory used for hugepages is locked down at boot time by an entry
in `/etc/default/grub`. As an admin, you can specify in the input model how to arrange this
memory on NUMA nodes. It can be spread across NUMA nodes or you can specify where you
want it. For example, if you have only one NIC, you would probably want all the hugepages
memory to be on the NUMA node closest to that NIC.

If you do not specify the `numa-node` settings in the `memory_models.yml` input model file and
use only the last entry indicating "size: 1G" and "count: 48" then this memory is spread evenly
across all NUMA nodes.

Also note that the hugepage service runs once at boot time and then goes to an inactive state so
you should not expect to see it running. If you decide to make changes to the NUMA memory
allocation, you will need to reboot the compute node for the changes to take effect.

9.3.14.4 DPDK Setup for Neutron Networking

9.3.14.4.1 Hardware requirements

- Intel-based compute node. DPDK is not available on AMD-based systems.

- The following BIOS settings must be enabled for DL360 Gen9:

  1. Virtualization Technology

  2. Intel(R) VT-d

  3. PCI-PT (Also see *Section 9.3.15.14, "Enabling PCI-PT on HPE DL360 Gen 9 Servers"*)

- Need adequate host memory to allow for hugepages. The examples below use 1G hugepages for the VMs

9.3.14.4.2 Limitations

- DPDK is supported on SLES only.

- Applies to HPE Helion OpenStack 8 only.

- Tenant network can be untagged vlan or untagged vxlan

- DPDK port names must be of the form 'dpdk<portid>' where port id is sequential and starts at 0

- No support for converting DPDK ports to non DPDK ports without rebooting compute node.

- No security group support, need userspace conntrack.

- No jumbo frame support.

9.3.14.4.3 Setup instructions

These setup instructions and example model are for a three-host system. There is one controller with Cloud Lifecycle Manager in cloud control plane and two compute hosts.

1. After initial run of site.yml all compute nodes must be rebooted to pick up changes in grub for hugepages and isolcpus

2. Changes to non-uniform memory access (NUMA) memory, isolcpu, or network devices must be followed by a reboot of compute nodes

3. Run sudo reboot to pick up libvirt change and hugepage/isocpus grub changes

```
tux > sudo reboot
```

4. Use the bash script below to configure nova aggregates, neutron networks, a new flavor, etc. And then it will spin up two VMs.

VM spin-up instructions

Before running the spin up script you need to get a copy of the cirros image to your Cloud Lifecycle Manager node. You can manually scp a copy of the cirros image to the system. You can copy it locallly with wget like so

```
ardana > wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

Save the following shell script in the home directory and run it. This should spin up two VMs, one on each compute node.

⬡ Warning

Make sure to change all network-specific information in the script to match your environment.

```
#!/usr/bin/env bash

source service.osrc

######## register glance image
glance image-create --name='cirros' --container-format=bare --disk-format=qcow2 < ~/
cirros-0.3.4-x86_64-disk.img

####### create nova aggregate and flavor for dpdk

MI_NAME=dpdk

nova aggregate-create $MI_NAME nova
nova aggregate-add-host $MI_NAME openstack-cp-comp0001-mgmt
nova aggregate-add-host $MI_NAME openstack-cp-comp0002-mgmt
```

```
nova aggregate-set-metadata $MI_NAME pinned=true


nova flavor-create $MI_NAME 6 1024 20 1
nova flavor-key $MI_NAME set hw:cpu_policy=dedicated
nova flavor-key $MI_NAME set aggregate_instance_extra_specs:pinned=true
nova flavor-key $MI_NAME set hw:mem_page_size=1048576


######## sec groups NOTE: no sec groups supported on DPDK.  This is in case we do non-
DPDK compute hosts.
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0


########  nova keys
nova keypair-add mykey >mykey.pem
chmod 400 mykey.pem


######## create neutron external network
neutron net-create ext-net --router:external --os-endpoint-type internalURL
neutron subnet-create ext-net 10.231.0.0/19 --gateway_ip=10.231.0.1  --ip-version=4 --
disable-dhcp  --allocation-pool start=10.231.17.0,end=10.231.17.255


########  neutron network
neutron net-create mynet1
neutron subnet-create mynet1 10.1.1.0/24 --name mysubnet1
neutron router-create myrouter1
neutron router-interface-add myrouter1 mysubnet1
neutron router-gateway-set myrouter1 ext-net
export MYNET=$(neutron net-list|grep mynet|awk '{print $2}')


######## spin up 2 VMs, 1 on each compute
nova boot --image cirros --nic net-id=${MYNET} --key-name mykey --flavor dpdk --
availability-zone nova:openstack-cp-comp0001-mgmt vm1
nova boot --image cirros --nic net-id=${MYNET} --key-name mykey --flavor dpdk --
availability-zone nova:openstack-cp-comp0002-mgmt vm2


######## create floating ip and attach to instance
export MYFIP1=$(nova floating-ip-create|grep ext-net|awk '{print $4}')
nova add-floating-ip vm1 ${MYFIP1}


export MYFIP2=$(nova floating-ip-create|grep ext-net|awk '{print $4}')
nova add-floating-ip vm2 ${MYFIP2}


nova list
```

9.3.14.5   DPDK Configurations

- *Section 9.3.14.5.1, "Base configuration"*

- *Section 9.3.14.5.2, "Performance considerations common to all NIC types"*

- *Section 9.3.14.5.3, "Multiqueue configuration"*

9.3.14.5.1   Base configuration

The following is specific to DL360 Gen9 and BIOS configuration as detailed in *Section 9.3.14.4, "DPDK Setup for Neutron Networking"*.

- EAL cores - 1, isolate: False in cpu-models

- PMD cores - 1 per NIC port

- Hugepages - 1G per PMD thread

- Memory channels - 4

- Global rx queues - based on needs

9.3.14.5.2   Performance considerations common to all NIC types

Compute host core frequency

Host CPUs should be running at maximum performance. The following is a script to set that. Note that in this case there are 24 cores. This needs to be modified to fit your environment. For a HP DL360 Gen9, the BIOS should be configured to use "OS Control Mode" which can be found on the iLO Power Settings page.

```
for i in `seq 0 23`; do echo "performance" > /sys/devices/system/cpu/cpu$i/cpufreq/
scaling_governor; done
```

IO non-posted prefetch

The DL360 Gen9 should have the IO non-posted prefetch disabled. Experimental evidence shows this yields an additional 6-8% performance boost.

9.3.14.5.3   Multiqueue configuration

In order to use multiqueue, a property must be applied to the Glance image and a setting inside the resulting VM must be applied. In this example we create a 4 vCPU flavor for DPDK using 1G hugepages.

```
MI_NAME=dpdk

nova aggregate-create $MI_NAME nova
nova aggregate-add-host $MI_NAME openstack-cp-comp0001-mgmt
nova aggregate-add-host $MI_NAME openstack-cp-comp0002-mgmt
nova aggregate-set-metadata $MI_NAME pinned=true

nova flavor-create $MI_NAME 6 1024 20 4
nova flavor-key $MI_NAME set hw:cpu_policy=dedicated
nova flavor-key $MI_NAME set aggregate_instance_extra_specs:pinned=true
nova flavor-key $MI_NAME set hw:mem_page_size=1048576
```

And set the hw_vif_multiqueue_enabled property on the Glance image

```
ardana > openstack image set --property hw_vif_multiqueue_enabled=true IMAGE UUID
```

Once the VM is booted using the flavor above, inside the VM, choose the number of combined rx and tx queues to be equal to the number of vCPUs

```
tux > sudo ethtool -L eth0 combined 4
```

On the hypervisor you can verify that multiqueue has been properly set by looking at the qemu process

```
-netdev type=vhost-user,id=hostnet0,chardev=charnet0,queues=4 -device virtio-net-
pci,mq=on,vectors=10,
```

Here you can see that 'mq = on' and vectors = 10. The formula for vectors is 2*num_queues + 2

9.3.14.6   Troubleshooting DPDK

- *Section 9.3.14.6.1, "Hardware configuration"*

- *Section 9.3.14.6.2, "System configuration"*

- *Section 9.3.14.6.3, "Input model configuration"*

- *Section 9.3.14.6.4, "Reboot requirements"*

- *Section 9.3.14.6.5, "Software configuration"*

- *Section 9.3.14.6.6, "DPDK runtime"*

- *Section 9.3.14.6.7, "Errors"*

9.3.14.6.1   Hardware configuration

Because there are several variations of hardware, it is up to you to verify that the hardware is configured properly.

- Only Intel based compute nodes are supported. There is no DPDK available for AMD-based CPUs.

- PCI-PT must be enabled for the NIC that will be used with DPDK.

- When using Intel Niantic and the igb_uio driver, the VT-d must be enabled in the BIOS.

- For DL360 Gen9 systems, the BIOS shared-memory *Section 9.3.15.14, "Enabling PCI-PT on HPE DL360 Gen 9 Servers"*.

- Adequate memory must be available for *Section 9.3.14.3, "Configuring Hugepages for DPDK in Neutron Networks"* usage.

- Hyper-threading can be enabled but is not required for base functionality.

- Determine the PCI slot that the DPDK NIC(s) are installed in to determine the associated NUMA node.

- Only the Intel Haswell, Broadwell, and Skylake microarchitectures are supported. These architectures are found in DL360 Gen9 and Gen10 hardware. Intel Sandy Bridge is not supported which is found in DL360 Gen8 hardware.

9.3.14.6.2 System configuration

- Only SLES12-SP3 compute nodes are supported.

- If a NIC port is used with PCI-PT, SRIOV-only, or PCI-PT + SRIOV, then it cannot be used with DPDK. They are mutually exclusive. This is because DPDK depends on an OvS bridge which does not exist if you use any combination of PCI-PT and SRIOV. You can use DPDK, SRIOV-only, and PCI-PT on difference interfaces of the same server.

- There is an association between the PCI slot for the NIC and a NUMA node. Make sure to use logical CPU cores that are on the NUMA node associated to the NIC. Use the following to determine which CPUs are on which NUMA node.

```
ardana > lscpu

Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2650L v3 @ 1.80GHz
Stepping:              2
CPU MHz:               1200.000
CPU max MHz:           1800.0000
CPU min MHz:           1200.0000
BogoMIPS:              3597.06
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0-11,24-35
NUMA node1 CPU(s):     12-23,36-47
```

9.3.14.6.3   Input model configuration

- If you do not specify a driver for a DPDK device, the igb_uio will be selected as default.

- DPDK devices must be named `dpdk<port-id>` where the port-id starts at 0 and increments sequentially.

- Tenant networks supported are untagged VXLAN and VLAN.

- Jumbo Frames MTU does not work with DPDK OvS. There is an upstream patch most likely showing up in OvS 2.6 and it cannot be backported due to changes this patch relies upon.

- Sample VXLAN model

- Sample VLAN model

9.3.14.6.4   Reboot requirements

A reboot of a compute node must be performed when an input model change causes the following:

1. After the initial `site.yml` play on a new OpenStack environment

2. Changes to an existing OpenStack environment that modify the `/etc/default/grub` file, such as

   - hugepage allocations

   - CPU isolation

   - iommu changes

3. Changes to a NIC port usage type, such as

   - moving from DPDK to any combination of PCI-PT and SRIOV

   - moving from DPDK to kernel based eth driver

9.3.14.6.5 Software configuration

The input model is processed by the Configuration Processor which eventually results in changes to the OS. There are several files that should be checked to verify the proper settings were applied. In addition, after the inital site.yml play is run all compute nodes must be rebooted in order to pickup changes to the `/etc/default/grub` file for hugepage reservation, CPU isolation and iommu settings.

Kernel settings

Check `/etc/default/grub` for the following

1. hugepages

2. CPU isolation

3. that iommu is in passthru mode if the igb_uio driver is in use

Open vSwitch settings

Check `/etc/default/openvswitch-switchf` for

1. using the `--dpdk` option

2. core 0 set aside for EAL and kernel to share

3. cores assigned to PMD drivers, at least two for each DPDK device

4. verify that memory is reserved with socket-mem option

5. Once VNETCORE-2509 (https://jira.hpcloud.net/browse/VNETCORE-2509)↗ merges also verify that the umask is 022 and the group is libvirt-qemu

DPDK settings

1. check `/etc/dpdk/interfacesf` for the correct DPDK devices

9.3.14.6.6 DPDK runtime

All non-bonded DPDK devices will be added to individual OvS bridges. The bridges will be named `br-dpdk0`, `br-dpdk1`, etc. The name of the OvS bridge for bonded DPDK devices will be `br-dpdkbond0`, `br-dpdkbond1`, etc.

1. Since each PMD thread is in a polling loop, it will use 100% of the CPU. Thus for two PMDs you would expect to see the ovs-vswitchd process running at 200%. This can be verified by running

```
ardana > top

top - 16:45:42 up 4 days, 22:24,  1 user,  load average: 2.03, 2.10, 2.14
Tasks: 384 total,   2 running, 382 sleeping,   0 stopped,   0 zombie
%Cpu(s):  9.0 us,  0.2 sy,  0.0 ni, 90.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  13171580+total, 10356851+used, 28147296 free,   257196 buffers
KiB Swap:        0 total,        0 used,        0 free.  1085868 cached Mem


  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1522 root      10 -10 6475196 287780  10192 S 200.4  0.2  14250:20 ovs-vswitchd
```

2. Verify that `ovs-vswitchd` is running with

   ```
   --dpdk option. ps -ef | grep ovs-vswitchd
   ```

3. PMD thread(s) are started when a DPDK port is added to an OvS bridge. Verify the port is on the bridge.

   ```
   tux > sudo ovs-vsctl show
   ```

4. A DPDK port cannot be added to an OvS bridge unless it is bound to a driver. Verify that the DPDK port is bound.

   ```
   tux > sudo dpdk_nic_bind -s
   ```

5. Verify that the proper number of hugepages is on the correct NUMA node

   ```
   tux > sudo virsh freepages --all
   ```

   or

   ```
   tux > sudo grep -R "" /sys/kernel/mm/hugepages/ /proc/sys/vm/*huge*
   ```

6. Verify that the VM and the DPDK PMD threads have both mapped the same hugepage(s)

   ```
   # this will yield 2 process ids, use the 2nd one
   tux > sudo ps -ef | grep ovs-vswitchd
   tux > sudo ls -l /proc/PROCESS-ID/fd | grep huge

   # if running more than 1 VM you will need to figure out which one to use
   tux > sudo ps -ef | grep qemu
   tux > sudo ls -l /proc/PROCESS-ID/fd | grep huge
   ```

9.3.14.6.7 Errors

VM does not get fixed IP

1. DPDK Poll Mode drivers (PMD) communicates with the VM by direct access of the VM hugepage. If a VM is not created using hugepages (see *Section 9.3.14.3, "Configuring Hugepages for DPDK in Neutron Networks"*), there is no way for DPDK to communicate with the VM and the VM will never be connected to the network.

2. It has been observed that the DPDK communication with VM fails if the shared-memory is not disabled in BIOS for DL360 Gen9.

Vestiges of non-existent DPDK devices

1. Incorrect input models that do not use the correct DPDK device name or do not use sequential port IDs starting at 0 may leave non-existent devices in the OvS database. While this does not affect proper functionality it may be confusing.

Startup issues

1. Running the following will help diagnose startup issues with ovs-vswitchd:

```
tux > sudo journalctl -u openvswitch.service --all
```

9.3.15 SR-IOV and PCI Passthrough Support

HPE Helion OpenStack supports both single-root I/O virtualization (SR-IOV) and PCI passthrough (PCIPT). Both technologies provide for better network performance.

This improves network I/O, decreases latency, and reduces processor overhead.

9.3.15.1 SR-IOV

A PCI-SIG Single Root I/O Virtualization and Sharing (SR-IOV) Ethernet interface is a physical PCI Ethernet NIC that implements hardware-based virtualization mechanisms to expose multiple virtual network interfaces that can be used by one or more virtual machines simultaneously. With SR-IOV based NICs, the traditional virtual bridge is no longer required. Each SR-IOV port is associated with a virtual function (VF).

When compared with a PCI Passthtrough Ethernet interface, an SR-IOV Ethernet interface:

- Provides benefits similar to those of a PCI Passthtrough Ethernet interface, including lower latency packet processing.

- Scales up more easily in a virtualized environment by providing multiple VFs that can be attached to multiple virtual machine interfaces.

- Shares the same limitations, including the lack of support for LAG, QoS, ACL, and live migration.

- Has the same requirements regarding the VLAN configuration of the access switches.

The process for configuring SR-IOV includes creating a VLAN provider network and subnet, then attaching VMs to that network.

With SR-IOV based NICs, the traditional virtual bridge is no longer required. Each SR-IOV port is associated with a virtual function (VF)

9.3.15.2 PCI passthrough Ethernet interfaces

A passthrough Ethernet interface is a physical PCI Ethernet NIC on a compute node to which a virtual machine is granted direct access. PCI passthrough allows a VM to have direct access to the hardware without being brokered by the hypervisor. This minimizes packet processing delays but at the same time demands special operational considerations. For all purposes, a PCI passthrough interface behaves as if it were physically attached to the virtual machine. Therefore any potential throughput limitations coming from the virtualized environment, such as the ones introduced by internal copying of data buffers, are eliminated. However, by bypassing the virtualized environment, the use of PCI passthrough Ethernet devices introduces several restrictions that must be taken into consideration. They include:

- no support for LAG, QoS, ACL, or host interface monitoring

- no support for live migration

- no access to the compute node's OVS switch

A passthrough interface bypasses the compute node's OVS switch completely, and is attached instead directly to the provider network's access switch. Therefore, proper routing of traffic to connect the passthrough interface to a particular tenant network depends entirely on the VLAN tagging options configured on both the passthrough interface and the access port on the switch (TOR).

The access switch routes incoming traffic based on a VLAN ID, which ultimately determines the tenant network to which the traffic belongs. The VLAN ID is either explicit, as found in incoming tagged packets, or implicit, as defined by the access port's default VLAN ID when the incoming packets are untagged. In both cases the access switch must be configured to process the proper VLAN ID, which therefore has to be known in advance

9.3.15.3   Leveraging PCI Passthrough

Two parts are necessary to leverage PCI passthrough on a SUSE OpenStack Cloud 8 Compute Node: preparing the Compute Node, preparing Nova and Glance.

1. **Preparing the Compute Node**

   1. There should be no kernel drivers or binaries with direct access to the PCI device. If there are kernel modules, they should be blacklisted.
      For example, it is common to have a `nouveau` driver from when the node was installed. This driver is a graphics driver for Nvidia-based GPUs. It must be blacklisted as shown in this example.

      ```
      ardana > echo 'blacklist nouveau' >> /etc/modprobe.d/nouveau-default.conf
      ```

      The file location and its contents are important; the filename is your choice. Other drivers can be blacklisted in the same manner, possibly including Nvidia drivers.

   2. On the host, `iommu_groups` should be enabled. To check if IOMMU is enabled:

      ```
      root # virt-host-validate
      .....
      QEMU: Checking if IOMMU is enabled by kernel
      : WARN (IOMMU appears to be disabled in kernel. Add intel_iommu=on to kernel
       cmdline arguments)
      .....
      ```

To modify the kernel cmdline as suggested in the warning, edit the file `/etc/default/grub` and append `intel_iommu=on` to the `GRUB_CMDLINE_LINUX_DEFAULT` variable. Then run `update-bootloader`.

A reboot will be required for `iommu_groups` to be enabled.

3. After the reboot, check that IOMMU is enabled:

```
root # virt-host-validate
.....
QEMU: Checking if IOMMU is enabled by kernel
: PASS
.....
```

4. Confirm IOMMU groups are available by finding the group associated with your PCI device (for example Nvidia GPU):

```
ardana > lspci -nn | grep -i nvidia
08:00.0 VGA compatible controller [0300]: NVIDIA Corporation GT218 [NVS 300]
 [10de:10d8] (rev a2)
08:00.1 Audio device [0403]: NVIDIA Corporation High Definition Audio
 Controller [10de:0be3] (rev a1)
```

In this example, `08:00.0` and `08:00.1` are addresses of the PCI device. The vendorID is `10de`. The productIDs are `10d8` and `0be3`.

5. Confirm that the devices are available for passthrough:

```
ardana > ls -ld /sys/kernel/iommu_groups/*/devices/*08:00.?/
drwxr-xr-x 3 root root 0 Feb 14 13:05 /sys/kernel/iommu_groups/20/
devices/0000:08:00.0/
drwxr-xr-x 3 root root 0 Feb 19 16:09 /sys/kernel/iommu_groups/20/
devices/0000:08:00.1/
```

☞ Note

With PCI passthrough, only an entire IOMMU group can be passed. Parts of the group cannot be passed. In this example, the IOMMU group is `20`.

2. **Preparing Nova and Glance for passthrough**

Information about configuring Nova and Glance is available in the documentation at https://docs.openstack.org/nova/pike/admin/pci-passthrough.html ↗. Both `nova-compute` and `nova-scheduler` must be configured.

9.3.15.4 Supported Intel 82599 Devices

TABLE 9.1: INTEL 82599 DEVICES SUPPORTED WITH SRIOV AND PCIPT

Vendor	Device	Title
Intel Corporation	10f8	82599 10 Gigabit Dual Port Backplane Connection
Intel Corporation	10f9	82599 10 Gigabit Dual Port Network Connection
Intel Corporation	10fb	82599ES 10-Gigabit SFI/ SFP+ Network Connection
Intel Corporation	10fc	82599 10 Gigabit Dual Port Network Connection

9.3.15.5   SRIOV PCIPT configuration

If you plan to take advantage of SR-IOV support in HPE Helion OpenStack you will need to plan in advance to meet the following requirements:

1. Use one of the supported NIC cards:

   - HP Ethernet 10Gb 2-port 560FLR-SFP+ Adapter (Intel Niantic). Product part number: 665243-B21 -- Same part number for the following card options:

     - FlexLOM card

     - PCI slot adapter card

2. Identify the NIC ports to be used for PCI Passthrough devices and SRIOV devices from each compute node

3. Ensure that:

   - SRIOV is enabled in the BIOS

   - HP Shared memory is disabled in the BIOS on the compute nodes.

   - The Intel boot agent is disabled on the compute (*Section 9.3.15.11, "Intel bootutils"* can be used to perform this)

> ☞ **Note**
>
> Because of Intel driver limitations, you cannot use a NIC port as an SRIOV NIC as well as a physical NIC. Using the physical function to carry the normal tenant traffic through the OVS bridge at the same time as assigning the VFs from the same NIC device as passthrough to the guest VM is not supported.

If the above prerequisites are met, then SR-IOV or PCIPT can be reconfigured at any time. There is no need to do it at install time.

9.3.15.6 Deployment use cases

The following are typical use cases that should cover your particular needs:

1. A device on the host needs to be enabled for both PCI-passthrough and PCI-SRIOV during deployment. At run time Nova decides whether to use physical functions or virtual function depending on vnic_type of the port used for booting the VM.

2. A device on the host needs to be configured only for PCI-passthrough.

3. A device on the host needs to be configured only for PCI-SRIOV virtual functions.

9.3.15.7 Input model updates

HPE Helion OpenStack 8 provides various options for the user to configure the network for tenant VMs. These options have been enhanced to support SRIOV and PCIPT.

the Cloud Lifecycle Manager input model changes to support SRIOV and PCIPT are as follows. If you were familiar with the configuration settings previously, you will notice these changes.

net_interfaces.yml: This file defines the interface details of the nodes. In it, the following fields have been added under the compute node interface section:

Key	Value
sriov_only:	Indicates that only SR-IOV be enabled on the interface. This should be set to true if you want to dedicate the NIC interface to support only SR-IOV functionality.
pci-pt:	When this value is set to true, it indicates that PCIPT should be enabled on the interface.
vf-count:	Indicates the number of VFs to be configured on a given interface.

In control_plane.yml, under Compute resource neutron-sriov-nic-agent has been added as service components

under resources:

Key	Value
name:	Compute
resource-prefix:	Comp
server-role:	COMPUTE-ROLE
allocation-policy:	Any
min-count:	0
service-components:	ntp-client
	nova-compute
	nova-compute-kvm
	neutron-l3-agent
	neutron-metadata-agent
	neutron-openvswitch-agent
	neutron-lbaasv2-agent
	- neutron-sriov-nic-agent*

nic_device_data.yml: This is the new file added with this release to support SRIOV and PCIPT configuration details. It contains information about the specifics of a nic, and is found here: `~/openstack/ardana/services/osconfig/nic_device_data.yml`. The fields in this file are as follows.

1. **nic-device-types:** The nic-device-types section contains the following key-value pairs:

Key	Value
name:	The name of the nic-device-types that will be referenced in nic_mappings.yml
family:	The name of the nic-device-families to be used with this nic_device_type
device_id:	Device ID as specified by the vendor for the particular NIC

Key	Value
type:	The value of this field can be "simple-port" or "multi-port". If a single bus address is assigned to more than one nic it will be multi-port, else if there is a one-to one mapping between bus address and the nic then it will be simple-port.

2. **nic-device-families:** The nic-device-families section contains the following key-value pairs:

Key	Value
name:	The name of the device family that can be used for reference in nic-device-types.
vendor-id:	Vendor ID of the NIC
config-script:	A script file used to create the virtual functions (VF) on the Compute node.
driver:	Indicates the NIC driver that needs to be used.
vf-count-type:	This value can be either "port" or "driver".
"port":	Indicates that the device supports per-port virtual function (VF) counts.
"driver:"	Indicates that all ports using the same driver will be configured with the same number of VFs, whether or not the interface model specifies a vf-count attribute for the port. If two or more ports specify different vf-count values, the config processor errors out.
Max-vf-count:	This field indicates the maximum VFs that can be configured on an interface as defined by the vendor.

control_plane.yml: This file provides the information about the services to be run on a particular node. To support SR-IOV on a particular compute node, you must run neutron-sriov-nic-agent on that node.

Mapping the use cases with various fields in input model

	Vf-count	SR-IOV	PCIPT	OVS bridge	Can be bonded	NIC Use case
sriov-only: true	Mandatory	Yes	No	No	No	Dedicated to SRIOV
pci-pt : true	Not Specified	No	Yes	No	No	Dedicated to PCI-PT
pci-pt : true	Specified	Yes	Yes	No	No	PCI-PT or SRIOV
pci-pt and sriov-only keywords are not specified	Specified	Yes	No	Yes	No	SRIOV with PF used by host
pci-pt and sriov-only keywords are not specified	Not Specified	No	No	Yes	Yes	Traditional/ Usual use case

9.3.15.8 Mappings between nic_mappings.yml and net_interfaces.yml

The following diagram shows which fields in nic_mappings.yml map to corresponding fields in net_interfaces.yml:
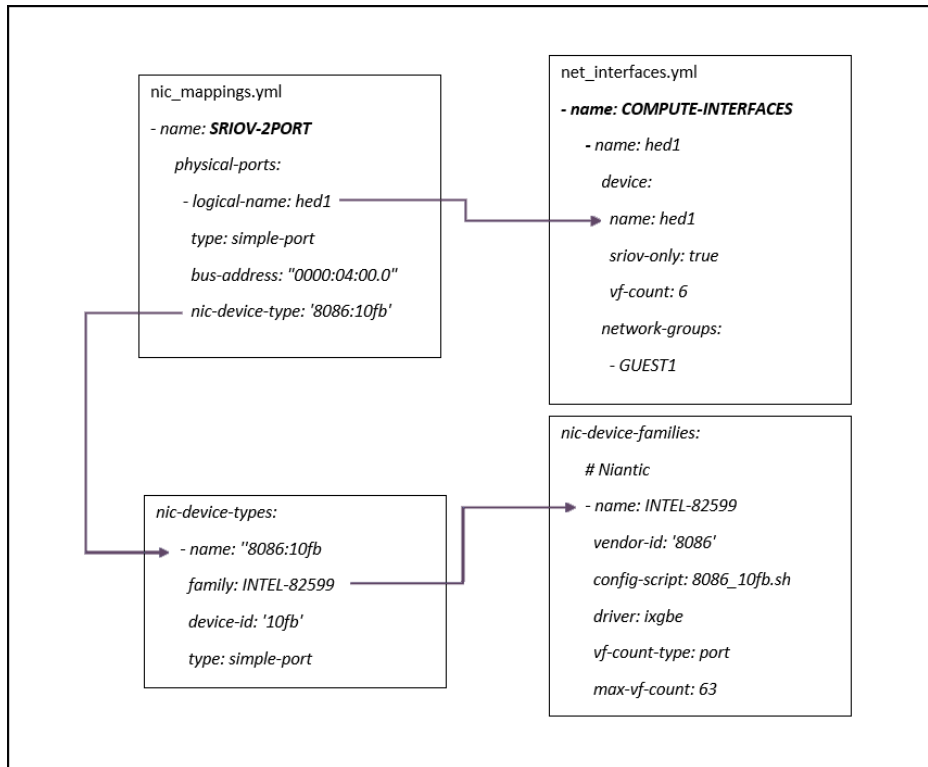
9.3.15.9 Example Use Cases for Intel

1. **Nic-device-types and nic-device-families** with Intel 82559 with ixgbe as the driver.

```
nic-device-types:
    - name: ''8086:10fb
      family: INTEL-82599
      device-id: '10fb'
      type: simple-port
nic-device-families:
    # Niantic
    - name: INTEL-82599
      vendor-id: '8086'
      config-script: intel-82599.sh
      driver: ixgbe
      vf-count-type: port
```

```
      max-vf-count: 63
```

2. **net_interfaces.yml** for the SRIOV-only use case:

```
- name: COMPUTE-INTERFACES
  - name: hed1
    device:
      name: hed1
      sriov-only: true
      vf-count: 6
    network-groups:
     - GUEST1
```

3. **net_interfaces.yml** for the PCIPT-only use case:

```
- name: COMPUTE-INTERFACES
  - name: hed1
    device:
      name: hed1
      pci-pt: true
   network-groups:
     - GUEST1
```

4. **net_interfaces.yml** for the SRIOV and PCIPT use case

```
 - name: COMPUTE-INTERFACES
   - name: hed1
     device:
       name: hed1
       pci-pt: true
       vf-count: 6
     network-groups:
     - GUEST1
```

5. **net_interfaces.yml** for SRIOV and Normal Virtio use case

```
- name: COMPUTE-INTERFACES
  - name: hed1
    device:
       name: hed1
       vf-count: 6
    network-groups:
     - GUEST1
```

6. **net_interfaces.yml** for PCI-PT (`hed1` and `hed4` refer to the DUAL ports of the PCI-PT NIC)

```
    - name: COMPUTE-PCI-INTERFACES
      network-interfaces:
      - name: hed3
        device:
          name: hed3
        network-groups:
          - MANAGEMENT
          - EXTERNAL-VM
        forced-network-groups:
          - EXTERNAL-API
      - name: hed1
        device:
          name: hed1
          pci-pt: true
        network-groups:
          - GUEST
      - name: hed4
        device:
          name: hed4
          pci-pt: true
        network-groups:
          - GUEST
```

9.3.15.10 Launching Virtual Machines

Provisioning a VM with SR-IOV NIC is a two-step process.

1. Create a Neutron port with `vnic_type = direct`.

   ```
   ardana > neutron port-create $net_id --name sriov_port --binding:vnic_type direct
   ```

2. Boot a VM with the created `port-id`.

   ```
   ardana > nova boot --flavor m1.large --image ubuntu_14.04 --nic port-id=$port_id
    test-sriov
   ```

Provisioning a VM with PCI-PT NIC is a two-step process.

1. Create two Neutron ports with `vnic_type = direct-physical`.

   ```
   ardana > neutron port-create net1 --name pci-port1 --vnic_type=direct-physical
   neutron port-create net1 --name pci-port2  --vnic_type=direct-physical
   ```

2. Boot a VM with the created ports.

```
ardana > nova boot --flavor 4 --image opensuse --nic port-id pci-port1-port-id \
--nic port-id pci-port2-port-id vm1-pci-passthrough
```

If PCI-PT VM gets stuck (hangs) at boot time when using an Intel NIC, the boot agent should be disabled.

9.3.15.11 Intel bootutils

When Intel cards are used for PCI-PT, a tenant VM can get stuck at boot time. When this happens, you should download Intel bootutils and use it to should disable bootagent.

1. Download Preebot.tar.gz from https://downloadcenter.intel.com/download/19186/Intel-Ethernet-Connections-Boot-Utility-Preboot-Images-and-EFI-Drivers ↗

2. Untar the `Preboot.tar.gz` on the compute node where the PCI-PT VM is to be hosted.

3. Go to ~/APPS/BootUtil/Linux_x64

   ```
   cd ~/APPS/BootUtil/Linux_x64
   ```

   and run following command

   ```
   ./bootutil64e -BOOTENABLE disable -all
   ```

4. Boot the PCI-PT VM and it should boot without getting stuck.

> 🎓 **Note**
>
> Here even though VM console shows VM getting stuck at PXE boot, it is not related to BIOS PXE settings.

9.3.15.12 Making input model changes and implementing PCI PT and SR-IOV

To implent the configuration you require, log into the Cloud Lifecycle Manager node and update the Cloud Lifecycle Manager model files to enable SR-IOV or PCIPT following the relevent use case explained above. You will need to edit

- net_interfaces.yml

- nic_device_data.yml

- control_plane.yml

To make the edits,

1. Check out the site branch of the local git repository and change to the correct directory:

```
ardana > git checkout site
ardana > cd ~/openstack/my_cloud/definition/data/
```

2. Open each file in vim or another editor and make the necessary changes. Save each file, then commit to the local git repository:

```
ardana > git add -A
ardana > git commit -m "your commit message goes here in quotes"
```

3. Here you will have the Cloud Lifecycle Manager enable your changes by running the necessary playbooks:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts site.yml
```

Note

After running the site.yml playbook above, you must reboot the compute nodes that are configured with Intel PCI devices.

Note

When a VM is running on an SRIOV port on a given compute node, reconfiguration is not supported.

You can set the number of virtual functions that must be enabled on a compute node at install time. You can update the number of virtual functions after deployment. If any VMs have been spawned before you change the number of virtual functions, those VMs may lose connectivity.

Therefore, it is always recommended that if any virtual function is used by any tenant VM, you should not reconfigure the virtual functions. Instead, you should delete/migrate all the VMs on that NIC before reconfiguring the number of virtual functions.

9.3.15.13 Limitations

- Security groups are not applicable for PCI-PT and SRIOV ports.

- Live migration is not supported for VMs with PCI-PT and SRIOV ports.

- Rate limiting (QoS) is not applicable on SRIOV and PCI-PT ports.

- SRIOV/PCIPT is not supported for VxLAN network.

- DVR is not supported with SRIOV/PCIPT.

- For Intel cards, the same NIC cannot be used for both SRIOV and normal VM boot.

- Current upstream OpenStack code does not support this hot plugin of SRIOV/PCIPT interface using the nova `attach_interface` command. See https://review.openstack.org/#/c/139910/ ↗ for more information.

- Neutron port-update when admin state is down will not work.

- SLES Compute Nodes with dual-port PCI-PT NICs, both ports should always be passed in the VM. It is not possible to split the dual port and pass through just a single port.

9.3.15.14 Enabling PCI-PT on HPE DL360 Gen 9 Servers

The HPE DL360 Gen 9 and HPE ProLiant systems with Intel processors use a region of system memory for sideband communication of management information. The BIOS sets up Reserved Memory Region Reporting (RMRR) to report these memory regions and devices to the operating system. There is a conflict between the Linux kernel and RMRR which causes problems with PCI pass-through (PCI-PT). This is needed for IOMMU use by DPDK. Note that this does not affect SR-IOV.

In order to enable PCI-PT on the HPE DL360 Gen 9 you must have a version of firmware that supports setting this and you must change a BIOS setting.

To begin, get the latest firmware and install it on your compute nodes.

Once the firmware has been updated:

1. Reboot the server and press `F9` (system utilities) during POST (power on self test)

2. Choose *System Configuration*

3. Select the NIC for which you want to enable PCI-PT

4. Choose *Device Level Configuration*

5. Disable the shared memory feature in the BIOS.

6. Save the changes and reboot server

9.3.16   Setting up VLAN-Aware VMs

Creating a VM with a trunk port will allow a VM to gain connectivity to one or more networks over the same virtual NIC (vNIC) through the use VLAN interfaces in the guest VM. Connectivity to different networks can be added and removed dynamically through the use of subports. The network of the parent port will be presented to the VM as the untagged VLAN, and the networks of the child ports will be presented to the VM as the tagged VLANs (the VIDs of which can be chosen arbitrarily as long as they are unique to that trunk). The VM will send/receive VLAN-tagged traffic over the subports, and Neutron will mux/demux the traffic onto the subport's corresponding network. This is not to be confused with VLAN transparency where a VM can pass VLAN-tagged traffic transparently across the network without interference from Neutron. VLAN transparency is not supported.

9.3.16.1   Terminology

- **Trunk**: a resource that logically represents a trunked vNIC and references a parent port.

- **Parent port**: a Neutron port that a Trunk is referenced to. Its network is presented as the untagged VLAN.

- **Subport**: a resource that logically represents a tagged VLAN port on a Trunk. A Subport references a child port and consists of the <port>,<segmentation-type>,<segmentation-id> tuple. Currently only the 'vlan' segmentation type is supported.

- **Child port**: a Neutron port that a Subport is referenced to. Its network is presented as a tagged VLAN based upon the segmentation-id used when creating/adding a Subport.

- **Legacy VM**: a VM that does not use a trunk port.

- **Legacy port**: a Neutron port that is not used in a Trunk.

- **VLAN-aware VM**: a VM that uses at least one trunk port.

9.3.16.2 Trunk CLI reference

Command	Action
network trunk create	Create a trunk.
network trunk delete	Delete a given trunk.
network trunk list	List all trunks.
network trunk show	Show information of a given trunk.
network trunk set	Add subports to a given trunk.
network subport list	List all subports for a given trunk.
network trunk unset	Remove subports from a given trunk.
network trunk set	Update trunk properties.

9.3.16.3 Enabling VLAN-aware VM capability

1. Edit `~/openstack/my_cloud/config/neutron/neutron.conf.j2` to add the "trunk" service_plugin:

```
service_plugins = {{ neutron_service_plugins }},trunk
```

2. Edit `~/openstack/my_cloud/config/neutron/ml2_conf.ini.j2` to enable the noop firewall driver:

```
[securitygroup]
firewall_driver = neutron.agent.firewall.NoopFirewallDriver
```

> **Note**
>
> This is a manual configuration step because it must be made apparent that this step disables Neutron security groups completely. The default HPE Helion OpenStack firewall_driver is `neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewall Driver` which does not implement security groups for trunk ports. Optionally, the HPE Helion OpenStack default firewall_driver may still be used (that is, skip this step), which would provide security groups for legacy VMs but not for VLAN-aware VMs. However, this mixed environment is not recommended. For more information, see *Section 9.3.16.6, "Firewall issues"*.

3. Commit the configuration changes:

```
ardana > git add -A
ardana > git commit -m "Enable vlan-aware VMs"
ardana > cd ~/openstack/ardana/ansible/
```

4. If this is an initial deployment, continue the rest of normal deployment process:

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts site.yml
```

5. If the cloud has already been deployed and this is a reconfiguration:

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
```

9.3.16.4 Use Cases

Creating a trunk port

Assume that a number of Neutron networks/subnets already exist: private, foo-net, and bar-net. This will create a trunk with two subports allocated to it. The parent port will be on the "private" network, while the two child ports will be on "foo-net" and "bar-net", respectively:

1. Create a port that will function as the trunk's parent port:

```
ardana > neutron port-create --name trunkparent private
```

2. Create ports that will function as the child ports to be used in subports:

```
ardana > neutron port-create --name subport1 foo-net
ardana > neutron port-create --name subport2 bar-net
```

3. Create a trunk port using the `openstack network trunk create` command, passing the parent port created in step 1 and child ports created in step 2:

```
ardana > openstack network trunk create --parent-port trunkparent --
subport port=subport1,segmentation-type=vlan,segmentation-id=1 --subport
 port=subport2,segmentation-type=vlan,segmentation-id=2 mytrunk
+-----------------
+-----------------------------------------------------------------------------------
+
| Field           | Value
                         |
+-----------------
+-----------------------------------------------------------------------------------
+
| admin_state_up  | UP
                         |
| created_at      | 2017-06-02T21:49:59Z
                         |
| description     |
                         |
| id              | bd822ebd-33d5-423e-8731-dfe16dcebac2
                         |
| name            | mytrunk
                         |
| port_id         | 239f8807-be2e-4732-9de6-c64519f46358
                         |
| project_id      | f51610e1ac8941a9a0d08940f11ed9b9
                         |
| revision_number | 1
                         |
| status          | DOWN
                         |
```

```
| sub_ports        | port_id='9d25abcf-d8a4-4272-9436-75735d2d39dc',
 segmentation_id='1', segmentation_type='vlan' |
|                  | port_id='e3c38cb2-0567-4501-9602-c7a78300461e',
 segmentation_id='2', segmentation_type='vlan' |
| tenant_id        | f51610e1ac8941a9a0d08940f11ed9b9
                                    |
| updated_at       | 2017-06-02T21:49:59Z
                                    |
+-----------------
+-------------------------------------------------------------------------------------------------
+

$ openstack network subport list --trunk mytrunk
+-------------------------------------+-------------------+-----------------+
| Port                                | Segmentation Type | Segmentation ID |
+-------------------------------------+-------------------+-----------------+
| 9d25abcf-d8a4-4272-9436-75735d2d39dc | vlan             |               1 |
| e3c38cb2-0567-4501-9602-c7a78300461e | vlan             |               2 |
+-------------------------------------+-------------------+-----------------+
```

Optionally, a trunk may be created without subports (they can be added later):

```
ardana > openstack network trunk create --parent-port trunkparent mytrunk
+-----------------+-------------------------------------+
| Field           | Value                               |
+-----------------+-------------------------------------+
| admin_state_up  | UP                                  |
| created_at      | 2017-06-02T21:45:35Z                |
| description     |                                     |
| id              | eb8a3c7d-9f0a-42db-b26a-ca15c2b38e6e |
| name            | mytrunk                             |
| port_id         | 239f8807-be2e-4732-9de6-c64519f46358 |
| project_id      | f51610e1ac8941a9a0d08940f11ed9b9    |
| revision_number | 1                                   |
| status          | DOWN                                |
| sub_ports       |                                     |
| tenant_id       | f51610e1ac8941a9a0d08940f11ed9b9    |
| updated_at      | 2017-06-02T21:45:35Z                |
+-----------------+-------------------------------------+
```

A port that is already bound (that is, already in use by a VM) cannot be upgraded to a trunk port. The port must be unbound to be eligible for use as a trunk's parent port. When adding subports to a trunk, the child ports must be unbound as well.

Checking a port's trunk details

Once a trunk has been created, its parent port will show the `trunk_details` attribute, which consists of the `trunk_id` and list of subport dictionaries:

```
ardana > neutron port-show -F trunk_details trunkparent
+---------------
+----------------------------------------------------------------------------------------+
| Field          | Value
            |
+---------------
+----------------------------------------------------------------------------------------+
| trunk_details | {"trunk_id": "bd822ebd-33d5-423e-8731-dfe16dcebac2", "sub_ports":
            |
|               | [{"segmentation_id": 2, "port_id": "e3c38cb2-0567-4501-9602-
c7a78300461e",       |
|               | "segmentation_type": "vlan", "mac_address": "fa:16:3e:11:90:d2"},
            |
|               | {"segmentation_id": 1, "port_id": "9d25abcf-
d8a4-4272-9436-75735d2d39dc",          |
|               | "segmentation_type": "vlan", "mac_address": "fa:16:3e:ff:de:73"}]}
            |
+---------------
+----------------------------------------------------------------------------------------+
```

Ports that are not trunk parent ports will not have a `trunk_details` field:

```
ardana > neutron port-show -F trunk_details subport1
need more than 0 values to unpack
```

Adding subports to a trunk

Assuming a trunk and new child port have been created already, the `trunk-subport-add` command will add one or more subports to the trunk.

1. Run `openstack network trunk set`

   ```
   ardana > openstack network trunk set --subport port=subport3,segmentation-
   type=vlan,segmentation-id=3 mytrunk
   ```

2. Run `openstack network subport list`

   ```
   ardana > openstack network subport list --trunk mytrunk
   +-------------------------------------+-------------------+-----------------+
   | Port                                | Segmentation Type | Segmentation ID |
   +-------------------------------------+-------------------+-----------------+
   | 9d25abcf-d8a4-4272-9436-75735d2d39dc | vlan              |               1 |
   | e3c38cb2-0567-4501-9602-c7a78300461e | vlan              |               2 |
   ```

```
| bf958742-dbf9-467f-b889-9f8f2d6414ad | vlan                |                3 |
+--------------------------------------+--------------------+------------------+
```

Note

The `--subport` option may be repeated multiple times in order to add multiple subports at a time.

Removing subports from a trunk

To remove a subport from a trunk, use `openstack network trunk unset` command:

```
ardana > openstack network trunk unset --subport subport3 mytrunk
```

Deleting a trunk port

To delete a trunk port, use the `openstack network trunk delete` command:

```
ardana > openstack network trunk delete mytrunk
```

Once a trunk has been created successfully, its parent port may be passed to the `nova boot` command, which will make the VM VLAN-aware:

```
ardana > nova boot --image ubuntu-server --flavor 1 --nic port-id=239f8807-
be2e-4732-9de6-c64519f46358 vlan-aware-vm
```

Note

A trunk cannot be deleted until its parent port is unbound. Mainly, this means you must delete the VM using the trunk port before you are allowed to delete the trunk.

9.3.16.5 VLAN-aware VM network configuration

This section illustrates how to configure the VLAN interfaces inside a VLAN-aware VM based upon the subports allocated to the trunk port being used.

1. Run `openstack network trunk subport list` to see the VLAN IDs in use on the trunk port:

```
ardana > openstack network subport list --trunk mytrunk
+--------------------------------------+------------------+------------------+
```

```
| Port                                  | Segmentation Type | Segmentation ID |
+---------------------------------------+-------------------+-----------------+
| e3c38cb2-0567-4501-9602-c7a78300461e | vlan              |               2 |
+---------------------------------------+-------------------+-----------------+
```

2. Run `neutron port-show` on the child port to get its mac_address:

```
ardana > neutron port-show -F mac_address 08848e38-50e6-4d22-900c-b21b07886fb7
+-------------+-------------------+
| Field       | Value             |
+-------------+-------------------+
| mac_address | fa:16:3e:08:24:61 |
+-------------+-------------------+
```

3. Log into the VLAN-aware VM and run the following commands to set up the VLAN interface:

```
tux > sudo ip link add link ens3 ens3.2 address fa:16:3e:11:90:d2 broadcast
 ff:ff:ff:ff:ff:ff type vlan id 2
$ sudo ip link set dev ens3.2 up
```

4. Note the usage of the mac_address from step 2 and VLAN ID from step 1 in configuring the VLAN interface:

```
tux > sudo ip link add link ens3 ens3.2 address fa:16:3e:11:90:d2 broadcast
 ff:ff:ff:ff:ff:ff type vlan id 2
```

5. Trigger a DHCP request for the new vlan interface to verify connectivity and retrieve its IP address. On an Ubuntu VM, this might be:

```
tux > sudo dhclient ens3.2
tux > sudo ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
 qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc pfifo_fast state UP group
 default qlen 1000
    link/ether fa:16:3e:8d:77:39 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.5/24 brd 10.10.10.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe8d:7739/64 scope link
```

```
        valid_lft forever preferred_lft forever
3: ens3.2@ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP
 group default qlen 1000
    link/ether fa:16:3e:11:90:d2 brd ff:ff:ff:ff:ff:ff
    inet 10.10.12.7/24 brd 10.10.12.255 scope global ens3.2
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe11:90d2/64 scope link
        valid_lft forever preferred_lft forever
```

9.3.16.6　Firewall issues

The HPE Helion OpenStack default firewall_driver is `neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver`. This default does not implement security groups for VLAN-aware VMs, but it does implement security groups for legacy VMs. For this reason, it is recommended to disable Neutron security groups altogether when using VLAN-aware VMs. To do so, set:

```
firewall_driver = neutron.agent.firewall.NoopFirewallDriver
```

Doing this will prevent having a mix of firewalled and non-firewalled VMs in the same environment, but it should be done with caution because all VMs would be non-firewalled.

10 Managing the Dashboard

Information about managing and configuring the Dashboard service.

10.1 Configuring the Dashboard Service

Horizon is the OpenStack service that serves as the basis for the HPE Helion OpenStack dashboards.

The dashboards provide a web-based user interface to HPE Helion OpenStack services including Compute, Volume Operations, Networking, and Identity.

Along the left side of the dashboard are sections that provide access to Project and Identity sections. If your login credentials have been assigned the 'admin' role you will also see a separate Admin section that provides additional system-wide setting options.

Across the top are menus to switch between projects and menus where you can access user settings.

10.1.1 Dashboard Service and TLS in HPE Helion OpenStack

By default, the Dashboard service is configured with TLS in the input model (ardana-input-model). You should not disable TLS in the input model for the Dashboard service. The normal use case for users is to have all services behind TLS, but users are given the freedom in the input model to take a service off TLS for troubleshooting or debugging. TLS should always be enabled for production environments.

Make sure that `horizon_public_protocol` and `horizon_private_protocol` are both be set to use https.

10.2 Changing the Dashboard Timeout Value

The default session timeout for the dashboard is 1800 seconds or 30 minutes. This is the recommended default and best practice for those concerned with security.

As an administrator, you can change the session timeout by changing the value of the SESSION_TIMEOUT to anything less than or equal to 14400, which is equal to four hours. Values greater than 14400 should not be used due to Keystone constraints.

⛉ Warning

Increasing the value of SESSION_TIMEOUT increases the risk of abuse.

10.2.1 How to Change the Dashboard Timeout Value

Follow these steps to change and commit the Horizon timeout value.

1. Log in to the Cloud Lifecycle Manager.

2. Edit the Dashboard config file at `~/openstack/my_cloud/config/horizon/local_settings.py` and, if it is not already present, add a line for `SESSION_TIMEOUT` above the line for `SESSION_ENGINE`.

    Here is an example snippet, in bold:

    ```
    SESSION_TIMEOUT = <timeout value>
    SESSION_ENGINE = 'django.contrib.sessions.backends.db'
    ```

    ## 🔔 Important

    Do not exceed the maximum value of 14400.

3. Commit the changes to git:

    ```
    git add -A
    git commit -a -m "changed Horizon timeout value"
    ```

4. Run the configuration processor:

    ```
    cd ~/openstack/ardana/ansible
    ansible-playbook -i hosts/localhost config-processor-run.yml
    ```

5. Update your deployment directory:

    ```
    cd ~/openstack/ardana/ansible
    ansible-playbook -i hosts/localhost ready-deployment.yml
    ```

6. Run the Dashboard reconfigure playbook:

    ```
    cd ~/scratch/ansible/next/ardana/ansible
    ansible-playbook -i hosts/verb_hosts horizon-reconfigure.yml
    ```

11 Managing Orchestration

Information about managing and configuring the Orchestration service, based on OpenStack Heat.

11.1 Configuring the Orchestration Service

Information about configuring the Orchestration service, based on OpenStack Heat.

The Orchestration service, based on OpenStack Heat, does not need any additional configuration to be used. This document describes some configuration options as well as reasons you may want to use them.

Heat Stack Tag Feature

Heat provides a feature called Stack Tags to allow attributing a set of simple string-based tags to stacks and optionally the ability to hide stacks with certain tags by default. This feature can be used for behind-the-scenes orchestration of cloud infrastructure, without exposing the cloud user to the resulting automatically-created stacks.

Additional details can be seen here: OpenStack - Stack Tags (https://specs.openstack.org/openstack/heat-specs/specs/kilo/stack-tags.html) ↗ .

In order to use the Heat stack tag feature, you need to use the following steps to define the `hidden_stack_tags` setting in the Heat configuration file and then reconfigure the service to enable the feature.

1. Log in to the Cloud Lifecycle Manager.

2. Edit the Heat configuration file, at this location:

   ```
   ~/openstack/my_cloud/config/heat/heat.conf.j2
   ```

3. Under the `[DEFAULT]` section, add a line for `hidden_stack_tags`. Example:

   ```
   [DEFAULT]
   hidden_stack_tags="<hidden_tag>"
   ```

4. Commit the changes to your local git:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add --all
   ```

```
ardana > git commit -m "enabling Heat Stack Tag feature"
```

5. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Reconfigure the Orchestration service:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts heat-reconfigure.yml
```

To begin using the feature, use these steps to create a Heat stack using the defined hidden tag. You will need to use credentials that have the Heat admin permissions. In the example steps below we are going to do this from the Cloud Lifecycle Manager using the `admin` credentials and a Heat template named `heat.yaml`:

1. Log in to the Cloud Lifecycle Manager.

2. Source the admin credentials:

```
ardana > source ~/service.osrc
```

3. Create a Heat stack using this feature:

```
ardana > openstack stack create -f heat.yaml hidden_stack_tags --tags hidden
```

4. If you list your Heat stacks, your hidden one will not show unless you use the `--hidden` switch.
   Example, not showing hidden stacks:

```
ardana > openstack stack list
```

Example, showing the hidden stacks:

```
ardana > openstack stack list --hidden
```

11.2 Autoscaling using the Orchestration Service

Autoscaling is a process that can be used to scale up and down your compute resources based on the load they are currently experiencing to ensure a balanced load.

11.2.1 What is autoscaling?

Autoscaling is a process that can be used to scale up and down your compute resources based on the load they are currently experiencing to ensure a balanced load across your compute environment.

🔔 Important

Autoscaling is only supported for KVM.

11.2.2 How does autoscaling work?

The monitoring service, Monasca, monitors your infrastructure resources and generates alarms based on their state. The orchestration service, Heat, talks to the Monasca API and offers the capability to templatize the existing Monasca resources, which are the Monasca Notification and Monasca Alarm definition. Heat can configure certain alarms for the infrastructure resources (compute instances and block storage volumes) it creates and can expect Monasca to notify continuously if a certain evaluation pattern in an alarm definition is met.

For example, Heat can tell Monasca that it needs an alarm generated if the average CPU utilization of the compute instance in a scaling group goes beyond 90%.

As Monasca continuously monitors all the resources in the cloud, if it happens to see a compute instance spiking above 90% load as configured by Heat, it generates an alarm and in turn sends a notification to Heat. Once Heat is notified, it will execute an action that was preconfigured in the template. Commonly, this action will be a scale up to increase the number of compute instances to balance the load that is being taken by the compute instance scaling group.

Monasca sends a notification every 60 seconds while the alarm is in the ALARM state.

11.2.3 Autoscaling template example

The following Monasca alarm definition template snippet is an example of instructing Monasca to generate an alarm if the average CPU utilization in a group of compute instances exceeds beyond 50%. If the alarm is triggered, it will invoke the `up_notification` webhook once the alarm evaluation expression is satisfied.

```
cpu_alarm_high:
  type: OS::Monasca::AlarmDefinition
  properties:
    name: CPU utilization beyond 50 percent
    description: CPU utilization reached beyond 50 percent
    expression:
    str_replace:
    template: avg(cpu.utilization_perc{scale_group=scale_group_id}) > 50 times 3
    params:
    scale_group_id: {get_param: "OS::stack_id"}
    severity: high
    alarm_actions:
      - {get_resource: up_notification }
```

The following Monasca notification template snippet is an example of creating a Monasca notification resource that will be used by the alarm definition snippet to notify Heat.

```
up_notification:
  type: OS::Monasca::Notification
  properties:
    type: webhook
    address: {get_attr: [scale_up_policy, alarm_url]}
```

11.2.4 Monasca Agent configuration options

There is a Monasca Agent configuration option which controls the behavior around compute instance creation and the measurements being received from the compute instance.

The variable is `monasca_libvirt_vm_probation` which is set in the `~/openstack/my_cloud/config/nova/libvirt-monitoring.yml` file. Here is a snippet of the file showing the description and variable:

```
# The period of time (in seconds) in which to suspend metrics from a
# newly-created VM. This is used to prevent creating and storing
# quickly-obsolete metrics in an environment with a high amount of instance
# churn (VMs created and destroyed in rapid succession).  Setting to 0
```

```
# disables VM probation and metrics will be recorded as soon as possible
# after a VM is created.  Decreasing this value in an environment with a high
# amount of instance churn can have a large effect on the total number of
# metrics collected and increase the amount of CPU, disk space and network
# bandwidth required for Monasca. This value may need to be decreased if
# Heat Autoscaling is in use so that Heat knows that a new VM has been
# created and is handling some of the load.
monasca_libvirt_vm_probation: 300
```

The default value is `300`. This is the time in seconds that a compute instance must live before the Monasca libvirt agent plugin will send measurements for it. This is so that the Monasca metrics database does not fill with measurements from short lived compute instances. However, this means that the Monasca threshold engine will not see measurements from a newly created compute instance for at least five minutes on scale up. If the newly created compute instance is able to start handling the load in less than five minutes, then Heat autoscaling may mistakenly create another compute instance since the alarm does not clear.

If the default `monasca_libvirt_vm_probation` turns out to be an issue, it can be lowered. However, that will affect all compute instances, not just ones used by Heat autoscaling which can increase the number of measurements stored in Monasca if there are many short lived compute instances. You should consider how often compute instances are created that live less than the new value of `monasca_libvirt_vm_probation`. If few, if any, compute instances live less than the value of `monasca_libvirt_vm_probation`, then this value can be decreased without causing issues. If many compute instances live less than the `monasca_libvirt_vm_probation` period, then decreasing `monasca_libvirt_vm_probation` can cause excessive disk, CPU and memory usage by Monasca.

If you wish to change this value, follow these steps:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `monasca_libvirt_vm_probation` value in this configuration file:

   ```
   ~/openstack/my_cloud/config/nova/libvirt-monitoring.yml
   ```

3. Commit your changes to the local git:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add --all
   ardana > git commit -m "changing Monasca Agent configuration option"
   ```

4. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ```

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run this playbook to reconfigure the Nova service and enact your changes:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

12 Managing Monitoring, Logging, and Usage Reporting

Information about the monitoring, logging, and metering services included with your HPE Helion OpenStack.

12.1 Monitoring

The HPE Helion OpenStack Monitoring service leverages OpenStack Monasca, which is a multitenant, scalable, fault tolerant monitoring service.

12.1.1 Getting Started with Monitoring

You can use the HPE Helion OpenStack Monitoring service to monitor the health of your cloud and, if necessary, to troubleshoot issues.

Monasca data can be extracted and used for a variety of legitimate purposes, and different purposes require different forms of data sanitization or encoding to protect against invalid or malicious data. Any data pulled from Monasca should be considered untrusted data, so users are advised to apply appropriate encoding and/or sanitization techniques to ensure safe and correct usage and display of data in a web browser, database scan, or any other use of the data.

12.1.1.1 Monitoring Service Overview

12.1.1.1.1 Installation

The monitoring service is automatically installed as part of the HPE Helion OpenStack installation.

No specific configuration is required to use Monasca. However, you can configure the database for storing metrics as explained in *Section 12.1.2, "Configuring the Monitoring Service"*.

12.1.1.1.2 Differences Between Upstream and HPE Helion OpenStack Implementations

In HPE Helion OpenStack, the OpenStack monitoring service, Monasca, is included as the monitoring solution, except for the following which are not included:

- Transform Engine

- Events Engine

- Anomaly and Prediction Engine

Note

Icinga was supported in previous HPE Helion OpenStack versions but it has been deprecated in HPE Helion OpenStack 8.

12.1.1.1.3 Diagram of Monasca Service
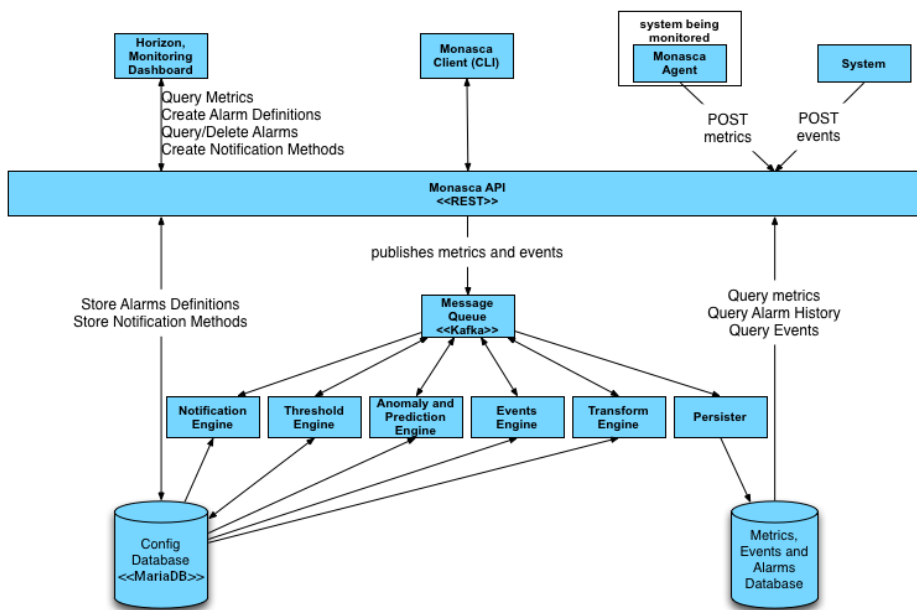
12.1.1.1.4 For More Information

For more details on OpenStack Monasca, see Monasca.io (http://monasca.io/) ↗

12.1.1.1.5   Back-end Database

The monitoring service default metrics database is Cassandra, which is a highly-scalable analytics database and the recommended database for HPE Helion OpenStack.

You can learn more about Cassandra at Apache Cassandra (http://cassandra.apache.org/) ↗.

12.1.1.2   Working with Monasca

Monasca-Agent

The **monasca-agent** is a Python program that runs on the control plane nodes. It runs the defined checks and then sends data onto the API. The checks that the agent runs include:

- System Metrics: CPU utilization, memory usage, disk I/O, network I/O, and filesystem utilization on the control plane and resource nodes.

- Service Metrics: the agent supports plugins such as MySQL, RabbitMQ, Kafka, and many others.

- VM Metrics: CPU utilization, disk I/O, network I/O, and memory usage of hosted virtual machines on compute nodes. Full details of these can be found https://github.com/openstack/monasca-agent/blob/master/docs/Plugins.md#per-instance-metrics ↗.

For a full list of packaged plugins that are included HPE Helion OpenStack, see Monasca Plugins (https://github.com/stackforge/monasca-agent/blob/master/docs/Plugins.md) ↗

You can further customize the monasca-agent to suit your needs, see Customizing the Agent (https://github.com/stackforge/monasca-agent/blob/master/docs/Customizations.md) ↗

12.1.1.3   Accessing the Monitoring Service

Access to the Monitoring service is available through a number of different interfaces.

12.1.1.3.1   Command-Line Interface

For users who prefer using the command line, there is the python-monascaclient, which is part of the default installation on your Cloud Lifecycle Manager node.

For details on the CLI, including installation instructions, see Python-Monasca Client (https://github.com/stackforge/python-monascaclient/blob/master/README.rst) ↗

Monasca API

If low-level access is desired, there is the Monasca REST API.

Full details of the Monasca API can be found on GitHub (https://github.com/stackforge/monasca-api/blob/master/docs/monasca-api-spec.md) ↗ .

12.1.1.3.2 Operations Console GUI

You can use the Operations Console (Ops Console) for HPE Helion OpenStack to view data about your HPE Helion OpenStack cloud infrastructure in a web-based graphical user interface (GUI) and ensure your cloud is operating correctly. By logging on to the console, HPE Helion OpenStack administrators can manage data in the following ways: **Triage alarm notifications.**

- Alarm Definitions and notifications now have their own screens and are collected under the **Alarm Explorer** menu item which can be accessed from the Central Dashboard. Central Dashboard now allows you to customize the view in the following ways:

  - Rename or re-configure existing alarm cards to include services different from the defaults

  - Create a new alarm card with the services you want to select

  - Reorder alarm cards using drag and drop

  - View all alarms that have no service dimension now grouped in an **Uncategorized Alarms** card

  - View all alarms that have a service dimension that does not match any of the other cards -now grouped in an **Other Alarms** card

- You can also easily access alarm data for a specific component. On the Summary page for the following components, a link is provided to an alarms screen specifically for that component:

  - Compute Instances: *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.3 "Managing Compute Hosts"*

  - Object Storage: *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.4 "Managing Swift Performance", Section 1.4.4 "Alarm Summary"*

12.1.1.3.3   Connecting to the Operations Console

To connect to Operations Console, perform the following:

- Ensure your login has the required access credentials: *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.2 "Connecting to the Operations Console", Section 1.2.1 "Required Access Credentials"*

- Connect through a browser: *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.2 "Connecting to the Operations Console", Section 1.2.2 "Connect Through a Browser"*

- Optionally use a Host name OR virtual IP address to access Operations Console: *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.2 "Connecting to the Operations Console", Section 1.2.3 "Optionally use a Hostname OR virtual IP address to access Operations Console"*

Operations Console will always be accessed over port 9095.

12.1.1.3.4   For More Information

For more details about the Operations Console, see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.1 "Operations Console Overview"*.

12.1.1.4   Service Alarm Definitions

HPE Helion OpenStack comes with some predefined monitoring alarms for the services installed. Full details of all service alarms can be found here: *Section 15.1.1, "Alarm Resolution Procedures"*. Each alarm will have one of the following statuses:

- *Critical* - Open alarms, identified by red indicator.

- *Warning* - Open alarms, identified by yellow indicator.

- *Unknown* - Open alarms, identified by gray indicator. Unknown will be the status of an alarm that has stopped receiving a metric. This can be caused by the following conditions:

  - An alarm exists for a service or component that is not installed in the environment.

  - An alarm exists for a virtual machine or node that previously existed but has been removed without the corresponding alarms being removed.

  - There is a gap between the last reported metric and the next metric.

- *Open* - Complete list of open alarms.

- *Total* - Complete list of alarms, may include Acknowledged and Resolved alarms.

When alarms are triggered it is helpful to review the service logs.

12.1.2   Configuring the Monitoring Service

The monitoring service, based on Monasca, allows you to configure an external SMTP server for email notifications when alarms trigger. You also have options for your alarm metrics database should you choose not to use the default option provided with the product.

In HPE Helion OpenStack you have the option to specify a SMTP server for email notifications and a database platform you want to use for the metrics database. These steps will assist in this process.

12.1.2.1   Configuring the Monitoring Email Notification Settings

The monitoring service, based on Monasca, allows you to configure an external SMTP server for email notifications when alarms trigger. In HPE Helion OpenStack, you have the option to specify a SMTP server for email notifications. These steps will assist in this process.

If you are going to use the email notifiication feature of the monitoring service, you must set the configuration options with valid email settings including an SMTP server and valid email addresses. The email server is not provided by HPE Helion OpenStack, but must be specified in the configuration file described below. The email server must support SMTP.

12.1.2.1.1 Configuring monitoring notification settings during initial installation

1. Log in to the Cloud Lifecycle Manager.

2. To change the SMTP server configuration settings edit the following file:

   ```
   ~/openstack/my_cloud/definition/cloudConfig.yml
   ```

   a. Enter your email server settings. Here is an example snippet showing the configuration file contents, uncomment these lines before entering your environment details.

   ```
   smtp-settings:
   #  server: mailserver.examplecloud.com
   #  port: 25
   #  timeout: 15
   # These are only needed if your server requires authentication
   #  user:
   #  password:
   ```

   This table explains each of these values:

Value	Description
Server (required)	The server entry must be uncommented and set to a valid hostname or IP Address.
Port (optional)	If your SMTP server is running on a port other than the standard 25, then uncomment the port line and set it your port.
Timeout (optional)	If your email server is heavily loaded, the timeout parameter can be uncommented and set to a larger value. 15 seconds is the default.

Value	Description
User / Password (optional)	If your SMTP server requires authentication, then you can configure user and password. Use double quotes around the password to avoid issues with special characters.

3. To configure the sending email addresses, edit the following file:

```
~/openstack/ardana/ansible/roles/monasca-notification/defaults/main.yml
```

Modify the following value to add your sending email address:

```
email_from_addr
```

> **Note**
>
> The default value in the file is `email_from_address: notification@exampleCloud.com` which you should edit.

4. [optional] To configure the receiving email addresses, edit the following file:

```
~/openstack/ardana/ansible/roles/monasca-default-alarms/defaults/main.yml
```

Modify the following value to configure a receiving email address:

```
notification_address
```

> **Note**
>
> You can also set the receiving email address via the Operations Console. Instructions for this are in the last section.

5. If your environment requires a proxy address then you can add that in as well:

```
# notification_environment can be used to configure proxies if needed.
# Below is an example configuration. Note that all of the quotes are required.
# notification_environment: '"http_proxy=http://<your_proxy>:<port>"
  "https_proxy=http://<your_proxy>:<port>"'
```

```
notification_environment: ''
```

6. Commit your configuration to the local Git repository (see *Book* "*Installing with Cloud Lifecycle Manager*"*, Chapter 10 "Using Git for Configuration Management*"), as follows:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "Updated monitoring service email notification settings"
```

7. Continue with your installation.

12.1.2.1.2   Monasca and Apache Commons validator

The Monasca notification uses a standard Apache Commons validator to validate the configured HPE Helion OpenStack domain names before sending the notification over webhook. Monasca notification supports some non-standard domain names, but not all. See the Domain Validator documentation for more information: https://commons.apache.org/proper/commons-validator/apidocs/org/apache/commons/validator/routines/DomainValidator.html ↗

You should ensure that any domains that you use are supported by IETF and IANA. As an example, **.local** is not listed by IANA and is invalid but **.gov** and **.edu** are valid.

- Internet Assigned Numbers Authority (IANA): https://www.iana.org/domains/root/db ↗

Failure to use supported domains will generate an unprocessable exception in Monasca notification create:

```
HTTPException code=422 message={"unprocessable_entity":
{"code":422,"message":"Address https://myopenstack.sample:8000/v1/signal/test is not of
 correct format","details":"","internal_code":"c6cf9d9eb79c3fc4"}
```

12.1.2.1.3   Configuring monitoring notification settings after the initial installation

If you need to make changes to the email notification settings after your initial deployment, you can change the "From" address using the configuration files but the "To" address will need to be changed in the Operations Console. The following section will describe both of these processes.

To change the sending email address:

1. Log in to the Cloud Lifecycle Manager.

2. To configure the sending email addresses, edit the following file:

   ```
   ~/openstack/ardana/ansible/roles/monasca-notification/defaults/main.yml
   ```

   Modify the following value to add your sending email address:

   ```
   email_from_addr
   ```

   > **Note**
   >
   > The default value in the file is `email_from_address: notification@exampleCloud.com` which you should edit.

3. Commit your configuration to the local Git repository (*Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "Updated monitoring service email notification settings"
   ```

4. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

5. Update your deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

6. Run the Monasca reconfigure playbook to deploy the changes:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts monasca-reconfigure.yml --tags
    notification
   ```

Note

You may need to use the `--ask-vault-pass` switch if you opted for encryption during the initial deployment.

To change the receiving email address via the Operations Console:

To configure the "To" email address, after installation,

1. Connect to and log in to the Operations Console. See *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.2 "Connecting to the Operations Console"* for assistance.

2. On the **Home** screen, click the menu represented by 3 horizontal lines (≡).

3. From the menu that slides in on the left side, click **Home**, and then **Alarm Explorer**.

4. On the **Alarm Explorer** page, at the top, click the **Notification Methods** text.

5. On the **Notification Methods** page, find the row with the **Default Email** notification.

6. In the **Default Email** row, click the details icon (⋯), then click **Edit**.

7. On the **Edit Notification Method: Default Email** page, in **Name**, **Type**, and **Address/Key**, type in the values you want to use.

8. On the **Edit Notification Method: Default Email** page, click **Update Notification**.

Important

Once the notification has been added, using the procedures using the Ansible playbooks will not change it.

12.1.2.2   Managing Notification Methods for Alarms

- *Section 12.1.2.2.1, "Enabling a Proxy for Webhook or Pager Duty Notifications"*

- *Section 12.1.2.2.2, "Creating a New Notification Method"*

- *Section 12.1.2.2.3, "Applying a Notification Method to an Alarm Definition"*

12.1.2.2.1   Enabling a Proxy for Webhook or Pager Duty Notifications

If your environment requires a proxy in order for communications to function then these steps will show you how you can enable one. These steps will only be needed if you are utilizing the webhook or pager duty notification methods.

These steps will require access to the Cloud Lifecycle Manager in your cloud deployment so you may need to contact your Administrator. You can make these changes during the initial configuration phase prior to the first installation or you can modify your existing environment, the only difference being the last step.

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/ardana/ansible/roles/monasca-notification/defaults/main.yml` file and edit the line below with your proxy address values:

   ```
   notification_environment: '"http_proxy=http://<proxy_address>:<port>"
    "https_proxy=<http://proxy_address>:<port>"'
   ```

   ### ☞ Note

   > There are single quotation marks around the entire value of this entry and then double quotation marks around the individual proxy entries. This formatting must exist when you enter these values into your configuration file.

3. If you are making these changes prior to your initial installation then you are done and can continue on with the installation. However, if you are modifying an existing environment, you will need to continue on with the remaining steps below.

4. Commit your configuration to the local Git repository (see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*), as follows:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

5. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

6. Generate an updated deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Monasca reconfigure playbook to enable these changes:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-reconfigure.yml --tags
 notification
```

12.1.2.2.2 Creating a New Notification Method

1. Log in to the Operations Console. For more information, see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.2 "Connecting to the Operations Console"*.

2. Use the navigation menu to go to the **Alarm Explorer** page:
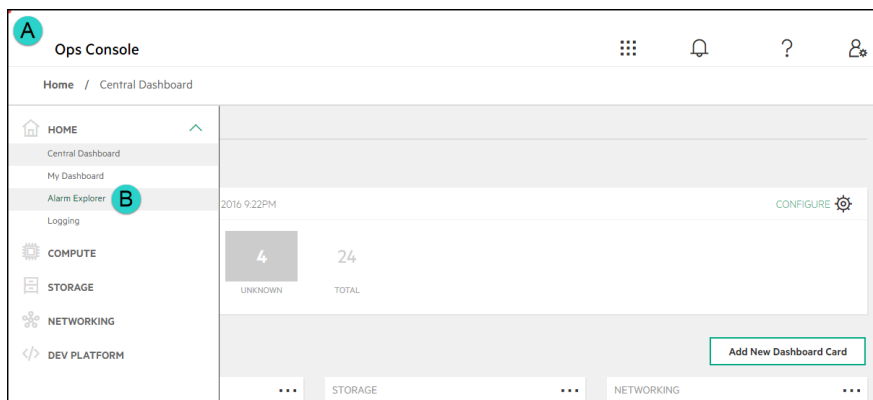
3. Select the **Notification Methods** menu and then click the **Create Notification Method** button:
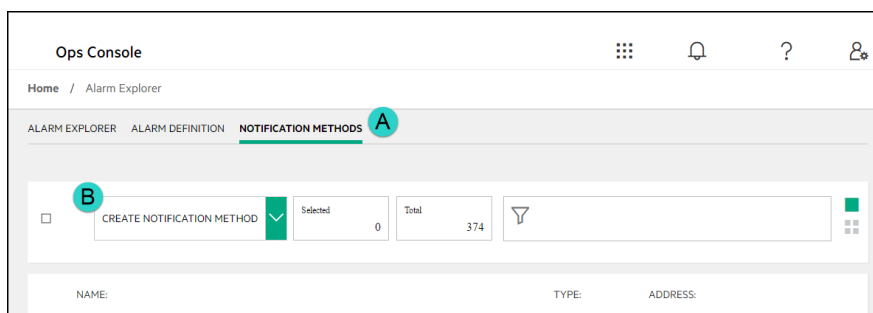
4. On the **Create Notification Method** window you will select your options and then click the **Create Notification** button.

A description of each of the fields you use for each notification method:

Field	Description
Name	Enter a unique name value for the notification method you are creating.
Type	Choose a type. Available values are **Webhook**, **Email**, or **Pager Duty**.
Address/Key	Enter the value corresponding to the type you chose.

12.1.2.2.3 Applying a Notification Method to an Alarm Definition

1. Log in to the Operations Console. For more informalfigure, see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.2 "Connecting to the Operations Console".*

2. Use the navigation menu to go to the **Alarm Explorer** page:
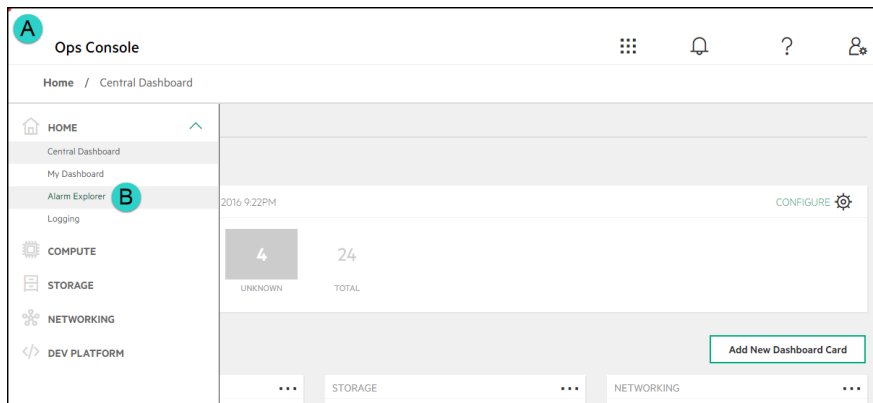
3. Select the **Alarm Definition** menu which will give you a list of each of the alarm definitions in your environment.
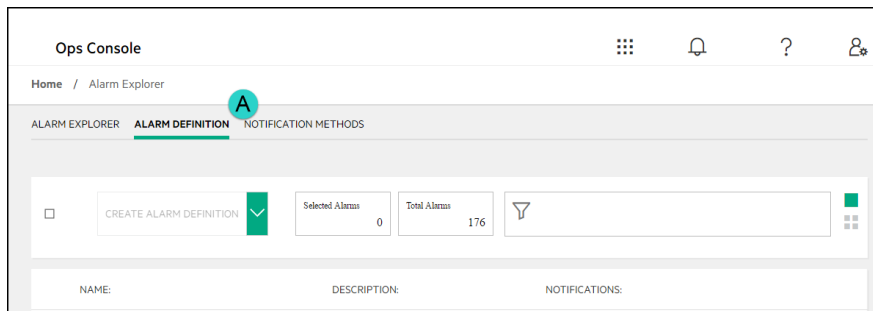
4. Locate the alarm you want to change the notification method for and click on its name to bring up the edit menu. You can use the sorting methods for assistance.

5. In the edit menu, scroll down to the **Notifications and Severity** section where you will select one or more **Notification Methods** before selecting the **Update Alarm Definition** button:

6. Repeat as needed until all of your alarms have the notification methods you desire.

12.1.2.3 Enabling the RabbitMQ Admin Console

The RabbitMQ Admin Console is off by default in HPE Helion OpenStack. You can turn on the console by following these steps:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/config/rabbitmq/main.yml` file. Under the `rabbit_plugins:` line, uncomment

```
- rabbitmq_management
```

3. Commit your configuration to the local Git repository (see *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "Enabled RabbitMQ Admin Console"
```

4. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the RabbitMQ reconfigure playbook to deploy the changes:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts rabbitmq-reconfigure.yml
```

To turn the RabbitMQ Admin Console off again, add the comment back and repeat steps 3 through 6.

12.1.2.4 Capacity Reporting and Monasca Transform

Capacity reporting is a new feature in HPE Helion OpenStack which will provide cloud operators overall capacity (available, used, and remaining) information via the Operations Console so that the cloud operator can ensure that cloud resource pools have sufficient capacity to meet the demands of users. The cloud operator is also able to set thresholds and set alarms to be notified when the thresholds are reached.

For Compute

- Host Capacity - CPU/Disk/Memory: Used, Available and Remaining Capacity - for the entire cloud installation or by host

- VM Capacity - CPU/Disk/Memory: Allocated, Available and Remaining - for the entire cloud installation, by host or by project

For Object Storage

- Disk Capacity - Used, Available and Remaining Capacity - for the entire cloud installation or by project

In addition to overall capacity, roll up views with appropriate slices provide views by a particular project, or compute node. Graphs also show trends and the change in capacity over time.

12.1.2.4.1 Monasca Transform Features

- Monasca Transform is a new component in Monasca which transforms and aggregates metrics using Apache Spark

- Aggregated metrics are published to Kafka and are available for other monasca components like monasca-threshold and are stored in monasca datastore

- Cloud operators can set thresholds and set alarms to receive notifications when thresholds are met.

- These aggregated metrics are made available to the cloud operators via Operations Console's new Capacity Summary (reporting) UI

- Capacity reporting is a new feature in HPE Helion OpenStack which will provides cloud operators an overall capacity (available, used and remaining) for Compute and Object Storage

- Cloud operators can look at Capacity reporting via Operations Console's Compute Capacity Summary and Object Storage Capacity Summary UI

- Capacity reporting allows the cloud operators the ability to ensure that cloud resource pools have sufficient capacity to meet demands of users. See table below for Service and Capacity Types.

- A list of aggregated metrics is provided in *Section 12.1.2.4.4, "New Aggregated Metrics"*.

- Capacity reporting aggregated metrics are aggregated and published every hour

- In addition to the overall capacity, there are graphs which show the capacity trends over time range (for 1 day, for 7 days, for 30 days or for 45 days)

- Graphs showing the capacity trends by a particular project or compute host are also provided.

- Monasca Transform is integrated with centralized monitoring (Monasca) and centralized logging

- Flexible Deployment

- Upgrade & Patch Support

Service	Type of Capacity	Description
Compute	Host Capacity	CPU/Disk/Memory: Used, Available and Remaining Capacity - for entire cloud installation or by compute host
	VM Capacity	CPU/Disk/Memory: Allocated, Available and Remaining - for entire cloud installation, by host or by project
Object Storage	Disk Capacity	Used, Available and Remaining Disk Capacity - for entire cloud installation or by project
	Storage Capacity	Utilized Storage Capacity - for entire cloud installation or by project

12.1.2.4.2 Architecture for Monasca Transform and Spark

Monasca Transform is a new component in Monasca. Monasca Transform uses Spark for data aggregation. Both Monasca Transform and Spark are depicted in the example diagram below.
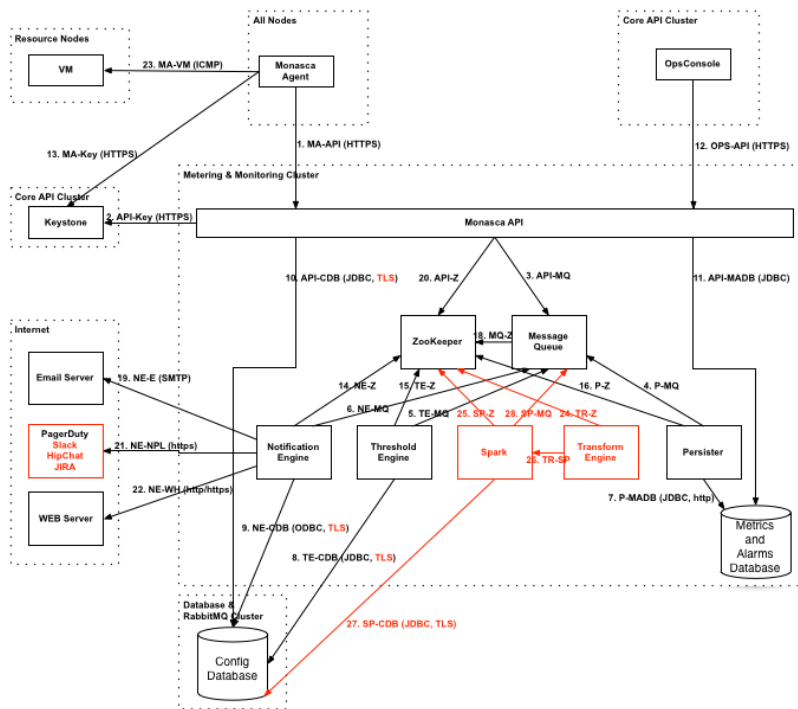
You can see that the Monasca components run on the Cloud Controller nodes, and the Monasca agents run on all nodes in the Mid-scale Example configuration.
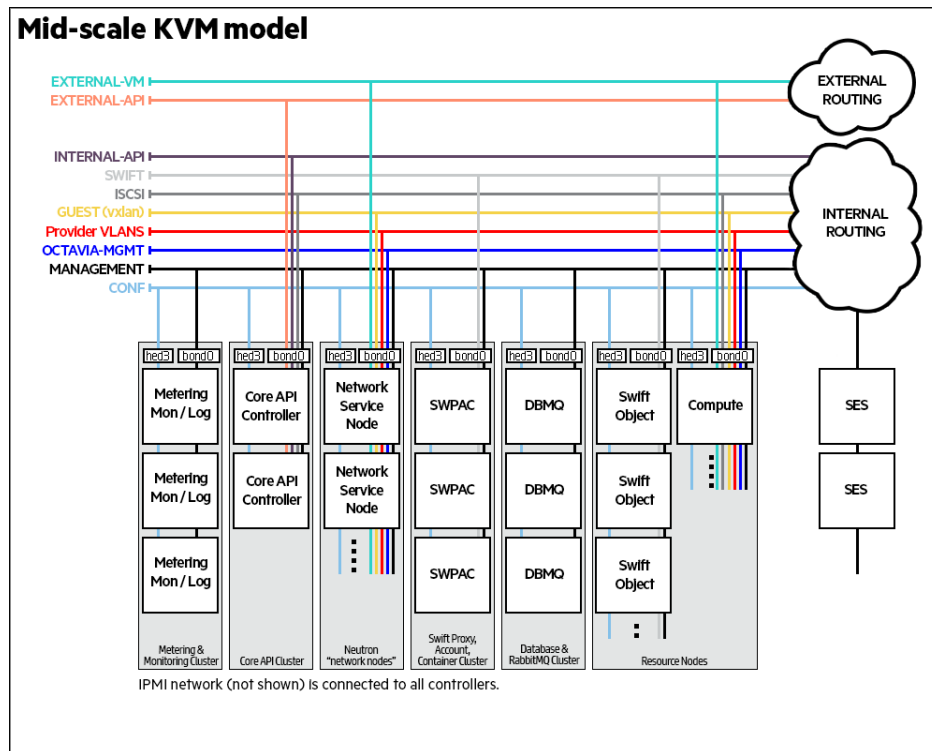
12.1.2.4.3   Components for Capacity Reporting

12.1.2.4.3.1   Monasca Transform: Data Aggregation Reporting

Monasca-transform is a new component which provides mechanism to aggregate or transform metrics and publish new aggregated metrics to Monasca.

Monasca Transform is a data driven Apache Spark based data aggregation engine which collects, groups and aggregates existing individual Monasca metrics according to business requirements and publishes new transformed (derived) metrics to the Monasca Kafka queue.

Since the new transformed metrics are published as any other metric in Monasca, alarms can be set and triggered on the transformed metric, just like any other metric.

12.1.2.4.3.2   Object Storage and Compute Capacity Summary Operations Console UI

A new "Capacity Summary" tab for Compute and Object Storage will displays all the aggregated metrics under the "Compute" and "Object Storage" sections.

Operations Console UI makes calls to Monasca API to retrieve and display various tiles and graphs on Capacity Summary tab in Compute and Object Storage Summary UI pages.

12.1.2.4.3.3   Persist new metrics and Trigger Alarms

New aggregated metrics will be published to Monasca's Kafka queue and will be ingested by monasca-persister. If thresholds and alarms have been set on the aggregated metrics, Monasca will generate and trigger alarms as it currently does with any other metric. No new/additional change is expected with persisting of new aggregated metrics or setting threshold/alarms.

12.1.2.4.4   New Aggregated Metrics

Following is the list of aggregated metrics produced by monasca transform in HPE Helion OpenStack

TABLE 12.1: AGGREGATED METRICS

	Metric Name	For	Description	Dimensions	Notes
1	cpu.utilized_logical_cores_agg	compute summary	utilized physical host cpu core capacity for one or all hosts by time interval (defaults to a hour)	aggregation_period: hourly host: all or <host name> project_id: all	Available as total or per host
2	cpu.total_logical_cores_agg	compute summary	total physical host cpu core capacity for one or all hosts by time interval (defaults to a hour)	aggregation_period: hourly host: all or <host name> project_id: all	Available as total or per host

	Metric Name	For	Description	Dimensions	Notes
3	mem.total_mb_agg	compute summary	total physical host memory capacity by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	
4	mem.usable_mb_agg	compute summary	usable physical host memory capacity by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	
5	disk.total_used_space_mb_agg	compute summary	utilized physical host disk capacity by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	
6	disk.total_space_mb_agg	compute summary	total physical host disk capacity by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	
7	nova.vm.cpu.total_allocated_agg	compute summary	cpus allocated across all VMs by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	
8	vcpus_agg	compute summary	virtual cpus allocated capacity for VMs of one or all projects by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all or <project ID>	Available as total or per project

	Metric Name	For	Description	Dimensions	Notes
9	nova.vm.mem.total_allocated_mb_agg	compute summary	memory allocated to all VMs by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	
10	vm.mem.used_mb_agg	compute summary	memory utilized by VMs of one or all projects by time interval (defaults to an hour)	aggregation_period: hourly host: all project_id: <project ID>	Available as total or per project
11	vm.mem.total_mb_agg	compute summary	memory allocated to VMs of one or all projects by time interval (defaults to an hour)	aggregation_period: hourly host: all project_id: <project ID>	Available as total or per project
12	vm.cpu.utilization_perc_agg	compute summary	cpu utilized by all VMs by project by time interval (defaults to an hour)	aggregation_period: hourly host: all project_id: <project ID>	
13	nova.vm.disk.total_allocated_gb_agg	compute summary	disk space allocated to all VMs by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	

Configuring the Monitoring Service   HPE Helion OpenStack 8

	Metric Name	For	Description	Dimensions	Notes
14	vm.disk.allocation_agg	compute summary	disk allocation for VMs of one or all projects by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all or < project ID >	Available as total or per project
15	swiftlm.diskusage.val.size_agg	object storage summary	total available object storage capacity by time interval (defaults to a hour)	aggregation_period: hourly host: all or < host name > project_id: all	Available as total or per host
16	swiftlm.diskusage.val.avail_agg	object storage summary	remaining object storage capacity by time interval (defaults to a hour)	aggregation_period: hourly host: all or < host name > project_id: all	Available as total or per host
17	swiftlm.diskusage.rate_agg	object storage summary	rate of change of object storage usage by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	
18	storage.objects.size_agg	object storage summary	used object storage capacity by time interval (defaults to a hour)	aggregation_period: hourly host: all project_id: all	

Configuring the Monitoring Service   HPE Helion OpenStack 8

12.1.2.4.5　Deployment

Monasca Transform and Spark will be deployed on the same control plane nodes along with Logging and Monitoring Service (Monasca).

Security Consideration during deployment of Monasca Transform and Spark

The HPE Helion OpenStack Monitoring system connects internally to the Kafka and Spark technologies without authentication. If you choose to deploy Monitoring, configure it to use only trusted networks such as the Management network, as illustrated on the network diagrams below for Entry Scale Deployment and Mid Scale Deployment.

Entry Scale Deployment

In Entry Scale Deployment Monasca Transform and Spark will be deployed on Shared Control Plane along with other Openstack Services along with Monitoring and Logging
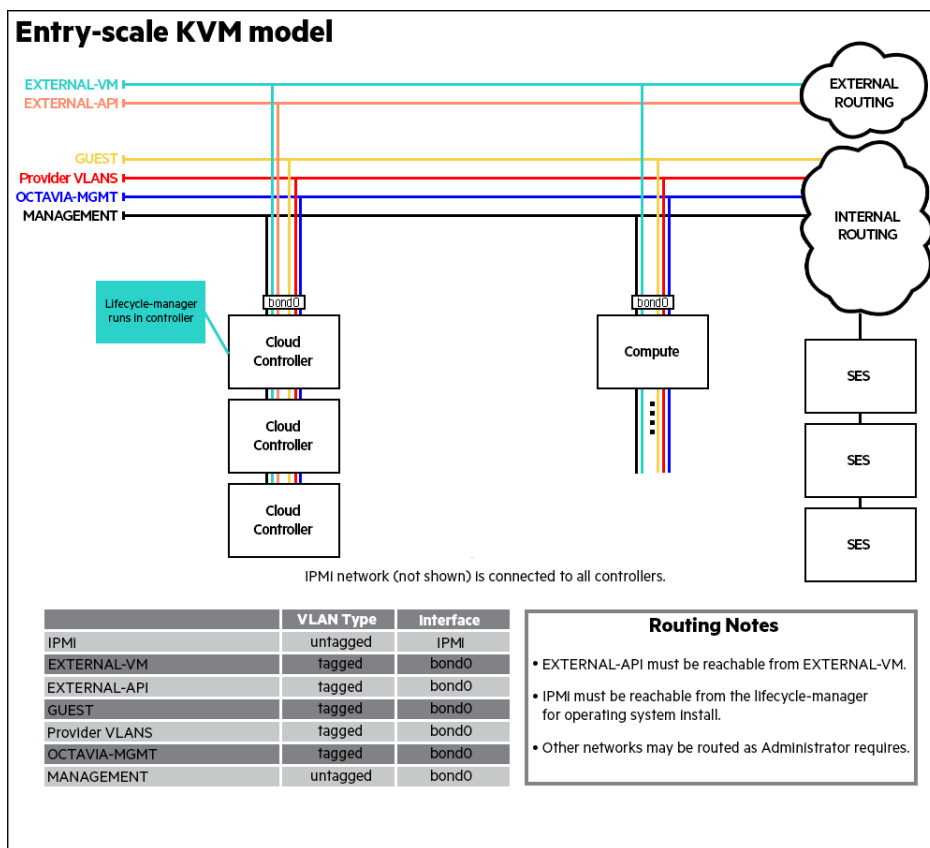
Mid scale Deployment

In a Mid Scale Deployment Monasca Transform and Spark will be deployed on dedicated Metering Monitoring and Logging (MML) control plane along with other data processing intensive services like Metering, Monitoring and Logging

Mid-scale KVM model

IPMI network (not shown) is connected to all controllers.
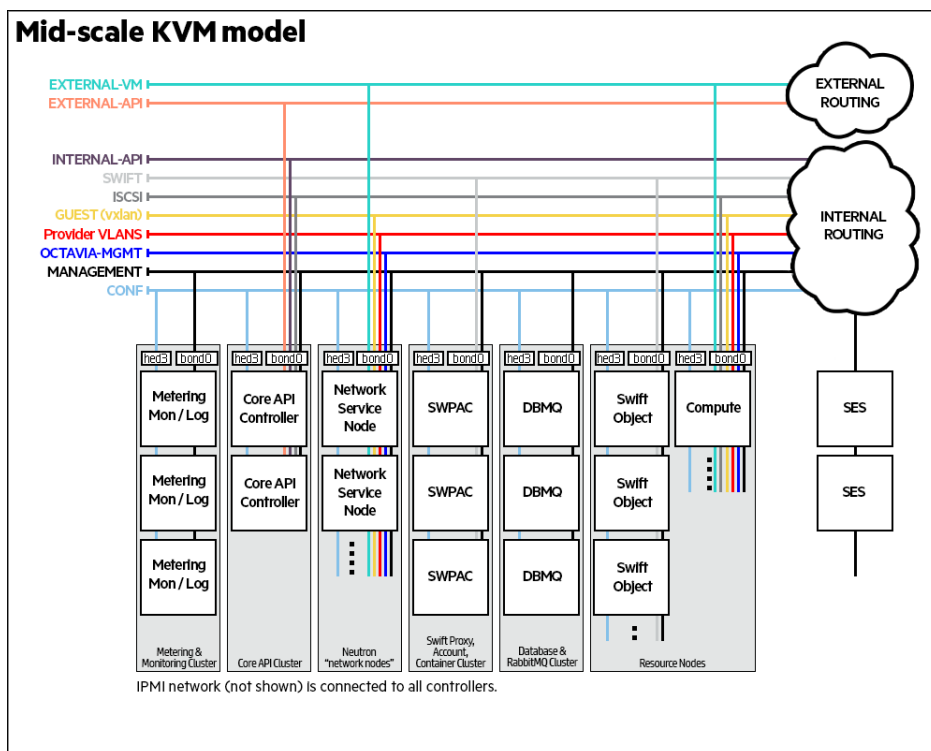
Multi Control Plane Deployment

In a Multi Control Plane Deployment, Monasca Transform and Spark will be deployed on the Shared Control plane along with rest of Monasca Components.

Start, Stop and Status for Monasca Transform and Spark processes

The service management methods for monasca-transform and spark follow the convention for services in the OpenStack platform. When executing from the deployer node, the commands are as follows:

Status

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts spark-status.yml
ardana > ansible-playbook -i hosts/verb_hosts monasca-transform-status.yml
```

Start

As monasca-transform depends on spark for the processing of the metrics spark will need to be started before monasca-transform.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts spark-start.yml
ardana > ansible-playbook -i hosts/verb_hosts monasca-transform-start.yml
```

Stop

Configuring the Monitoring Service   HPE Helion OpenStack 8

As a precaution, stop the monasca-transform service before taking spark down. Interruption to the spark service altogether while monasca-transform is still running can result in a monasca-transform process that is unresponsive and needing to be tidied up.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-transform-stop.yml
ardana > ansible-playbook -i hosts/verb_hosts spark-stop.yml
```

12.1.2.4.6   Reconfigure

The reconfigure process can be triggered again from the deployer. Presuming that changes have been made to the variables in the appropriate places execution of the respective ansible scripts will be enough to update the configuration. The spark reconfigure process alters the nodes serially meaning that spark is never down altogether, each node is stopped in turn and zookeeper manages the leaders accordingly. This means that monasca-transform may be left running even while spark is upgraded.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts spark-reconfigure.yml
ardana > ansible-playbook -i hosts/verb_hosts monasca-transform-reconfigure.yml
```

12.1.2.4.7   Adding Monasca Transform and Spark to HPE Helion OpenStack Deployment

Since Monasca Transform and Spark are optional components, the users might elect to not install these two components during their initial HPE Helion OpenStack install. The following instructions provide a way the users can add Monasca Transform and Spark to their existing HPE Helion OpenStack deployment.

Steps

1. Add Monasca Transform and Spark to the input model. Monasca Transform and Spark on a entry level cloud would be installed on the common control plane, for mid scale cloud which has a MML (Metering, Monitoring and Logging) cluster, Monasca Transform and Spark will should be added to MML cluster.

   ```
   ardana > cd ~/openstack/my_cloud/definition/data/
   ```

   Add spark and monasca-transform to input model, control_plane.yml

```
clusters
      - name: core
        cluster-prefix: c1
        server-role: CONTROLLER-ROLE
        member-count: 3
        allocation-policy: strict
        service-components:

          [...]

          - zookeeper
          - kafka
          - cassandra
          - storm
          - spark
          - monasca-api
          - monasca-persister
          - monasca-notifier
          - monasca-threshold
          - monasca-client
          - monasca-transform

          [...]
```

2. Run the Configuration Processor

```
ardana > cd ~/openstack/my_cloud/definition
ardana > git add -A
ardana > git commit -m "Adding Monasca Transform and Spark"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

3. Run Ready Deployment

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

4. Run Cloud Lifecycle Manager Deploy

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-deploy.yml
```

Verify Deployment

Login to each controller node and run

```
tux > sudo service monasca-transform status
tux > sudo service spark-master status
tux > sudo service spark-worker status
```

```
tux > sudo service monasca-transform status
● monasca-transform.service - Monasca Transform Daemon
  Loaded: loaded (/etc/systemd/system/monasca-transform.service; disabled)
  Active: active (running) since Wed 2016-08-24 00:47:56 UTC; 2 days ago
Main PID: 7351 (bash)
  CGroup: /system.slice/monasca-transform.service
          ├─ 7351 bash /etc/monasca/transform/init/start-monasca-transform.sh
          ├─ 7352 /opt/stack/service/monasca-transform/venv//bin/python /opt/monasca/
monasca-transform/lib/service_runner.py
          ├─27904 /bin/sh -c export SPARK_HOME=/opt/stack/service/spark/venv/bin/../
current && spark-submit --supervise --master spark://omega-cp1-c1-m1-mgmt:7077,omega-cp1-
c1-m2-mgmt:7077,omega-cp1-c1...
          ├─27905 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /opt/stack/service/
spark/venv/lib/drizzle-jdbc-1.3.jar:/opt/stack/service/spark/venv/bin/../current/conf/:/
opt/stack/service/spark/v...
          └─28355 python /opt/monasca/monasca-transform/lib/driver.py
Warning: Journal has been rotated since unit was started. Log output is incomplete or
 unavailable.


tux > sudo service spark-worker status
● spark-worker.service - Spark Worker Daemon
  Loaded: loaded (/etc/systemd/system/spark-worker.service; disabled)
  Active: active (running) since Wed 2016-08-24 00:46:05 UTC; 2 days ago
Main PID: 63513 (bash)
  CGroup: /system.slice/spark-worker.service
          ├─ 7671 python -m pyspark.daemon
          ├─28948 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /opt/stack/service/
spark/venv/bin/../current/conf/:/opt/stack/service/spark/venv/bin/../current/lib/spark-
assembly-1.6.1-hadoop2.6.0...
          ├─63513 bash /etc/spark/init/start-spark-worker.sh &
          └─63514 /usr/bin/java -cp /opt/stack/service/spark/venv/bin/../current/conf/:/
opt/stack/service/spark/venv/bin/../current/lib/spark-assembly-1.6.1-hadoop2.6.0.jar:/
opt/stack/service/spark/ven...
Warning: Journal has been rotated since unit was started. Log output is incomplete or
 unavailable.



tux > sudo service spark-master status
● spark-master.service - Spark Master Daemon
  Loaded: loaded (/etc/systemd/system/spark-master.service; disabled)
  Active: active (running) since Wed 2016-08-24 00:44:24 UTC; 2 days ago
```

```
Main PID: 55572 (bash)
  CGroup: /system.slice/spark-master.service
          ├─55572 bash /etc/spark/init/start-spark-master.sh &
          └─55573 /usr/bin/java -cp /opt/stack/service/spark/venv/bin/../current/conf/:/
opt/stack/service/spark/venv/bin/../current/lib/spark-assembly-1.6.1-hadoop2.6.0.jar:/
opt/stack/service/spark/ven...
Warning: Journal has been rotated since unit was started. Log output is incomplete or
 unavailable.
```

12.1.2.4.8   Increase Monasca Transform Scale

Monasca Transform in the default configuration can scale up to estimated data for 100 node cloud deployment. Estimated maximum rate of metrics from a 100 node cloud deployment is 120M/hour.

You can further increase the processing rate to 180M/hour. Making the Spark configuration change will increase the CPU's being used by Spark and Monasca Transform from average of around 3.5 to 5.5 CPU's per control node over a 10 minute batch processing interval.

To increase the processing rate to 180M/hour the customer will have to make following spark configuration change.

Steps

1. Edit /var/lib/ardana/openstack/my_cloud/config/spark/spark-defaults.conf.j2 and set spark.cores.max to 6 and spark.executor.cores 2
   Set spark.cores.max to 6

   ```
   spark.cores.max {{ spark_cores_max }}
   ```

   to

   ```
   spark.cores.max 6
   ```

   Set spark.executor.cores to 2

   ```
   spark.executor.cores {{ spark_executor_cores }}
   ```

   to

   ```
   spark.executor.cores 2
   ```

2. Edit ~/openstack/my_cloud/config/spark/spark-env.sh.j2

Set SPARK_WORKER_CORES to 2

```
export SPARK_WORKER_CORES={{ spark_worker_cores }}
```

to

```
export SPARK_WORKER_CORES=2
```

3. Edit ~/openstack/my_cloud/config/spark/spark-worker-env.sh.j2
   Set SPARK_WORKER_CORES to 2

```
export SPARK_WORKER_CORES={{ spark_worker_cores }}
```

to

```
export SPARK_WORKER_CORES=2
```

4. Run Configuration Processor

```
ardana > cd ~/openstack/my_cloud/definition
ardana > git add -A
ardana > git commit -m "Changing Spark Config increase scale"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Run Ready Deployment

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run spark-reconfigure.yml and monasca-transform-reconfigure.yml

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts spark-reconfigure.yml
ardana > ansible-playbook -i hosts/verb_hosts monasca-transform-reconfigure.yml
```

12.1.2.4.9 Change Compute Host Pattern Filter in Monasca Transform

Monasca Transform identifies compute host metrics by pattern matching on hostname dimension in the incoming monasca metrics. The default pattern is of the form `compNNN`. For example, `comp001`, `comp002`, etc. To filter for it in the transformation specs, use the expression `-comp[0-9]+-`. In case the compute host names follow a different pattern other than the standard pattern above, the filter by expression when aggregating metrics will have to be changed.

Steps

1. On the deployer: Edit `~/openstack/my_cloud/config/monasca-transform/transform_specs.json.j2`

2. Look for all references of `-comp[0-9]+-` and change the regular expression to the desired pattern say for example `-compute[0-9]+-`.

```
{"aggregation_params_map":{"aggregation_pipeline":{"source":"streaming",
 "usage":"fetch_quantity", "setters":["rollup_quantity",
 "set_aggregated_metric_name", "set_aggregated_period"],
 "insert":["prepare_data","insert_data_pre_hourly"]},
 "aggregated_metric_name":"mem.total_mb_agg", "aggregation_period":"hourly",
 "aggregation_group_by_list": ["host", "metric_id", "tenant_id"],
 "usage_fetch_operation": "avg", "filter_by_list": [{"field_to_filter":
 "host", "filter_expression": "-comp[0-9]+", "filter_operation":
 "include"}], "setter_rollup_group_by_list":[], "setter_rollup_operation":
 "sum", "dimension_list":["aggregation_period", "host", "project_id"],
 "pre_hourly_operation":"avg", "pre_hourly_group_by_list":["default"]},
 "metric_group":"mem_total_all", "metric_id":"mem_total_all"}
```

to

```
{"aggregation_params_map":{"aggregation_pipeline":{"source":"streaming",
 "usage":"fetch_quantity", "setters":["rollup_quantity",
 "set_aggregated_metric_name", "set_aggregated_period"], "insert":["prepare_data",
 "insert_data_pre_hourly"]}, "aggregated_metric_name":"mem.total_mb_agg",
 "aggregation_period":"hourly", "aggregation_group_by_list": ["host", "metric_id",
 "tenant_id"],"usage_fetch_operation": "avg","filter_by_list": [{"field_to_filter":
 "host","filter_expression": "-compute[0-9]+", "filter_operation":
 "include"}], "setter_rollup_group_by_list":[], "setter_rollup_operation":
 "sum", "dimension_list":["aggregation_period", "host", "project_id"],
 "pre_hourly_operation":"avg", "pre_hourly_group_by_list":["default"]},
 "metric_group":"mem_total_all", "metric_id":"mem_total_all"}
```

> 📖 **Note**
>
> The filter_expression has been changed to the *new* pattern.

3. To change all host metric transformation specs in the same JSON file, repeat *Step 2*.
   Transformation specs will have to be changed for following metric_ids namely "mem_total_all", "mem_usable_all", "disk_total_all", "disk_usable_all", "cpu_total_all", "cpu_total_host", "cpu_util_all", "cpu_util_host"

4. Run the Configuration Processor:

```
ardana > cd ~/openstack/my_cloud/definition
ardana > git add -A
ardana > git commit -m "Changing Monasca Transform specs"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Run Ready Deployment:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run Monasca Transform Reconfigure:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-transform-reconfigure.yml
```

12.1.2.5   Configuring Availability of Alarm Metrics

Using the Monasca agent tuning knobs, you can choose which alarm metrics are available in your environment.

The addition of the libvirt and OVS plugins to the Monasca agent provides a number of additional metrics that can be used. Most of these metrics are included by default, but others are not. You have the ability to use tuning knobs to add or remove these metrics to your environment based on your individual needs in your cloud.

We will list these metrics along with the tuning knob name and instructions for how to adjust these.

12.1.2.5.1 Libvirt plugin metric tuning knobs

The following metrics are added as part of the libvirt plugin:

🎓 Note

For a description of each of these metrics, see *Section 12.1.4.11, "Libvirt Metrics"*.

Tuning Knob	Default Setting	Admin Metric Name	Project Metric Name
vm_cpu_check_enable	True	vm.cpu.time_ns	cpu.time_ns
		vm.cpu.utilization_norm_perc	cpu.utilization_norm_perc
		vm.cpu.utilization_perc	cpu.utilization_perc
vm_disks_check_enable	True Creates 20 disk metrics per disk device per virtual machine.	vm.io.errors	io.errors
		vm.io.errors_sec	io.errors_sec
		vm.io.read_bytes	io.read_bytes
		vm.io.read_bytes_sec	io.read_bytes_sec
		vm.io.read_ops	io.read_ops
		vm.io.read_ops_sec	io.read_ops_sec
		vm.io.write_bytes	io.write_bytes
		vm.io.write_bytes_sec	io.write_bytes_sec
		vm.io.write_ops	io.write_ops
		vm.io.write_ops_sec	io.write_ops_sec
vm_network_check_enable	True Creates 16 network metrics per NIC per virtual machine.	vm.net.in_bytes	net.in_bytes
		vm.net.in_bytes_sec	net.in_bytes_sec
		vm.net.in_packets	net.in_packets
		vm.net.in_packets_sec	net.in_packets_sec
		vm.net.out_bytes	net.out_bytes

Tuning Knob	Default Setting	Admin Metric Name	Project Metric Name
		vm.net.out_bytes_sec	net.out_bytes_sec
		vm.net.out_packets	net.out_packets
		vm.net.out_packets_sec	net.out_packets_sec
vm_ping_check_enable	True	vm.ping_status	ping_status
vm_extended_disks_check_enable	True Creates 6 metrics per device per virtual machine.	vm.disk.allocation	disk.allocation
		vm.disk.capacity	disk.capacity
		vm.disk.physical	disk.physical
	True Creates 6 aggregate metrics per virtual machine.	vm.disk.allocation_total	disk.allocation_total
		vm.disk.capacity_total	disk.capacity.total
		vm.disk.physical_total	disk.physical_total
vm_disks_check_enable vm_extended_disks_check_enable	True Creates 20 aggregate metrics per virtual machine.	vm.io.errors_total	io.errors_total
		vm.io.errors_total_sec	io.errors_total_sec
		vm.io.read_bytes_total	io.read_bytes_total
		vm.io.read_bytes_total_sec	io.read_bytes_total_sec
		vm.io.read_ops_total	io.read_ops_total
		vm.io.read_ops_total_sec	io.read_ops_total_sec
		vm.io.write_bytes_total	io.write_bytes_total
		vm.io.write_bytes_total_sec	io.write_bytes_total_sec
		vm.io.write_ops_total	io.write_ops_total
		vm.io.write_ops_total_sec	io.write_ops_total_sec

12.1.2.5.1.1 Configuring the libvirt metrics using the tuning knobs

Use the following steps to configure the tuning knobs for the libvirt plugin metrics.

1. Log in to the Cloud Lifecycle Manager.

2. Edit the following file:

```
~/openstack/my_cloud/config/nova/libvirt-monitoring.yml
```

3. Change the value for each tuning knob to the desired setting, `True` if you want the metrics created and `False` if you want them removed. Refer to the table above for which metrics are controlled by each tuning knob.

```
vm_cpu_check_enable: <true or false>
vm_disks_check_enable: <true or false>
vm_extended_disks_check_enable: <true or false>
vm_network_check_enable: <true or false>
vm_ping_check_enable: <true or false>
```

4. Commit your configuration to the local Git repository (see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "configuring libvirt plugin tuning knobs"
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the Nova reconfigure playbook to implement the changes:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

Note

If you modify either of the following files, then the monasca tuning parameters should be adjusted to handle a higher load on the system.

```
~/openstack/my_cloud/config/nova/libvirt-monitoring.yml
~/openstack/my_cloud/config/neutron/monasca_ovs_plugin.yaml.j2
```

Tuning parameters are located in `~/openstack/my_cloud/config/monasca/configuration.yml`. The parameter `monasca_tuning_selector_override` should be changed to the `extra-large` setting.

12.1.2.5.2 OVS plugin metric tuning knobs

The following metrics are added as part of the OVS plugin:

📖 Note

For a description of each of these metrics, see *Section 12.1.4.16, "Open vSwitch (OVS) Metrics"*.

Tuning Knob	Default Setting	Admin Metric Name	Project Metric Name
use_rate_metrics	False	ovs.vrouter.in_bytes_sec	vrouter.in_bytes_sec
		ovs.vrouter.in_packets_sec	vrouter.in_packets_sec
		ovs.vrouter.out_bytes_sec	vrouter.out_bytes_sec
		ovs.vrouter.out_packets_sec	vrouter.out_packets_sec
use_absolute_metrics	True	ovs.vrouter.in_bytes	vrouter.in_bytes
		ovs.vrouter.in_packets	vrouter.in_packets
		ovs.vrouter.out_bytes	vrouter.out_bytes
		ovs.vrouter.out_packets	vrouter.out_packets
use_health_metrics with use_rate_metrics	False	ovs.vrouter.in_dropped_sec	vrouter.in_dropped_sec
		ovs.vrouter.in_errors_sec	vrouter.in_errors_sec
		ovs.vrouter.out_dropped_sec	vrouter.out_dropped_sec
		ovs.vrouter.out_errors_sec	vrouter.out_errors_sec
use_health_metrics with use_absolute_metrics	False	ovs.vrouter.in_dropped	vrouter.in_dropped
		ovs.vrouter.in_errors	vrouter.in_errors
		ovs.vrouter.out_dropped	vrouter.out_dropped

Tuning Knob	Default Setting	Admin Metric Name	Project Metric Name
		ovs.vrouter.out_errors	vrouter.out_errors

12.1.2.5.2.1 Configuring the OVS metrics using the tuning knobs

Use the following steps to configure the tuning knobs for the libvirt plugin metrics.

1. Log in to the Cloud Lifecycle Manager.

2. Edit the following file:

   ```
   ~/openstack/my_cloud/config/neutron/monasca_ovs_plugin.yaml.j2
   ```

3. Change the value for each tuning knob to the desired setting, `True` if you want the metrics created and `False` if you want them removed. Refer to the table above for which metrics are controlled by each tuning knob.

   ```
   init_config:
       use_absolute_metrics: <true or false>
       use_rate_metrics: <true or false>
       use_health_metrics: <true or false>
   ```

4. Commit your configuration to the local Git repository (see *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "configuring OVS plugin tuning knobs"
   ```

5. Update your deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

6. Run the Neutron reconfigure playbook to implement the changes:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
   ```

12.1.3 Integrating HipChat, Slack, and JIRA

Monasca, the HPE Helion OpenStack monitoring and notification service, includes three default notification methods, **email**, **PagerDuty**, and **webhook**. Monasca also supports three other notification plugins which allow you to send notifications to **HipChat**, **Slack**, and **JIRA**. Unlike the default notification methods, the additional notification plugins must be manually configured.

This guide details the steps to configure each of the three non-default notification plugins. This guide also assumes that your cloud is fully deployed and functional.

12.1.3.1 Configuring the HipChat Plugin

To configure the HipChat plugin you will need the following four pieces of information from your HipChat system.

- The URL of your HipChat system.

- A token providing permission to send notifications to your HipChat system.

- The ID of the HipChat room you wish to send notifications to.

- A HipChat user account. This account will be used to authenticate any incoming notifications from your HPE Helion OpenStack cloud.

Obtain a token

Use the following instructions to obtain a token from your Hipchat system.

1. Log in to HipChat as the user account that will be used to authenticate the notifications.

2. Navigate to the following URL: `https://<your_hipchat_system>/account/api`. Replace `<your_hipchat_system>` with the fully-qualified-domain-name of your HipChat system.

3. Select the **Create token** option. Ensure that the token has the "SendNotification" attribute.

Obtain a room ID

Use the following instructions to obtain the ID of a HipChat room.

1. Log in to HipChat as the user account that will be used to authenticate the notifications.

2. Select **My account** from the application menu.

3. Select the **Rooms** tab.

4. Select the room that you want your notifications sent to.

5. Look for the API ID field in the room information. This is the room ID.

Create HipChat notification type

Use the following instructions to create a HipChat notification type.

1. Begin by obtaining the API URL for the HipChat room that you wish to send notifications to. The format for a URL used to send notifications to a room is as follows:

   `/v2/room/{room_id_or_name}/notification`

2. Use the Monasca API to create a new notification method. The following example demonstrates how to create a HipChat notification type named **MyHipChatNotification**, for room **ID 13**, using an example API URL and auth token.

   ```
   ardana > monasca notification-create  NAME TYPE ADDRESS
   ardana > monasca notification-create  MyHipChatNotification HIPCHAT https://
   hipchat.hpe.net/v2/room/13/notification?auth_token=1234567890
   ```

   The preceding example creates a notification type with the following characteristics

   - NAME: MyHipChatNotification

   - TYPE: HIPCHAT

   - ADDRESS: https://hipchat.hpe.net/v2/room/13/notification

   - auth_token: 1234567890

☞ Note

The Horizon dashboard can also be used to create a HipChat notification type.

12.1.3.2 Configuring the Slack Plugin

Configuring a Slack notification type requires four pieces of information from your Slack system.

- Slack server URL

- Authentication token

- Slack channel

- A Slack user account. This account will be used to authenticate incoming notifications to Slack.

Identify a Slack channel

1. Log in to your Slack system as the user account that will be used to authenticate the notifications to Slack.

2. In the left navigation panel, under the **CHANNELS** section locate the channel that you wish to receive the notifications. The instructions that follow will use the example channel **#general**.

Create a Slack token

1. Log in to your Slack system as the user account that will be used to authenticate the notifications to Slack

2. Navigate to the following URL: https://api.slack.com/docs/oauth-test-tokens ↗

3. Select the **Create token** button.

Create a Slack notification type

1. Begin by identifying the structure of the API call to be used by your notification method. The format for a call to the Slack Web API is as follows:
   ```
   https://slack.com/api/METHOD
   ```
   You can authenticate a Web API request by using the token that you created in the previous **Create a Slack Token**section. Doing so will result in an API call that looks like the following.
   ```
   https://slack.com/api/METHOD?token=auth_token
   ```
   You can further refine your call by specifying the channel that the message will be posted to. Doing so will result in an API call that looks like the following.
   ```
   https://slack.com/api/METHOD?token=AUTH_TOKEN&channel=#channel
   ```
   The following example uses the `chat.postMessage` method, the token `1234567890`, and the channel `#general`.

   ```
   https://slack.com/api/chat.postMessage?token=1234567890&channel=#general
   ```

Find more information on the Slack Web API here: https://api.slack.com/web ↗

2. Use the CLI on your Cloud Lifecycle Manager to create a new Slack notification type, using the API call that you created in the preceding step. The following example creates a notification type named **MySlackNotification**, using token **1234567890**, and posting to channel **#general**.

```
ardana > monasca notification-create  MySlackNotification SLACK https://slack.com/
api/chat.postMessage?token=1234567890&channel=#general
```

🎓 Note

Notification types can also be created in the Horizon dashboard.

12.1.3.3 Configuring the JIRA Plugin

Configuring the JIRA plugin requires three pieces of information from your JIRA system.

- The URL of your JIRA system.

- Username and password of a JIRA account that will be used to authenticate the notifications.

- The name of the JIRA project that the notifications will be sent to.

Create JIRA notification type

You will configure the Monasca service to send notifications to a particular JIRA project. You must also configure JIRA to create new issues for each notification it receives to this project, however, that configuration is outside the scope of this document.

The Monasca JIRA notification plugin supports only the following two JIRA issue fields.

- `PROJECT`. This is the only supported "mandatory" JIRA issue field.

- `COMPONENT`. This is the only supported "optional" JIRA issue field.

The JIRA issue type that your notifications will create may only be configured with the "Project" field as mandatory. If your JIRA issue type has any other mandatory fields, the Monasca plugin will not function correctly. Currently, the Monasca plugin only supports the single optional "component" field.

Creating the JIRA notification type requires a few more steps than other notification types covered in this guide. Because the Python and YAML files for this notification type are not yet included in HPE Helion OpenStack 8, you must perform the following steps to manually retrieve and place them on your Cloud Lifecycle Manager.

1. Configure the JIRA plugin by adding the following block to the `/etc/monasca/notification.yaml` file, under the `notification_types` section, and adding the username and password of the JIRA account used for the notifications to the respective sections.

```
plugins:

 - monasca_notification.plugins.jira_notifier:JiraNotifier

jira:
    user:

    password:

    timeout: 60
```

After adding the necessary block, the `notification_types` section should look like the following example. Note that you must also add the username and password for the JIRA user related to the notification type.

```
notification_types:
    plugins:

     - monasca_notification.plugins.jira_notifier:JiraNotifier

    jira:
        user:

        password:

        timeout: 60

    webhook:
        timeout: 5

    pagerduty:
        timeout: 5
```

```
          url: "https://events.pagerduty.com/generic/2010-04-15/create_event.json"
```

2. Create the JIRA notification type. The following command example creates a JIRA notification type named `MyJiraNotification`, in the JIRA project `HISO`.

```
ardana > monasca notification-create  MyJiraNotification JIRA https://
jira.hpcloud.net/?project=HISO
```

The following command example creates a JIRA notification type named `MyJiraNotification`, in the JIRA project `HISO`, and adds the optional *component* field with a value of `keystone`.

```
ardana > monasca notification-create MyJiraNotification JIRA https://
jira.hpcloud.net/?project=HISO&component=keystone
```

Note

There is a slash (`/`) separating the URL path and the query string. The slash is required if you have a query parameter without a path parameter.

Note

Notification types may also be created in the Horizon dashboard.

12.1.4 Alarm Metrics

You can use the available metrics to create custom alarms to further monitor your cloud infrastructure and facilitate autoscaling features.

For details on how to create customer alarms using the Operations Console, see *Book "Operations Console", Chapter 1 "Alarm Definition"*.

12.1.4.1 Apache Metrics

A list of metrics associated with the Apache service.

Metric Name	Dimensions	Description
apache.net.hits	`hostname`	Total accesses

Metric Name	Dimensions	Description
	service=apache component=apache	
apache.net.kbytes_sec	hostname service=apache component=apache	Total Kbytes per second
apache.net.requests_sec	hostname service=apache component=apache	Total accesses per second
apache.net.total_kbytes	hostname service=apache component=apache	Total Kbytes
apache.performance.busy_work	hostname service=apache component=apache	The number of workers serving requests
apache.performance.cpu_load_p	hostname service=apache component=apache	The current percentage of CPU used by each worker and in total by all workers combined
apache.performance.idle_worke	hostname service=apache component=apache	The number of idle workers
apache.status	apache_port hostname service=apache component=apache	Status of Apache port

12.1.4.2 Ceilometer Metrics

A list of metrics associated with the Ceilometer service.

Metric Name	Dimensions	Description
disk.total_space_mb_agg	aggregation_period=hourly,	Total space of disk

Metric Name	Dimensions	Description
	host=all, project_id=all	
disk.total_used_space_mb_agg	aggregation_period=hourly, host=all, project_id=all	Total used space of disk
swiftlm.diskusage.rate_agg	aggregation_period=hourly, host=all, project_id=all	
swiftlm.diskusage.val.avail_agg	aggregation_period=hourly, host, project_id=all	
swiftlm.diskusage.val.size_agg	aggregation_period=hourly, host, project_id=all	
image	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=image, source=openstack	Existence of the image
image.delete	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=image, source=openstack	Delete operation on this image
image.size	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge,	Size of the uploaded image

Metric Name	Dimensions	Description
	unit=B, source=openstack	
image.update	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=image, source=openstack	Update operation on this image
image.upload	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=image, source=openstack	Upload operation on this image
instance	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=instance, source=openstack	Existence of instance
disk.ephemeral.size	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=GB, source=openstack	Size of ephemeral disk on this instance
disk.root.size	user_id, region, resource_id, datasource=ceilometer,	Size of root disk on this instance

Metric Name	Dimensions	Description
	project_id, type=gauge, unit=GB, source=openstack	
memory	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=MB, source=openstack	Size of memory on this instance
ip.floating	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=ip, source=openstack	Existence of IP
ip.floating.create	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=ip, source=openstack	Create operation on this fip
ip.floating.update	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=ip, source=openstack	Update operation on this fip
mem.total_mb_agg	aggregation_period=hourly, host=all,	Total space of memory

Metric Name	Dimensions	Description
	`project_id=all`	

mem.usable_mb_agg	`aggregation_period=hourly,`
	`host=all,`
	`project_id=all`
		Available space of memory

network	`user_id,`
	`region,`
	`resource_id,`
	`datasource=ceilometer,`
	`project_id,`
	`type=gauge,`
	`unit=network,`
	`source=openstack`
		Existence of network

network.create	`user_id,`
	`region,`
	`resource_id,`
	`datasource=ceilometer,`
	`project_id,`
	`type=delta,`
	`unit=network,`
	`source=openstack`
		Create operation on this network

network.update	`user_id,`
	`region,`
	`resource_id,`
	`datasource=ceilometer,`
	`project_id,`
	`type=delta,`
	`unit=network,`
	`source=openstack`
		Update operation on this network

network.delete	`user_id,`
	`region,`
	`resource_id,`
	`datasource=ceilometer,`
	`project_id,`
	`type=delta,`
	`unit=network,`
	`source=openstack`
		Delete operation on this network

port	`user_id,`	Existence of port

Metric Name	Dimensions	Description
	region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=port, source=openstack	
port.create	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=port, source=openstack	Create operation on this port
port.delete	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=port, source=openstack	Delete operation on this port
port.update	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=port, source=openstack	Update operation on this port
router	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=router, source=openstack	Existence of router

Metric Name	Dimensions	Description
router.create	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=router, source=openstack	Create operation on this router
router.delete	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=router, source=openstack	Delete operation on this router
router.update	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=router, source=openstack	Update operation on this router
snapshot	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=snapshot, source=openstack	Existence of the snapshot
snapshot.create.end	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=snapshot,	Create operation on this snapshot

Metric Name	Dimensions	Description
	source=openstack	
snapshot.delete.end	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=snapshot, source=openstack	Delete operation on this snapshot
snapshot.size	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=GB, source=openstack	Size of this snapshot
subnet	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=subnet, source=openstack	Existence of the subnet
subnet.create	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=subnet, source=openstack	Create operation on this subnet
subnet.delete	user_id, region, resource_id, datasource=ceilometer, project_id,	Delete operation on this subnet

Metric Name	Dimensions	Description
	type=delta, unit=subnet, source=openstack	
subnet.update	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=subnet, source=openstack	Update operation on this subnet
vcpus	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=vcpus, source=openstack	Number of virtual CPUs allocated to the instance
vcpus_agg	aggregation_period=hourly, host=all, project_id	Number of vcpus used by a project
volume	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=volume, source=openstack	Existence of the volume
volume.create.end	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=volume, source=openstack	Create operation on this volume

Metric Name	Dimensions	Description
volume.delete.end	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=volume, source=openstack	Delete operation on this volume
volume.resize.end	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=volume, source=openstack	Resize operation on this volume
volume.size	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=GB, source=openstack	Size of this volume
volume.update.end	user_id, region, resource_id, datasource=ceilometer, project_id, type=delta, unit=volume, source=openstack	Update operation on this volume
storage.objects	user_id, region, resource_id, datasource=ceilometer, project_id, type=gauge, unit=object,	Number of objects

Metric Name	Dimensions	Description
	`source=openstack`	

storage.objects.size	`user_id,`
	`region,`
	`resource_id,`
	`datasource=ceilometer,`
	`project_id,`
	`type=gauge,`
	`unit=B,`
	`source=openstack`
		Total size of stored objects

storage.objects.containers	`user_id,`
	`region,`
	`resource_id,`
	`datasource=ceilometer,`
	`project_id,`
	`type=gauge,`
	`unit=container,`
	`source=openstack`
		Number of containers

12.1.4.3   Cinder Metrics

A list of metrics associated with the Cinder service.

Metric Name	Dimensions	Description
cinderlm.cinder.backend.physical.list	service = block-storage, hostname, cluster, cloud_name, control_plane, component, backends	List of physical backends
cinderlm.cinder.backend.total.avail	service = block-storage, hostname, cluster, cloud_name, control_plane, component, backendname	Total available capacity metric per backend

Metric Name	Dimensions	Description
cinderlm.cinder.backend.total.size	service = block-storage, hostname, cluster, cloud_name, control_plane, component, backendname	Total capacity metric per backend
cinderlm.cinder.cinder_services	service = block-storage, hostname, cluster, cloud_name, control_plane, component	Status of a cinder-volume service
cinderlm.hp_hardware.hpssacli.logical_drive	service = block-storage, hostname, cluster, cloud_name, control_plane, component, sub_component, logical_drive, controller_slot, array The HPE Smart Storage Administrator (HPE SSA) CLI component will have to be installed for SSACLI status to be reported. To download and install the SSACLI utility to enable management of disk controllers, please refer to: https://support.hpe.com/ hpsc/swd/public/detail? swItemId=MTX_3d16386b418a443388c18da82f ↗	Status of a logical drive
cinderlm.hp_hardware.hpssacli.physical_drive	service = block-storage, hostname, cluster, cloud_name, control_plane, component, box, bay, controller_slot	Status of a logical drive
cinderlm.hp_hardware.hpssacli.smart_array	service = block-storage, hostname, cluster, cloud_name, control_plane, component, sub_component, model	Status of smart array

Metric Name	Dimensions	Description
cinderlm.hp_hardware.hpssacli.smart_array_firmware	service=block storage, hostname, cluster, cloud_name, control_plane, component, model	Checks firmware version

12.1.4.4 Compute Metrics

Note

Compute instance metrics are listed in *Section 12.1.4.11, "Libvirt Metrics"*.

A list of metrics associated with the Compute service.

Metric Name	Dimensions	Description
nova.heartbeat	service=compute cloud_name hostname component control_plane cluster	Checks that all services are running heartbeats (uses nova user and to list services then sets up checks for each. For example, nova-scheduler, nova-conductor, nova-consoleauth, nova-compute)
nova.vm.cpu.total_allocated	service=compute hostname component control_plane cluster	Total CPUs allocated across all VMs
nova.vm.disk.total_allocated_gb	service=compute hostname component control_plane cluster	Total Gbytes of disk space allocated to all VMs
nova.vm.mem.total_allocated_m	service=compute hostname component	Total Mbytes of memory allocated to all VMs

Metric Name	Dimensions	Description
	control_plane cluster	

12.1.4.5 Crash Metrics

A list of metrics associated with the Crash service.

Metric Name	Dimensions	Description
crash.dump_count	service=system hostname cluster	Number of crash dumps found

12.1.4.6 Directory Metrics

A list of metrics associated with the Directory service.

Metric Name	Dimensions	Description
directory.files_count	service hostname path	Total number of files under a specific directory path
directory.size_bytes	service hostname path	Total size of a specific directory path

12.1.4.7 Elasticsearch Metrics

A list of metrics associated with the Elasticsearch service.

Metric Name	Dimensions	Description
elasticsearch.active_primary_sha	service=logging url hostname	Indicates the number of primary shards in your cluster. This is an aggregate total across all indices.

Metric Name	Dimensions	Description

elasticsearch.active_shards	`service=logging`
	`url`
	`hostname`
		Aggregate total of all shards across all indices, which includes replica shards.

elasticsearch.cluster_status	`service=logging`
	`url`
	`hostname`
		Cluster health status.

elasticsearch.initializing_shards	`service=logging`
	`url`
	`hostname`
		The count of shards that are being freshly created.

elasticsearch.number_of_data_no	`service=logging`
	`url`
	`hostname`
		Number of data nodes.

elasticsearch.number_of_nodes	`service=logging`
	`url`
	`hostname`
		Number of nodes.

elasticsearch.relocating_shards	`service=logging`
	`url`
	`hostname`
		Shows the number of shards that are currently moving from one node to another node.

elasticsearch.unassigned_shards	`service=logging`
	`url`
	`hostname`
		The number of unassigned shards from the master node.

12.1.4.8 HAProxy Metrics

A list of metrics associated with the HAProxy service.

Metric Name	Dimensions	Description
haproxy.backend.bytes.in_rate		
haproxy.backend.bytes.out_rate		

Metric Name	Dimensions	Description
haproxy.backend.denied.req_rate		
haproxy.backend.denied.resp_rate		
haproxy.backend.errors.con_rate		
haproxy.backend.errors.resp_rate		
haproxy.backend.queue.current		
haproxy.backend.response.1xx		
haproxy.backend.response.2xx		
haproxy.backend.response.3xx		
haproxy.backend.response.4xx		
haproxy.backend.response.5xx		
haproxy.backend.response.other		
haproxy.backend.session.current		
haproxy.backend.session.limit		
haproxy.backend.session.pct		
haproxy.backend.session.rate		
haproxy.backend.warnings.redis_rate		
haproxy.backend.warnings.retr_rate		
haproxy.frontend.bytes.in_rate		
haproxy.frontend.bytes.out_rate		
haproxy.frontend.denied.req_rate		
haproxy.frontend.denied.resp_rate		
haproxy.frontend.errors.req_rate		
haproxy.frontend.requests.rate		

Metric Name	Dimensions	Description
haproxy.frontend.response.1xx		
haproxy.frontend.response.2xx		
haproxy.frontend.response.3xx		
haproxy.frontend.response.4xx		
haproxy.frontend.response.5xx		
haproxy.frontend.response.other		
haproxy.frontend.session.current		
haproxy.frontend.session.limit		
haproxy.frontend.session.pct		
haproxy.frontend.session.rate		

12.1.4.9   HTTP Check Metrics

A list of metrics associated with the HTTP Check service:

TABLE 12.2: HTTP CHECK METRICS

Metric Name	Dimensions	Description
| http_response_time | ```
url
hostname
service
component
``` | The response time in seconds of the http endpoint call. |
| http_status | ```
url
hostname
service
``` | The status of the http endpoint call (0 = success, 1 = failure). |

For each component and HTTP metric name there are two separate metrics reported, one for the local URL and another for the virtual IP (VIP) URL:

TABLE 12.3: HTTP METRIC COMPONENTS

Component	Dimensions	Description

account-server	`service=object-storage`
	`component=account-server`
	`url`
		swift account-server http endpoint status and response time

barbican-api	`service=key-manager`
	`component=barbican-api`
	`url`
		barbican-api http endpoint status and response time

ceilometer-api	`service=telemetry`
	`component=ceilometer-api`
	`url`
		ceilometer-api http endpoint status and response time

cinder-api	`service=block-storage`
	`component=cinder-api`
	`url`
		cinder-api http endpoint status and response time

container-server	`service=object-storage`
	`component=container-server`
	`url`
		swift container-server http endpoint status and response time

designate-api	`service=dns`
	`component=designate-api`
	`url`
		designate-api http endpoint status and response time

freezer-api	`service=backup`
	`component=freezer-api`
	`url`
		freezer-api http endpoint status and response time

glance-api	`service=image-service`
	`component=glance-api`
	`url`
		glance-api http endpoint status and response time

glance-registry	`service=image-service`
	`component=glance-registry`
	`url`
		glance-registry http endpoint status and response time

Component	Dimensions	Description

heat-api	`service=orchestration`
	`component=heat-api`
	`url`
		heat-api http endpoint status and response time

heat-api-cfn	`service=orchestration`
	`component=heat-api-cfn`
	`url`
		heat-api-cfn http endpoint status and response time

heat-api-cloudwatch	`service=orchestration`
	`component=heat-api-`
	`cloudwatch`
	`url`
		heat-api-cloudwatch http endpoint status and response time

ardana-ux-services	`service=ardana-ux-services`
	`component=ardana-ux-`
	`services`
	`url`
		ardana-ux-services http endpoint status and response time

horizon	`service=web-ui`
	`component=horizon`
	`url`
		horizon http endpoint status and response time

keystone-api	`service=identity-service`
	`component=keystone-api`
	`url`
		keystone-api http endpoint status and response time

monasca-api	`service=monitoring`
	`component=monasca-api`
	`url`
		monasca-api http endpoint status

monasca-persister	`service=monitoring`
	`component=monasca-persister`
	`url`
		monasca-persister http endpoint status

neutron-server	`service=networking`
	`component=neutron-server`
	`url`
		neutron-server http endpoint status and response time

neutron-server-vip	`service=networking`
	`component=neutron-server-`
	`vip`
		neutron-server-vip http endpoint status and response time

Component	Dimensions	Description
	`url`	

nova-api	`service=compute`
	`component=nova-api`
	`url`
		nova-api http endpoint status and response time

nova-vnc	`service=compute`
	`component=nova-vnc`
	`url`
		nova-vnc http endpoint status and response time

object-server	`service=object-storage`
	`component=object-server`
	`url`
		object-server http endpoint status and response time

object-storage-vip	`service=object-storage`
	`component=object-storage-`
	`vip`
	`url`
		object-storage-vip http endpoint status and response time

octavia-api	`service=octavia`
	`component=octavia-api`
	`url`
		octavia-api http endpoint status and response time

ops-console-web	`service=ops-console`
	`component=ops-console-web`
	`url`
		ops-console-web http endpoint status and response time

proxy-server	`service=object-storage`
	`component=proxy-server`
	`url`
		proxy-server http endpoint status and response time

12.1.4.10   Kafka Metrics

A list of metrics associated with the Kafka service.

Metric Name	Dimensions	Description

kafka.consumer_lag	`topic`
	`service`
	`component=kafka`
		Hostname consumer offset lag from broker offset

Metric Name	Dimensions	Description
	`consumer_group` `hostname`	

12.1.4.11 Libvirt Metrics

🎓 Note

For information on how to turn these metrics on and off using the tuning knobs, see *Section 12.1.2.5.1, "Libvirt plugin metric tuning knobs"*.

A list of metrics associated with the Libvirt service.

TABLE 12.4: **TUNABLE LIBVIRT METRICS**

Admin Metric Name	Project Metric Name	Dimensions	Description

vm.cpu.time_ns	cpu.time_ns	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Cumulative CPU time (in ns)

vm.cpu.utilization_norm_perc	cpu.utilization_norm_perc	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Normalized CPU utilization (percentage)

vm.cpu.utilization_perc	cpu.utilization_perc	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Overall CPU utilization (percentage)

vm.io.errors	io.errors	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Overall disk I/O errors

Admin Metric Name	Project Metric Name	Dimensions	Description
vm.io.errors_sec	io.errors_sec	``` zone service resource_id hostname component ```	Disk I/O errors per second
vm.io.read_bytes	io.read_bytes	``` zone service resource_id hostname component ```	Disk I/O read bytes value
vm.io.read_bytes_sec	io.read_bytes_sec	``` zone service resource_id hostname component ```	Disk I/O read bytes per second
vm.io.read_ops	io.read_ops	``` zone service resource_id hostname component ```	Disk I/O read operations value
vm.io.read_ops_sec	io.read_ops_sec	``` zone service resource_id hostname component ```	Disk I/O write operations per second
vm.io.write_bytes	io.write_bytes	``` zone service resource_id hostname component ```	Disk I/O write bytes value
vm.io.write_bytes_sec	io.write_bytes_sec	``` zone service resource_id hostname ```	Disk I/O write bytes per second

Admin Metric Name	Project Metric Name	Dimensions	Description
		`component`	

vm.io.write_ops	io.write_ops	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Disk I/O write operations value

vm.io.write_ops_sec	io.write_ops_sec	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Disk I/O write operations per second

vm.net.in_bytes	net.in_bytes	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
		`device`
		`port_id`
			Network received total bytes

vm.net.in_bytes_sec	net.in_bytes_sec	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
		`device`
		`port_id`
			Network received bytes per second

vm.net.in_packets	net.in_packets	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
		`device`
		`port_id`
			Network received total packets

vm.net.in_packets_sec	net.in_packets_sec	`zone`
		`service`
		`resource_id`
		`hostname`
			Network received packets per second

Admin Metric Name	Project Metric Name	Dimensions	Description
| | | ```
component
device
port_id
``` | |
| vm.net.out_bytes | net.out_bytes | ```
zone
service
resource_id
hostname
component
device
port_id
``` | Network transmitted total bytes |
| vm.net.out_bytes_sec | net.out_bytes_sec | ```
zone
service
resource_id
hostname
component
device
port_id
``` | Network transmitted bytes per second |
| vm.net.out_packets | net.out_packets | ```
zone
service
resource_id
hostname
component
device
port_id
``` | Network transmitted total packets |
| vm.net.out_packets_sec | net.out_packets_sec | ```
zone
service
resource_id
hostname
component
device
port_id
``` | Network transmitted packets per second |
| vm.ping_status | ping_status | ```
zone
service
resource_id
hostname
component
``` | 0 for ping success, 1 for ping failure |

Admin Metric Name	Project Metric Name	Dimensions	Description
| vm.disk.allocation | disk.allocation | ```
zone
service
resource_id
hostname
component
``` | Total Disk allocation for a device |
| vm.disk.allocation_total | disk.allocation_total | ```
zone
service
resource_id
hostname
component
``` | Total Disk allocation across devices for instances |
| vm.disk.capacity | disk.capacity | ```
zone
service
resource_id
hostname
component
``` | Total Disk capacity for a device |
| vm.disk.capacity_total | disk.capacity_total | ```
zone
service
resource_id
hostname
component
``` | Total Disk capacity across devices for instances |
| vm.disk.physical | disk.physical | ```
zone
service
resource_id
hostname
component
``` | Total Disk usage for a device |
| vm.disk.physical_total | disk.physical_total | ```
zone
service
resource_id
hostname
component
``` | Total Disk usage across devices for instances |
| vm.io.errors_total | io.errors_total | ```
zone
service
resource_id
hostname
``` | Total Disk I/O errors across all devices |

Admin Metric Name	Project Metric Name	Dimensions	Description
		`component`	

vm.io.errors_total_sec	io.errors_total_sec	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O errors per second across all devices

vm.io.read_bytes_total	io.read_bytes_total	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O read bytes across all devices

vm.io.read_bytes_total_sec	io.read_bytes_total_sec	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O read bytes per second across devices

vm.io.read_ops_total	io.read_ops_total	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O read operations across all devices

vm.io.read_ops_total_sec	io.read_ops_total_sec	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O read operations across all devices per sec

vm.io.write_bytes_total	io.write_bytes_total	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O write bytes across all devices

vm.io.write_bytes_total_sec	io.write_bytes_total_sec	`zone`
		`service`
		`resource_id`
			Total Disk I/O Write bytes per second across devices

Admin Metric Name	Project Metric Name	Dimensions	Description
		`hostname` `component`	

vm.io.write_ops_total	io.write_ops_total	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O write operations across all devices

vm.io.write_ops_total_sec	io.write_ops_total_sec	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			Total Disk I/O write operations across all devices per sec

These metrics in libvirt are always enabled and cannot be disabled using the tuning knobs.

TABLE 12.5: UNTUNABLE LIBVIRT METRICS

Admin Metric Name	Project Metric Name	Dimensions	Description

vm.host_alive_status	host_alive_status	`zone`
		`service`
		`resource_id`
		`hostname`
		`component`
			-1 for no status, 0 for Running / OK, 1 for Idle / blocked, 2 for Paused,
			
			3 for Shutting down, 4 for Shut off or Nova suspend 5 for Crashed,
			
			6 for Power management suspend (S3 state)

vm.mem.free_mb	mem.free_mb	`cluster`
		`service`
		`hostname`
			Free memory in Mbytes

Admin Metric Name	Project Metric Name	Dimensions	Description

vm.mem.free_perc	mem.free_perc	`cluster`
		`service`
		`hostname`
			Percent of memory free

vm.mem.resident_mb		`cluster`
		`service`
		`hostname`
			Total memory used on host, an Operations-only metric

vm.mem.swap_used_mb	mem.swap_used_mb	`cluster`
		`service`
		`hostname`
			Used swap space in Mbytes

vm.mem.total_mb	mem.total_mb	`cluster`
		`service`
		`hostname`
			Total memory in Mbytes

vm.mem.used_mb	mem.used_mb	`cluster`
		`service`
		`hostname`
			Used memory in Mbytes

12.1.4.12 Monitoring Metrics

A list of metrics associated with the Monitoring service.

Metric Name	Dimensions	Description

alarm-state-transitions-added-to-batch-counter	`service=monitoring`
	`url`
	`hostname`
	`component=monasca-persister`	

jvm.memory.total.max	`service=monitoring`
	`url`
	`hostname`
	`component`
		Maximum JVM overall memory

jvm.memory.total.used	`service=monitoring`	Used JVM overall memory

Metric Name	Dimensions	Description
	url hostname component	
metrics-added-to-batch-counter	service=monitoring url hostname component=monasca-persister	
metrics.published	service=monitoring url hostname component=monasca-api	Total number of published metrics
monasca.alarms_finished_count	hostname component=monasca-notification service=monitoring	Total number of alarms received
monasca.checks_running_too_lor	hostname component=monasca-agent service=monitoring cluster	Only emitted when collection time for a check is too long
monasca.collection_time_sec	hostname component=monasca-agent service=monitoring cluster	Collection time in monasca-agent
monasca.config_db_time	hostname component=monasca-notification service=monitoring	
monasca.created_count	hostname component=monasca-notification service=monitoring	Number of notifications created
monasca.invalid_type_count	hostname component=monasca-notification	Number of notifications with invalid type

Metric Name	Dimensions	Description
	service=monitoring	
monasca.log.in_bulks_rejected	hostname component=monasca-log-api service=monitoring version	
monasca.log.in_logs	hostname component=monasca-log-api service=monitoring version	
monasca.log.in_logs_bytes	hostname component=monasca-log-api service=monitoring version	
monasca.log.in_logs_rejected	hostname component=monasca-log-api service=monitoring version	
monasca.log.out_logs	hostname component=monasca-log-api service=monitoring	
monasca.log.out_logs_lost	hostname component=monasca-log-api service=monitoring	
monasca.log.out_logs_truncated	hostname component=monasca-log-api service=monitoring	
monasca.log.processing_time_m	hostname component=monasca-log-api service=monitoring	
monasca.log.publish_time_ms	hostname component=monasca-log-api service=monitoring	

Metric Name	Dimensions	Description
| monasca.thread_count | ```
service=monitoring
process_name
hostname
component
``` | Number of threads monasca is using |
| raw-sql.time.avg | ```
service=monitoring
url
hostname
component
``` | Average raw sql query time |
| raw-sql.time.max | ```
service=monitoring
url
hostname
component
``` | Max raw sql query time |

12.1.4.13 Monasca Aggregated Metrics

A list of the aggregated metrics associated with the Monasca Transform feature.

Metric Name	For	Dimensions	Description
| cpu.utilized_logical_cores_agg | Compute summary | ```
aggregation_period:
 hourly
host: all or
 <hostname>
project_id: all
``` | Utilized physical host cpu core capacity for one or all hosts by time interval (defaults to a hour). Available as total or per host |
| cpu.total_logical_cores_agg | Compute summary | ```
aggregation_period:
 hourly
host: all or
 <hostname>
project_id: all
``` | Total physical host cpu core capacity for one or all hosts by time interval (defaults to a hour) Available as total or per host |

Metric Name	For	Dimensions	Description
mem.total_mb_agg	Compute summary	```aggregation_period: hourly host: all project_id: all```	Total physical host memory capacity by time interval (defaults to a hour)
mem.usable_mb_agg	Compute summary	```aggregation_period: hourly host: all project_id: all```	Usable physical host memory capacity by time interval (defaults to a hour)
disk.total_used_space_mb_agg	Compute summary	```aggregation_period: hourly host: all project_id: all```	Utilized physical host disk capacity by time interval (defaults to a hour)
disk.total_space_mb_agg	Compute summary	```aggregation_period: hourly host: all project_id: all```	Total physical host disk capacity by time interval (defaults to a hour)
nova.vm.cpu.total_allocated_agg	Compute summary	```aggregation_period: hourly host: all project_id: all```	CPUs allocated across all virtual machines by time interval (defaults to a hour)
vcpus_agg	Compute summary	```aggregation_period: hourly host: all project_id: all or <project ID>```	Virtual CPUs allocated capacity for virtual machines of one or all projects by time interval (defaults to a hour) Available as total or per host

Metric Name	For	Dimensions	Description
| nova.vm.mem.total_allocated_mb_agg | Compute summary | ```
aggregation_period:
 hourly
host: all
project_id: all
``` | Memory allocated to all virtual machines by time interval (defaults to a hour) |
| vm.mem.used_mb_agg | Compute summary | ```
aggregation_period:
 hourly
host: all
project_id: all or
 <project ID>
``` | Memory utilized by virtual machines of one or all projects by time interval (defaults to an hour) Available as total or per host |
| vm.mem.total_mb_agg | Compute summary | ```
aggregation_period:
 hourly
host: all
project_id: all or
 <project ID>
``` | Memory allocated to virtual machines of one or all projects by time interval (defaults to an hour) Available as total or per host |
| vm.cpu.utilization_perc_agg | Compute summary | ```
aggregation_period:
 hourly
host: all
project_id: all or
 <project ID>
``` | CPU utilized by all virtual machines by project by time interval (defaults to an hour) |
| nova.vm.disk.total_allocated_gb_agg | Compute summary | ```
aggregation_period:
 hourly
host: all
project_id: all
``` | Disk space allocated to all virtual machines by time interval (defaults to an hour) |

Metric Name	For	Dimensions	Description
vm.disk.allocation_agg	Compute summary	``` aggregation_period: hourly host: all project_id: all or <project ID> ```	Disk allocation for virtual machines of one or all projects by time interval (defaults to a hour) Available as total or per host
swiftlm.diskusage.val.size_agg	Object Storage summary	``` aggregation_period: hourly host: all or <hostname> project_id: all ```	Total available object storage capacity by time interval (defaults to a hour) Available as total or per host
swiftlm.diskusage.val.avail_agg	Object Storage summary	``` aggregation_period: hourly host: all or <hostname> project_id: all ```	Remaining object storage capacity by time interval (defaults to a hour) Available as total or per host
swiftlm.diskusage.rate_agg	Object Storage summary	``` aggregation_period: hourly host: all project_id: all ```	Rate of change of object storage usage by time interval (defaults to a hour)
storage.objects.size_agg	Object Storage summary	``` aggregation_period: hourly host: all project_id: all ```	Used object storage capacity by time interval (defaults to a hour)

12.1.4.14 MySQL Metrics

A list of metrics associated with the MySQL service.

Metric Name	Dimensions	Description
mysql.innodb.buffer_pool_free	hostname mode service=mysql	The number of free pages, in bytes. This value is calculated by multiplying `Innodb_buffer_pool_pages_free` and `Innodb_page_size` of the server status variable.
mysql.innodb.buffer_pool_total	hostname mode service=mysql	The total size of buffer pool, in bytes. This value is calculated by multiplying `Innodb_buffer_pool_pages_total` and `Innodb_page_size` of the server status variable.
mysql.innodb.buffer_pool_used	hostname mode service=mysql	The number of used pages, in bytes. This value is calculated by subtracting `Innodb_buffer_pool_pages_total` away from `Innodb_buffer_pool_pages_free` of the server status variable.
mysql.innodb.current_row_locks	hostname mode service=mysql	Corresponding to current row locks of the server status variable.
mysql.innodb.data_reads	hostname mode service=mysql	Corresponding to `Innodb_data_reads` of the server status variable.
mysql.innodb.data_writes	hostname mode service=mysql	Corresponding to `Innodb_data_writes` of the server status variable.

Metric Name	Dimensions	Description
| mysql.innodb.mutex_os_waits | ```
hostname
mode
service=mysql
``` | Corresponding to the OS waits of the server status variable. |
| mysql.innodb.mutex_spin_round | ```
hostname
mode
service=mysql
``` | Corresponding to spinlock rounds of the server status variable. |
| mysql.innodb.mutex_spin_waits | ```
hostname
mode
service=mysql
``` | Corresponding to the spin waits of the server status variable. |
| mysql.innodb.os_log_fsyncs | ```
hostname
mode
service=mysql
``` | Corresponding to `Innodb_os_log_fsyncs` of the server status variable. |
| mysql.innodb.row_lock_time | ```
hostname
mode
service=mysql
``` | Corresponding to `Innodb_row_lock_time` of the server status variable. |
| mysql.innodb.row_lock_waits | ```
hostname
mode
service=mysql
``` | Corresponding to `Innodb_row_lock_waits` of the server status variable. |
| mysql.net.connections | ```
hostname
mode
service=mysql
``` | Corresponding to `Connections` of the server status variable. |
| mysql.net.max_connections | ```
hostname
mode
service=mysql
``` | Corresponding to `Max_used_connections` of the server status variable. |
| mysql.performance.com_delete | ```
hostname
mode
service=mysql
``` | Corresponding to `Com_delete` of the server status variable. |

Metric Name	Dimensions	Description
mysql.performance.com_delete_	hostname mode service=mysql	Corresponding to `Com_delete_multi` of the server status variable.
mysql.performance.com_insert	hostname mode service=mysql	Corresponding to `Com_insert` of the server status variable.
mysql.performance.com_insert_	hostname mode service=mysql	Corresponding to `Com_insert_select` of the server status variable.
mysql.performance.com_replace	hostname mode service=mysql	Corresponding to `Com_replace_select` of the server status variable.
mysql.performance.com_select	hostname mode service=mysql	Corresponding to `Com_select` of the server status variable.
mysql.performance.com_update	hostname mode service=mysql	Corresponding to `Com_update` of the server status variable.
mysql.performance.com_update	hostname mode service=mysql	Corresponding to `Com_update_multi` of the server status variable.
mysql.performance.created_tmp	hostname mode service=mysql	Corresponding to `Created_tmp_disk_tables` of the server status variable.
mysql.performance.created_tmp	hostname mode service=mysql	Corresponding to `Created_tmp_files` of the server status variable.

Metric Name	Dimensions	Description
mysql.performance.created_tmp	hostname mode service=mysql	Corresponding to `Created_tmp_tables` of the server status variable.
mysql.performance.kernel_time	hostname mode service=mysql	The kernel time for the databases performance, in seconds.
mysql.performance.open_files	hostname mode service=mysql	Corresponding to `Open_files` of the server status variable.
mysql.performance.qcache_hits	hostname mode service=mysql	Corresponding to `Qcache_hits` of the server status variable.
mysql.performance.queries	hostname mode service=mysql	Corresponding to `Queries` of the server status variable.
mysql.performance.questions	hostname mode service=mysql	Corresponding to `Question` of the server status variable.
mysql.performance.slow_querie	hostname mode service=mysql	Corresponding to `Slow_queries` of the server status variable.
mysql.performance.table_locks_	hostname mode service=mysql	Corresponding to `Table_locks_waited` of the server status variable.
mysql.performance.threads_con	hostname mode service=mysql	Corresponding to `Threads_connected` of the server status variable.

Metric Name	Dimensions	Description
| mysql.performance.user_time | ```
hostname
mode
service=mysql
``` | The CPU user time for the databases performance, in seconds. |

12.1.4.15 NTP Metrics

A list of metrics associated with the NTP service.

Metric Name	Dimensions	Description
| ntp.connection_status | ```
hostname
ntp_server
``` | Value of ntp server connection status (0 = Healthy) |
| ntp.offset | ```
hostname
ntp_server
``` | Time offset in seconds |

12.1.4.16 Open vSwitch (OVS) Metrics

A list of metrics associated with the OVS service.

☞ Note

For information on how to turn these metrics on and off using the tuning knobs, see *Section 12.1.2.5.2, "OVS plugin metric tuning knobs"*.

TABLE 12.6: PER-ROUTER METRICS

Admin Metric Name	Project Metric Name	Dimensions	Description
| ovs.vrouter.in_bytes_sec | vrouter.in_bytes_sec | ```
service=networking
resource_id
component=ovs
router_name
port_id
``` | Inbound bytes per second for the router (if `network_use_bits` is false) |

Admin Metric Name	Project Metric Name	Dimensions	Description
| ovs.vrouter.in_packets_sec | vrouter.in_packets_sec | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Incoming packets per second for the router |
| ovs.vrouter.out_bytes_sec | vrouter.out_bytes_sec | ```
service=networking
resource_id
component=ovs
router_name
port_id
``` | Outgoing bytes per second for the router (if `network_use_bits` is false) |
| ovs.vrouter.out_packets_sec | vrouter.out_packets_sec | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Outgoing packets per second for the router |
| ovs.vrouter.in_bytes | vrouter.in_bytes | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Inbound bytes for the router (if `network_use_bits` is false) |
| ovs.vrouter.in_packets | vrouter.in_packets | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Incoming packets for the router |
| ovs.vrouter.out_bytes | vrouter.out_bytes | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Outgoing bytes for the router (if `network_use_bits` is false) |

Admin Metric Name	Project Metric Name	Dimensions	Description

ovs.vrouter.out_packets	vrouter.out_packets	`service=networking`
		`resource_id`
		`tenant_id`
		`component=ovs`
		`router_name`
		`port_id`
			Outgoing packets for the router

ovs.vrouter.in_dropped_sec	vrouter.in_dropped_sec	`service=networking`
		`resource_id`
		`tenant_id`
		`component=ovs`
		`router_name`
		`port_id`
			Incoming dropped packets per second for the router

ovs.vrouter.in_errors_sec	vrouter.in_errors_sec	`service=networking`
		`resource_id`
		`component=ovs`
		`router_name`
		`port_id`
			Number of incoming errors per second for the router

ovs.vrouter.out_dropped_sec	vrouter.out_dropped_sec	`service=networking`
		`resource_id`
		`component=ovs`
		`router_name`
		`port_id`
			Outgoing dropped packets per second for the router

ovs.vrouter.out_errors_sec	vrouter.out_errors_sec	`service=networking`
		`resource_id`
		`component=ovs`
		`router_name`
		`port_id`
			Number of outgoing errors per second for the router

ovs.vrouter.in_dropped	vrouter.in_dropped	`service=networking`
		`resource_id`
		`tenant_id`
		`component=ovs`
		`router_name`
		`port_id`
			Incoming dropped packets for the router

ovs.vrouter.in_errors	vrouter.in_errors	`service=networking` `resource_id`	Number of incoming errors for the router

Admin Metric Name	Project Metric Name	Dimensions	Description
		component=ovs router_name port_id	
ovs.vrouter.out_dropped	vrouter.out_dropped	service=networking resource_id component=ovs router_name port_id	Outgoing dropped packets for the router
ovs.vrouter.out_errors	vrouter.out_errors	service=networking resource_id tenant_id component=ovs router_name port_id	Number of outgoing errors for the router

TABLE 12.7: PER-DHCP PORT AND RATE METRICS

Admin Metric Name	Tenant Metric Name	Dimensions	Description
ovs.vswitch.in_bytes_sec	vswitch.in_bytes_sec	service=networking resource_id component=ovs router_name port_id	Incoming Bytes per second on DHCP port(if `network_use_bits` is false)
ovs.vswitch.in_packets_sec	vswitch.in_packets_sec	service=networking resource_id tenant_id component=ovs router_name port_id	Incoming packets per second for the DHCP port
ovs.vswitch.out_bytes_sec	vswitch.out_bytes_sec	service=networking resource_id component=ovs router_name port_id	Outgoing Bytes per second on DHCP port(if `network_use_bits` is false)

Admin Metric Name	Tenant Metric Name	Dimensions	Description
| ovs.vswitch.out_packets_sec | vswitch.out_packets_sec | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Outgoing packets per second for the DHCP port |
| ovs.vswitch.in_bytes | vswitch.in_bytes | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Inbound bytes for the DHCP port (if `network_use_bits` is false) |
| ovs.vswitch.in_packets | vswitch.in_packets | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Incoming packets for the DHCP port |
| ovs.vswitch.out_bytes | vswitch.out_bytes | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Outgoing bytes for the DHCP port (if `network_use_bits` is false) |
| ovs.vswitch.out_packets | vswitch.out_packets | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Outgoing packets for the DHCP port |
| ovs.vswitch.in_dropped_sec | vswitch.in_dropped_sec | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Incoming dropped per second for the DHCP port |

Alarm Metrics   HPE Helion OpenStack 8

Admin Metric Name	Tenant Metric Name	Dimensions	Description
| ovs.vswitch.in_errors_sec | vswitch.in_errors_sec | ```
service=networking
resource_id
component=ovs
router_name
port_id
``` | Incoming errors per second for the DHCP port |
| ovs.vswitch.out_dropped_sec | vswitch.out_dropped_sec | ```
service=networking
resource_id
component=ovs
router_name
port_id
``` | Outgoing dropped packets per second for the DHCP port |
| ovs.vswitch.out_errors_sec | vswitch.out_errors_sec | ```
service=networking
resource_id
component=ovs
router_name
port_id
``` | Outgoing errors per second for the DHCP port |
| ovs.vswitch.in_dropped | vswitch.in_dropped | ```
service=networking
resource_id
tenant_id
component=ovs
router_name
port_id
``` | Incoming dropped packets for the DHCP port |
| ovs.vswitch.in_errors | vswitch.in_errors | ```
service=networking
resource_id
component=ovs
router_name
port_id
``` | Errors received for the DHCP port |
| ovs.vswitch.out_dropped | vswitch.out_dropped | ```
service=networking
resource_id
component=ovs
router_name
port_id
``` | Outgoing dropped packets for the DHCP port |
| ovs.vswitch.out_errors | vswitch.out_errors | ```
service=networking
resource_id
tenant_id
component=ovs
``` | Errors transmitted for the DHCP port |

Admin Metric Name	Tenant Metric Name	Dimensions	Description
| | | ```
router_name
port_id
``` | |

12.1.4.17 Process Metrics

A list of metrics associated with processes.

Metric Name	Dimensions	Description
| process.cpu_perc | ```
hostname
service
process_name
component
``` | Percentage of cpu being consumed by a process |
| process.io.read_count | ```
hostname
service
process_name
component
``` | Number of reads by a process |
| process.io.read_kbytes | ```
hostname
service
process_name
component
``` | Kbytes read by a process |
| process.io.write_count | ```
hostname
service
process_name
component
``` | Number of writes by a process |
| process.io.write_kbytes | ```
hostname
service
process_name
component
``` | Kbytes written by a process |
| process.mem.rss_mbytes | ```
hostname
service
process_name
component
``` | Amount of physical memory allocated to a process, including memory from shared libraries in Mbytes |

Metric Name	Dimensions	Description

process.open_file_descriptors	`hostname`
	`service`
	`process_name`
	`component`
		Number of files being used by a process

process.pid_count	`hostname`
	`service`
	`process_name`
	`component`
		Number of processes that exist with this process name

process.thread_count	`hostname`
	`service`
	`process_name`
	`component`
		Number of threads a process is using

12.1.4.17.1 process.cpu_perc, process.mem.rss_mbytes, process.pid_count and process.thread_count metrics

Component Name	Dimensions	Description

apache-storm	`service=monitoring`
	`process_name=monasca-thresh`
	`process_user=storm`
		apache-storm process info: cpu percent, memory, pid count and thread count

barbican-api	`service=key-manager` `process_name=barbican-api`	barbican-api process info: cpu percent, momory, pid count and thread count
ceilometer-agent-notification	`service=telemetry` `process_name=ceilometer-agent-notification`	ceilometer-agent-notification process info: cpu percent, momory, pid count and thread count
ceilometer-api	`service=telemetry` `process_name=ceilometer-api`	ceilometer-api process info: cpu percent, momory, pid count and thread count

Component Name	Dimensions	Description

ceilometer-polling	`service=telemetry`
	`process_name=ceilometer-`
	`polling`
		ceilometer-polling process info: cpu percent, momory, pid count and thread count

cinder-api	`service=block-storage` `process_name=cinder-api`	cinder-api process info: cpu percent, momory, pid count and thread count

cinder-scheduler	`service=block-storage`
	`process_name=cinder-`
	`scheduler`
		cinder-scheduler process info: cpu percent, momory, pid count and thread count

designate-api	`service=dns` `process_name=designate-api`	designate-api process info: cpu percent, momory, pid count and thread count

designate-central	`service=dns`
	`process_name=designate-`
	`central`
		designate-central process info: cpu percent, momory, pid count and thread count

designate-mdns	`service=dns` `process_name=designate-mdns`	designate-mdns process cpu percent, momory, pid count and thread count

designate-pool-manager	`service=dns`
	`process_name=designate-`
	`pool-manager`
		designate-pool-manager process info: cpu percent, momory, pid count and thread count

freezer-scheduler	`service=backup`
	`process_name=freezer-`
	`scheduler`
		freezer-scheduler process info: cpu percent, momory, pid count and thread count

heat-api	`service=orchestration` `process_name=heat-api`	heat-api process cpu percent, momory, pid count and thread count

Component Name	Dimensions	Description
heat-api-cfn	`service=orchestration` `process_name=heat-api-cfn`	heat-api-cfn process info: cpu percent, momory, pid count and thread count

heat-api-cloudwatch	`service=orchestration`
	`process_name=heat-api-`
	`cloudwatch`
		heat-api-cloudwatch process cpu percent, momory, pid count and thread count

heat-engine	`service=orchestration` `process_name=heat-engine`	heat-engine process info: cpu percent, momory, pid count and thread count
ipsec/charon	`service=networking` `process_name=ipsec/charon`	ipsec/charon process info: cpu percent, momory, pid count and thread count
keystone-admin	`service=identity-service` `process_name=keystone-admin`	keystone-admin process info: cpu percent, momory, pid count and thread count
keystone-main	`service=identity-service` `process_name=keystone-main`	keystone-main process info: cpu percent, momory, pid count and thread count
monasca-agent	`service=monitoring` `process_name=monasca-agent`	monasca-agent process info: cpu percent, momory, pid count and thread count
monasca-api	`service=monitoring` `process_name=monasca-api`	monasca-api process info: cpu percent, momory, pid count and thread count

monasca-notification	`service=monitoring`
	`process_name=monasca-`
	`notification`
		monasca-notification process info: cpu percent, momory, pid count and thread count

monasca-persister	`service=monitoring`
	`process_name=monasca-`
	`persister`
		monasca-persister process info: cpu percent, momory, pid count and thread count

Component Name	Dimensions	Description

monasca-transform	`service=monasca-transform`
	`process_name=monasca-`
	`transform`
		monasca-transform process info: cpu percent, momory, pid count and thread count

neutron-dhcp-agent	`service=networking`
	`process_name=neutron-dhcp-`
	`agent`
		neutron-dhcp-agent process info: cpu percent, momory, pid count and thread count

neutron-l3-agent	`service=networking`
	`process_name=neutron-l3-`
	`agent`
		neutron-l3-agent process info: cpu percent, momory, pid count and thread count

neutron-lbaasv2-agent	`service=networking`
	`process_name:neutron-`
	`lbaasv2-agent`
		neutron-lbaasv2-agent process info: cpu percent, momory, pid count and thread count

neutron-metadata-agent	`service=networking`
	`process_name=neutron-`
	`metadata-agent`
		neutron-metadata-agent process info: cpu percent, momory, pid count and thread count

neutron-openvswitch-agent	`service=networking`
	`process_name=neutron-`
	`openvswitch-agent`
		neutron-openvswitch-agent process info: cpu percent, momory, pid count and thread count

neutron-rootwrap	`service=networking`
	`process_name=neutron-`
	`rootwrap`
		neutron-rootwrap process info: cpu percent, momory, pid count and thread count

neutron-server	`service=networking` `process_name=neutron-server`	neutron-server process info: cpu percent, momory, pid count and thread count

neutron-vpn-agent	`service=networking`
	`process_name=neutron-vpn-`
	`agent`
		neutron-vpn-agent process info: cpu percent, momory, pid count and thread count

Component Name	Dimensions	Description
nova-api	`service=compute` `process_name=nova-api`	nova-api process info: cpu percent, momory, pid count and thread count
nova-compute	`service=compute` `process_name=nova-compute`	nova-compute process info: cpu percent, momory, pid count and thread count
nova-conductor	`service=compute` `process_name=nova-conductor`	nova-conductor process info: cpu percent, momory, pid count and thread count

nova-consoleauth	`service=compute`
	`process_name=nova-`
	`consoleauth`
		nova-consoleauth process info: cpu percent, momory, pid count and thread count

nova-novncproxy	`service=compute`
	`process_name=nova-`
	`novncproxy`
		nova-novncproxy process info: cpu percent, momory, pid count and thread count

nova-scheduler	`service=compute` `process_name=nova-scheduler`	nova-scheduler process info: cpu percent, momory, pid count and thread count
octavia-api	`service=octavia` `process_name=octavia-api`	octavia-api process info: cpu percent, momory, pid count and thread count

octavia-health-manager	`service=octavia`
	`process_name=octavia-`
	`health-manager`
		octavia-health-manager process info: cpu percent, momory, pid count and thread count

octavia-housekeeping	`service=octavia`
	`process_name=octavia-`
	`housekeeping`
		octavia-housekeeping process info: cpu percent, momory, pid count and thread count

Component Name	Dimensions	Description
octavia-worker	`service=octavia` `process_name=octavia-worker`	octavia-worker process info: cpu percent, memory, pid count and thread count
org.apache.spark.deploy.master	`service=spark` `process_name=org.apache.spark.deploy.master.Master`	org.apache.spark.deploy.master.Master process info: cpu percent, momory, pid count and thread count
org.apache.spark.executor.Coar	`service=monasca-transform` `process_name=org.apache.spark.executor.CoarseGrainedExecutorBackend`	org.apache.spark.executor.CoarseGrainedEx process info: cpu percent, momory, pid count and thread count
pyspark	`service=monasca-transform` `process_name=pyspark`	pyspark process info: cpu percent, momory, pid count and thread count
transform/lib/driver	`service=monasca-transform` `process_name=transform/lib/driver`	transform/lib/driver process info: cpu percent, momory, pid count and thread count
cassandra	`service=cassandra` `process_name=cassandra`	cassandra process info: cpu percent, momory, pid count and thread count

12.1.4.17.2   process.io.*, process.open_file_descriptors metrics

Component Name	Dimensions	Description

monasca-agent	`service=monitoring`
	`process_name=monasca-agent`
	`process_user=mon-agent`
		monasca-agent process info: number of reads, number of writes,number of files being used

12.1.4.18   RabbitMQ Metrics

A list of metrics associated with the RabbitMQ service.

Metric Name	Dimensions	Description
rabbitmq.exchange.messages.pu	hostname exchange vhost type service=rabbitmq	Value of the "publish_out" field of "message_stats" object
rabbitmq.exchange.messages.pu	hostname exchange vhost type service=rabbitmq	Value of the "rate" field of "message_stats/ publish_out_details" object
rabbitmq.exchange.messages.re	hostname exchange vhost type service=rabbitmq	Value of the "publish_in" field of "message_stats" object
rabbitmq.exchange.messages.re	hostname exchange vhost type service=rabbitmq	Value of the "rate" field of "message_stats/ publish_in_details" object
rabbitmq.node.fd_used	hostname node service=rabbitmq	Value of the "fd_used" field in the response of /api/nodes
rabbitmq.node.mem_used	hostname node service=rabbitmq	Value of the "mem_used" field in the response of /api/nodes
rabbitmq.node.run_queue	hostname node service=rabbitmq	Value of the "run_queue" field in the response of /api/nodes

Metric Name	Dimensions	Description
| rabbitmq.node.sockets_used | ```
hostname
node
service=rabbitmq
``` | Value of the "sockets_used" field in the response of /api/ nodes |
| rabbitmq.queue.messages | ```
hostname
queue
vhost
service=rabbitmq
``` | Sum of ready and unacknowledged messages (queue depth) |
| rabbitmq.queue.messages.deliv | ```
hostname
queue
vhost
service=rabbitmq
``` | Value of the "rate" field of "message_stats/ deliver_details" object |
| rabbitmq.queue.messages.publi | ```
hostname
queue
vhost
service=rabbitmq
``` | Value of the "rate" field of "message_stats/ publish_details" object |
| rabbitmq.queue.messages.redel | ```
hostname
queue
vhost
service=rabbitmq
``` | Value of the "rate" field of "message_stats/ redeliver_details" object |

12.1.4.19 Swift Metrics

A list of metrics associated with the Swift service.

Metric Name	Dimensions	Description
| swiftlm.access.host.operation.ge | ```
service=object-storage
``` | This metric is the number of bytes read from objects in GET requests processed by this host during the last minute. Only successful GET requests to objects are |

Metric Name	Dimensions	Description
		counted. GET requests to the account or container is not included.
swiftlm.access.host.operation.op	`service=object-storage`	This metric is a count of the all the API requests made to Swift that were processed by this host during the last minute.
swiftlm.access.host.operation.project.get.bytes		
swiftlm.access.host.operation.project.ops		
swiftlm.access.host.operation.project.put.bytes		
swiftlm.access.host.operation.pr	`service=object-storage`	This metric is the number of bytes written to objects in PUT or POST requests processed by this host during the last minute. Only successful requests to objects are counted. Requests to the account or container is not included.
swiftlm.access.host.operation.status		
swiftlm.access.project.operation	`service=object-storage`	This metric reports whether the swiftlm-access-log-tailer program is running normally.
swiftlm.access.project.operation	`tenant_id` `service=object-storage`	This metric is a count of the all the API requests made to Swift that were processed by this host during the last minute to a given project id.

Metric Name	Dimensions	Description
swiftlm.access.project.operation	`tenant_id` `service=object-storage`	This metric is the number of bytes read from objects in GET requests processed by this host for a given project during the last minute. Only successful GET requests to objects are counted. GET requests to the account or container is not included.
swiftlm.access.project.operation	`tenant_id` `service=object-storage`	This metric is the number of bytes written to objects in PUT or POST requests processed by this host for a given project during the last minute. Only successful requests to objects are counted. Requests to the account or container is not included.
swiftlm.async_pending.cp.total.	`observer_host` `service=object-storage`	This metric reports the total length of all async pending queues in the system. When a container update fails, the update is placed on the async pending queue. An update may fail becuase the container server is too busy or because the server is down or failed. Later the system will "replay" updates from the queue – so eventually,

Metric Name	Dimensions	Description
		the container listings will show all objects known to the system.

If you know that container servers are down, it is normal to see the value of async pending increase. Once the server is restored, the value should return to zero.

A non-zero value may also indicate that containers are too large. Look for "lock timeout" messages in /var/log/swift/swift.log. If you find such messages consider reducing the container size or enable rate limiting. |
| swiftlm.check.failure | ```
check
error
component
service=object-storage
``` | The total exception string is truncated if longer than 1919 characters and an ellipsis is prepended in the first three characters of the message. If there is more than one error reported, the list of errors is paired to the last reported error and the operator is expected to resolve failures until no more are reported. Where there are no further reported errors, the Value Class is emitted as 'Ok'. |

Metric Name	Dimensions	Description
swiftlm.diskusage.cp.avg.usage	`observer_host` `service=object-storage`	Is the average utilization of all drives in the system. The value is a percentage (example: 30.0 means 30% of the total space is used).
swiftlm.diskusage.cp.max.usage	`observer_host` `service=object-storage`	Is the highest utilization of all drives in the system. The value is a percentage (example: 80.0 means at least one drive is 80% utilized). The value is just as important as swiftlm.diskusage.usage.avg. For example, if swiftlm.diskusage.usage.avg is 70% you might think that there is plenty of space available. However, if swiftlm.diskusage.usage.max is 100%, this means that some objects cannot be stored on that drive. Swift will store replicas on other drives. However, this will create extra overhead.
swiftlm.diskusage.cp.min.usage	`observer_host` `service=object-storage`	Is the lowest utilization of all drives in the system. The value is a percentage (example: 10.0 means at least one drive is 10% utilized)

Metric Name	Dimensions	Description
swiftlm.diskusage.cp.total.avail	`observer_host` `service=object-storage`	Is the size in bytes of available (unused) space of all drives in the system. Only drives used by Swift are included.
swiftlm.diskusage.cp.total.size	`observer_host` `service=object-storage`	Is the size in bytes of raw size of all drives in the system.
swiftlm.diskusage.cp.total.used	`observer_host` `service=object-storage`	Is the size in bytes of used space of all drives in the system. Only drives used by Swift are included.
swiftlm.diskusage.host.avg.usage	`hostname` `service=object-storage`	This metric reports the average percent usage of all Swift filesystems on a host.
swiftlm.diskusage.host.max.usa	`hostname` `service=object-storage`	This metric reports the percent usage of a Swift filesystem that is most used (full) on a host. The value is the max of the percentage used of all Swift filesystems.
swiftlm.diskusage.host.min.usag	`hostname` `service=object-storage`	This metric reports the percent usage of a Swift filesystem that is least used (has free space) on a host. The value is the min of the percentage used of all Swift filesystems.

Metric Name	Dimensions	Description

swiftlm.diskusage.host.val.avail	`hostname`
	`service=object-storage`
	`mount`
	`device`
	`label`
		This metric reports the number of bytes available (free) in a Swift filesystem. The value is an integer (units: Bytes)

swiftlm.diskusage.host.val.size	`hostname`
	`service=object-storage`
	`mount`
	`device`
	`label`
		This metric reports the size in bytes of a Swift filesystem. The value is an integer (units: Bytes)

swiftlm.diskusage.host.val.usag	`hostname`
	`service=object-storage`
	`mount`
	`device`
	`label`
		This metric reports the percent usage of a Swift filesystem. The value is a floating point number in range 0.0 to 100.0

swiftlm.diskusage.host.val.used	`hostname`
	`service=object-storage`
	`mount`
	`device`
	`label`
		This metric reports the number of used bytes in a Swift filesystem. The value is an integer (units: Bytes)

swiftlm.load.cp.avg.five	`observer_host` `service=object-storage`	This is the averaged value of the five minutes system load average of all nodes in the Swift system.
swiftlm.load.cp.max.five	`observer_host` `service=object-storage`	This is the five minute load average of the busiest host in the Swift system.
swiftlm.load.cp.min.five	`observer_host` `service=object-storage`	This is the five minute load average of the least loaded host in the Swift system.

Metric Name	Dimensions	Description
swiftlm.load.host.val.five	`hostname` `service=object-storage`	This metric reports the 5 minute load average of a host. The value is derived from `/proc/loadavg`.
swiftlm.md5sum.cp.check.ring_	`observer_host` `service=object-storage`	If you are in the middle of deploying new rings, it is normal for this to be in the failed state. However, if you are not in the middle of a deployment, you need to investigate the cause. Use "swift-recon –md5 -v" to identify the problem hosts.
swiftlm.replication.cp.avg.accou	`observer_host` `service=object-storage`	This is the average across all servers for the account replicator to complete a cycle. As the system becomes busy, the time to complete a cycle increases. The value is in seconds.
swiftlm.replication.cp.avg.conta	`observer_host` `service=object-storage`	This is the average across all servers for the container replicator to complete a cycle. As the system becomes busy, the time to complete a cycle increases. The value is in seconds.
swiftlm.replication.cp.avg.objec	`observer_host` `service=object-storage`	This is the average across all servers for the object replicator to complete a

Metric Name	Dimensions	Description
		cycle. As the system becomes busy, the time to complete a cycle increases. The value is in seconds.

swiftlm.replication.cp.max.acco	`hostname`
	`path`
	`service=object-storage`
		This is the number of seconds since the account replicator last completed a scan on the host that has the oldest completion time. Normally the replicators runs periodically and hence this value will decrease whenever a replicator completes. However, if a replicator is not completing a cycle, this value increases (by one second for each second that the replicator is not completing). If the value remains high and increasing for a long period of time, it indicates that one of the hosts is not completing the replication cycle.

swiftlm.replication.cp.max.cont	`hostname`
	`path`
	`service=object-storage`
		This is the number of seconds since the container replicator last completed a scan on the host that has the oldest completion time. Normally the replicators runs periodically and hence this value will decrease whenever a replicator completes.

Metric Name	Dimensions	Description
		However, if a replicator is not completing a cycle, this value increases (by one second for each second that the replicator is not completing). If the value remains high and increasing for a long period of time, it indicates that one of the hosts is not completing the replication cycle.

swiftlm.replication.cp.max.obje	`hostname`
	`path`
	`service=object-storage`
		This is the number of seconds since the object replicator last completed a scan on the host that has the oldest completion time. Normally the replicators runs periodically and hence this value will decrease whenever a replicator completes. However, if a replicator is not completing a cycle, this value increases (by one second for each second that the replicator is not completing). If the value remains high and increasing for a long period of time, it indicates that one of the hosts is not completing the replication cycle.

Metric Name	Dimensions	Description
swiftlm.swift.drive_audit	``` hostname service=object-storage mount_point kernel_device ```	If an unrecoverable read error (URE) occurs on a filesystem, the error is logged in the kernel log. The swift-drive-audit program scans the kernel log looking for patterns indicating possible UREs. To get more information, log onto the node in question and run: ``` sudoswift-drive-audit/etc/ swift/drive-audit.conf ``` UREs are common on large disk drives. They do not necessarily indicate that the drive is failed. You can use the xfs_repair command to attempt to repair the filesystem. Failing this, you may need to wipe the filesystem. If UREs occur very often on a specific drive, this may indicate that the drive is about to fail and should be replaced.
swiftlm.swift.file_ownership.cor	``` hostname path service ```	This metric reports if a directory or file has the appropriate owner. The check looks at Swift configuration directories and

Metric Name	Dimensions	Description
		files. It also looks at the top-level directories of mounted file systems (for example, /srv/node/disk0 and /srv/node/disk0/objects).

swiftlm.swift.file_ownership.dat	`hostname`
	`path`
	`service`
		This metric reports if a directory or file has the appropriate owner. The check looks at Swift configuration directories and files. It also looks at the top-level directories of mounted file systems (for example, /srv/node/disk0 and /srv/node/disk0/objects).

swiftlm.swiftlm_check	`hostname` `service=object-storage`	This indicates of the Swiftlm Monasca Agent Plug-in is running normally. If the status is failed, it probable that some or all metrics are no longer being reported.
swiftlm.swift.replication.accour	`hostname` `service=object-storage`	This reports how long (in seconds) since the replicator process last finished a replication run. If the replicator is stuck, the time will keep increasing forever. The time a replicator normally takes depends on disk sizes and how much

Metric Name	Dimensions	Description
		data needs to be replicated. However, a value over 24 hours is generally bad.
swiftlm.swift.replication.contai	`hostname` `service=object-storage`	This reports how long (in seconds) since the replicator process last finished a replication run. If the replicator is stuck, the time will keep increasing forever. The time a replicator normally takes depends on disk sizes and how much data needs to be replicated. However, a value over 24 hours is generally bad.
swiftlm.swift.replication.object.	`hostname` `service=object-storage`	This reports how long (in seconds) since the replicator process last finished a replication run. If the replicator is stuck, the time will keep increasing forever. The time a replicator normally takes depends on disk sizes and how much data needs to be replicated. However, a value over 24 hours is generally bad.
swiftlm.swift.swift_services	`hostname` `service=object-storage`	This metric reports of the process as named in the component dimension and the msg value_meta is running or not.

Metric Name	Dimensions	Description
		Use the `swift-start.yml` playbook to attempt to restart the stopped process (it will start any process that has stopped – you do not need to specifically name the process).

swiftlm.swift.swift_services.chec	`hostname`
	`service=object-storage`
	`component`
		Reports if a service is listening to the correct ip and port.

swiftlm.systems.check_mounts	`hostname`
	`service=object-storage`
	`mount`
	`device`
	`label`
		This metric reports the mount state of each drive that should be mounted on this node.

swiftlm.systems.connectivity.co	`observer_host`
	`url`
	`target_port`
	`service=object-storage`
		This metric reports if a server can connect to a VIPs. Currently the following VIPs are checked:
		
		• The Keystone VIP used to validate tokens (normally port 5000)

swiftlm.systems.connectivity.me	`observer_host`
	`hostname`
	`target_port`
	`service=object-storage`
		This metric reports if memcached on the host as specified by the hostname dimension is accepting connections from the host

Metric Name	Dimensions	Description
		running the check. The following value_meta.msg are used: We successfully connected to <hostname> on port <target_port>

```
{
  "dimensions": {
    "hostname": "ardana-
ccp-c1-m1-mgmt",
    "observer_host":
 "ardana-ccp-c1-m1-mgmt",
    "service": "object-
storage",
    "target_port": "11211"
  },
  "metric":
 "swiftlm.systems.connectivity.memcache_c
  "timestamp": 1449084058,
  "value": 0,
  "value_meta": {
    "msg": "ardana-ccp-c1-
m1-mgmt:11211 ok"
  }
}
```

We failed to connect to <hostname> on port <target_port>

```
{
  "dimensions": {
    "fail_message": "[Errno
 111] Connection refused",
    "hostname": "ardana-
ccp-c1-m1-mgmt",
    "observer_host":
 "ardana-ccp-c1-m1-mgmt",
    "service": "object-
storage",
    "target_port": "11211"
```

Metric Name	Dimensions	Description
| | | ```
},
"metric":
"swiftlm.systems.connectivity.memcache_cl
"timestamp": 1449084150,
"value": 2,
"value_meta": {
"msg": "ardana-ccp-c1-
m1-mgmt:11211 [Errno 111]
Connection refused"
}
}
``` |
| swiftlm.systems.connectivity.rsy | ```
observer_host
hostname
target_port
service=object-storage
``` | This metric reports if rsyncd on the host as specified by the hostname dimension is accepting connections from the host running the check. The following value_meta.msg are used:

We successfully connected to <hostname> on port <target_port>:

```
{
  "dimensions": {
    "hostname": "ardana-
ccp-c1-m1-mgmt",
    "observer_host":
 "ardana-ccp-c1-m1-mgmt",
    "service": "object-
storage",
    "target_port": "873"
  },
  "metric":
 "swiftlm.systems.connectivity.rsync_checl
  "timestamp": 1449082663,
  "value": 0,
  "value_meta": {
    "msg": "ardana-ccp-c1-
m1-mgmt:873 ok"
  }
``` |

Metric Name	Dimensions	Description
		}
		We failed to connect to <hostname> on port <target_port>:

		``
		{
		  "dimensions": {
		    "fail_message": "[Errno 111] Connection refused",
		    "hostname": "ardana-ccp-c1-m1-mgmt",
		    "observer_host": "ardana-ccp-c1-m1-mgmt",
		    "service": "object-storage",
		    "target_port": "873"
		  },
		  "metric": "swiftlm.systems.connectivity.rsync_chec
		  "timestamp": 1449082860,
		  "value": 2,
		  "value_meta": {
		    "msg": "ardana-ccp-c1-m1-mgmt:873 [Errno 111] Connection refused"
		  }
		}
		``

swiftlm.umon.target.avg.latency	``` component hostname observer_host service=object-storage url ```	Reports the average value of N-iterations of the latency values recorded for a component.
swiftlm.umon.target.check.state	``` component hostname observer_host service=object-storage url ```	This metric reports the state of each component after N-iterations of checks. If the initial check succeeds, the checks move onto the next component until all

Metric Name	Dimensions	Description
		components are queried, then the checks sleep for 'main_loop_interval' seconds. If a check fails, it is retried every second for 'retries' number of times per component. If the check fails 'retries' times, it is reported as a fail instance.

A successful state will be reported in JSON:

```
{
    "dimensions": {
        "component": "rest-
api",
        "hostname":
 "ardana-ccp-vip-admin-SWF-
PRX-mgmt",
        "observer_host":
 "ardana-ccp-c1-m1-mgmt",
        "service": "object-
storage",
        "url": "http://
ardana-ccp-vip-admin-SWF-
PRX-mgmt:8080"
    },
    "metric":
 "swiftlm.umon.target.check.state",
    "timestamp":
 1453111805,
    "value": 0
},
```

A failed state will report a "fail" value and the value_meta will provide the http response error.

```
{
```

Metric Name	Dimensions	Description

		"dimensions": {
		    "component": "rest-api",
		    "hostname":
		"ardana-ccp-vip-admin-SWF-PRX-mgmt",
		    "observer_host":
		"ardana-ccp-c1-m1-mgmt",
		    "service": "object-storage",
		    "url": "http://ardana-ccp-vip-admin-SWF-PRX-mgmt:8080"
		    },
		    "metric":
		"swiftlm.umon.target.check.state",
		    "timestamp":
		1453112841,
		    "value": 2,
		    "value_meta": {
		        "msg":
		"HTTPConnectionPool(host='192.168.245.9'
		port=8080): Max retries
		exceeded with url: /v1/
		AUTH_76538ce683654a35983b62e333001b47
		(Caused by
		NewConnectionError('<requests.packages.u
		object at 0x7fd857d7f550>:
		Failed to establish a new
		connection: [Errno 110]
		Connection timed out',))"
		    }
		}

swiftlm.umon.target.max.latenc	component hostname observer_host service=object-storage url	This metric reports the maximum response time in seconds of a REST call from the observer to the component REST API listening on the reported host

Metric Name	Dimensions	Description
		A response time query will be reported in JSON:

```
{
    "dimensions": {
        "component": "rest-
api",
        "hostname":
 "ardana-ccp-vip-admin-SWF-
PRX-mgmt",
        "observer_host":
 "ardana-ccp-c1-m1-mgmt",
        "service": "object-
storage",
        "url": "http://
ardana-ccp-vip-admin-SWF-
PRX-mgmt:8080"
    },
    "metric":
 "swiftlm.umon.target.max.latency_sec",
    "timestamp":
 1453111805,
    "value":
 0.2772650718688965
}
```

A failed query will have a much longer time value:

```
{
    "dimensions": {
        "component": "rest-
api",
        "hostname":
 "ardana-ccp-vip-admin-SWF-
PRX-mgmt",
        "observer_host":
 "ardana-ccp-c1-m1-mgmt",
        "service": "object-
storage",
        "url": "http://
ardana-ccp-vip-admin-SWF-
PRX-mgmt:8080"
    },
```

Metric Name	Dimensions	Description

		``
		    "metric":
		  "swiftlm.umon.target.max.latency_sec",
		    "timestamp":
		  1453112841,
		    "value":
		  127.288015127182
		  }
		``

swiftlm.umon.target.min.latenc	```
	component
	hostname
	observer_host
	service=object-storage
	url
	``
		This metric reports the minimum response time in seconds of a REST call from the observer to the component REST API listening on the reported host
		
		A response time query will be reported in JSON:
		
		``
		{
		    "dimensions": {
		      "component": "rest-
		api",
		      "hostname":
		  "ardana-ccp-vip-admin-SWF-
		PRX-mgmt",
		      "observer_host":
		  "ardana-ccp-c1-m1-mgmt",
		      "service": "object-
		storage",
		      "url": "http://
		ardana-ccp-vip-admin-SWF-
		PRX-mgmt:8080"
		    },
		    "metric":
		  "swiftlm.umon.target.min.latency_sec",
		    "timestamp":
		  1453111805,
		    "value":
		  0.10025882720947266
		  }
		``

Metric Name	Dimensions	Description
		A failed query will have a much longer time value:
| | | ```
{
    "dimensions": {
        "component": "rest-
api",
        "hostname":
 "ardana-ccp-vip-admin-SWF-
PRX-mgmt",
        "observer_host":
 "ardana-ccp-c1-m1-mgmt",
        "service": "object-
storage",
        "url": "http://
ardana-ccp-vip-admin-SWF-
PRX-mgmt:8080"
    },
    "metric":
 "swiftlm.umon.target.min.latency_sec",
    "timestamp":
 1453112841,
    "value":
 127.25378203392029
}
``` |
| swiftlm.umon.target.val.avail_d | ```
component
hostname
observer_host
service=object-storage
url
``` | This metric reports the average of all the collected records in the swiftlm.umon.target.val.avail_minute metric data. This is a walking average data set of these approximately per-minute states of the Swift Object Store. The most basic case is a whole day of successful per-minute records, which will average to 100% availability. If there is any downtime |

Alarm Metrics   HPE Helion OpenStack 8

Metric Name	Dimensions	Description
		throughout the day resulting in gaps of data which are two minutes or longer, the per-minute availability data will be "back filled" with an assumption of a down state for all the per-minute records which did not exist during the non-reported time. Because this is a walking average of approximately 24 hours worth of data, any outtage will take 24 hours to be purged from the dataset.
		A 24-hour average availability report:

```
{
    "dimensions": {
        "component": "rest-
api",
        "hostname":
 "ardana-ccp-vip-admin-SWF-
PRX-mgmt",
        "observer_host":
 "ardana-ccp-c1-m1-mgmt",
        "service": "object-
storage",
        "url": "http://
ardana-ccp-vip-admin-SWF-
PRX-mgmt:8080"
    },
    "metric":
 "swiftlm.umon.target.val.avail_day",
    "timestamp":
 1453645405,
    "value":
 7.894736842105263
}
```

Metric Name	Dimensions	Description
swiftlm.umon.target.val.avail_m	component hostname observer_host service=object-storage url	A value of 100 indicates that swift-uptime-monitor was able to get a token from Keystone and was able to perform operations against the Swift API during the reported minute. A value of zero indicates that either Keystone or Swift failed to respond successfully. A metric is produced every minute that swift-uptime-monitor is running. An "up" minute report value will report 100 [percent]:

```
{
    "dimensions": {
        "component": "rest-
api",
        "hostname":
 "ardana-ccp-vip-admin-SWF-
PRX-mgmt",
        "observer_host":
 "ardana-ccp-c1-m1-mgmt",
        "service": "object-
storage",
        "url": "http://
ardana-ccp-vip-admin-SWF-
PRX-mgmt:8080"
    },
    "metric":
 "swiftlm.umon.target.val.avail_minute",
    "timestamp":
 1453645405,
    "value": 100.0
}
```

Metric Name	Dimensions	Description

		A "down" minute report value will report 0 [percent]:
		
		``
		{
		    "dimensions": {
		        "component": "rest-api",
		        "hostname":
		 "ardana-ccp-vip-admin-SWF-PRX-mgmt",
		        "observer_host":
		 "ardana-ccp-c1-m1-mgmt",
		        "service": "object-storage",
		        "url": "http://ardana-ccp-vip-admin-SWF-PRX-mgmt:8080"
		    },
		    "metric":
		 "swiftlm.umon.target.val.avail_minute",
		    "timestamp":
		 1453649139,
		    "value": 0.0
		}
		``

swiftlm.hp_hardware.hpssacli.sr	``` component hostname service=object-storage component model controller_slot ```	This metric reports the firmware version of a component of a Smart Array controller.
swiftlm.hp_hardware.hpssacli.sr	``` component hostname service=object-storage component sub_component model controller_slot ```	This reports the status of various sub-components of a Smart Array Controller. A failure is considered to have occured if: • Controller is failed • Cache is not enabled or has failed

Metric Name	Dimensions	Description
		• Battery or capacitor is not installed • Battery or capacitor has failed

swiftlm.hp_hardware.hpssacli.p	`component`
	`hostname`
	`service=object-storage`
	`component`
	`controller_slot`
	`box`
	`bay`
		This reports the status of a disk drive attached to a Smart Array controller.

swiftlm.hp_hardware.hpssacli.lc	`component`
	`hostname`
	`observer_host`
	`service=object-storage`
	`controller_slot`
	`array`
	`logical_drive`
	`sub_component`
		This reports the status of a LUN presented by a Smart Array controller.
		
		A LUN is considered failed if the LUN has failed or if the LUN cache is not enabled and working.

⊱ Note

- HPE Smart Storage Administrator (HPE SSA) CLI component will have to be installed on all control nodes that are Swift nodes, in order to generate the following Swift metrics:

    - swiftlm.hp_hardware.hpssacli.smart_array

    - swiftlm.hp_hardware.hpssacli.logical_drive

- swiftlm.hp_hardware.hpssacli.smart_array.firmware

- swiftlm.hp_hardware.hpssacli.physical_drive

- HPE-specific binaries that are not based on open source are distributed directly from and supported by HPE. To download and install the SSACLI utility, please refer to: https://support.hpe.com/hpsc/swd/public/detail?swItemId=MTX_3d16386b418a443388c18da82f ↗

- After the HPE SSA CLI component is installed on the Swift nodes, the metrics will be generated automatically during the next agent polling cycle. Manual reboot of the node is not required.

12.1.4.20 System Metrics

A list of metrics associated with the System.

TABLE 12.8: CPU METRICS

Metric Name	Dimensions	Description

cpu.frequency_mhz	`cluster`
	`hostname`
	`service=system`
		Maximum MHz value for the cpu frequency.
		
		🎓 **Note**
		This value is dynamic, and driven by CPU governor depending on current resource need.

cpu.idle_perc	`cluster`
	`hostname`
	`service=system`
		Percentage of time the CPU is idle when no I/O requests are in progress

Metric Name	Dimensions	Description
cpu.idle_time	cluster hostname service=system	Time the CPU is idle when no I/O requests are in progress
cpu.percent	cluster hostname service=system	Percentage of time the CPU is used in total
cpu.stolen_perc	cluster hostname service=system	Percentage of stolen CPU time, that is, the time spent in other OS contexts when running in a virtualized environment
cpu.system_perc	cluster hostname service=system	Percentage of time the CPU is used at the system level
cpu.system_time	cluster hostname service=system	Time the CPU is used at the system level
cpu.time_ns	cluster hostname service=system	Time the CPU is used at the host level
cpu.total_logical_cores	cluster hostname service=system	Total number of logical cores available for an entire node (Includes hyper threading). ☞ Note: This is an optional metric that is only sent when send_rollup_stats is set to true.

Metric Name	Dimensions	Description
| cpu.user_perc | ```
cluster
hostname
service=system
``` | Percentage of time the CPU is used at the user level |
| cpu.user_time | ```
cluster
hostname
service=system
``` | Time the CPU is used at the user level |
| cpu.wait_perc | ```
cluster
hostname
service=system
``` | Percentage of time the CPU is idle AND there is at least one I/O request in progress |
| cpu.wait_time | ```
cluster
hostname
service=system
``` | Time the CPU is idle AND there is at least one I/O request in progress |

TABLE 12.9: DISK METRICS

Metric Name	Dimensions	Description
| disk.inode_used_perc | ```
mount_point
service=system
hostname
cluster
device
``` | The percentage of inodes that are used on a device |
| disk.space_used_perc | ```
mount_point
service=system
hostname
cluster
device
``` | The percentage of disk space that is being used on a device |
| disk.total_space_mb | ```
mount_point
service=system
hostname
cluster
device
``` | The total amount of disk space in Mbytes aggregated across all the disks on a particular node. |

Metric Name	Dimensions	Description
		Note This is an optional metric that is only sent when send_rollup_stats is set to true.
disk.total_used_space_mb	``` mount_point service=system hostname cluster device ```	The total amount of used disk space in Mbytes aggregated across all the disks on a particular node. **Note** This is an optional metric that is only sent when send_rollup_stats is set to true.
io.read_kbytes_sec	``` mount_point service=system hostname cluster device ```	Kbytes/sec read by an io device
io.read_req_sec	``` mount_point service=system hostname cluster device ```	Number of read requests/sec to an io device
io.read_time_sec	``` mount_point service=system hostname cluster device ```	Amount of read time in seconds to an io device

Metric Name	Dimensions	Description

io.write_kbytes_sec	`mount_point`
	`service=system`
	`hostname`
	`cluster`
	`device`
		Kbytes/sec written by an io device

io.write_req_sec	`mount_point`
	`service=system`
	`hostname`
	`cluster`
	`device`
		Number of write requests/sec to an io device

io.write_time_sec	`mount_point`
	`service=system`
	`hostname`
	`cluster`
	`device`
		Amount of write time in seconds to an io device

TABLE 12.10: LOAD METRICS

Metric Name	Dimensions	Description

load.avg_15_min	`service=system`
	`hostname`
	`cluster`
		The normalized (by number of logical cores) average system load over a 15 minute period

load.avg_1_min	`service=system`
	`hostname`
	`cluster`
		The normalized (by number of logical cores) average system load over a 1 minute period

load.avg_5_min	`service=system`
	`hostname`
	`cluster`
		The normalized (by number of logical cores) average system load over a 5 minute period

TABLE 12.11: MEMORY METRICS

Metric Name	Dimensions	Description
mem.free_mb	service=system hostname cluster	Mbytes of free memory
mem.swap_free_mb	service=system hostname cluster	Percentage of free swap memory that is free
mem.swap_free_perc	service=system hostname cluster	Mbytes of free swap memory that is free
mem.swap_total_mb	service=system hostname cluster	Mbytes of total physical swap memory
mem.swap_used_mb	service=system hostname cluster	Mbytes of total swap memory used
mem.total_mb	service=system hostname cluster	Total Mbytes of memory
mem.usable_mb	service=system hostname cluster	Total Mbytes of usable memory
mem.usable_perc	service=system hostname cluster	Percentage of total memory that is usable
mem.used_buffers	service=system hostname cluster	Number of buffers in Mbytes being used by the kernel for block io
mem.used_cache	service=system hostname cluster	Mbytes of memory used for the page cache

Metric Name	Dimensions	Description
mem.used_mb	service=system hostname cluster	Total Mbytes of used memory

Metric Name	Dimensions	Description
net.in_bytes_sec	service=system hostname device	Number of network bytes received per second
net.in_errors_sec	service=system hostname device	Number of network errors on incoming network traffic per second
net.in_packets_dropped_sec	service=system hostname device	Number of inbound network packets dropped per second
net.in_packets_sec	service=system hostname device	Number of network packets received per second
net.out_bytes_sec	service=system hostname device	Number of network bytes sent per second
net.out_errors_sec	service=system hostname device	Number of network errors on outgoing network traffic per second
net.out_packets_dropped_sec	service=system hostname device	Number of outbound network packets dropped per second
net.out_packets_sec	service=system hostname device	Number of network packets sent per second

12.1.4.21   Zookeeper Metrics

A list of metrics associated with the Zookeeper service.

Metric Name	Dimensions	Description
| zookeeper.avg_latency_sec | ```
hostname
mode
service=zookeeper
``` | Average latency in second |
| zookeeper.connections_count | ```
hostname
mode
service=zookeeper
``` | Number of connections |
| zookeeper.in_bytes | ```
hostname
mode
service=zookeeper
``` | Received bytes |
| zookeeper.max_latency_sec | ```
hostname
mode
service=zookeeper
``` | Maximum latency in second |
| zookeeper.min_latency_sec | ```
hostname
mode
service=zookeeper
``` | Minimum latency in second |
| zookeeper.node_count | ```
hostname
mode
service=zookeeper
``` | Number of nodes |
| zookeeper.out_bytes | ```
hostname
mode
service=zookeeper
``` | Sent bytes |
| zookeeper.outstanding_bytes | ```
hostname
mode
service=zookeeper
``` | Outstanding bytes |
| zookeeper.zxid_count | ```
hostname
mode
service=zookeeper
``` | Count number |
| zookeeper.zxid_epoch | ```
hostname
``` | Epoch number |

Metric Name	Dimensions	Description
	mode service=zookeeper	

12.2 Centralized Logging Service

You can use the Centralized Logging Service to evaluate and troubleshoot your distributed cloud environment from a single location.

12.2.1 Getting Started with Centralized Logging Service

A typical cloud consists of multiple servers which makes locating a specific log from a single server difficult. The Centralized Logging feature helps the administrator evaluate and troubleshoot the distributed cloud deployment from a single location.

The Logging API is a component in the centralized logging architecture. It works between log producers and log storage. In most cases it works by default after installation with no additional configuration. To use Logging API with logging-as-a-service, you must configure an end-point. This component adds flexibility and supportability for features in the future.

Do I need to Configure monasca-log-api? If you are only using Cloud Lifecycle Manager , then the default configuration is ready to use.

Important

If you are using logging in any of the following deployments, then you will need to query Keystone to get an end-point to use.

- Logging as a Service

- Platform as a Service

The Logging API is protected by Keystone's role-based access control. To ensure that logging is allowed and Monasca alarms can be triggered, the user must have the monasca-user role. **To get an end-point from Keystone:**

1. Log on to Cloud Lifecycle Manager (deployer node).

2. To list the Identity service catalog, run:

```
ardana > source ./service.osrc
ardana > openstack catalog list
```

3. In the output, find Kronos. For example:

Name	Type	Endpoints
kronos	region0	public: http://myardana.test:5607/v3.0, admin: http://192.168.245.5:5607/v3.0, internal: http://192.168.245.5:5607/v3.0

4. Use the same port number as found in the output. In the example, you would use port 5607.

In HPE Helion OpenStack, the logging-ansible restart playbook has been updated to manage the start,stop, and restart of the Centralized Logging Service in a specific way. This change was made to ensure the proper stop, start, and restart of Elasticsearch.

Important

It is recommended that you only use the logging playbooks to perform the start, stop, and restart of the Centralized Logging Service. Manually mixing the start, stop, and restart operations with the logging playbooks will result in complex failures.

For more information, see *Section 12.2.4, "Managing the Centralized Logging Feature"*.

12.2.1.1   For More Information

For more information about the centralized logging components, see the following sites:

- Logstash (https://www.elastic.co/guide/en/logstash/current/introduction.html) ↗

- Elasticsearch Guide (http://www.elasticsearch.org/guide) ↗

- Elasticsearch Scripting and Security (http://www.elasticsearch.org/blog/scripting-security) ↗

- Beaver (https://python-beaver.readthedocs.io/en/latest/) ↗

- Kibana Dashboard (http://www.elasticsearch.org/guide/en/kibana/current/index.html) ↗

- Apache Kafka (http://kafka.apache.org/) ↗

12.2.2   Understanding the Centralized Logging Service

The Centralized Logging feature collects logs on a central system, rather than leaving the logs scattered across the network. The administrator can use a single Kibana interface to view log information in charts, graphs, tables, histograms, and other forms.

12.2.2.1   What Components are Part of Centralized Logging?

Centralized logging consists of several components, detailed below:

- **Administrator's Browser:**   Operations Console can be used to access logging alarms or to access Kibana's dashboards to review logging data.

- **Apache Website for Kibana:**   A standard Apache website that proxies web/REST requests to the Kibana NodeJS server.

- **Beaver:**   A Python daemon that collects information in log files and sends it to the Logging API (monasca-log API) over a secure connection.

- **Cloud Auditing Data Federation (CADF):**   Defines a standard, full-event model anyone can use to fill in the essential data needed to certify, self-manage and self-audit application security in cloud environments.

- **Centralized Logging and Monitoring (CLM):**   Used to evaluate and troubleshoot your HPE Helion OpenStack distributed cloud environment from a single location.

- **Curator:** a tool provided by Elasticsearch to manage indices.

- **Elasticsearch:**   A data store offering fast indexing and querying.

- **HPE Helion OpenStack:**   Provides public, private, and managed cloud solutions to get you moving on your cloud journey.

- **JavaScript Object Notation (JSON) log file:** A file stored in the JSON format and used to exchange data. JSON uses JavaScript syntax, but the JSON format is text only. Text can be read and used as a data format by any programming language. This format is used by the Beaver and Logstash components.

- **Kafka:** A messaging broker used for collection of HPE Helion OpenStack centralized logging data across nodes. It is highly available, scalable and performant. Kafka stores logs in disk instead of memory and is therefore more tolerant to consumer down times.

  ## ⌂ Important

  Make sure not to undersize your Kafka partition or the data retention period may be lower than expected. If the Kafka partition capacity is lower than 85%, the retention period will increase to 30 minutes. Over time Kafka will also eject old data.

- **Kibana:** A client/server application with rich dashboards to visualize the data in Elasticsearch through a web browser. Kibana enables you to create charts and graphs using the log data.

- **Logging API (monasca-log-api):** HPE Helion OpenStack API provides a standard REST interface to store logs. It uses Keystone authentication and role-based access control support.

- **Logstash:** A log processing system for receiving, processing and outputting logs. Logstash retrieves logs from Kafka, processes and enriches the data, then stores the data in Elasticsearch.

- **MML Service Node:** Metering, Monitoring, and Logging (MML) service node. All services associated with metering, monitoring, and logging run on a dedicated three-node cluster. Three nodes are required for high availability with quorum.

- **Monasca:** OpenStack monitoring at scale infrastructure for the cloud that supports alarms and reporting.

- **OpenStack Service.** An OpenStack service process that requires logging services.

- **Oslo.log.** An OpenStack library for log handling. The library functions automate configuration, deployment and scaling of complete, ready-for-work application platforms. Some PaaS solutions, such as Cloud Foundry, combine operating systems, containers, and orchestrators with developer tools, operations utilities, metrics, and security to create a developer-rich solution.

- **Text log:** A type of file used in the logging process that contains human-readable records.

These components are configured to work out-of-the-box and the admin should be able to view log data using the default configurations.

In addition to each of the services, Centralized Logging also processes logs for the following features:

- HAProxy

- Syslog

- keepalived

The purpose of the logging service is to provide a common logging infrastructure with centralized user access. Since there are numerous services and applications running in each node of a HPE Helion OpenStack cloud, and there could be hundreds of nodes, all of these services and applications can generate enough log files to make it very difficult to search for specific events in log files across all of the nodes. Centralized Logging addresses this issue by sending log messages in real time to a central Elasticsearch, Logstash, and Kibana cluster. In this cluster they are indexed and organized for easier and visual searches. The following illustration describes the architecture used to collect operational logs.

☞ Note

The arrows come from the active (requesting) side to the passive (listening) side. The active side is always the one providing credentials, so the arrows may also be seen as coming from the credential holder to the application requiring authentication.

12.2.2.2   Steps 1- 2

Services configured to generate log files record the data. Beaver listens for changes to the files and sends the log files to the Logging Service. The first step the Logging service takes is to re-format the original log file to a new log file with text only and to remove all network operations. In Step 1a, the Logging service uses the Oslo.log library to re-format the file to text-only. In Step 1b, the Logging service uses the Python-Logstash library to format the original audit log file to a JSON file.

Step 1a

Beaver watches configured service operational log files for changes and reads incremental log changes from the files.

Step 1b

Beaver watches configured service operational log files for changes and reads incremental log changes from the files.

Step 2a

The monascalog transport of Beaver makes a token request call to Keystone passing in credentials. The token returned is cached to avoid multiple network round-trips.

Step 2b

The monascalog transport of Beaver batches multiple logs (operational or audit) and posts them to the monasca-log-api VIP over a secure connection. Failure logs are written to the local Beaver log.

Step 2c

The REST API client for monasca-log-api makes a token-request call to Keystone passing in credentials. The token returned is cached to avoid multiple network round-trips.

Step 2d

The REST API client for monasca-log-api batches multiple logs (operational or audit) and posts them to the monasca-log-api VIP over a secure connection.

12.2.2.3   Steps 3a- 3b

The Logging API (monasca-log API) communicates with Keystone to validate the incoming request, and then sends the logs to Kafka.

Step 3a

The monasca-log-api WSGI pipeline is configured to validate incoming request tokens with Keystone. The keystone middleware used for this purpose is configured to use the monasca-log-api admin user, password and project that have the required keystone role to validate a token.

Step 3b

Monasca-log-api sends log messages to Kafka using a language-agnostic TCP protocol.

12.2.2.4 Steps 4- 8

Logstash pulls messages from Kafka, identifies the log type, and transforms the messages into either the audit log format or operational format. Then Logstash sends the messages to Elasticsearch, using either an audit or operational indices.

Step 4

> Logstash input workers pull log messages from the Kafka-Logstash topic using TCP.

Step 5

> This Logstash filter processes the log message in-memory in the request pipeline. Logstash identifies the log type from this field.

Step 6

> This Logstash filter processes the log message in-memory in the request pipeline. If the message is of audit-log type, Logstash transforms it from the monasca-log-api envelope format to the original CADF format.

Step 7

> This Logstash filter determines which index should receive the log message. There are separate indices in Elasticsearch for operational versus audit logs.

Step 8

> Logstash output workers write the messages read from Kafka to the daily index in the local Elasticsearch instance.

12.2.2.5 Steps 9- 12

When an administrator who has access to the guest network accesses the Kibana client and makes a request, Apache forwards the request to the Kibana NodeJS server. Then the server uses the Elasticsearch REST API to service the client requests.

Step 9

> An administrator who has access to the guest network accesses the Kibana client to view and search log data. The request can originate from the external network in the cloud through a tenant that has a pre-defined access route to the guest network.

Step 10

> An administrator who has access to the guest network uses a web browser and points to the Kibana URL. This allows the user to search logs and view Dashboard reports.

Understanding the Centralized Logging Service HPE Helion OpenStack 8

Step 11

The authenticated request is forwarded to the Kibana NodeJS server to render the required dashboard, visualization, or search page.

Step 12

The Kibana NodeJS web server uses the Elasticsearch REST API in localhost to service the UI requests.

12.2.2.6 Steps 13- 15

Log data is backed-up and deleted in the final steps.

Step 13

A daily cron job running in the ELK node runs curator to prune old Elasticsearch log indices.

Step 14

The curator configuration is done at the deployer node through the Ansible role logging-common. Curator is scripted to then prune or clone old indices based on this configuration.

Step 15

The audit logs are configured to be backed up by the HPE Helion OpenStack Freezer product. For more information about Freezer (and Bura), see *Chapter 14, Backup and Restore*.

12.2.2.7 How Long are Log Files Retained?

The logs that are centrally stored are saved to persistent storage as Elasticsearch indices. These indices are stored in the partition `/var/lib/elasticsearch` on each of the Elasticsearch cluster nodes. Out of the box, logs are stored in one Elasticsearch index per service. As more days go by, the number of indices stored in this disk partition grows. Eventually the partition fills up. If they are **open**, each of these indices takes up CPU and memory. If these indices are left unattended they will continue to consume system resources and eventually deplete them.

Elasticsearch, by itself, does not prevent this from happening.

HPE Helion OpenStack uses a tool called curator that is developed by the Elasticsearch community to handle these situations. HPE Helion OpenStack installs and uses a curator in conjunction with several configurable settings. This curator is called by cron and performs the following checks:

- **First Check.** The hourly cron job checks to see if the currently used Elasticsearch partition size is over the value set in:

  ```
  curator_low_watermark_percent
  ```

  If it is higher than this value, the curator deletes old indices according to the value set in:

  ```
  curator_num_of_indices_to_keep
  ```

- **Second Check.** Another check is made to verify if the partition size is below the high watermark percent. If it is still too high, curator will delete all indices except the current one that is over the size as set in:

  ```
  curator_max_index_size_in_gb
  ```

- **Third Check.** A third check verifies if the partition size is still too high. If it is, curator will delete all indices except the current one.

- **Final Check.** A final check verifies if the partition size is still high. If it is, an error message is written to the log file but the current index is NOT deleted.

In the case of an extreme network issue, log files can run out of disk space in under an hour. To avoid this HPE Helion OpenStack uses a shell script called `logrotate_if_needed.sh`. The cron process runs this script every 5 minutes to see if the size of `/var/log` has exceeded the high_watermark_percent (95% of the disk, by default). If it is at or above this level, `logrotate_if_needed.sh` runs the `logrotate` script to rotate logs and to free up extra space. This script helps to minimize the chance of running out of disk space on `/var/log`.

12.2.2.8 How Are Logs Rotated?

HPE Helion OpenStack uses the cron process which in turn calls Logrotate to provide rotation, compression, and removal of log files. Each log file can be rotated hourly, daily, weekly, or monthly. If no rotation period is set then the log file will only be rotated when it grows too large.

Rotating a file means that the Logrotate process creates a copy of the log file with a new extension, for example, the .1 extension, then empties the contents of the original file. If a .1 file already exists, then that file is first renamed with a .2 extension. If a .2 file already exists, it is renamed to .3, etc., up to the maximum number of rotated files specified in the settings file. When Logrotate reaches the last possible file extension, it will delete the last file first on the next rotation. By the time the Logrotate process needs to delete a file, the results will have been copied to Elasticsearch, the central logging database.

The log rotation setting files can be found in the following directory

```
~/scratch/ansible/next/ardana/ansible/roles/logging-common/vars
```

These files allow you to set the following options:

Service

> The name of the service that creates the log entries.

Rotated Log Files

> List of log files to be rotated. These files are kept locally on the server and will continue to be rotated. If the file is also listed as Centrally Logged, it will also be copied to Elasticsearch.

Frequency

> The timing of when the logs are rotated. Options include:hourly, daily, weekly, or monthly.

Max Size

> The maximum file size the log can be before it is rotated out.

Rotation

> The number of log files that are rotated.

Centrally Logged Files

> These files will be indexed by Elasticsearch and will be available for searching in the Kibana user interface.

As an example, Freezer, the Backup and Restore (BURA) service, may be configured to create log files by setting the Rotated Log Files section to contain:

```
/var/log/freezer/freezer-scheduler.log
```

This configuration means that in the /var/log/freezer-agent directory, in a live environment, there should be a file called freezer-scheduler.log. As the log file grows, the cron process runs every hour to check the log file size against the settings in the configuration files. The example freezer-agent settings are described below.

Service	Node Type	Rotated Log Files	Frequenc	Max Size	Rotation	Centrally Logged Files
Freezer	Control	/var/log/freezer/ freezer-scheduler.log /var/log/freezer/ freezer-agent-json.log	Daily	45 MB	7	/var/log/freezer-agent/freezer-agent-json.log

For the `freezer-scheduler.log` file specifically, the information in the table tells the Logrotate process that the log file is to be rotated daily, and it can have a maximum size of 45 MB. After a week of log rotation, you might see something similar to this list:

```
freezer-scheduler.log at 10K
freezer-scheduler.log.1 at 123K
freezer-scheduler.log.2.gz at 13K
freezer-scheduler.log.3.gz at 17K
freezer-scheduler.log.4.gz at 128K
freezer-scheduler.log.5.gz at 22K
freezer-scheduler.log.6.gz at 323K
freezer-scheduler.log.7.gz at 123K
```

Since the Rotation value is set to 7 for this log file, there will never be a `freezer-scheduler.log.8.gz`. When the cron process runs its checks, if the `freezer-scheduler.log` size is more than 45 MB, then Logrotate rotates the file.

In this example, the following log files are rotated:

```
/var/log/freezer/freezer-scheduler.log
/var/log/freezer/freezer-agent-json.log
```

However, in this example, only the following file is centrally logged with Elasticsearch:

```
/var/log/freezer/freezer-agent-json.log
```

Only files that are listed in the **Centrally Logged Files** section are copied to Elasticsearch.

All of the variables for the Logrotate process are found in the following file:

```
~/scratch/ansible/next/ardana/ansible/roles/logging-ansible/logging-common/defaults/
main.yml
```

Cron runs Logrotate hourly. Every 5 minutes another process is run called **"logrotate_if_needed"** which uses a watermark value to determine if the Logrotate process needs to be run. If the **"high watermark"** has been reached, and the /var/log partition is more than 95% full (by default - this can be adjusted), then Logrotate will be run within 5 minutes.

12.2.2.9 Are Log Files Backed-Up To Elasticsearch?

While centralized logging is enabled out of the box, the backup of these logs is not. The reason is because Centralized Logging relies on the Elasticsearch FileSystem Repository plugin, which in turn requires shared disk partitions to be configured and accessible from each of the Elasticsearch nodes. Since there are multiple ways to setup a shared disk partition, HPE Helion OpenStack allows you to choose an approach that works best for your deployment before enabling the back-up of log files to Elasticsearch.

If you enable automatic back-up of centralized log files, then all the logs collected from the cloud nodes will be backed-up to Elasticsearch. Every hour, in the management controller nodes where Elasticsearch is setup, a cron job runs to check if Elasticsearch is running low on disk space. If the check succeeds, it further checks if the backup feature is enabled. If enabled, the cron job saves a snapshot of the Elasticsearch indices to the configured shared disk partition using curator. Next, the script starts deleting the oldest index and moves down from there checking each time if there is enough space for Elasticsearch. A check is also made to ensure that the backup runs only once a day.

For steps on how to enable automatic back-up, see *Section 12.2.5, "Configuring Centralized Logging"*.

12.2.3 Accessing Log Data

All logging data in HPE Helion OpenStack is managed by the Centralized Logging Service and can be viewed or analyzed by Kibana. Kibana is the only graphical interface provided with HPE Helion OpenStack to search or create a report from log data. Operations Console provides only a link to the Kibana Logging dashboard.

The following two methods allow you to access the Kibana Logging dashboard to search log data:

- *Section 12.2.3.1, "Use the Operations Console Link"*

- *Section 12.2.3.2, "Using Kibana to Access Log Data"*

To learn more about Kibana, read the Getting Started with Kibana (https://www.elastic.co/guide/en/kibana/current/getting-started.html) ⬈ guide.

12.2.3.1 Use the Operations Console Link

Operations Console allows you to access Kibana in the same tool that you use to manage the other HPE Helion OpenStack resources in your deployment. To use Operations Console, you must have the correct permissions. For more about permission requirements, see *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.2 "Connecting to the Operations Console"*.

To use Operations Console:

1. In a browser, open the Operations Console.

2. On the login page, enter the user name, and the **Password**, and then click **LOG IN**.

3. On the **Home/Central Dashboard** page, click the menu represented by 3 horizontal lines (≡).

4. From the menu that slides in on the left, select **Home**, and then select **Logging**.

5. On the **Home/Logging** page, click **View Logging Dashboard**.

Important

In HPE Helion OpenStack, Kibana usually runs on a different network than Operations Console. Due to this configuration, it is possible that using Operations Console to access Kibana will result in an "404 not found" error. This error only occurs if the user has access only to the public facing network.

12.2.3.2 Using Kibana to Access Log Data

Kibana is an open-source, data-visualization plugin for Elasticsearch. Kibana provides visualization capabilities using the log content indexed on an Elasticsearch cluster. Users can create bar and pie charts, line and scatter plots, and maps using the data collected by HPE Helion OpenStack in the cloud log files.

While creating Kibana dashboards is beyond the scope of this document, it is important to know that the dashboards you create are JSON files that you can modify or create new dashboards based on existing dashboards.

📖 Note

Kibana is client-server software. To operate properly, the browser must be able to access port 5601 on the control plane.

Field	Default Value	Description
user	kibana	Username that will be required for logging into the Kibana UI.
password	random password is generated	Password generated during installation that is used to login to the Kibana UI.

12.2.3.3 Logging into Kibana

To log into Kibana to view data, you must make sure you have the required login configuration.

1. Verify login credentials: *Section 12.2.3.3.1, "Verify Login Credentials"*

2. Find the randomized password: *Section 12.2.3.3.2, "Find the Randomized Password"*

3. Access Kibana using a direct link: *Section 12.2.3.3.3, "Access Kibana Using a Direct Link:"*

12.2.3.3.1 Verify Login Credentials

During the installation of Kibana, a password is automatically set and it is randomized. Therefore, unless an administrator has already changed it, you need to retrieve the default password from a file on the control plane node.

12.2.3.3.2 Find the Randomized Password

1. To find the Kibana password, run:

```
ardana > grep kibana ~/scratch/ansible/next/my_cloud/stage/internal/CloudModel.yaml
```

12.2.3.3.3 Access Kibana Using a Direct Link:

This section helps you verify the Horizon virtual IP (VIP) address that you should use.

1. To find hostname, run:

```
ardana > grep -i log-svr /etc/hosts
```

2. Navigate to the following directory:

```
ardana > ~/openstack/my_cloud/definition/data
```

> **Note**
>
> The file `network_groups.yml` in the `~/openstack/my_cloud/definition/data`
> directory is the input model file that may be copied automatically to other
> directories.

3. Open the following file for editing:

```
network_groups.yml
```

4. Find the following entry:

```
external-name
```

5. If your administrator set a hostname value in the *EXTERNAL_NAME* field during the
   configuration process for your cloud, then Kibana will be accessed over port 5601 on that
   hostname.

6. If your administrator did not set a hostname value, then to determine which IP address to
   use, from your Cloud Lifecycle Manager, run:

```
ardana > grep HZN-WEB /etc/hosts
```

The output of the grep command should show you the virtual IP address for Kibana that you should use.

Important

If nothing is returned by the grep command, you can open the following file to look for the IP address manually:

```
/etc/hosts
```

Access to Kibana will be over port 5601 of that virtual IP address. Example:

```
https://VIP:5601
```

12.2.4   Managing the Centralized Logging Feature

No specific configuration tasks are required to use Centralized Logging, as it is enabled by default after installation. However, you can configure the individual components as needed for your environment.

12.2.4.1   How Do I Stop and Start the Logging Service?

Although you might not need to stop and start the logging service very often, you may need to if, for example, one of the logging services is not behaving as expected or not working.

You cannot enable or disable centralized logging across all services unless you stop all centralized logging. Instead, it is recommended that you enable or disable individual log files in the <service>-clr.yml files and then reconfigure logging. You would enable centralized logging for a file when you want to make sure you are able to monitor those logs in Kibana.

In HPE Helion OpenStack, the logging-ansible restart playbook has been updated to manage the start,stop, and restart of the Centralized Logging Service in a specific way. This change was made to ensure the proper stop, start, and restart of Elasticsearch.

> **Important**
>
> It is recommended that you only use the logging playbooks to perform the start, stop, and restart of the Centralized Logging Service. Manually mixing the start, stop, and restart operations with the logging playbooks will result in complex failures.

The steps in this section only impact centralized logging. Logrotate is an essential feature that keeps the service log files from filling the disk and will not be affected.

> **Important**
>
> These playbooks must be run from the Cloud Lifecycle Manager.

To stop the Logging service:

1. To change to the directory containing the ansible playbook, run

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ```

2. To run the ansible playbook that will stop the logging service, run:

   ```
   ardana > ansible-playbook -i hosts/verb_hosts logging-stop.yml
   ```

To start the Logging service:

1. To change to the directory containing the ansible playbook, run

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ```

2. To run the ansible playbook that will stop the logging service, run:

   ```
   ardana > ansible-playbook -i hosts/verb_hosts logging-start.yml
   ```

12.2.4.2 How Do I Enable or Disable Centralized Logging For a Service?

To enable or disable Centralized Logging for a service you need to modify the configuration for the service, set the **enabled** flag to **true** or **false**, and then reconfigure logging.

Important

There are consequences if you enable too many logging files for a service. If there is not enough storage to support the increased logging, the retention period of logs in Elasticsearch is decreased. Alternatively, if you wanted to increase the retention period of log files or if you did not want those logs to show up in Kibana, you would disable centralized logging for a file.

To enable Centralized Logging for a service:

1. Use the documentation provided with the service to ensure it is not configured for logging.

2. To find the HPE Helion OpenStack file to edit, run:

   ```
   ardana > find ~/openstack/my_cloud/config/logging/vars/ -name "*service-name*"
   ```

3. Edit the file for the service for which you want to enable logging.

4. To enable Centralized Logging, find the following code and change the enabled flag to **true**, to disable, change the enabled flag to **false**:

   ```
   logging_options:
    - centralized_logging:
          enabled: true
          format: json
   ```

5. Save the changes to the file.

6. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

7. To reconfigure logging, run:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts kronos-reconfigure.yml
   ardana > cd ~/openstack/ardana/ansible/
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

Sample of a Freezer file enabled for Centralized logging:

```
---
sub_service:
   hosts: FRE-AGN
   name: freezer-agent
   service: freezer
   monitoring:
      enabled: true
      external_name: backup
      logging_dir: /var/log/freezer
   logging_options:
    - files:
       - /var/log/freezer/freezer-agent.log
       - /var/log/freezer/freezer-scheduler.log
       - centralized_logging:
          enabled: true
          format: json
```

12.2.5 Configuring Centralized Logging

You can adjust the settings for centralized logging when you are troubleshooting problems with a service or to decrease log size and retention to save on disk space. For steps on how to configure logging settings, refer to the following tasks:

- *Section 12.2.5.1, "Configuration Files"*

- *Section 12.2.5.2, "Planning Resource Requirements"*

- *Section 12.2.5.3, "Backing Up Elasticsearch Log Indices"*

- *Section 12.2.5.4, "Restoring Logs From an Elasticsearch Backup"*

- *Section 12.2.5.5, "Tuning Logging Parameters"*

12.2.5.1 Configuration Files

Centralized Logging settings are stored in the configuration files in the following directory on the Cloud Lifecycle Manager: `~/openstack/my_cloud/config/logging/`

The configuration files and their use are described below:

File	Description
main.yml	Main configuration file for all centralized logging components.
elasticsearch.yml.j2	Main configuration file for Elasticsearch.
elasticsearch-default.j2	Default overrides for the Elasticsearch init script.
kibana.yml.j2	Main configuration file for Kibana.
kibana-apache2.conf.j2	Apache configuration file for Kibana.
logstash.conf.j2	Logstash inputs/outputs configuration.
logstash-default.j2	Default overrides for the Logstash init script.
beaver.conf.j2	Main configuration file for Beaver.
vars	Path to logrotate configuration files.

12.2.5.2 Planning Resource Requirements

The Centralized Logging service needs to have enough resources available to it to perform adequately for different scale environments. The base logging levels are tuned during installation according to the amount of RAM allocated to your control plane nodes to ensure optimum performance.

These values can be viewed and changed in the `~/openstack/my_cloud/config/logging/main.yml` file, but you will need to run a reconfigure of the Centralized Logging service if changes are made.

⬡ Warning

The total process memory consumption for Elasticsearch will be the above allocated heap value (in `~/openstack/my_cloud/config/logging/main.yml`) plus any Java Virtual Machine (JVM) overhead.

Setting Disk Size Requirements

In the entry-scale models, the disk partition sizes on your controller nodes for the logging and Elasticsearch data are set as a percentage of your total disk size. You can see these in the following file on the Cloud Lifecycle Manager (deployer): `~/openstack/my_cloud/` `definition/data/<controller_disk_files_used>`

Sample file settings:

```
# Local Log files.
- name: log
  size: 13%
  mount: /var/log
  fstype: ext4
  mkfs-opts: -O large_file

# Data storage for centralized logging. This holds log entries from all
# servers in the cloud and hence can require a lot of disk space.
- name: elasticsearch
  size: 30%
  mount: /var/lib/elasticsearch
  fstype: ext4
```

🔔 Important

The disk size is set automatically based on the hardware configuration. If you need to adjust it, you can set it manually with the following steps.

To set disk sizes:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Open the following file:

   ```
   ~/openstack/my_cloud/definition/data/disks.yml
   ```

3. Make any desired changes.

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A git
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. To create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. To run the logging reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts kronos-reconfigure.yml
```

12.2.5.3 Backing Up Elasticsearch Log Indices

The log files that are centrally collected in HPE Helion OpenStack are stored by Elasticsearch on disk in the `/var/lib/elasticsearch` partition. However, this is distributed across each of the Elasticsearch cluster nodes as shards. A cron job runs periodically to see if the disk partition runs low on space, and, if so, it runs curator to delete the old log indices to make room for new logs. This deletion is permanent and the logs are lost forever. If you want to backup old logs, for example to comply with certain regulations, you can configure automatic backup of Elasticsearch indices.

🔔 Important

If you need to restore data that was archived prior to HPE Helion OpenStack 8 and used the older versions of Elasticsearch, then this data will need to be restored to a separate deployment of Elasticsearch.

This can be accomplished using the following steps:

1. Deploy a separate distinct Elasticsearch instance version matching the version in HPE Helion OpenStack.

2. Configure the backed-up data using NFS or some other share mechanism to be available to the Elasticsearch instance matching the version in HPE Helion OpenStack.

Before enabling automatic back-ups, make sure you understand how much disk space you will need, and configure the disks that will store the data. Use the following checklist to prepare your deployment for enabling automatic backups:

#	Item
☐	Add a shared disk partition to each of the Elasticsearch controller nodes. The default partition name used for backup is `/var/lib/esbackup` You can change this by: 1. Open the following file: `my_cloud/config/logging/main.yml` 2. Edit the following variable `curator_es_backup_partition`
☐	Ensure the shared disk has enough storage to retain backups for the desired retention period.

To enable automatic back-up of centralized logs to Elasticsearch:

1. Log in to the Cloud Lifecyle Manager (deployer node).

2. Open the following file in a text editor:

   ```
   ~/openstack/my_cloud/config/logging/main.yml
   ```

3. Find the following variables:

   ```
   curator_backup_repo_name: "es_{{host.my_dimensions.cloud_name}}"
   curator_es_backup_partition: /var/lib/esbackup
   ```

4. To enable backup, change the **curator_enable_backup** value to **true** in the curator section:

   ```
   curator_enable_backup: true
   ```

5. Save your changes and re-run the configuration processor:

   ```
   ardana > cd ~/openstack
   ardana > git add -A
   # Verify the added files
   ```

```
ardana > git status
ardana > git commit -m "Enabling Elasticsearch Backup"

$ cd ~/openstack/ardana/ansible
$ ansible-playbook -i hosts/localhost config-processor-run.yml
$ ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. To re-configure logging:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts kronos-reconfigure.yml
```

7. To verify that the indices are backed up, check the contents of the partition:

```
ardana > ls /var/lib/esbackup
```

12.2.5.4 Restoring Logs From an Elasticsearch Backup

To restore logs from an Elasticsearch backup, see https://www.elastic.co/guide/en/elasticsearch/
reference/2.4/modules-snapshots.html ↗.

☞ Note

We do not recommend restoring to the original HPE Helion OpenStack Centralized
Logging cluster as it may cause storage/capacity issues. We rather recommend setting up
a separate ELK cluster of the same version and restoring the logs there.

12.2.5.5 Tuning Logging Parameters

When centralized logging is installed in HPE Helion OpenStack, parameters for Elasticsearch
heap size and logstash heap size are automatically configured based on the amount of RAM on
the system. These values are typically the required values, but they may need to be adjusted if
performance issues arise, or disk space issues are encountered. These values may also need to
be adjusted if hardware changes are made after an installation.

These values are defined at the top of the following file `.../logging-common/defaults/`
`main.yml`. An example of the contents of the file is below:

```
1. Select heap tunings based on system RAM
```

```
#------------------------------------------------------------------------------
threshold_small_mb: 31000
threshold_medium_mb: 63000
threshold_large_mb: 127000
tuning_selector: " {% if ansible_memtotal_mb < threshold_small_mb|int %}
demo
{% elif ansible_memtotal_mb < threshold_medium_mb|int %}
small
{% elif ansible_memtotal_mb < threshold_large_mb|int %}
medium
{% else %}
large
{%endif %}
"

logging_possible_tunings:
2. RAM < 32GB
demo:
elasticsearch_heap_size: 512m
logstash_heap_size: 512m
3. RAM < 64GB
small:
elasticsearch_heap_size: 8g
logstash_heap_size: 2g
4. RAM < 128GB
medium:
elasticsearch_heap_size: 16g
logstash_heap_size: 4g
5. RAM >= 128GB
large:
elasticsearch_heap_size: 31g
logstash_heap_size: 8g
logging_tunings: "{{ logging_possible_tunings[tuning_selector] }}"
```

This specifies thresholds for what a **small**, **medium**, or **large** system would look like, in terms of
memory. To see what values will be used, see what RAM your system uses, and see where it fits
in with the thresholds to see what values you will be installed with. To modify the values, you
can either adjust the threshold values so that your system will change from a **small** configuration
to a **medium** configuration, for example, or keep the threshold values the same, and modify
the heap_size variables directly for the selector that your system is set for. For example, if your
configuration is a **medium** configuration, which sets heap_sizes to 16 GB for Elasticsearch and
4 GB for logstash, and you want twice as much set aside for logstash, then you could increase
the 4 GB for logstash to 8 GB.

12.2.6 Configuring Settings for Other Services

When you configure settings for the Centralized Logging Service, those changes impact all services that are enabled for centralized logging. However, if you only need to change the logging configuration for one specific service, you will want to modify the service's files instead of changing the settings for the entire Centralized Logging service. This topic helps you complete the following tasks:

- *Section 12.2.6.1, "Setting Logging Levels for Services"*

- *Section 12.2.6.19, "Selecting Files for Centralized Logging"*

- *Section 12.2.6.20, "Controlling Disk Space Allocation and Retention of Log Files"*

- *Section 12.2.6.21, "Configuring Elasticsearch for Centralized Logging"*

- *Section 12.2.6.22, "Safeguards for the Log Partitions Disk Capacity"*

12.2.6.1 Setting Logging Levels for Services

When it is necessary to increase the logging level for a specific service to troubleshoot an issue, or to decrease logging levels to save disk space, you can edit the service's config file and then reconfigure logging. All changes will be made to the service's files and not to the Centralized Logging service files.

Messages only appear in the log files if they are the same as or more severe than the log level you set. The DEBUG level logs everything. Most services default to the INFO logging level, which lists informational events, plus warnings, errors, and critical errors. Some services provide other logging options which will narrow the focus to help you debug an issue, receive a warning if an operation fails, or if there is a serious issue with the cloud.

For more information on logging levels, see the OpenStack Logging Guidelines (http://specs.openstack.org/openstack/openstack-specs/specs/log-guidelines.html) documentation.

12.2.6.2 Configuring the Logging Level for a Service

If you want to increase or decrease the amount of details that are logged by a service, you can change the current logging level in the configuration files. Most services support, at a minimum, the DEBUG and INFO logging levels. For more information about what levels are supported by a service, check the documentation or Website for the specific service.

12.2.6.3 Barbican

Service	Sub-component	Supported Logging Levels
Barbican	barbican-api	INFO (default) DEBUG

To change the Barbican logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Open the following file:

   ```
   ardana > cd ~/openstack/my_cloud/config/barbican/barbican_deploy_config.yml
   ```

3. To change the logging level, use ALL CAPS to set the desired level in the following lines:

   ```
   barbican_loglevel:  {{ openstack_loglevel | default('INFO') }}
   barbican_logstash_loglevel:  {{ openstack_loglevel | default('INFO') }}
   ```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

7. To create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

8. To run the reconfigure playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ```

```
ardana > ansible-playbook -i hosts/verb_hosts barbican-reconfigure.yml
```

12.2.6.4 Block Storage (Cinder)

Service	Sub-component	Supported Logging Levels
Cinder	cinder-local cinder-logstash	INFO DEBUG (default)

To enable Cinder logging:

1. On each Control Node, edit `/opt/stack/service/cinder-volume-CURRENT_VENV/etc/volume-logging.conf`
   In the `Writes to disk` section, change `WARNING` to `DEBUG`.

   ```
   # Writes to disk
   [handler_watchedfile]
   class: handlers.WatchedFileHandler
   args: ('/var/log/cinder/cinder-volume.log',)
   formatter: context
   # level: WARNING
   level: DEBUG
   ```

2. On the Cloud Lifecycle Manager (deployer) node, edit `/var/lib/ardana/openstack/my_cloud/config/cinder/volume.conf.j2`, adding a line `debug = TRUE` to the default section.

   ```
   [DEFAULT]
   log_config_append={{cinder_volume_conf_dir }}/volume-logging.conf
   debug = True
   ```

3. Run the following commands:

   ```
   ardana > cd ~/openstack/ardana/ansible/
   ardana > git commit -am "Enable Cinder Debug"
   ardana > ansible-playbook config-processor-run.yml
   ardana > ansible-playbook ready-deployment.yml
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook cinder-reconfigure.yml
   ardana > sudo grep -i debug /opt/stack/service/cinder-volume-CURRENT_VENV/etc/volume.conf
   ```

```
debug = True
```

🔔 Important

Leaving debugs enabled is not recommended. After collecting necessary logs, disable debug with the following steps:

1. On the Cloud Lifecycle Manager (deployer) node, edit `/var/lib/ardana/openstack/my_cloud/config/cinder/volume.conf.j2`, comment the line `debug = TRUE` in the default section.

   ```
   [DEFAULT]
   log_config_append={{cinder_volume_conf_dir }}/volume-logging.conf
   #debug = True
   ```

2. Run the following commands:

   ```
   ardana > cd ~/openstack/ardana/ansible/
   ardana > git commit -am "Disable Cinder Debug"
   ardana > ansible-playbook config-processor-run.yml
   ardana > ansible-playbook ready-deployment.yml
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook cinder-reconfigure.yml
   ardana > sudo grep -i debug /opt/stack/service/cinder-volume-CURRENT_VENV/etc/
   volume.conf
   ```

   ```
   #debug = True
   ```

12.2.6.5 Ceilometer

Service	Sub-component	Supported Logging Levels
Ceilometer	ceilometer-api ceilometer-collector ceilometer-agent-notification ceilometer-agent-central ceilometer-expirer	INFO (default) DEBUG

To change the Ceilometer logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Open the following file:

   ```
   ~/openstack/ardana/ansible/roles/_CEI-CMN/defaults/main.yml
   ```

3. To change the logging level, use ALL CAPS to set the desired level in the following lines:

   ```
   ceilometer_loglevel:  INFO
   ceilometer_logstash_loglevel:  INFO
   ```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

7. To create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

8. To run the reconfigure playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts ceilometer-reconfigure.yml
   ```

12.2.6.6 Compute (Nova)

Service	Sub-component	Supported Logging Levels
nova		INFO (default)

Service	Sub-component	Supported Logging Levels
		DEBUG

To change the Nova logging level:

1. Log in to the Cloud Lifecycle Manager.

2. The Neutron service component logging can be changed by modifying the following files:

   ```
   ~/openstack/my_cloud/config/nova/novncproxy-logging.conf.j2
   ~/openstack/my_cloud/config/nova/api-logging.conf.j2
   ~/openstack/my_cloud/config/nova/compute-logging.conf.j2
   ~/openstack/my_cloud/config/nova/conductor-logging.conf.j2
   ~/openstack/my_cloud/config/nova/consoleauth-logging.conf.j2
   ~/openstack/my_cloud/config/nova/scheduler-logging.conf.j2
   ```

3. To change the logging level, use ALL CAPS to set the desired level in the following line in the [handler_logstash] section:

   ```
   level: INFO
   ```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

7. To create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

8. To run the reconfigure playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ```

```
ardana > ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

12.2.6.7 Designate

Service	Sub-component	Supported Logging Levels
Designate	designate-api	INFO (default)
	designate-central	DEBUG
	designate-mdns	
	designate-pool-manager	
	designate-zone-manager	
	designate-api-json	
	designate-central-json	
	designate-mdns-json	
	designate-pool-manager-json	
	designate-zone-manager-json	

○ Important

To change the logging level, see the OpenStack Designate documentation (http://docs.openstack.org/developer/designate/) ↗ .

12.2.6.8 Freezer

Service	Sub-component	Supported Logging Levels
Freezer	freezer-agent	INFO (default)
	freezer-api	
	freezer-scheduler	

○ Important

Currently the freezer service does not support any level other than INFO.

12.2.6.9 ARDANA-UX-Services

Service	Sub-component	Supported Logging Levels
ARDANA-UX-Services		INFO (default) DEBUG

To change the ARDANA-UX-Services logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Open the following file:

   ```
   ~/openstack/ardana/ansible/roles/HUX-SVC/defaults/main.yml
   ```

3. To change the logging level, set the desired level in the following line:

   ```
   hux_svc_default_log_level: info
   ```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

7. To create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

8. To run the reconfigure playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts ardana-ux-services-reconfigure.yml
   ```

12.2.6.10 Identity (Keystone)

Service	Sub-component	Supported Logging Levels
Keystone	key-api	INFO (default) DEBUG WARN ERROR

To change the Keystone logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Open the following file:

   ```
   ~/openstack/my_cloud/config/keystone/keystone_deploy_config.yml
   ```

3. To change the logging level, use ALL CAPS to set the desired level in the following lines:

   ```
   keystone_loglevel: INFO
   keystone_logstash_loglevel: INFO
   ```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

7. To create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

8. To run the reconfigure playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
   ```

12.2.6.11 Image (Glance)

Service	Sub-component	Supported Logging Levels
Glance	glance-api glance-registry	INFO (default) DEBUG

To change the Glance logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Open the following file:

   ```
   ~/openstack/my_cloud/config/glance/glance-[api,registry]-logging.conf.j2
   ```

3. To change the logging level, use ALL CAPS to set the desired level in the following line in the [handler_logstash] section:

   ```
   level: INFO
   ```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

7. To create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

8. To run the reconfigure playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts glance-reconfigure.yml
   ```

12.2.6.12 Ironic

Service	Sub-component	Supported Logging Levels
ironic	ironic-api-logging.conf.j2 ironic-conductor-logging.conf.j2	INFO (default) DEBUG

To change the Ironic logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Change to the following directory:

   ```
   ~/openstack/my_cloud/config/ironic
   ```

3. To change the logging for one of the sub-components, open one of the following files:

   ```
   ironic-api-logging.conf.j2
   ironic-conductor-logging.conf.j2
   ```

4. To change the logging level, use ALL CAPS to set the desired level in the following line in the [handler_logstash] section:

   ```
   level: INFO
   ```

5. Save the changes to the file.

6. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

7. To run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

8. To create a deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ```

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

9. To run the reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

12.2.6.13 Monitoring (Monasca)

Service	Sub-component	Supported Logging Levels
monasca	monasca-persister zookeeper storm monasca-notification monasca-api kafka monasca-agent	WARN (default) INFO

To change the Monasca logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Monitoring service component logging can be changed by modifying the following files:

```
~/openstack/ardana/ansible/roles/monasca-persister/defaults/main.yml
~/openstack/ardana/ansible/roles/zookeeper/defaults/main.yml
~/openstack/ardana/ansible/roles/storm/defaults/main.yml
~/openstack/ardana/ansible/roles/monasca-notification/defaults/main.yml
~/openstack/ardana/ansible/roles/monasca-api/defaults/main.yml
~/openstack/ardana/ansible/roles/kafka/defaults/main.yml
~/openstack/ardana/ansible/roles/monasca-agent/defaults/main.yml (For this file, you
 will need to add the variable)
```

3. To change the logging level, use ALL CAPS to set the desired level in the following line:

```
monasca_log_level: WARN
```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "My config or other commit message"
```

6. To run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. To create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. To run the reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-reconfigure.yml
```

12.2.6.14   Networking (Neutron)

Service	Sub-component	Supported Logging Levels
neutron	neutron-server dhcp-agent l3-agent lbaas-agent metadata-agent openvswitch-agent vpn-agent	INFO (default) DEBUG

To change the Neutron logging level:

1. Log in to the Cloud Lifecycle Manager.

2. The Neutron service component logging can be changed by modifying the following files:

```
~/openstack/ardana/ansible/roles/neutron-common/templates/dhcp-agent-logging.conf.j2
```

```
~/openstack/ardana/ansible/roles/neutron-common/templates/l3-agent-logging.conf.j2
~/openstack/ardana/ansible/roles/neutron-common/templates/lbaas-agent-
logging.conf.j2
~/openstack/ardana/ansible/roles/neutron-common/templates/metadata-agent-
logging.conf.j2
~/openstack/ardana/ansible/roles/neutron-common/templates/openvswitch-agent-
logging.conf.j2
~/openstack/ardana/ansible/roles/neutron-common/templates/vpn-agent-logging.conf.j2
```

3. To change the logging level, use ALL CAPS to set the desired level in the following line in the [handler_logstash] section:

```
level: INFO
```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

```
ardana > cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

6. To run the configuration processor:

```
cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. To create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. To run the reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts neutron-reconfigure.yml
```

12.2.6.15   Object Storage (Swift)

Service	Sub-component	Supported Logging Levels
swift		INFO (default)

Service	Sub-component	Supported Logging Levels
		DEBUG

📖 Note

Currently it is not recommended to log at any level other than INFO.

12.2.6.16 Octavia

Service	Sub-component	Supported Logging Levels
octavia	Octavia-api Octavia-worker Octavia-hk Octavia-hm	INFO (default) DEBUG

To change the Octavia logging level:

1. Log in to the Cloud Lifecycle Manager.

2. The Octavia service component logging can be changed by modifying the following files:

   ```
   ~/openstack/my_cloud/config/octavia/octavia-api.conf.j2
   ~/openstack/my_cloud/config/octavia/octavia-worker.conf.j2
   ~/openstack/my_cloud/config/octavia/octavia-hk-logging.conf.j2
   ~/openstack/my_cloud/config/octavia/Octavia-hm-logging.conf.j2
   ```

3. To change the logging level, use ALL CAPS to set the desired level in the following line in the [handler_logstash] section:

   ```
   level: INFO
   ```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > git add -A
   ardana > git commit -m "My config or other commit message"
   ```

6. To run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. To create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. To run the reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts octavia-reconfigure.yml
```

12.2.6.17 Operations Console

Service	Sub-component	Supported Logging Levels
opsconsole	ops-web	INFO (default)
	ops-mon	DEBUG

To change the Operations Console logging level:

1. Log in to the Cloud Lifecycle Manager.

2. Open the following file:

```
~/openstack/ardana/ansible/roles/OPS-WEV/defaults/main.yml
```

3. To change the logging level, use ALL CAPS to set the desired level in the following line:

```
ops_console_loglevel: "{{ openstack_loglevel | default('INFO') }}"
```

4. To commit the changes to your local git repository:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "My config or other commit message"
```

5. To run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. To create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. To run the reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ops-console-reconfigure.yml
```

12.2.6.18   Orchestration (Heat)

Service	Sub-component	Supported Logging Levels
heat	api-cfn api-cloudwatch api-logging engine	INFO (default) DEBUG

To change the Heat logging level:

1. Log in to the Cloud Lifecycle Manager (deployer).

2. Open the following file:

```
~/openstack/my_cloud/config/heat/*-logging.conf.j2
```

3. To change the logging level, use ALL CAPS to set the desired level in the following line in the [handler_logstash] section:

```
level: INFO
```

4. Save the changes to the file.

5. To commit the changes to your local git repository:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
```

```
ardana > git commit -m "My config or other commit message"
```

6. To run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. To create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. To run the reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts heat-reconfigure.yml
```

12.2.6.19   Selecting Files for Centralized Logging

As you use HPE Helion OpenStack, you might find a need to redefine which log files are rotated on disk or transferred to centralized logging. These changes are all made in the centralized logging definition files.

HPE Helion OpenStack uses the logrotate service to provide rotation, compression, and removal of log files. All of the tunable variables for the logrotate process itself can be controlled in the following file: `~/openstack/ardana/ansible/roles/logging-common/defaults/main.yml`

You can find the centralized logging definition files for each service in the following directory: `~/openstack/ardana/ansible/roles/logging-common/vars`

You can change log settings for a service by following these steps.

1. Log in to the Cloud Lifecycle Manager.
   Open the *.yml file for the service or sub-component that you want to modify.
   Using Freezer, the Backup, Restore, and Archive service as an example:

```
ardana > vi ~/openstack/ardana/ansible/roles/logging-common/vars/freezer-agent-
clr.yml
```

Consider the opening clause of the file:

```
sub_service:
  hosts: FRE-AGN
```

```
    name: freezer-agent
    service: freezer
```

The **hosts** setting defines the role which will trigger this logrotate definition being applied to a particular host. It can use regular expressions for pattern matching, that is, **NEU-.***. The **service** setting identifies the high-level service name associated with this content, which will be used for determining log files' collective quotas for storage on disk.

2. Verify logging is enabled by locating the following lines:

```
centralized_logging:
  enabled: true
  format: rawjson
```

☞ Note

When possible, centralized logging is most effective on log files generated using logstash-formatted JSON. These files should specify *format: rawjson*. When only plaintext log files are available, *format: json* is appropriate. (This will cause their plaintext log lines to be wrapped in a json envelope before being sent to centralized logging storage.)

3. Observe log files selected for rotation:

```
- files:
  - /var/log/freezer/freezer-agent.log
  - /var/log/freezer/freezer-scheduler.log
  log_rotate:
  - daily
  - compress
  - missingok
  - notifempty
  - copytruncate
  - maxsize 80M
  - rotate 14
```

☞ Note

With the introduction of dynamic log rotation, the frequency (that is, *daily*) and file size threshold (that is, *maxsize*) settings no longer have any effect. The *rotate* setting may be easily overridden on a service-by-service basis.

4. Commit any changes to your local git repository:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "My config or other commit message"
```

5. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Create a deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the logging reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts kronos-reconfigure.yml
```

12.2.6.20   Controlling Disk Space Allocation and Retention of Log Files

Each service is assigned a weighted allocation of the `/var/log` filesystem's capacity. When all its log files' cumulative sizes exceed this allocation, a rotation is triggered for that service's log files according to the behavior specified in the `/etc/logrotate.d/*` specification.

These specification files are auto-generated based on YML sources delivered with the Cloud Lifecycle Manager codebase. The source files can be edited and reapplied to control the allocation of disk space across services or the behavior during a rotation.

Disk capacity is allocated as a percentage of the total weighted value of all services running on a particular node. For example, if 20 services run on the same node, all with a default weight of **100**, they will each be granted 1/20th of the log filesystem's capacity. If the configuration is updated to change one service's weight to **150**, all the services' allocations will be adjusted to make it possible for that one service to consume 150% of the space available to other individual services.

These policies are enforced by the script `/opt/kronos/rotate_if_exceeded_quota.py`, which will be executed every 5 minutes via a cron job and will rotate the log files of any services which have exceeded their respective quotas. When log rotation takes place for a service, logs are generated to describe the activity in `/var/log/kronos/check_if_exceeded_quota.log`.

When logrotate is performed on a service, its existing log files are compressed and archived to make space available for fresh log entries. Once the number of archived log files exceeds that service's retention thresholds, the oldest files are deleted. Thus, longer retention thresholds (that is, 10 to 15) will result in more space in the service's allocated log capacity being used for historic logs, while shorter retention thresholds (that is, 1 to 5) will keep more space available for its active plaintext log files.

Use the following process to make adjustments to services' log capacity allocations or retention thresholds:

1. Navigate to the following directory on your Cloud Lifecycle Manager:

   ```
   ~/stack/scratch/ansible/next/ardana/ansible
   ```

2. Open and edit the service weights file:

   ```
   ardana > vi roles/kronos-logrotation/vars/rotation_config.yml
   ```

3. Edit the service parameters to set the desired parameters. Example:

   ```
   cinder:
     weight: 300
     retention: 2
   ```

   > 🎓 **Note**
   >
   > The retention setting of *default* will use recommend defaults for each services' log files.

4. Run the kronos-logrotation-deploy playbook:

   ```
   ardana > ansible-playbook -i hosts/verb_hosts kronos-logrotation-deploy.yml
   ```

5. Verify the changes to the quotas have been changed:
   Login to a node and check the contents of the file /opt/kronos/service_info.yml to see the active quotas for that node, and the specifications in /etc/logrotate.d/* for rotation thresholds.

12.2.6.21   Configuring Elasticsearch for Centralized Logging

Elasticsearch includes some tunable options exposed in its configuration. HPE Helion OpenStack uses these options in Elasticsearch to prioritize indexing speed over search speed. HPE Helion OpenStack also configures Elasticsearch for optimal performance in low RAM environments. The options that HPE Helion OpenStack modifies are listed below along with an explanation about why they were modified.

These configurations are defined in the `~/openstack/my_cloud/config/logging/main.yml` file and are implemented in the Elasticsearch configuration file `~/openstack/my_cloud/config/logging/elasticsearch.yml.j2`.

12.2.6.22   Safeguards for the Log Partitions Disk Capacity

Because the logging partitions are at a high risk of filling up over time, a condition which can cause many negative side effects on services running, it is important to safeguard against log files consuming 100 % of available capacity.

This protection is implemented by pairs of low/high **watermark** thresholds, with values established in `~/stack/scratch/ansible/next/ardana/ansible/roles/logging-common/defaults/main.yml` and applied by the `kronos-logrotation-deploy` playbook.

- **var_log_low_watermark_percent** (default: 80) sets a capacity level for the contents of the `/var/log` partition beyond which alarms will be triggered (visible to administrators in Monasca).

- **var_log_high_watermark_percent** (default: 95) defines how much capacity of the `/var/log` partition to make available for log rotation (in calculating weighted service allocations).

- **var_audit_low_watermark_percent** (default: 80) sets a capacity level for the contents of the `/var/audit` partition beyond which alarm notifications will be triggered.

- **var_audit_high_watermark_percent** (default: 95) sets a capacity level for the contents of the `/var/audit` partition which will cause log rotation to be forced according to the specification in `/etc/auditlogrotate.conf`.

12.2.7 Audit Logging Overview

Existing OpenStack service logging varies widely across services. Generally, log messages do not have enough detail about who is requesting the application program interface (API), or enough context-specific details about an action performed. Often details are not even consistently logged across various services, leading to inconsistent data formats being used across services. These issues make it difficult to integrate logging with existing audit tools and processes.

To help you monitor your workload and data in compliance with your corporate, industry or regional policies, HPE Helion OpenStack provides auditing support as a basic security feature. The audit logging can be integrated with customer Security Information and Event Management (SIEM) tools and support your efforts to correlate threat forensics.

The HPE Helion OpenStack audit logging feature uses Audit Middleware for Python services. This middleware service is based on OpenStack services which use the Paste Deploy system. Most OpenStack services use the paste deploy mechanism to find and configure WSGI servers and applications. Utilizing the paste deploy system provides auditing support in services with minimal changes.

By default, audit logging as a post-installation feature is disabled in the cloudConfig file on the Cloud Lifecycle Manager and it can only be enabled after HPE Helion OpenStack installation or upgrade.

The tasks in this section explain how to enable services for audit logging in your environment. HPE Helion OpenStack provides audit logging for the following services:

- Nova

- Barbican

- Keystone

- Cinder

- Ceilometer

- Neutron

- Glance

- Heat

For audit log backup information see *Section 14.13, "Backing up and Restoring Audit Logs"*

12.2.7.1 Audit Logging Checklist

Before enabling audit logging, make sure you understand how much disk space you will need, and configure the disks that will store the logging data. Use the following table to complete these tasks:

#	Item
	Section 12.2.7.1.1, "Frequently Asked Questions"
	Section 12.2.7.1.2, "Estimate Disk Size"
	Section 12.2.7.1.3, "Add disks to the controller nodes"
	Section 12.2.7.1.4, "Update the disk template for the controller nodes"
	Section 12.2.7.1.5, "Save your changes"

12.2.7.1.1 Frequently Asked Questions

How are audit logs generated?

The audit logs are created by services running in the cloud management controller nodes. The events that create auditing entries are formatted using a structure that is compliant with Cloud Auditing Data Federation (CADF) policies. The formatted audit entries are then saved to disk files. For more information, see the Cloud Auditing Data Federation Website. (http://www.dmtf.org/standards/cadf) ↗

Where are audit logs stored?

We strongly recommend adding a dedicated disk volume for `/var/audit`.

If the disk templates for the controllers are not updated to create a separate volume for `/var/audit`, the audit logs will still be created in the root partition under the folder `/var/audit`. This could be problematic if the root partition does not have adequate space to hold the audit logs.

⬡ Warning

We recommend that you do **not** store audit logs in the `/var/log` volume. The `/var/log` volume is used for storing operational logs and logrotation/alarms have been preconfigured for various services based on the size of this volume. Adding audit logs here may impact these causing undesired alarms. This would also impact the retention times for the operational logs.

Are audit logs centrally stored?

Yes. The existing operational log profiles have been configured to centrally log audit logs as well, once their generation has been enabled. The audit logs will be stored in separate Elasticsearch indices separate from the operational logs.

How long are audit log files retained?

By default, audit logs are configured to be retained for 7 days on disk. The audit logs are rotated each day and the rotated files are stored in a compressed format and retained up to 7 days (configurable). The backup service has been configured to back up the audit logs to a location outside of the controller nodes for much longer retention periods.

Do I lose audit data if a management controller node goes down?

Yes. For this reason, it is strongly recommended that you back up the audit partition in each of the management controller nodes for protection against any data loss.

12.2.7.1.2   Estimate Disk Size

The table below provides estimates from each service of audit log size generated per day. The estimates are provided for environments with 100 nodes, 300 nodes, and 500 nodes.

Service	Log File Size: 100 nodes	Log File Size: 300 nodes	Log File Size: 500 nodes
Barbican	2.6 MB	4.2 MB	5.6 MB
Keystone	96 - 131 MB	288 - 394 MB	480 - 657 MB
Nova	186 (with a margin of 46) MB	557 (with a margin of 139) MB	928 (with a margin of 232) MB
Ceilometer	12 MB	12 MB	12 MB

Service	Log File Size: 100 nodes	Log File Size: 300 nodes	Log File Size: 500 nodes
Cinder	2 - 250 MB	2 - 250 MB	2 - 250 MB
Neutron	145 MB	433 MB	722 MB
Glance	20 (with a margin of 8) MB	60 (with a margin of 22) MB	100 (with a margin of 36) MB
Heat	432 MB (1 transaction per second)	432 MB (1 transaction per second)	432 MB (1 transaction per second)
Swift	33 GB (700 transactions per second)	102 GB (2100 transactions per second)	172 GB (3500 transactions per second)

12.2.7.1.3 Add disks to the controller nodes

You need to add disks for the audit log partition to store the data in a secure manner. The steps to complete this task will vary depending on the type of server you are running. Please refer to the manufacturer's instructions on how to add disks for the type of server node used by the management controller cluster. If you already have extra disks in the controller node, you can identify any unused one and use it for the audit log partition.

12.2.7.1.4 Update the disk template for the controller nodes

Since audit logging is disabled by default, the audit volume groups in the disk templates are commented out. If you want to turn on audit logging, the template needs to be updated first. If it is not updated, there will be no back-up volume group. To update the disk template, you will need to copy templates from the examples folder to the definition folder and then edit the disk controller settings. Changes to the disk template used for provisioning cloud nodes must be made prior to deploying the nodes.

To update the disk controller template:

1. Log in to your Cloud Lifecycle Manager.

2. To copy the example templates folder, run the following command:

   > **Important**
   >
   > If you already have the required templates in the definition folder, you can skip this step.

   ```
   ardana > cp -r ~/openstack/examples/entry-scale-esx/* ~/openstack/my_cloud/
   definition/
   ```

3. To change to the data folder, run:

   ```
   ardana > cd ~/openstack/my_cloud/definition/
   ```

4. To edit the disks controller settings, open the file that matches your server model and disk model in a text editor:

Model	File
entry-scale-kvm	`disks_controller_1TB.yml`
	`disks_controller_600GB.yml`
mid-scale	`disks_compute.yml`
	`disks_control_common_600GB.yml`
	`disks_dbmq_600GB.yml`
	`disks_mtrmon_2TB.yml`
	`disks_mtrmon_4.5TB.yml`
	`disks_mtrmon_600GB.yml`
	`disks_swobj.yml`

Model	File
	`disks_swpac.yml`

5. To update the settings and enable an audit log volume group, edit the appropriate file(s) listed above and remove the '#' comments from these lines, confirming that they are appropriate for your environment.

```
- name: audit-vg
  physical-volumes:
    - /dev/sdz
  logical-volumes:
    - name: audit
      size: 95%
      mount: /var/audit
      fstype: ext4
      mkfs-opts: -O large_file
```

12.2.7.1.5 Save your changes

To save your changes you will use the GIT repository to add the setup disk files.

To save your changes:

1. To change to the openstack directory, run:

```
ardana > cd ~/openstack
```

2. To add the new and updated files, run:

```
ardana > git add -A
```

3. To verify the files are added, run:

```
ardana > git status
```

4. To commit your changes, run:

```
ardana > git commit -m "Setup disks for audit logging"
```

12.2.7.2 Enable Audit Logging

To enable audit logging you must edit your cloud configuration settings, save your changes and re-run the configuration processor. Then you can run the playbooks to create the volume groups and configure them.

In the `~/openstack/my_cloud/definition/cloudConfig.yml` file, service names defined under enabled-services or disabled-services override the default setting.

The following is an example of your audit-settings section:

```
# Disc space needs to be allocated to the audit directory before enabling
# auditing.
# Default can be either "disabled" or "enabled". Services listed in
# "enabled-services" and "disabled-services" override the default setting.
audit-settings:
   default: disabled
   #enabled-services:
   #  - keystone
   #  - barbican
   disabled-services:
     - nova
     - barbican
     - keystone
     - cinder
     - ceilometer
     - neutron
```

In this example, although the default setting for all services is set to **disabled**, keystone and barbican may be explicitly enabled by removing the comments from these lines and this setting overrides the default.

12.2.7.2.1 To edit the configuration file:

1. Log in to your Cloud Lifecycle Manager.

2. To change to the cloud definition folder, run:

   ```
   ardana > cd ~/openstack/my_cloud/definition
   ```

3. To edit the auditing settings, in a text editor, open the following file:

   ```
   cloudConfig.yml
   ```

4. To enable audit logging, begin by uncommenting the "enabled-services:" block.

   - enabled-service:

   - any service you want to enable for audit logging.

   For example, Keystone has been enabled in the following text:

Default cloudConfig.yml file	Enabling Keystone audit logging
   | ```
audit-
settings:
default:
 disabled
enabled-
services:
#  - keystone
``` | ```
audit-settings:
default: disabled
enabled-services:
  - keystone
``` |

5. To move the services you want to enable, comment out the service in the disabled section and add it to the enabled section. For example, Barbican has been enabled in the following text:

cloudConfig.yml file	Enabling Barbican audit logging
   | ```
audit-settings:
default: disabled
enabled-services:
  - keystone
disabled-services:
  - nova
 # - keystone
  - barbican
  - cinder
``` | ```
audit-settings:
default: disabled
enabled-services:
 - keystone
 - barbican
disabled-services:
 - nova
 # - barbican
 # - keystone
 - cinder
``` |

12.2.7.2.2 To save your changes and run the configuration processor:

1. To change to the openstack directory, run:

   ```
   ardana > cd ~/openstack
   ```

2. To add the new and updated files, run:

   ```
   ardana > git add -A
   ```

3. To verify the files are added, run:

   ```
   ardana > git status
   ```

4. To commit your changes, run:

   ```
   ardana > git commit -m "Enable audit logging"
   ```

5. To change to the directory with the ansible playbooks, run:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ```

6. To rerun the configuration processor, run:

   ```
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

12.2.7.2.3 To create the volume group:

1. To change to the directory containing the osconfig playbook, run:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ```

2. To remove the stub file that osconfig uses to decide if the disks are already configured, run:

   ```
   ardana > ansible -i hosts/verb_hosts KEY-API -a 'sudo rm -f /etc/hos/osconfig-ran'
   ```

Important

The osconfig playbook uses the stub file to mark already configured disks as "idempotent." To stop osconfig from identifying your new disk as already configured, you must remove the stub file /etc/hos/osconfig-ran before re-running the osconfig playbook.

3. To run the playbook that enables auditing for a service, run:

```
ardana > ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit KEY-API
```

Important

The variable KEY-API is used as an example to cover the management controller cluster. To enable auditing for a service that is not run on the same cluster, add the service to the –limit flag in the above command. For example:

```
ardana > ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit KEY-
API:NEU-SVR
```

12.2.7.2.4   To Reconfigure services for audit logging:

1. To change to the directory containing the service playbooks, run:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

2. To run the playbook that reconfigures a service for audit logging, run:

```
ardana > ansible-playbook -i hosts/verb_hosts SERVICE_NAME-reconfigure.yml
```

For example, to reconfigure Keystone for audit logging, run:

```
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

3. Repeat steps 1 and 2 for each service you need to reconfigure.

Important

You must reconfigure each service that you changed to be enabled or disabled in the cloudConfig.yml file.

12.2.8 Troubleshooting

For information on troubleshooting Central Logging, see *Section 15.7.1, "Troubleshooting Centralized Logging"*.

12.3 Metering Service (Ceilometer) Overview

The HPE Helion OpenStack metering service collects and provides access to OpenStack usage data that can be used for billing reporting such as showback, and chargeback. The metering service can also provide general usage reporting. Ceilometer acts as the central collection and data access service to the meters provided by all the OpenStack services. The data collected is available both through the Monasca API and the Ceilometer V2 API.

Important

Ceilometer V2 API has been deprecated in Pike release upstream. Although the Ceilometer V2 API is still available with HPE Helion OpenStack, to prepare for eventual removal of Ceilometer V2 API in next release we recommend that users switch to the Monasca API to access data.

12.3.1 Metering Service New Functionality

12.3.1.1 New Metering Functionality in HPE Helion OpenStack 8

- Ceilometer is now integrated with Monasca to use it as the datastore. Ceilometer API also now queries the Monasca datastore using the Monasca API (query) instead of the MySQL database

- The default meters and other items configured for the Ceilometer API can now be modified and additional meters can be added. It is highly recommended that customers test overall HPE Helion OpenStack performance prior to deploying any Ceilometer modifications to ensure the addition of new notifications or polling events does not negatively affect overall system performance.

- Ceilometer Central Agent (pollster) is now called Polling Agent and is configured to support HA (Active-Active)

- Notification Agent has built-in HA (Active-Active) with support for pipeline transformers, but workload partitioning has been disabled in HPE Helion OpenStack

- SWIFT Poll-based account level meters will be enabled by default with an hourly collection cycle.

- Integration with centralized monitoring (Monasca) and centralized logging

- Support for upgrade and reconfigure operations

12.3.1.2 Limitations

- The Ceilometer Post Meter API is disabled by default.

- The Ceilometer Events and Traits API is not supported and disabled by default.

- The Number of metadata attributes that can be extracted from resource_metadata has a maximum of 16. This is the number of fields in the metadata section of the **monasca_field_definitions.yaml** file for any service. It is also the number that is equal to fields in metadata.common and fields in metadata.<service.meters> sections. The total number of these fields cannot be more than 16.

- Several network-related attributes are accessible using a colon ":" but are returned as a period ".". For example, you can access a sample list using the following command:

```
ardana > source ~/service.osrc
ardana > ceilometer --debug sample-list network -q "resource_id=421d50a5-156e-4cb9-
b404-
d2ce5f32f18b;resource_metadata.provider.network_type=flat"
```

However, in response you will see the following:

```
provider.network_type
```

instead of

```
provider:network_type
```

This limitation is known for the following attributes:

```
provider:network_type
provider:physical_network
provider:segmentation_id
```

- Ceilometer Expirer is unsupported. Data retention expiration is now handled by Monasca with a default retention period of 45 days.

- Ceilometer Collector is unsupported.

- The Ceilometer Alarms API is disabled by default. HPE Helion OpenStack 8 provides an alternative operations monitoring service that will provide support for operations monitoring, alerts, and notifications use cases.

12.3.2 Understanding the Metering Service Concepts

12.3.2.1 Ceilometer Introduction

Before configuring the Ceilometer Metering Service, make sure you understand how it works.

12.3.2.1.1 Metering Architecture

HPE Helion OpenStack automatically configures Ceilometer to use Logging and Monitoring Service (Monasca) as its backend. Ceilometer is deployed on the same control plane nodes as Monasca.

The installation of Celiometer creates several management nodes running different metering components.

Ceilometer Components on Controller nodes

This controller node is the first of the High Available (HA) cluster. In this node there is an instance of the Ceilometer API running under the HA Proxy Virtual IP address.

Ceilometer Sample Polling

Sample Polling is part of the Polling Agent. Now that Ceilometer API uses Monasca API (query) instead of the MySQL database, messages are posted by Notification Agent directly to Monasca API.

Ceilometer Polling Agent

The Polling Agent is responsible for coordinating the polling activity. It parses the **pipeline.yml** configuration file and identifies all the sources that need to be polled. The sources are then evaluated using a discovery mechanism and all the sources are translated to resources where a dedicated pollster can retrieve and publish data. At each identified interval the discovery mechanism is triggered, the resource list is composed, and the data is polled and sent to the queue.

Ceilometer Collector No Longer Required

In previous versions, the collector was responsible for getting the samples/events from the RabbitMQ service and storing it in the main database. The Ceilometer Collector is no longer enabled. Now that Notification Agent posts the data directly to Monasca API, the collector is no longer required

12.3.2.1.2 Meter Reference

The Ceilometer API collects basic information grouped into categories known as meters. A meter is the unique resource-usage measurement of a particular OpenStack service. Each OpenStack service defines what type of data is exposed for metering.

Each meter has the following characteristics:

Attribute	Description
Name	Description of the meter
Unit of Measurement	The method by which the data is measured. For example: storage meters are defined in Gigabytes (GB) and network bandwidth is measured in Gigabits (Gb).
Type	The origin of the meter's data. OpenStack defines the following origins: • Cumulative - Increasing over time (instance hours) • Gauge - a discrete value. For example: the number of floating IP addresses or image uploads. • Delta - Changing over time (bandwidth)

A meter is defined for every measurable resource. A meter can exist beyond the actual existence of a particular resource, such as an active instance, to provision long-cycle use cases such as billing.

🔔 Important

For a list of meter types and default meters installed with HPE Helion OpenStack, see *Section 12.3.3, "Ceilometer Metering Available Meter Types"*

The most common meter submission method is notifications. With this method, each service sends the data from their respective meters on a periodic basis to a common notifications bus.

Ceilometer, in turn, pulls all of the events from the bus and saves the notifications in a Ceilometer-specific database. The period of time that the data is collected and saved is known as the Ceilometer expiry and is configured during Ceilometer installation. Each meter is collected from one or more samples, gathered from the messaging queue or polled by agents. The samples are represented by counter objects. Each counter has the following fields:

Attribute	Description
counter_name	Description of the counter

Attribute	Description
counter_unit	The method by which the data is measured. For example: data can be defined in Gigabytes (GB) or for network bandwidth, measured in Gigabits (Gb).
counter_typee	The origin of the counter's data. OpenStack defines the following origins: • Cumulative - Increasing over time (instance hours) • Gauge - a discrete value. For example: the number of floating IP addresses or image uploads. • Delta - Changing over time (bandwidth)
counter_volume	The volume of data measured (CPU ticks, bytes transmitted, etc.). Not used for gauge counters. Set to a default value such as 1.
resource_id	The identifier of the resource measured (UUID)
project_id	The project (tenant) ID to which the resource belongs.
user_id	The ID of the user who owns the resource.
resource_metadata	Other data transmitted in the metering notification payload.

12.3.2.1.3 Role Based Access Control (RBAC)

A user with the **admin role** can access all API functions across all projects by default. Ceilometer also supports the ability to assign access to a specific API function by **project** and **UserID**. User access is configured in the Ceilometer policy file and enables you to grant specific API functions to specific users for a specific project.

For instructions on how to configure role-based access, see *Section 12.3.7, "Ceilometer Metering Setting Role-based Access Control"*.

12.3.3   Ceilometer Metering Available Meter Types

The Metering service contains three types of meters:

Cumulative

    A cumulative meter measures data over time (for example, instance hours).

Gauge

    A gauge measures discrete items (for example, floating IPs or image uploads) or fluctuating values (such as disk input or output).

Delta

    A delta measures change over time, for example, monitoring bandwidth.

Each meter is populated from one or more *samples*, which are gathered from the messaging queue (listening agent), polling agents, or push agents. Samples are populated by *counter* objects. Each counter contains the following *fields*:

name

    the name of the meter

type

    the type of meter (cumulative, gauge, or delta)

amount

    the amount of data measured

unit

    the unit of measure

resource

    the resource being measured

project ID

    the project the resource is assigned to

user

    the user the resource is assigned to.

Note: The metering service shares the same High-availability proxy, messaging, and database clusters with the other Information services. To avoid unnecessarily high loads, *Section 12.3.9, "Optimizing the Ceilometer Metering Service"*.

12.3.3.1 HPE Helion OpenStack Default Meters

These meters are installed and enabled by default during an HPE Helion OpenStack installation.

Detailed information on the Ceilometer API can be found on the following page:

Ceilometer Web API (http://docs.openstack.org/developer/ceilometer/webapi/v2.html) ↗ .

12.3.3.2 Compute (Nova) Meters

Meter	Type	Unit	Resource	Origin	Note
vcpus	Gauge	vcpu	Instance ID	Notification	Number of virtual CPUs allocated to the instance
memory	Gauge	MB	Instance ID	Notification	Volume of RAM allocated to the instance
memory.resident	Gauge	MB	Instance ID	Pollster	Volume of RAM used by the instance on the physical machine
memory.usage	Gauge	MB	Instance ID	Pollster	Volume of RAM used by the instance from the amount of its allocated memory
cpu	Cumulative	ns	Instance ID	Pollster	CPU time used

Meter	Type	Unit	Resource	Origin	Note
cpu_util	Gauge	%	Instance ID	Pollster	Average CPU utilization
disk.read.requests	Cumulative	request	Instance ID	Pollster	Number of read requests
disk.read.requests.rate	Gauge	request/s	Instance ID	Pollster	Average rate of read requests
disk.write.requests	Cumulative	request	Instance ID	Pollster	Number of write requests
disk.write.requests.rate	Gauge	request/s	Instance ID	Pollster	Average rate of write requests
disk.read.bytes	Cumulative	B	Instance ID	Pollster	Volume of reads
disk.read.bytes.rate	Gauge	B/s	Instance ID	Pollster	Average rate of reads
disk.write.bytes	Cumulative	B	Instance ID	Pollster	Volume of writes
disk.write.bytes.rate	Gauge	B/s	Instance ID	Pollster	Average rate of writes
disk.root.size	Gauge	GB	Instance ID	Notification	Size of root disk
disk.ephemeral.size	Gauge	GB	Instance ID	Notification	Size of ephemeral disk
disk.device.read.requests	Cumulative	request	Disk ID	Pollster	Number of read requests

Meter	Type	Unit	Resource	Origin	Note
disk.device.read.requests.rate	Gauge	request/s	Disk ID	Pollster	Average rate of read requests
disk.device.write.requests	Cumulative	request	Disk ID	Pollster	Number of write requests
disk.device.write.requests.rate	Gauge	request/s	Disk ID	Pollster	Average rate of write requests
disk.device.read.bytes	Cumulative	B	Disk ID	Pollster	Volume of reads
disk.device.read.bytes .rate	Gauge	B/s	Disk ID	Pollster	Average rate of reads
disk.device.write.bytes	Cumulative	B	Disk ID	Pollster	Volume of writes
disk.device.write.bytes .rate	Gauge	B/s	Disk ID	Pollster	Average rate of writes
disk.capacity	Gauge	B	Instance ID	Pollster	The amount of disk that the instance can see
disk.allocation	Gauge	B	Instance ID	Pollster	The amount of disk occupied by the instance on the host machine
disk.usage	Gauge	B	Instance ID	Pollster	The physical size in bytes of the image

Meter	Type	Unit	Resource	Origin	Note
					container on the host
disk.device.capacity	Gauge	B	Disk ID	Pollster	The amount of disk per device that the instance can see
disk.device.allocation	Gauge	B	Disk ID	Pollster	The amount of disk per device occupied by the instance on the host machine
disk.device.usage	Gauge	B	Disk ID	Pollster	The physical size in bytes of the image container on the host per device
network.incoming.bytes	Cumulative	B	Interface ID	Pollster	Number of incoming bytes
network.outgoing.bytes	Cumulative	B	Interface ID	Pollster	Number of outgoing bytes
network.incoming.packets	Cumulative	packet	Interface ID	Pollster	Number of incoming packets

Meter	Type	Unit	Resource	Origin	Note
network.outgoing.packets	Cumulative	packet	Interface ID	Pollster	Number of outgoing packets

12.3.3.3 Compute Host Meters

Meter	Type	Unit	Resource	Origin	Note
compute.node.cpu.frequency	Gauge	MHz	Host ID	Notification	CPU frequency
compute.node.cpu.kernel.time	Cumulative	ns	Host ID	Notification	CPU kernel time
compute.node.cpu.idle.time	Cumulative	ns	Host ID	Notification	CPU idle time
compute.node.cpu.user.time	Cumulative	ns	Host ID	Notification	CPU user mode time
compute.node.cpu.iowait.time	Cumulative	ns	Host ID	Notification	CPU I/O wait time
compute.node.cpu.kernel.percent	Gauge	%	Host ID	Notification	CPU kernel percentage
compute.node.cpu.idle.percent	Gauge	%	Host ID	Notification	CPU idle percentage
compute.node.cpu.user.percent	Gauge	%	Host ID	Notification	CPU user mode percentage
compute.node.cpu.iowait.percent	Gauge	%	Host ID	Notification	CPU I/O wait percentage
compute.node.cpu.percent	Gauge	%	Host ID	Notification	CPU utilization

12.3.3.4 Image (Glance) Meters

Meter	Type	Unit	Resource	Origin	Note
image.size	Gauge	B	Image ID	Notification	Uploaded image size
image.update	Delta	Image	Image ID	Notification	Number of uploads of the image
image.upload	Delta	Image	image ID	notification	Number of uploads of the image
image.delete	Delta	Image	Image ID	Notification	Number of deletes on the image

12.3.3.5 Volume (Cinder) Meters

Meter	Type	Unit	Resource	Origin	Note
volume.size	Gauge	GB	Vol ID	Notification	Size of volume
snapshot.size	Gauge	GB	Snap ID	Notification	Size of snapshot's volume

12.3.3.6 Storage (Swift) Meters

Meter	Type	Unit	Resource	Origin	Note
storage.objects	Gauge	Object	Storage ID	Pollster	Number of objects
storage.objects.size	Gauge	B	Storage ID	Pollster	Total size of stored objects

Meter	Type	Unit	Resource	Origin	Note
storage.objects.containers	Gauge	Container	Storage ID	Pollster	Number of containers

The `resource_id` for any Ceilometer query is the `tenant_id` for the Swift object because Swift usage is rolled up at the tenant level.

12.3.4   Metering API Reference

Ceilometer uses a polling agent to communicate with an API to collect information at a regular interval, as shown in the diagram below.

Ceilometer query APIs can put a significant load on the database leading to unexpected results or failures. Therefore it is important to understand how the Ceilometer API works and how to change the configuration to protect against failures.

12.3.4.1   Ceilometer API Changes

The following changes have been made in the latest release of Ceilometer for HPE Helion OpenStack:

- **Ceilometer API** supports a default of 100 queries. This limit is configurable in the ceilometer.conf configuration file. The option is in the `DEFAULT` section and is named `default_api_return_limit`.

- Flexible configuration for pollster and notifications has been added. Ceilometer can now list different event types differently for these services.

- **Query-sample API** is now supported in HPE Helion OpenStack.

- **Meter-list API** can now return a unique list of meter names with no duplicates. To create this list, when running the list command, use the `--unique` option.

The following limitations exist in the latest release of Ceilometer for HPE Helion OpenStack:

- **Event API** is disabled by default and is unsupported in HPE Helion OpenStack.

- **Trait API** is disabled by default and is unsupported in HPE Helion OpenStack.

- **Post Sample API** is disabled by default and is unsupported in HPE Helion OpenStack.

- **Alarm API** is disabled by default and is unsupported in HPE Helion OpenStack.

- **Sample-Show API** is unsupported in HPE Helion OpenStack.

- **Meter-List API** does not support filtering with metadata.

- **Query-Sample API** (Complex query) does not support using the following operators in the same query:

```
order by argument
NOT
```

- **Query-Sample API** requires you to specify a meter name. Complex queries will be analyzed as several simple queries according to the AND/OR logic. As meter-list is a constraint, each simple query must specify a meter name. If this condition is not met, you will receive a detailed 400 error.

- Due to a Monasca API limitation, microsecond is no longer supported. In the **Resource-List API**, **Sample-List API**, **Statistics API** and **Query-Samples API**, the `timestamp` field now only supports measuring down to the millisecond.

- **Sample-List API** does not support `message_id` as a valid search parameter. This parameter is also not included in the output.

- **Sample-List API** now requires the meter name as a positional parameter.

- **Sample-List API** returns a sample with an empty `message_signature` field.

12.3.4.2   Disabled APIs

The following Ceilometer metering APIs are disabled in this release:

- Event API

- Trait API

- Ceilometer Alarms API

- Post Samples API

These APIs are disabled through a custom rule called `hp_disabled_rule:not_implemented`. This rule is added to each disabled API in Ceilometer's policy.json file **/etc/ceilometer/ policy.json** on controller nodes. Attempts to access any of the disabled APIs will result in an HTTP response 501 Not Implemented.

To manually enable any of the APIs, remove the corresponding rule and restart Apache

```
{
 "context_is_admin": "role:admin",
 "context_is_project": "project_id:%(target.project_id)s",
 "context_is_owner": "user_id:%(target.user_id)s",
 "segregation": "rule:context_is_admin",

 "telemetry:create_samples": "hp_disabled_rule:not_implemented",

 "telemetry:get_alarm": "hp_disabled_rule:not_implemented",
 "telemetry:change_alarm": "hp_disabled_rule:not_implemented",
 "telemetry:delete_alarm": "hp_disabled_rule:not_implemented",
 "telemetry:alarm_history": "hp_disabled_rule:not_implemented",
 "telemetry:change_alarm_state": "hp_disabled_rule:not_implemented",
 "telemetry:get_alarm_state": "hp_disabled_rule:not_implemented",
 "telemetry:create_alarm": "hp_disabled_rule:not_implemented",
 "telemetry:get_alarms": "hp_disabled_rule:not_implemented",
 "telemetry:query_sample":"hp_disabled_rule:not_implemented",
 "default": ""
}
```

The following Alarm APIs are disabled

- POST /v2/alarms

- GET /v2/alarms

- GET /v2/alarms/(alarm_id)

- PUT /v2/alarms/(alarm_id)

- DELETE /v2/alarms/(alarm_id)

- GET /v2/alarms/(alarm_id)/history

- PUT /v2/alarms/(alarm_id)/state

- GET /v2/alarms/(alarm_id)/state

- POST /v2/query/alarms

- POST /v2/query/alarms/history

In addition, these APIs are disabled:

- Post Samples API: POST /v2/meters/(meter_name)

- Query Sample API: POST /v2/query/samples

12.3.4.3 Improving Reporting API Responsiveness

Reporting APIs are the main access to the Metering data stored in Ceilometer. These APIs are accessed by Horizon to provide basic usage data and information. However, Horizon Resources Usage Overview / Stats panel shows usage metrics with the following limitations:

- No metric option is available until you actually create a resource (such as an instance, Swift container, etc).

- Only specific meters are displayed for a selection after resources have been created. For example, only the Cinder volume and volume.size meters are displayed if only a Cinder volume has been created (for example, if no compute instance or Swift containers were created yet)

- Only the top 20 meters associated with the sample query results are displayed.

- Period duration selection should be much less than the default retention period (currently 7 days), to get statistics for multiple groups.

HPE Helion OpenStack uses the Apache2 Web Server to provide API access. It is possible to tune performance to optimize the front end as well as the back-end database. Experience indicates that an excessive increase of concurrent access to the front-end tends to put a strain in the database.

12.3.4.4 Reconfiguring Apache2, Horizon and Keystone

The ceilometer-api is now running as part of the Apache2 service together with Horizon and Keystone. To remove them from the active list so that changes can be made and then re-instate them, use the following commands.

1. Disable the Ceilometer API on the active sites.

   ```
   tux > sudo rm /etc/apache2/vhosts.d/ceilometer_modwsgi.conf
   ```

   ```
   tux > sudo systemctl reload apache2.service
   ```

2. Perform all necessary changes. The Ceilometer API will not be served until it is re-enabled.

3. Re-enable the Ceilometer API on the active sites.

   ```
   tux > sudo ln -s  /etc/apache2/vhosts.d/ceilometer_modwsgi.vhost /etc/apache2/
   vhosts.d/ceilometer_modwsgi.conf
   ```

   ```
   tux > sudo systemctl reload apache2.service
   ```

4. The new changes need to be picked up by Apache2. If possible, force a reload rather than a restart. Unlike a restart, the reload waits for currently active sessions to gracefully terminate or complete.

   ```
   tux > sudo systemctl reload apache2.service
   ```

12.3.4.5 Data Access API

Ceilometer provides a complete API for data access only and not for data visualization or aggregation. These functions are provided by external, downstream applications that support various use cases like usage billing and software license policy adherence.

Each application calls the specific Ceilometer API needed for their use case. The resulting data is then aggregated and visualized based on the unique functions provided by each application.

For more information, see the OpenStack Developer documentation for V2 Web API (http://docs.openstack.org/developer/ceilometer/webapi/v2.html) ↗.

12.3.4.6   Post Samples API

The Post Sample API is disabled by default in HPE Helion OpenStack 8 and it requires a separate pipeline.yml for Ceilometer. This is because it uses a pipeline configuration different than the agents. Also by default, the API pipeline has no meters enabled. When the Post Samples API is enabled, you need to configure the meters.

> **Important**
>
> Use caution when adding meters to the API pipeline. Ensure that only meters already present in the notification agent and the polling agent pipeline are added to the Post Sample API pipeline.

The Ceilometer API pipeline configuration file is located in the following directory:

```
/opt/stack/service/ceilometer-api/etc/pipeline-api.yml
```

Sample API pipeline file:

```
---
sources:
    - name: meter_source
      interval: 30
      meters:
          - "instance"
          - "ip.floating"
          - "network"
          - "network.create"
          - "network.update"
      sinks:
          - meter_sink
    - name: image_source
      interval: 30
      meters:
          - "image"
          - "image.size"
          - "image.upload"
          - "image.delete"
      sinks:
          - meter_sink
    - name: volume_source
      interval: 30
      meters:
          - "volume"
```

```
            - "volume.size"
            - "snapshot"
            - "snapshot.size"
        sinks:
            - meter_sink
    - name: swift_source
      interval: 3600
      meters:
            - "storage.objects"
            - "storage.objects.size"
            - "storage.objects.containers"
        sinks:
            - meter_sink
 sinks:
    - name: meter_sink
      transformers:
      publishers:
          - notifier://
```

12.3.4.7 Resource API

The Ceilometer Resource API provides a list of resources associated with meters that Ceilometer polls. By default, all meter links are generated for each resource.

🔔 Important

Be aware that this functionality has a high cost. For a large deployment, in order to reduce the response time, it is recommended that you do not return meter links. You can disable links in the output using the following filter in your query: (for the REST API only)

```
meter_links=0
```

The `resource-list` (/v2/resources) API can be filtered by the following parameters:

- project_id

- user_id

- source

- resource_id

- timestamp

- metadata

🔔 Important

It is highly recommended that you use one or both of the following query filters to get a quick response in a scaled deployment:

- project_id

- timestamp

Example Query:

```
ardana > ceilometer resource-list -q
 "project_id=7aa0fe3f02ff4e11a70a41e97d0db5e3;timestamp>=2015-10-22T15:44:00;timestamp<=2015-10-23T15:4
```

12.3.4.8   Sample API

Ceilometer Sample has two APIs:

- ceilometer sample-list(/v2/samples)

- ceilometer query-sample (/v2/query/samples)

Sample-list API allows querying based on the following values:

- meter name

- user_id

- project_id

- sample source

- resource_id

- sample timestamp (range)

- sample message_id

- resource metadata attributes

Sample-list API uses the AND operator implicitly. However, the query-sample API allows for finer control over the filter expression. This is because query-sample API allows the use of AND, OR, and NOT operators over any of the sample, meter or resource attributes.

Limitations:

- Ceilometer query-sample API does not support the JOIN operator for stability of the system. This is due to the fact that query-sample API uses an anonymous/alias table to cache the JOIN query results and concurrent requests to this API. This can use up the disk space quickly and cause service interruptions.

- Ceilometer sample-list API uses the AND operator implicitly for all queries. However, sample-list API does allow you to query on resource metadata field of samples.

Sample queries from the command line:

```
ardana > ceilometer sample-list -m METER_NAME -q
 '<field1><operator1><value1>;...;<field_n><operator_n><value_n>'
```

where operators can be: $<$, $<=$, $=$, $!=$, $>=$ $>$

🔔 Important

All the key value pairs will be combined with the implicit AND operator.

Example usage for the sample-list API

```
ardana > ceilometer sample-list --meter image.serve -q 'resource_id=a1ec2585'
```

```
ardana > ceilometer sample-list --meter instance -q
 'resource_id=<ResourceID>;metadata.event_type=<eventType>'
```

12.3.4.9 Statistics API

Ceilometer Statistics is an open-ended query API that performs queries on the table of data collected from a meter. The Statistics API obtains the minimum and maximum timestamp for the meter that is being queried.

The Statistics API also provides a set of statistical functions. These functions perform basic aggregation for meter-specific data over a period of time. Statistics API includes the following functions:

Count

> the number of discrete samples collected in each period

Maximum

> the sample with the maximum value in a selected time period

Minimum

> the sample with the minimum value in a selected time period

Average

> the average value of a samples within a selected time period

Sum

> the total value of all samples within a selected time period added together

🔔 Important

> The Statistics API can put a significant load on the database leading to unexpected results and or failures. Therefore, you should be careful about restricting your queries.

Limitations of Statistics-list API

- filtering with metadata is not supported

- the `groupby` option is only supported with only one parameter. That single parameter has to be one of the following:

```
user_id
project_id
resource_id
source
```

- only the following are supported as aggregate functions: **average**, **minimum**, **maximum**, **sum**, and **count**

- when no time period is specified in the query, a default period of 300 seconds is used to aggregate measurements (samples)

- the **meter name** is a required positional parameter

- when a closed time range is specified, results may contain an extra row with **duration**, **duration start**, **duration end** assigned with a value of **None**. This row has a start and end time period that fall outside the requested time range and can be ignored. Ceilometer does not remove this row because it is by design inside the back-end Monasca.

Statistical Query Best Practices

By default, the Statistics API will return a limited number of statistics. You can control the output using the **period "."** parameter.

Without a period parameter

> only a few statistics: **minimum**, **maximum**, **avgerage** and **sum**

With a period parameter "."

> the range is divided into equal periods and Statistics API finds the **count**, **minimum**, **maximum**, **average**, and **sum** for each of the periods

> ## Important
>
> It is recommended that you provide a `timestamp` parameter with every query, regardless of whether a period paramter is used. For example:
>
> ```
> timestamp>={$start-timestamp} and timestamp<{$end-timestamp}
> ```
>
> It is also recommended that you query a period of time that covers at most 1 day (24 hours).

Examples

Without period parameter

```
ardana > ceilometer statistics -q
 "timestamp>=2014-12-11T00:00:10;timestamp<2014-12-11T23:00:00" -m "instance"
```

With the period parameter "."

```
ardana > ceilometer statistics -q
 "timestamp>=2014-12-11T00:00:10;timestamp<2014-12-11T23:00:00" -m "instance" -p
 3600
```

If the **query** and **timestamp** parameters are not provided, all records in the database will be queried. This is not recommended. Use the following recommended values for **query (-q)** parameter and **period (-p)** parameters:

-q

Always provide a **timestamp** range, with the following guidelines:

- recommended maximum time period to query is one day (24 hours)

- do not set the timestamp range to greater than a day

- it is better to provide no time stamp range than to set the time period for more than 1 day

- example of an acceptable range:

```
-q "timestamp>=2014-12-11T00:00:10;timestamp<2014-12-11T23:00:00"
```

-p

Provide a large number in seconds, with the following guidelines:

- recommended minimum value is 3600 or more (1 hour or more)

- providing a period of less than 3600 is not recommended

- Use this parameter to divide the overall time range into smaller intervals. A small period parameter value will translate into a very large number of queries against the database.

- Example of an acceptable range:

```
-p 3600
```

12.3.5 Configure the Ceilometer Metering Service

HPE Helion OpenStack 8 automatically deploys Ceilometer to use the Monasca database. Ceilometer is deployed on the same control plane nodes along with other OpenStack services such as Keystone, Nova, Neutron, Glance, and Swift.

The Metering Service can be configured using one of the procedures described below.

12.3.5.1   Run the Upgrade Playbook

Follow Standard Service upgrade mechanism available in the Cloud Lifecycle Manager distribution. For Ceilometer, the playbook included with HPE Helion OpenStack is **ceilometer-upgrade.yml**

12.3.5.2   Configure Apache2 for the Ceilometer API

Reporting APIs provide access to the metering data stored in Ceilometer. These APIs are accessed by Horizon to provide basic usage data and information.HPE Helion OpenStack uses Apache2 Web Server to provide the API access.

> **Important**
>
> To improve API responsiveness you can increase the number of threads and processes in the Ceilometer configuration file. The Ceilometer API runs as an WSGI processes. Each process can have a certain amount of threads managing the filters and applications, which can comprise the processing pipeline.

To configure Apache:

1. Edit the Ceilometer configuration files.

2. Reload and verify Apache2.

Edit the Ceilometer Configuration Files

To create a working file for Ceilometer with the correct settings:

1. To add the configuration file to the correct folder, copy the following file:

   ```
   ceilometer.conf
   ```

   to the following directory:

   ```
   /etc/apache2/vhosts.d/
   ```

2. To verify the settings, in a text editor, open the `ceilometer_modwsgi.vhost` file.

3. The ceilometer_modwsgi.conf file should have the following data. If it does not exist, add it to the file.

```
Listen <ipaddress>:8777
<VirtualHost *:8777>
  WSGIScriptAlias / /srv/www/ceilometer/ceilometer-api
  WSGIDaemonProcess ceilometer user=ceilometer group=ceilometer processes=4
 threads=5 socket-timeout=600 python-path=/opt/stack/service/ceilometer-api/
venv:/opt/stack/service/ceilometer-api/venv/lib/python2.7/site-packages/ display-
name=ceilometer-api
  WSGIApplicationGroup %{GLOBAL}
  WSGIProcessGroup ceilometer

  ErrorLog /var/log/ceilometer/ceilometer_modwsgi.log
  LogLevel INFO
  CustomLog /var/log/ceilometer/ceilometer_access.log combined

  <Directory /opt/stack/service/ceilometer-api/venv/lib/python2.7/site-packages/
ceilometer>
    Options Indexes FollowSymLinks MultiViews
    Require all granted
    AllowOverride None
    Order allow,deny
    allow from all
    LimitRequestBody 102400
  </Directory>
</VirtualHost>
```

Note

The WSGIDaemon Recommended Settings are to use four processes running in parallel:

```
processes=4
```

Five threads for each process is also recommended:

```
threads=5
```

4. To add a softlink for the ceilometer.conf, run:

```
tux > sudo ln -s /etc/apache2/vhosts.d/ceilometer_modwsgi.vhost /etc/apache2/
vhosts.d/ceilometer_modwsgi.conf
```

Reload and Verify Apache2

For the changes to take effect, the Apache2 service needs to be reloaded. This ensures that all the configuration changes are saved and the service has applied them. The system administrator can change the configuration of processes and threads and experiment if alternative settings are necessary.

Once the Apache2 service has been reloaded you can verify that the Ceilometer APIs are running and able to receive incoming traffic. The Ceilometer APIs are listening on port 8777.

To reload and verify the Apache2 service:

1. To reload Apache2, run:

   ```
   tux > sudo systemctl reload apache2.service
   ```

2. To verify the service is running, run:

   ```
   tux > sudo systemctl status apache2.service
   ```

> **Important**
>
> In a working environment, the list of entries in the output should match the number of processes in the configuration file. In the example configuration file, the recommended number of 4 is used, and the number of Running Instances is also 4.

You can also verify that Apache2 is accepting incoming traffic using the following procedure:

1. To verify traffic on port 8777, run:

   ```
   tux > sudo netstat -tulpn | grep 8777
   ```

2. Verify your output is similar to the following example:

   ```
   tcp6 0 0 :::8777 :::* LISTEN 8959/apache2
   ```

Important

If Ceilometer fails to deploy:

- check the proxy setting

- unset the https_proxy, for example:

```
unset http_proxy HTTP_PROXY HTTPS_PROXY
```

12.3.5.3   Enable Services for Messaging Notifications

After installation of HPE Helion OpenStack, the following services are enabled by default to send notifications:

- Nova

- Cinder

- Glance

- Neutron

- Swift

The list of meters for these services are specified in the Notification Agent or Polling Agent's pipeline configuration file.

For steps on how to edit the pipeline configuration files, see: *Section 12.3.6, "Ceilometer Metering Service Notifications"*

12.3.5.4   Restart the Polling Agent

The Polling Agent is responsible for coordinating the polling activity. It parses the **pipeline.yml** configuration file and identifies all the sources where data is collected. The sources are then evaluated and are translated to resources that a dedicated pollster can retrieve. The Polling Agent follows this process:

1. At each identified interval, the **pipeline.yml** configuration file is parsed.

2. The resource list is composed.

3. The pollster collects the data.

4. The pollster sends data to the queue.

Metering processes should normally be operating at all times. This need is addressed by the Upstart event engine which is designed to run on any Linux system. Upstart creates events, handles the consequences of those events, and starts and stops processes as required. Upstart will continually attempt to restart stopped processes even if the process was stopped manually. To stop or start the Polling Agent and avoid the conflict with Upstart, using the following steps.

To restart the Polling Agent:

1. To determine whether the process is running, run:

```
tux > sudo systemctl status ceilometer-agent-notification
#SAMPLE OUTPUT:
ceilometer-agent-notification.service - ceilometer-agent-notification Service
   Loaded: loaded (/etc/systemd/system/ceilometer-agent-notification.service;
 enabled; vendor preset: disabled)
   Active: active (running) since Tue 2018-06-12 05:07:14 UTC; 2 days ago
 Main PID: 31529 (ceilometer-agen)
    Tasks: 69
   CGroup: /system.slice/ceilometer-agent-notification.service
           ├─31529 ceilometer-agent-notification: master process [/opt/stack/
service/ceilometer-agent-notification/venv/bin/ceilometer-agent-notification --
config-file /opt/stack/service/ceilometer-agent-noti...
           └─31621 ceilometer-agent-notification: NotificationService worker(0)

Jun 12 05:07:14 ardana-qe201-cp1-c1-m2-mgmt systemd[1]: Started ceilometer-agent-
notification Service.
```

2. To stop the process, run:

```
tux > sudo systemctl stop ceilometer-agent-notification
```

3. To start the process, run:

```
tux > sudo systemctl start ceilometer-agent-notification
```

12.3.5.5 Replace a Logging, Monitoring, and Metering Controller

In a medium-scale environment, if a metering controller has to be replaced or rebuilt, use the following steps:

1. *Section 13.1.2.1, "Replacing a Controller Node"*.

2. If the Ceilometer nodes are not on the shared control plane, to implement the changes and replace the controller, you must reconfigure Ceilometer. To do this, run the ceilometer-reconfigure.yml ansible playbook **without** the limit option

12.3.5.6 Configure Monitoring

The Monasca HTTP Process monitors the Ceilometer API service. Ceilometer's notification and polling agents are also monitored. If these agents are down, Monasca monitoring alarms are triggered. You can use the notification alarms to debug the issue and restart the notifications agent. However, for Central-Agent (polling) and Collector the alarms need to be deleted. These two processes are not started after an upgrade so when the monitoring process checks the alarms for these components, they will be in UNDETERMINED state. HPE Helion OpenStack does not monitor these processes anymore so the best option to resolve this issue is to manually delete alarms that are no longer used but are installed.

To resolve notification alarms, first check the **ceilometer-agent-notification** logs for errors in the **/var/log/ceilometer** directory. You can also use the Operations Console to access Kibana and check the logs. This will help you understand and debug the error.

To restart the service, run the **ceilometer-start.yml**. This playbook starts the ceilometer processes that has stopped and only restarts during install, upgrade or reconfigure which is what is needed in this case. Restarting the process that has stopped will resolve this alarm because this Monasca alarm means that ceilometer-agent-notification is no longer running on certain nodes.

You can access Ceilometer data through Monasca. Ceilometer publishes samples to Monasca with credentials of the following accounts:

- **ceilometer** user

- **services**

Data collected by Ceilometer can also be retrieved by the Monasca REST API. Make sure you use the following guidelines when requesting data from the Monasca REST API:

- Verify you have the monasca-admin role. This role is configured in the monasca-api configuration file.

- Specify the `tenant id` of the **services** project.

For more details, read the Monasca API Specification (https://github.com/openstack/monasca-api/blob/master/docs/monasca-api-spec.md) ↗ .

To run Monasca commands at the command line, you must be have the **admin** role. This allows you to use the Ceilometer account credentials to replace the default admin account credentials defined in the **service.osrc** file. When you use the Ceilometer account credentials, Monasca commands will only return data collected by Ceilometer. At this time, Monasca command line interface (CLI) does not support the data retrieval of other tenants or projects.

12.3.6 Ceilometer Metering Service Notifications

Ceilometer uses the notification agent to listen to the message queue, convert notifications to Events and Samples, and apply pipeline actions.

12.3.6.1 Manage Whitelisting and Polling

HPE Helion OpenStack is designed to reduce the amount of data that is stored. HPE Helion OpenStack's use of a SQL-based cluster, which is not recommended for big data, means you must control the data that Ceilometer collects. You can do this by filtering (whitelisting) the data or by using the configuration files for the Ceilometer Polling Agent and the Ceilometer Notificfoation Agent.

Whitelisting is used in a rule specification as a positive filtering parameter. Whitelist is only included in rules that can be used in direct mappings, for identity service issues such as service discovery, provisioning users, groups, roles, projects, domains as well as user authentication and authorization.

You can run tests against specific scenarios to see if filtering reduces the amount of data stored. You can create a test by editing or creating a run filter file (whitelist). For steps on how to do this, see: *Book "Installing with Cloud Lifecycle Manager", Chapter 27 "Cloud Verification", Section 27.1 "API Verification".*

Ceilometer Polling Agent (polling agent) and Ceilometer Notification Agent (notification agent) use different pipeline.yaml files to configure meters that are collected. This prevents accidentally polling for meters which can be retrieved by the polling agent as well as the notification agent. For example, glance image and image.size are meters which can be retrieved both by polling and notifications.

In both of the separate configuration files, there is a setting for `interval`. The interval attribute determines the frequency, in seconds, of how often data is collected. You can use this setting to control the amount of resources that are used for notifications and for polling. For example, you want to use more resources for notifications and less for polling. To accomplish this you would set the `interval` in the polling configuration file to a large amount of time, such as 604800 seconds, which polls only once a week. Then in the notifications configuration file, you can set the `interval` to a higher amount, such as collecting data every 30 seconds.

🔔 Important

Swift account data will be collected using the polling mechanism in an hourly interval.

Setting this interval to manage both notifications and polling is the recommended procedure when using a SQL cluster back-end.

Sample Ceilometer Polling Agent file:

```
#File: ~/opt/stack/service/ceilometer-polling/etc/pipeline-polling.yaml
---
sources:
    - name: swift_source
      interval: 3600
      meters:
          - "storage.objects"
          - "storage.objects.size"
          - "storage.objects.containers"
      resources:
      discovery:
      sinks:
          - meter_sink
sinks:
```

```
    - name: meter_sink
      transformers:
      publishers:
          - notifier://
```

Sample Ceilometer Notification Agent(notification agent) file:

```
#File:    ~/opt/stack/service/ceilometer-agent-notification/etc/pipeline-agent-
notification.yaml
---
sources:
    - name: meter_source
      interval: 30
      meters:
          - "instance"
          - "image"
          - "image.size"
          - "image.upload"
          - "image.delete"
          - "volume"
          - "volume.size"
          - "snapshot"
          - "snapshot.size"
          - "ip.floating"
          - "network"
          - "network.create"
          - "network.update"
resources:
discovery:
sinks:
          - meter_sink
sinks:
    - name: meter_sink
      transformers:
      publishers:
          - notifier://
```

Both of the pipeline files have two major sections:

Sources

represents the data that is collected either from notifications posted by services or through polling. In the Sources section there is a list of meters. These meters define what kind of data is collected. For a full list refer to the Ceilometer documentation available at: Telemetry Measurements (http://docs.openstack.org/admin-guide/telemetry-measurements.html) ↗

Sinks

represents how the data is modified before it is published to the internal queue for collection and storage.

You will only need to change a setting in the Sources section to control the data collection interval.

For more information, see Telemetry Measurements (http://docs.openstack.org/admin-guide-cloud/telemetry-measurements.html) ↗

To change the Ceilometer Polling Agent interval setting:

1. To find the polling agent configuration file, run:

```
cd ~/opt/stack/service/ceilometer-polling/etc
```

2. In a text editor, open the following file:

```
pipeline-polling.yaml
```

3. In the following section, change the value of `interval` to the desired amount of time:

```
---
sources:
    - name: swift_source
      interval: 3600
      meters:
          - "storage.objects"
          - "storage.objects.size"
          - "storage.objects.containers"
      resources:
      discovery:
      sinks:
          - meter_sink
 sinks:
    - name: meter_sink
      transformers:
      publishers:
          - notifier://
```

In the sample code above, the polling agent will collect data every 600 seconds, or 10 minutes.

To change the Ceilometer Notification Agent (notification agent) interval setting:

1. To find the notification agent configuration file, run:

```
cd /opt/stack/service/ceilometer-agent-notification
```

2. In a text editor, open the following file:

```
pipeline-agent-notification.yaml
```

3. In the following section, change the value of `interval` to the desired amount of time:

```
sources:
    - name: meter_source
      interval: 30
      meters:
          - "instance"
          - "image"
          - "image.size"
          - "image.upload"
          - "image.delete"
          - "volume"
          - "volume.size"
          - "snapshot"
          - "snapshot.size"
          - "ip.floating"
          - "network"
          - "network.create"
          - "network.update"
```

In the sample code above, the notification agent will collect data every 30 seconds.

☞ Note

The `pipeline-agent-notification.yaml` file needs to be changed on all controller nodes to change the white-listing and polling strategy.

12.3.6.2 Edit the List of Meters

The number of enabled meters can be reduced or increased by editing the pipeline configuration of the notification and polling agents. To deploy these changes you must then restart the agent. If pollsters and notifications are both modified, then you will have to restart both the Polling Agent and the Notification Agent. Ceilometer Collector will also need to be restarted. The following code is an example of a compute-only Ceilometer Notification Agent (notification agent) **pipeline-agent-notification.yaml** file:

```
---
sources:
    - name: meter_source
      interval: 86400
      meters:
          - "instance"
          - "memory"
          - "vcpus"
          - "compute.instance.create.end"
          - "compute.instance.delete.end"
          - "compute.instance.update"
          - "compute.instance.exists"
      sinks:
          - meter_sink
sinks:
    - name: meter_sink
      transformers:
      publishers:
          - notifier://
```

🔔 Important

> If you enable meters at the container level in this file, every time the polling interval triggers a collection, at least 5 messages per existing container in Swift are collected.

The following table illustrates the amount of data produced hourly in different scenarios:

Swift Containers	Swift Objects per container	Samples per Hour	Samples stored per 24 hours
10	10	500	12000
10	100	5000	120000

| 100 | 100 | 50000 | 1200000 |
| 100 | 1000 | 500000 | 12000000 |

The data in the table shows that even a very small Swift storage with 10 containers and 100 files will store 120,000 samples in 24 hours, generating a total of 3.6 million samples.

Important

The size of each file does not have any impact on the number of samples collected. As shown in the table above, the smallest number of samples results from polling when there are a small number of files and a small number of containers. When there are a lot of small files and containers, the number of samples is the highest.

12.3.6.3 Add Resource Fields to Meters

By default, not all the resource metadata fields for an event are recorded and stored in Ceilometer. If you want to collect metadata fields for a consumer application, for example, it is easier to add a field to an existing meter rather than creating a new meter. If you create a new meter, you must also reconfigure Ceilometer.

Important

Consider the following information before you add or edit a meter:

- You can add a maximum of 12 new fields.

- Adding or editing a meter causes all non-default meters to STOP receiving notifications. You will need to restart Ceilometer.

- New meters added to the `pipeline-polling.yaml.j2` file must also be added to the `pipeline-agent-notification.yaml.j2` file. This is due to the fact that polling meters are drained by the notification agent and not by the collector.

- After HPE Helion OpenStack is installed, services like compute, cinder, glance, and neutron are configured to publish Ceilometer meters by default. Other meters can also be enabled after the services are configured to start publishing the meter. The only requirement for publishing a meter is that the `origin` must

have a value of `notification`. For a complete list of meters, see the OpenStack documentation on Measurements (http://docs.openstack.org/admin-guide/telemetry-measurements.html) ↗.

- Not all meters are supported. Meters collected by Ceilometer Compute Agent or any agent other than Ceilometer Polling are not supported or tested with HPE Helion OpenStack.

- Identity meters are disabled by Keystone.

- To enable Ceilometer to start collecting meters, some services require you enable the meters you need in the service first before enabling them in Ceilometer. Refer to the documentation for the specific service before you add new meters or resource fields.

To add Resource Metadata fields:

1. Log on to the Cloud Lifecycle Manager (deployer node).

2. To change to the Ceilometer directory, run:

   ```
   ardana > cd ~/openstack/my_cloud/config/ceilometer
   ```

3. In a text editor, open the target configuration file (for example, monasca-field-definitions.yaml.j2).

4. In the metadata section, either add a new meter or edit an existing one provided by HPE Helion OpenStack.

5. Include the metadata fields you need. You can use the `instance meter` in the file as an example.

6. Save and close the configuration file.

7. To save your changes in HPE Helion OpenStack, run:

   ```
   ardana > cd ~/openstack
   ardana > git add -A
   ardana > git commit -m "My config"
   ```

8. If you added a new meter, reconfigure Ceilometer:

   ```
   ardana > cd ~/openstack/ardana/ansible/
   ```

```
# To run the config-processor playbook:
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
#To run the ready-deployment playbook:
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts ceilometer-reconfigure.yml
```

12.3.6.4 Update the Polling Strategy and Swift Considerations

Polling can be very taxing on the system due to the sheer volume of data thtyat the system may have to process. It also has a severe impact on queries since the database will now have a very large amount of data to scan to respond to the query. This consumes a great amount of cpu and memory. This can result in long wait times for query responses, and in extreme cases can result in timeouts.

There are 3 polling meters in Swift:

- storage.objects

- storage.objects.size

- storage.objects.containers

Here is an example of `pipeline.yml` in which Swift polling is set to occur hourly.

```
---
    sources:
    - name: swift_source
    interval: 3600
    meters:
    - "storage.objects"
    - "storage.objects.size"
    - "storage.objects.containers"
    resources:
    discovery:
    sinks:
    - meter_sink
    sinks:
    - name: meter_sink
    transformers:
    publishers:
    - notifier://
```

With this configuration above, we did not enable polling of container based meters and we only collect 3 messages for any given tenant, one for each meter listed in the configuration files. Since we have 3 messages only per tenant, it does not create a heavy load on the MySQL database as it would have if container-based meters were enabled. Hence, other APIs are not hit because of this data collection configuration.

12.3.7 Ceilometer Metering Setting Role-based Access Control

Role Base Access Control (RBAC) is a technique that limits access to resources based on a specific set of roles associated with each user's credentials.

Keystone has a set of users that are associated with each project. Each user has one or more roles. After a user has authenticated with Keystone using a valid set of credentials, Keystone will augment that request with the Roles that are associated with that user. These roles are added to the Request Header under the X-Roles attribute and are presented as a comma-separated list.

12.3.7.1 Displaying All Users

To discover the list of users available in the system, an administrator can run the following command using the Keystone command-line interface:

```
ardana > source ~/service.osrc
ardana > openstack user list
```

The output should resemble this response, which is a list of all the users currently available in this system.

```
+----------------------------------+-----------+---------+---------------------+----+
|                id                |   name    | enabled |        email        |    |
+----------------------------------+-----------+---------+---------------------+----+
| 1c20d327c92a4ea8bb513894ce26f1f1 |   admin   |  True   | admin.example.com   |
| 0f48f3cc093c44b4ad969898713a0d65 | ceilometer |  True   | nobody@example.com  |
| 85ba98d27b1c4c8f97993e34fcd14f48 |   cinder  |  True   | nobody@example.com  |
| d2ff982a0b6547d0921b94957db714d6 |    demo   |  True   |  demo@example.com   |
| b2d597e83664489ebd1d3c4742a04b7c |    ec2    |  True   | nobody@example.com  |
| 2bd85070ceec4b608d9f1b06c6be22cb |   glance  |  True   | nobody@example.com  |
| 0e9e2daebbd3464097557b87af4afa4c |    heat   |  True   | nobody@example.com  |
| 0b466ddc2c0f478aa139d2a0be314467 |   neutron |  True   | nobody@example.com  |
| 5cda1a541dee4555aab88f36e5759268 |    nova   |  True   | nobody@example.com ||
| 5cda1a541dee4555aab88f36e5759268 |    nova   |  True   | nobody@example.com  |
| 1cefd1361be8437d9684eb2add8bdbfa |   swift   |  True   | nobody@example.com  |
```

```
| f05bac3532c44414a26c0086797dab23 | user20141203213957|True| nobody@example.com |
| 3db0588e140d4f88b0d4cc8b5ca86a0b | user20141205232231|True| nobody@example.com |
+---------------------------------+----------------------------------------------+----+
```

12.3.7.2 Displaying All Roles

To see all the roles that are currently available in the deployment, an administrator (someone with the admin role) can run the following command:

```
ardana > source ~/service.osrc
ardana > openstack role list
```

The output should resemble the following response:

```
+----------------------------------+-------------------------------------+
|                id                |                name                 |
+----------------------------------+-------------------------------------+
| 507bface531e4ac2b7019a1684df3370 |             ResellerAdmin            |
| 9fe2ff9ee4384b1894a90878d3e92bab |              _member_                |
| e00e9406b536470dbde2689ce1edb683 |               admin                  |
| aa60501f1e664ddab72b0a9f27f96d2c |           heat_stack_user            |
| a082d27b033b4fdea37ebb2a5dc1a07b |              service                 |
| 8f11f6761534407585feecb5e896922f |            swiftoperator             |
+----------------------------------+-------------------------------------+
```

12.3.7.3 Assigning a Role to a User

In this example, we want to add the role **ResellerAdmin** to the demo user who has the ID **d2ff982a0b6547d0921b94957db714d6**.

1. Determine which Project/Tenant the user belongs to.

   ```
   ardana > source ~/service.osrc
   ardana > openstack user show d2ff982a0b6547d0921b94957db714d6
   ```

   The response should resemble the following output:

   ```
   +--------------------+----------------------------------+
   | Field              | Value                            |
   +--------------------+----------------------------------+
   | domain_id          | default                          |
   | enabled            | True                             |
   |    id              | d2ff982a0b6547d0921b94957db714d6 |
   ```

```
| name              | admin                          |
| options           | {}                             |
| password_expires_at | None                         |
+-------------------+--------------------------------+
```

2. We need to link the ResellerAdmin Role to a Project/Tenant. To start, determine which tenants are available on this deployment.

```
ardana > source ~/service.osrc
ardana > openstack project list
```

The response should resemble the following output:

```
+----------------------------------+-------------------------------+--+
|                id                |            name   | enabled |
+----------------------------------+-------------------------------+--+
| 4a8f4207a13444089a18dc524f41b2cf |           admin   |   True  |
| 00cbaf647bf24627b01b1a314e796138 |           demo    |   True  |
| 8374761f28df43b09b20fcd3148c4a08 |           gf1     |   True  |
| 0f8a9eef727f4011a7c709e3fbe435fa |           gf2     |   True  |
| 6eff7b888f8e470a89a113acfcca87db |           gf3     |   True  |
| f0b5d86c7769478da82cdeb180aba1b0 |           jaq1    |   True  |
| a46f1127e78744e88d6bba20d2fc6e23 |           jaq2    |   True  |
| 977b9b7f9a6b4f59aaa70e5a1f4ebf0b |           jaq3    |   True  |
| 4055962ba9e44561ab495e8d4fafa41d |           jaq4    |   True  |
| 33ec7f15476545d1980cf90b05e1b5a8 |           jaq5    |   True  |
| 9550570f8bf147b3b9451a635a1024a1 |          service  |   True  |
+----------------------------------+-------------------------------+--+
```

3. Now that we have all the pieces, we can assign the ResellerAdmin role to this User on the Demo project.

```
ardana > openstack role add --user d2ff982a0b6547d0921b94957db714d6 --project
  00cbaf647bf24627b01b1a314e796138 507bface531e4ac2b7019a1684df3370
```

This will produce no response if everything is correct.

4. Validate that the role has been assigned correctly. Pass in the user and tenant ID and request a list of roles assigned.

```
ardana > openstack role list --user d2ff982a0b6547d0921b94957db714d6 --project
  00cbaf647bf24627b01b1a314e796138
```

Note that all members have the *member* role as a default role in addition to any other roles that have been assigned.

```
+-------------------------------+---------------
+-------------------------------+---------------------------------+
|                id             |     name      |            user_id
  |   tenant_id           |
+-------------------------------+---------------
+-------------------------------+---------------------------------+
| 507bface531e4ac2b7019a1684df3370 | ResellerAdmin |
 d2ff982a0b6547d0921b94957db714d6 | 00cbaf647bf24627b01b1a314e796138 |
| 9fe2ff9ee4384b1894a90878d3e92bab |   _member_   |
 d2ff982a0b6547d0921b94957db714d6 | 00cbaf647bf24627b01b1a314e796138 |
+-------------------------------+---------------
+-------------------------------+---------------------------------+
```

12.3.7.4 Creating a New Role

In this example, we will create a Level 3 Support role called **L3Support**.

1. Add the new role to the list of roles.

   ```
   ardana > openstack role create L3Support
   ```

   The response should resemble the following output:

   ```
   +----------+----------------------------------+
   | Property |              Value               |
   +----------+----------------------------------+
   |    id    | 7e77946db05645c4ba56c6c82bf3f8d2 |
   |   name   |             L3Support            |
   +----------+----------------------------------+
   ```

2. Now that we have the new role's ID, we can add that role to the Demo user from the previous example.

   ```
   ardana > openstack role add --user d2ff982a0b6547d0921b94957db714d6  --project
    00cbaf647bf24627b01b1a314e796138 7e77946db05645c4ba56c6c82bf3f8d2
   ```

   This will produce no response if everything is correct.

3. Verify that the user Demo has both the ResellerAdmin and L3Support roles.

   ```
   ardana > openstack role list --user d2ff982a0b6547d0921b94957db714d6 --project
    00cbaf647bf24627b01b1a314e796138
   ```

4. The response should resemble the following output. Note that this user has the L3Support role, the ResellerAdmin role, and the default member role.

```
+--------------------------------+---------------
+--------------------------------+--------------------------------+
|               id               |     name     |             user_id
  |            tenant_id          |     |
+--------------------------------+--------------
+--------------------------------+--------------------------------+
| 7e77946db05645c4ba56c6c82bf3f8d2 |  L3Support   |
 d2ff982a0b6547d0921b94957db714d6 | 00cbaf647bf24627b01b1a314e796138 |
| 507bface531e4ac2b7019a1684df3370 | ResellerAdmin |
 d2ff982a0b6547d0921b94957db714d6 | 00cbaf647bf24627b01b1a314e796138 |
| 9fe2ff9ee4384b1894a90878d3e92bab |   _member_   |
 d2ff982a0b6547d0921b94957db714d6 | 00cbaf647bf24627b01b1a314e796138 |
+--------------------------------+---------------
+--------------------------------+--------------------------------+
```

12.3.7.5 Access Policies

Before introducing RBAC, Ceilometer had very simple access control. There were two types of user: admins and users. Admins will be able to access any API and perform any operation. Users will only be able to access non-admin APIs and perform operations only on the Project/Tenant where they belonged.

12.3.7.6 New RBAC Policy File

This is the policy file for Ceilometer without RBAC (**etc/ceilometer/policy.json**)

```
{
  "context_is_admin": "role:admin"
}
```

With the RBAC-enhanced code it is possible to control access to each API command. The new policy file (*rbac_policy.json*) looks like this.

```
{
    "context_is_admin": "role:admin",
    "telemetry:get_samples": "rule:context_is_admin",
    "telemetry:get_sample": "rule:context_is_admin",
    "telemetry:query_sample": "rule:context_is_admin",
    "telemetry:create_samples": "rule:context_is_admin",
    "telemetry:compute_statistics": "rule:context_is_admin",
```

```
     "telemetry:get_meters": "rule:context_is_admin",
     "telemetry:get_resource": "rule:context_is_admin",
     "telemetry:get_resources": "rule:context_is_admin",
     "telemetry:get_alarm": "rule:context_is_admin",
     "telemetry:query_alarm": "rule:context_is_admin",
     "telemetry:get_alarm_state": "rule:context_is_admin",
     "telemetry:get_alarms": "rule:context_is_admin",
     "telemetry:create_alarm": "rule:context_is_admin",
     "telemetry:set_alarm": "rule:service_role",
     "telemetry:delete_alarm": "rule:context_is_admin",
     "telemetry:alarm_history": "rule:context_is_admin",
     "telemetry:change_alarm_state": "rule:context_is_admin",
     "telemetry:query_alarm_history": "rule:context_is_admin"
}
```

Note that the API action names are namespaced using the **telemetry:** prefix. This avoids potential confusion if other services have policies with the same name.

12.3.7.7 Applying Policies to Roles

Copy the **rbac_policy.json** file over the **policy.json** file and make any required changes.

12.3.7.8 Apply a policy to multiple roles

For example, the ResellerAdmin role could also be permitted to access **compute_statistics**. This change would require the following changes in the **rbac_policy.json** policy file:

```
{
    "context_is_admin": "role:admin",
    "i_am_reseller": "role:ResellerAdmin",
    "telemetry:get_samples": "rule:context_is_admin",
    "telemetry:get_sample": "rule:context_is_admin",
    "telemetry:query_sample": "rule:context_is_admin",
    "telemetry:create_samples": "rule:context_is_admin",
    "telemetry:compute_statistics": "rule:context_is_admin or rule:i_am_reseller",
    ...
}
```

After a policy change has been made all the API services will need to be restarted .

12.3.7.9 Apply a policy to a non-default role only

Another example: assign the L3Support role to the **get_meters** API and exclude all other roles.

```
{
    "context_is_admin": "role:admin",
    "i_am_reseller": "role:ResellerAdmin",
    "l3_support": "role:L3Support",
    "telemetry:get_samples": "rule:context_is_admin",
    "telemetry:get_sample": "rule:context_is_admin",
    "telemetry:query_sample": "rule:context_is_admin",
    "telemetry:create_samples": "rule:context_is_admin",
    "telemetry:compute_statistics": "rule:context_is_admin or rule:i_am_reseller",
    "telemetry:get_meters": "rule:l3_support",
    ...
}
```

12.3.7.10   Writing a Policy

The Policy Engine capabilities are as expressible using a set of rules and guidelines. For a complete reference, please see the OSLO policy documentation (https://github.com/openstack/oslo.policy/blob/master/oslo_policy/policy.py) ↗.

Policies can be expressed in one of two forms: A list of lists, or a string written in the new policy language.

In the list-of-lists representation, each check inside the innermost list is combined with an **and** conjunction - for that check to pass, **all** the specified checks must pass. These innermost lists are then combined as with an **or** conjunction.

As an example, take the following rule, expressed in the list-of-lists representation:

```
[["role:admin"], ["project_id:%(project_id)s", "role:projectadmin"]]
```

In the policy language, each check is specified the same way as in the list-of-lists representation: a simple [a:b] pair that is matched to the correct class to perform that check.

- User's Role

  ```
  role:admin
  ```

- Rules already defined on policy

  ```
  rule:admin_required
  ```

- Against a URL (URL checking must return TRUE to be valid)

  ```
  http://my-url.org/check
  ```

- User attributes (obtained through the token: user_id, domain_id, project_id)

```
project_id:%(target.project.id)s
```

- Strings

```
<variable>:'xpto2035abc'
'myproject':<variable>
```

- Literals

```
project_id:xpto2035abc
domain_id:20
True:%(user.enabled)s
```

Conjunction operators are also available, allowing for more flexibility in crafting policies. So, in the policy language, the previous check in list-of-lists becomes:

```
role:admin or (project_id:%(project_id)s and role:projectadmin)
```

The policy language also has the NOT operator, allowing for richer policy rules:

```
project_id:%(project_id)s and not role:dunce
```

Attributes sent along with API calls can be used by the policy engine (on the right side of the expression), by using the following syntax:

```
<some_value>:%(user.id)s
```

Note: two special policy checks should be mentioned; the policy check **@** will **always accept** an access, and the policy check **!** will **always reject** an access.

12.3.8   Ceilometer Metering Failover HA Support

In the HPE Helion OpenStack environment, the Ceilometer metering service supports native Active-Active high-availability (HA) for the notification and polling agents. Implementing HA support includes workload-balancing, workload-distribution and failover.

Tooz is the coordination engine that is used to coordinate workload among multiple active agent instances. It also maintains the knowledge of active-instance-to-handle failover and group membership using hearbeats (pings).

Zookeeper is used as the coordination backend. Zookeeper uses Tooz to expose the APIs that manage group membership and retrieve workload specific to each agent.

The following section in the configuration file is used to implement high-availability (HA):

```
[coordination]
backend_url = <IP address of Zookeeper host: port> (port is usually 2181 as a zookeeper
 default)
heartbeat = 1.0
check_watchers = 10.0
```

For the notification agent to be configured in HA mode, additional configuration is needed in the configuration file:

```
[notification]
workload_partitioning = true
```

The HA notification agent distributes workload among multiple queues that are created based on the number of unique source:sink combinations. The combinations are configured in the notification agent pipeline configuration file. If there are additional services to be metered using notifications, then the recommendation is to use a separate source for those events. This is recommended especially if the expected load of data from that source is considered high. Implementing HA support should lead to better workload balancing among multiple active notification agents.

Ceilometer-expirer is also an Active-Active HA. Tooz is used to pick an expirer process that acquires a lock when there are multiple contenders and the winning process runs. There is no failover support, as expirer is not a daemon and is scheduled to run at pre-determined intervals.

🔔 Important

You must ensure that a single expirer process runs when multiple processes are scheduled to run at the same time. This must be done using cron-based scheduling. on multiple controller nodes

The following configuration is needed to enable expirer HA:

```
[coordination]
backend_url = <IP address of Zookeeper host: port> (port is usually 2181 as a zookeeper
 default)
heartbeat = 1.0
```

```
check_watchers = 10.0
```

The notification agent HA support is mainly designed to coordinate among notification agents so that correlated samples can be handled by the same agent. This happens when samples get transformed from other samples. The HPE Helion OpenStack Ceilometer pipeline has no transformers, so this task of coordination and workload partitioning does not need to be enabled. The notification agent is deployed on multiple controller nodes and they distribute workload among themselves by randomly fetching the data from the queue.

To disable coordination and workload partitioning by OpenStack, set the following value in the configuration file:

```
[notification]
workload_partitioning = False
```

Important

When a configuration change is made to an API running under the HA Proxy, that change needs to be replicated in **all** controllers.

12.3.9 Optimizing the Ceilometer Metering Service

You can improve API and database responsiveness by configuring metering to store only the data you are require. This topic provides strategies for getting the most out of metering while not overloading your resources.

12.3.9.1 Change the List of Meters

The list of meters can be easily reduced or increased by editing the pipeline.yaml file and restarting the polling agent.

Sample compute-only pipeline.yaml file with the daily poll interval:

```
---
sources:
    - name: meter_source
      interval: 86400
      meters:
          - "instance"
```

```
        - "memory"
        - "vcpus"
        - "compute.instance.create.end"
        - "compute.instance.delete.end"
        - "compute.instance.update"
        - "compute.instance.exists"
    sinks:
        - meter_sink
sinks:
    - name: meter_sink
      transformers:
      publishers:
          - notifier://
```

🎓 Note

This change will cause all non-default meters to stop receiving notifications.

12.3.9.2 Enable Nova Notifications

You can configure Nova to send notifications by enabling the setting in the configuration file. When enabled, Nova will send information to Ceilometer related to its usage and VM status. You must restart Nova for these changes to take effect.

The Openstack notification daemon, also known as a polling agent, monitors the message bus for data being provided by other OpenStack components such as Nova. The notification daemon loads one or more listener plugins, using the `ceilometer.notification` namespace. Each plugin can listen to any topic, but by default it will listen to the `notifications.info` topic. The listeners grab messages off the defined topics and redistribute them to the appropriate plugins (endpoints) to be processed into Events and Samples. After the Nova service is restarted, you should verify that the notification daemons are receiving traffic.

For a more in-depth look at how information is sent over *openstack.common.rpc*, refer to the OpenStack Ceilometer documentation (http://docs.openstack.org/developer/ceilometer/measurements.html) ⬈.

Nova can be configured to send following data to Ceilometer:

Name	Unit	Type	Resource	Note
instance	g	instance	inst ID	Existence of instance

instance: type	g	instance	inst ID	Existence of instance of type (Where type is a valid OpenStack type.)
memory	g	MB	inst ID	Amount of allocated RAM. Measured in MB.
vcpus	g	vcpu	inst ID	Number of VCPUs
disk.root.size	g	GB	inst ID	Size of root disk. Measured in GB.
disk.ephemeral.size	g	GB	inst ID	Size of ephemeral disk. Measured in GB.

To enable Nova to publish notifications:

1. In a text editor, open the following file:

   ```
   nova.conf
   ```

2. Compare the example of a working configuration file with the necessary changes to your configuration file. If there is anything missing in your file, add it, and then save the file.

   ```
   notification_driver=messaging
   notification_topics=notifications
   notify_on_state_change=vm_and_task_state
   instance_usage_audit=True
   instance_usage_audit_period=hour
   ```

   🔔 Important

   The instance_usage_audit_period interval can be set to check the instance's status every hour, once a day, once a week or once a month. Every time the audit period elapses, Nova sends a notification to Ceilometer to record whether or not the instance is alive and running. Metering this statistic is critical if billing depends on usage.

3. To restart Nova service, run:

   ```
   tux > sudo systemctl restart nova-api.service
   tux > sudo systemctl restart nova-conductor.service
   tux > sudo systemctl restart nova-scheduler.service
   ```

```
tux > sudo systemctl restart nova-novncproxy.service
```

Important

Different platforms may use their own unique command to restart nova-compute services. If the above command does not work, please refer to the documentation for your specific platform.

4. To verify successful launch of each process, list the service components:

```
ardana > source ~/service.osrc
ardana > nova service-list
+----+-----------------+------------+----------+--------+-------
+--------------------------+----------------+
| Id | Binary          | Host       | Zone     | Status | State | Updated_at
          | Disabled Reason |
+----+-----------------+------------+----------+--------+-------
+--------------------------+----------------+
| 1  | nova-conductor  | controller | internal | enabled | up    |
 2014-09-16T23:54:02.000000 | -              |
| 2  | nova-consoleauth | controller | internal | enabled | up    |
 2014-09-16T23:54:04.000000 | -              |
| 3  | nova-scheduler  | controller | internal | enabled | up    |
 2014-09-16T23:54:07.000000 | -              |
| 4  | nova-cert       | controller | internal | enabled | up    |
 2014-09-16T23:54:00.000000 | -              |
| 5  | nova-compute    | compute1   | nova     | enabled | up    |
 2014-09-16T23:54:06.000000 | -              |
+----+-----------------+------------+----------+--------+-------
+--------------------------+----------------+
```

12.3.9.3 Improve Reporting API Responsiveness

Reporting APIs are the main access to the metering data stored in Ceilometer. These APIs are accessed by Horizon to provide basic usage data and information.

HPE Helion OpenStack uses Apache2 Web Server to provide the API access. This topic provides some strategies to help you optimize the front-end and back-end databases.

To improve the responsiveness you can increase the number of threads and processes in the ceilometer configuration file. The Ceilometer API runs as an WSGI processes. Each process can have a certain amount of threads managing the filters and applications, which can comprise the processing pipeline.

To configure Apache2 to use increase the number of threads, use the steps in *Section 12.3.5, "Configure the Ceilometer Metering Service"*

⛨ Warning

The resource usage panel could take some time to load depending on the number of metrics selected.

12.3.9.4   Update the Polling Strategy and Swift Considerations

Polling can put an excessive amount of strain on the system due to the amount of data the system may have to process. Polling also has a severe impact on queries since the database can have very large amount of data to scan before responding to the query. This process usually consumes a large amount of CPU and memory to complete the requests. Clients can also experience long waits for queries to come back and, in extreme cases, even timeout.

There are 3 polling meters in Swift:

- storage.objects

- storage.objects.size

- storage.objects.containers

Sample section of the pipeline.yaml configuration file with Swift polling on an hourly interval:

```
---
sources:
    - name: swift_source
      interval: 3600
      sources:
            meters:
          - "storage.objects"
          - "storage.objects.size"
          - "storage.objects.containers"
sinks:
    - name: meter_sink
      transformers:
      publishers:
          - notifier://
```

Every time the polling interval occurs, at least 3 messages per existing object/container in Swift are collected. The following table illustrates the amount of data produced hourly in different scenarios:

Swift Containers	Swift Objects per container	Samples per Hour	Samples stored per 24 hours
10	10	500	12000
10	100	5000	120000
100	100	50000	1200000
100	1000	500000	12000000

Looking at the data we can see that even a very small Swift storage with 10 containers and 100 files will store 120K samples in 24 hours, bringing it to a total of 3.6 million samples.

☞ Note

The file size of each file does not have any impact on the number of samples collected. In fact the smaller the number of containers or files, the smaller the sample size. In the scenario where there a large number of small files and containers, the sample size is also large and the performance is at its worst.

12.3.10 Metering Service Samples

Samples are discrete collections of a particular meter or the actual usage data defined by a meter description. Each sample is time-stamped and includes a variety of data that varies per meter but usually includes the project ID and UserID of the entity that consumed the resource represented by the meter and sample.

In a typical deployment, the number of samples can be in the tens of thousands if not higher for a specific collection period depending on overall activity.

Sample collection and data storage expiry settings are configured in Ceilometer. Use cases that include collecting data for monthly billing cycles are usually stored over a period of 45 days and require a large, scalable, back-end database to support the large volume of samples generated by production OpenStack deployments.

Example configuration:

```
[database]
metering_time_to_live=-1
```

In our example use case, to construct a complete billing record, an external billing application must collect all pertinent samples. Then the results must be sorted, summarized, and combine with the results of other types of metered samples that are required. This function is known as aggregation and is external to the Ceilometer service.

Meter data, or samples, can also be collected directly from the service APIs by individual Ceilometer polling agents. These polling agents directly access service usage by calling the API of each service.

OpenStack services such as Swift currently only provide metered data through this function and some of the other OpenStack services provide specific metrics only through a polling action.

13 System Maintenance

Information about managing and configuring your cloud as well as procedures for performing node maintenance.

This section contains the following sections to help you manage, configure, and maintain your HPE Helion OpenStack cloud.

13.1 Planned System Maintenance

Planned maintenance tasks for your cloud. See sections below for:

- *Section 13.1.2, "Planned Control Plane Maintenance"*

- *Section 13.1.3, "Planned Compute Maintenance"*

- *Section 13.1.4, "Planned Network Maintenance"*

- *Section 13.1.5, "Planned Storage Maintenance"*

- *Section 13.1.6, "Updating MariaDB with Galera"*

13.1.1 Whole Cloud Maintenance

Planned maintenance procedures for your whole cloud.

13.1.1.1 Bringing Down Your Cloud: Services Down Method

🔔 Important

If you have a planned maintenance and need to bring down your entire cloud, update and reboot all nodes in the cloud one by one. Start with the deployer node, then follow the order recommended in *Section 13.1.1.2, "Rolling Reboot of the Cloud"*. This method will bring down all of your services.

If you wish to use a method utilizing rolling reboots where your cloud services will continue running then see *Section 13.1.1.2, "Rolling Reboot of the Cloud"*.

To perform backups prior to these steps, visit the backup and restore pages first at *Chapter 14, Backup and Restore*.

13.1.1.1.1   Gracefully Bringing Down and Restarting Your Cloud Environment

You will do the following steps from your Cloud Lifecycle Manager.

1. Log in to your Cloud Lifecycle Manager.

2. Gracefully shut down your cloud by running the `ardana-stop.yml` playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts ardana-stop.yml
   ```

3. Shut down your nodes. You should shut down your controller nodes last and bring them up first after the maintenance.
   There are multiple ways you can do this:

   a. You can SSH to each node and use `sudo reboot -f` to reboot the node.

   b. From the Cloud Lifecycle Manager, you can use the `bm-power-down.yml` and `bm-power-up.yml` playbooks.

   c. You can shut down the nodes and then physically restart them either via a power button or the IPMI.

4. Perform the necessary maintenance.

5. After the maintenance is complete, power your Cloud Lifecycle Manager back up and then SSH to it.

6. Determine the current power status of the nodes in your environment:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts bm-power-status.yml
   ```

7. If necessary, power up any nodes that are not already powered up, ensuring that you power up your controller nodes first. You can target specific nodes with the `-e nodelist=<node_name>` switch.

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ```

```
ansible-playbook -i hosts/verb_hosts bm-power-up.yml [-e nodelist=<node_name>]
```

🎓 Note

Obtain the <node_name> by using the **`sudo cobbler system list`** command from the Cloud Lifecycle Manager.

8. Bring the databases back up:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts galera-bootstrap.yml
```

9. Gracefully bring up your cloud services by running the `ardana-start.yml` playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-start.yml
```

10. Pause for a few minutes and give the cloud environment time to come up completely and then verify the status of the individual services using this playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-status.yml
```

11. If any services did not start properly, you can run playbooks for the specific services having issues.

For example:

If RabbitMQ fails, run:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts rabbitmq-start.yml
```

You can check the status of RabbitMQ afterwards with this:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts rabbitmq-status.yml
```

If the recovery had failed, you can run:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts rabbitmq-disaster-recovery.yml
```

Each of the other services have playbooks in the `~/scratch/ansible/next/ardana/ansible` directory in the format of `<service>-start.yml` that you can run. One example, for the compute service, is `nova-start.yml`.

12. Continue checking the status of your HPE Helion OpenStack 8 cloud services until there are no more failed or unreachable nodes:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-status.yml
```

13.1.1.2 Rolling Reboot of the Cloud

If you have a planned maintenance and need to bring down your entire cloud and restart services while minimizing downtime, follow the steps here to safely restart your cloud. If you do not mind your services being down, then another option for planned maintenance can be found at *Section 13.1.1.1, "Bringing Down Your Cloud: Services Down Method"*.

13.1.1.2.1 Recommended node reboot order

To ensure that rebooted nodes reintegrate into the cluster, the key is having enough time between controller reboots.

The recommended way to achieve this is as follows:

1. Reboot controller nodes one-by-one with a suitable interval in between. If you alternate between controllers and compute nodes you will gain more time between the controller reboots.

2. Reboot of compute nodes (if present in your cloud).

3. Reboot of Swift nodes (if present in your cloud).

4. Reboot of ESX nodes (if present in your cloud).

13.1.1.2.2 Rebooting controller nodes

Turn off the Keystone Fernet Token-Signing Key Rotation

Before rebooting any controller node, you need to ensure that the Keystone Fernet token-signing key rotation is turned off. Run the following command:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts keystone-stop-fernet-auto-rotation.yml
```

Migrate singleton services first

☞ Note

> If you have previously rebooted your Cloud Lifecycle Manager for any reason, ensure that the `apache2` service is running before continuing. To start the `apache2` service, use this command:
>
> ```
> sudo systemctl start apache2
> ```

The first consideration before rebooting any controller nodes is that there are a few services that run as singletons (non-HA), thus they will be unavailable while the controller they run on is down. Typically this is a very small window, but if you want to retain the service during the reboot of that server you should take special action to maintain service, such as migrating the service.

For these steps, if your singleton services are running on controller1 and you move them to controller2, then ensure you move them back to controller1 before proceeding to reboot controller2.

For the `cinder-volume` **singleton service:**

Execute the following command on each controller node to determine which node is hosting the cinder-volume singleton. It should be running on only one node:

```
ps auxww | grep cinder-volume | grep -v grep
```

Run the `cinder-migrate-volume.yml` playbook - details about the Cinder volume and backup migration instructions can be found in *Section 7.1.3, "Managing Cinder Volume and Backup Services"*.

For the `nova-consoleauth` **singleton service:**

The `nova-consoleauth` component runs by default on the first controller node, that is, the host with `consoleauth_host_index=0`. To move it to another controller node before rebooting controller 0, run the ansible playbook `nova-start.yml` and pass it the index of the next controller node. For example, to move it to controller 2 (index of 1), run:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts nova-start.yml --extra-vars
 "consoleauth_host_index=1"
```

After you run this command you may now see two instances of the `nova-consoleauth` service, which will show as being in `disabled` state, when you run the `nova service-list` command. You can then delete the service using these steps.

1. Obtain the service ID for the duplicated nova-consoleauth service:

   ```
   nova service-list
   ```

   Example:

   ```
   $ nova service-list
   +----+-----------------+-------------------------+----------+----------+-------
   +--------------------------+-----------------+
   | Id | Binary          | Host                    | Zone     | Status   | State |
    Updated_at                | Disabled Reason |
   +----+-----------------+-------------------------+----------+----------+-------
   +--------------------------+-----------------+
   | 1  | nova-conductor  | ...a-cp1-c1-m1-mgmt     | internal | enabled  | up    |
    2016-08-25T12:11:48.000000 | -              |
   | 10 | nova-conductor  | ...a-cp1-c1-m3-mgmt     | internal | enabled  | up    |
    2016-08-25T12:11:47.000000 | -              |
   | 13 | nova-conductor  | ...a-cp1-c1-m2-mgmt     | internal | enabled  | up    |
    2016-08-25T12:11:48.000000 | -              |
   | 16 | nova-scheduler  | ...a-cp1-c1-m1-mgmt     | internal | enabled  | up    |
    2016-08-25T12:11:39.000000 | -              |
   | 19 | nova-scheduler  | ...a-cp1-c1-m2-mgmt     | internal | enabled  | up    |
    2016-08-25T12:11:41.000000 | -              |
   | 22 | nova-scheduler  | ...a-cp1-c1-m3-mgmt     | internal | enabled  | up    |
    2016-08-25T12:11:44.000000 | -              |
   | 25 | nova-consoleauth | ...a-cp1-c1-m1-mgmt    | internal | enabled  | up    |
    2016-08-25T12:11:45.000000 | -              |
   | 49 | nova-compute    | ...a-cp1-comp0001-mgmt | nova     | enabled  | up    |
    2016-08-25T12:11:48.000000 | -              |
   ```

```
| 52 | nova-compute     | ...a-cp1-comp0002-mgmt | nova     | enabled  | up    |
 2016-08-25T12:11:41.000000 | -               |
| 55 | nova-compute     | ...a-cp1-comp0003-mgmt | nova     | enabled  | up    |
 2016-08-25T12:11:43.000000 | -               |
| 70 | nova-consoleauth | ...a-cp1-c1-m3-mgmt    | internal | disabled | down  |
 2016-08-25T12:10:40.000000 | -               |
+----+----------------+--------------------------+----------+----------+-------
+--------------------------+----------------+
```

2. Delete the disabled duplicate service with this command:

```
nova service-delete <service_ID>
```

Given the example in the previous step, the command could be:

```
nova service-delete 70
```

For the SNAT namespace singleton service:

If you reboot the controller node hosting the SNAT namespace service on it, Compute instances without floating IPs will lose network connectivity when that controller is rebooted. To prevent this from happening, you can use these steps to determine which controller node is hosting the SNAT namespace service and migrate it to one of the other controller nodes while that node is rebooted.

1. Locate the SNAT node where the router is providing the active snat_service:

   a. From the Cloud Lifecycle Manager, list out your ports to determine which port is serving as the router gateway:

   ```
   source ~/service.osrc
   neutron port-list --device_owner network:router_gateway
   ```

   Example:

   ```
   $ neutron port-list --device_owner network:router_gateway
   +------------------------------------+------+------------------
   +-----------------------------------------------------------------------------
   +
   | id                                 | name | mac_address      | fixed_ips
                                        |
   +------------------------------------+------+------------------
   +-----------------------------------------------------------------------------
   +
   ```

```
| 287746e6-7d82-4b2c-914c-191954eba342 |        | fa:16:3e:2e:26:ac |
 {"subnet_id": "f4152001-2500-4ebe-ba9d-a8d6149a50df", "ip_address":
 "10.247.96.29"} |
+------------------------------------+------+------------------
+----------------------------------------------------------------------------------
+
```

b. Look at the details of this port to determine what the `binding:host_id` value is, which will point to the host in which the port is bound to:

```
neutron port-show <port_id>
```

Example, with the value you need in bold:

```
$ neutron port-show 287746e6-7d82-4b2c-914c-191954eba342
+----------------------
+----------------------------------------------------------------------------------
+
| Field                 | Value
                                                                               |
+----------------------
+----------------------------------------------------------------------------------
+
| admin_state_up        | True
                                                                               |
| allowed_address_pairs |
                                                                               |
| binding:host_id       | ardana-cp1-c1-m2-mgmt
                                                                               |
| binding:profile       | {}
                                                                               |
| binding:vif_details   | {"port_filter": true, "ovs_hybrid_plug": true}
                                                                               |
| binding:vif_type      | ovs
                                                                               |
| binding:vnic_type     | normal
                                                                               |
| device_id             | e122ea3f-90c5-4662-bf4a-3889f677aacf
                                                                               |
| device_owner          | network:router_gateway
                                                                               |
| dns_assignment        | {"hostname": "host-10-247-96-29", "ip_address":
 "10.247.96.29", "fqdn": "host-10-247-96-29.openstacklocal."} |
| dns_name              |
                                                                               |
```

```
| extra_dhcp_opts        |
                                                        |
| fixed_ips              | {"subnet_id": "f4152001-2500-4ebe-ba9d-
a8d6149a50df", "ip_address": "10.247.96.29"}                        |
| id                     | 287746e6-7d82-4b2c-914c-191954eba342
                                                        |
| mac_address            | fa:16:3e:2e:26:ac
                                                        |
| name                   |
                                                        |
| network_id             | d3cb12a6-a000-4e3e-82c4-ee04aa169291
                                                        |
| security_groups        |
                                                        |
| status                 | DOWN
                                                        |
| tenant_id              |
                                                        |
+----------------------
+--------------------------------------------------------------------------
+
```

In this example, the `ardana-cp1-c1-m2-mgmt` is the node hosting the SNAT namespace service.

2. SSH to the node hosting the SNAT namespace service and check the SNAT namespace, specifying the router_id that has the interface with the subnet that you are interested in:

```
ssh <IP_of_SNAT_namespace_host>
sudo ip netns exec snat-<router_ID> bash
```

Example:

```
sudo ip netns exec snat-e122ea3f-90c5-4662-bf4a-3889f677aacf bash
```

3. Obtain the ID for the L3 Agent for the node hosting your SNAT namespace:

```
source ~/service.osrc
neutron agent-list
```

Example, with the entry you need given the examples above:

```
$ neutron agent-list
+------------------------------------+----------------------
+-------------------------+-------+---------------+--------------------------+
```

```
| id                                   | agent_type           | host
      | alive | admin_state_up | binary                      |
+--------------------------------------+----------------------
+--------------------------+-------+----------------+--------------------------+
| 0126bbbf-5758-4fd0-84a8-7af4d93614b8 | DHCP agent           | ardana-cp1-c1-m3-
mgmt    | :-)   | True           | neutron-dhcp-agent          |
| 33dec174-3602-41d5-b7f8-a25fd8ff6341 | Metadata agent       | ardana-cp1-c1-m2-
mgmt    | :-)   | True           | neutron-metadata-agent      |
| 3bc28451-c895-437b-999d-fdcff259b016 | L3 agent             | ardana-cp1-c1-m1-
mgmt    | :-)   | True           | neutron-vpn-agent           |
| 4af1a941-61c1-4e74-9ec1-961cebd6097b | L3 agent             | ardana-cp1-comp0001-
mgmt | :-)   | True           | neutron-l3-agent            |
| 58f01f34-b6ca-4186-ac38-b56ee376ffeb | Loadbalancerv2 agent | ardana-cp1-comp0001-
mgmt | :-)   | True           | neutron-lbaasv2-agent       |
| 65bcb3a0-4039-4d9d-911c-5bb790953297 | Open vSwitch agent   | ardana-cp1-c1-m1-
mgmt    | :-)   | True           | neutron-openvswitch-agent   |
| 6981c0e5-5314-4ccd-bbad-98ace7db7784 | L3 agent             | ardana-cp1-c1-m3-
mgmt    | :-)   | True           | neutron-vpn-agent           |
| 7df9fa0b-5f41-411f-a532-591e6db04ff1 | Metadata agent       | ardana-cp1-comp0001-
mgmt | :-)   | True           | neutron-metadata-agent      |
| 92880ab4-b47c-436c-976a-a605daa8779a | Metadata agent       | ardana-cp1-c1-m1-
mgmt    | :-)   | True           | neutron-metadata-agent      |
| a209c67d-c00f-4a00-b31c-0db30e9ec661 | L3 agent             | ardana-cp1-c1-m2-
mgmt    | :-)   | True           | neutron-vpn-agent           |
| a9467f7e-ec62-4134-826f-366292c1f2d0 | DHCP agent           | ardana-cp1-c1-m1-
mgmt    | :-)   | True           | neutron-dhcp-agent          |
| b13350df-f61d-40ec-b0a3-c7c647e60f75 | Open vSwitch agent   | ardana-cp1-c1-m2-
mgmt    | :-)   | True           | neutron-openvswitch-agent   |
| d4c07683-e8b0-4a2b-9d31-b5b0107b0b31 | Open vSwitch agent   | ardana-cp1-comp0001-
mgmt | :-)   | True           | neutron-openvswitch-agent   |
| e91d7f3f-147f-4ad2-8751-837b936801e3 | Open vSwitch agent   | ardana-cp1-c1-m3-
mgmt    | :-)   | True           | neutron-openvswitch-agent   |
| f33015c8-f4e4-4505-b19b-5a1915b6e22a | DHCP agent           | ardana-cp1-c1-m2-
mgmt    | :-)   | True           | neutron-dhcp-agent          |
| fe43c0e9-f1db-4b67-a474-77936f7acebf | Metadata agent       | ardana-cp1-c1-m3-
mgmt    | :-)   | True           | neutron-metadata-agent      |
+--------------------------------------+----------------------
+--------------------------+-------+----------------+--------------------------+
```

4. Also obtain the ID for the L3 Agent of the node you are going to move the SNAT namespace service to using the same commands as the previous step.

5. Use these commands to move the SNAT namespace service, with the `router_id` being the same value as the ID for router:

   a. Remove the L3 Agent for the old host:

      ```
      neutron l3-agent-router-remove <agent_id_of_snat_namespace_host>
       <qrouter_uuid>
      ```

      Example:

      ```
      $ neutron l3-agent-router-remove a209c67d-c00f-4a00-b31c-0db30e9ec661
       e122ea3f-90c5-4662-bf4a-3889f677aacf
      Removed router e122ea3f-90c5-4662-bf4a-3889f677aacf from L3 agent
      ```

   b. Remove the SNAT namespace:

      ```
      sudo ip netns delete snat-<router_id>
      ```

      Example:

      ```
      $ sudo ip netns delete snat-e122ea3f-90c5-4662-bf4a-3889f677aacf
      ```

   c. Create a new L3 Agent for the new host:

      ```
      neutron l3-agent-router-add <agent_id_of_new_snat_namespace_host>
       <qrouter_uuid>
      ```

      Example:

      ```
      $ neutron l3-agent-router-add 3bc28451-c895-437b-999d-fdcff259b016
       e122ea3f-90c5-4662-bf4a-3889f677aacf
      Added router e122ea3f-90c5-4662-bf4a-3889f677aacf to L3 agent
      ```

   Confirm that it has been moved by listing the details of your port from step 1b above, noting the value of `binding:host_id` which should be updated to the host you moved your SNAT namespace to:

   ```
   neutron port-show <port_ID>
   ```

   Example:

   ```
   $ neutron port-show 287746e6-7d82-4b2c-914c-191954eba342
   ```

```
+----------------------
+---------------------------------------------------------------------------------------
+
| Field                 | Value
                                                    |
+----------------------
+---------------------------------------------------------------------------------------
+
| admin_state_up        | True
                                                    |
| allowed_address_pairs |
                                                    |
| binding:host_id       | ardana-cp1-c1-m1-mgmt
                                                    |
| binding:profile       | {}
                                                    |
| binding:vif_details   | {"port_filter": true, "ovs_hybrid_plug": true}
                                                    |
| binding:vif_type      | ovs
                                                    |
| binding:vnic_type     | normal
                                                    |
| device_id             | e122ea3f-90c5-4662-bf4a-3889f677aacf
                                                    |
| device_owner          | network:router_gateway
                                                    |
| dns_assignment        | {"hostname": "host-10-247-96-29", "ip_address":
 "10.247.96.29", "fqdn": "host-10-247-96-29.openstacklocal."} |
| dns_name              |
                                                    |
| extra_dhcp_opts       |
                                                    |
| fixed_ips             | {"subnet_id": "f4152001-2500-4ebe-ba9d-a8d6149a50df",
 "ip_address": "10.247.96.29"}                              |
| id                    | 287746e6-7d82-4b2c-914c-191954eba342
                                                    |
| mac_address           | fa:16:3e:2e:26:ac
                                                    |
| name                  |
                                                    |
| network_id            | d3cb12a6-a000-4e3e-82c4-ee04aa169291
                                                    |
| security_groups       |
                                                    |
| status                | DOWN
                                                    |
```

```
| tenant_id            |
                              |
+----------------------
+----------------------------------------------------------------------------------
+
```

Reboot the controllers

In order to reboot the controller nodes, you must first retrieve a list of nodes in your cloud running control plane services.

```
for i in $(grep -w cluster-prefix ~/openstack/my_cloud/definition/data/control_plane.yml
 | awk '{print $2}'); do grep $i ~/scratch/ansible/next/ardana/ansible/hosts/verb_hosts |
 grep ansible_ssh_host | awk '{print $1}'; done
```

Then perform the following steps from your Cloud Lifecycle Manager for each of your controller nodes:

1. If any singleton services are active on this node, they will be unavailable while the node is down. If you want to retain the service during the reboot, you should take special action to maintain the service, such as migrating the service as appropriate as noted above.

2. Stop all services on the controller node that you are rebooting first:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts ardana-stop.yml --limit <controller node>
   ```

3. Reboot the controller node, e.g. run the following command on the controller itself:

   ```
   sudo reboot
   ```

   Note that the current node being rebooted could be hosting the lifecycle manager.

4. Wait for the controller node to become ssh-able and allow an additional minimum of five minutes for the controller node to settle. Start all services on the controller node:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit <controller node>
   ```

5. Verify that the status of all services on that is OK on the controller node:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts ardana-status.yml --limit <controller node>
   ```

6. When above start operation has completed successfully, you may proceed to the next controller node. Ensure that you migrate your singleton services off the node first.

☞ Note

It is important that you not begin the reboot procedure for a new controller node until the reboot of the previous controller node has been completed successfully (that is, the ardana-status playbook has completed without error).

Reenable the Keystone Fernet Token-Signing Key Rotation

After all the controller nodes are successfully updated and back online, you need to re-enable the Keystone Fernet token-signing key rotation job by running the `keystone-reconfigure.yml` playbook. On the deployer, run:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

13.1.1.2.3   Rebooting compute nodes

To reboot a compute node the following operations will need to be performed:

- Disable provisioning of the node to take the node offline to prevent further instances being scheduled to the node during the reboot.

- Identify instances that exist on the compute node, and then either:

    - Live migrate the instances off the node before actioning the reboot. OR

    - Stop the instances

- Reboot the node

- Restart the Nova services

1. Disable provisioning:

```
nova service-disable --reason "<describe reason>" <node name> nova-compute
```

If the node has existing instances running on it these instances will need to be migrated or stopped prior to re-booting the node.

2. Live migrate existing instances. Identify the instances on the compute node. Note: The following command must be run with nova admin credentials.

```
nova list --host <hostname> --all-tenants
```

3. Migrate or Stop the instances on the compute node.

Migrate the instances off the node by running one of the following commands for each of the instances:

If your instance is booted from a volume and has any number of Cinder volume attached, use the nova live-migration command:

```
nova live-migration <instance uuid> [<target compute host>]
```

If your instance has local (ephemeral) disk(s) only, you can use the --block-migrate option:

```
nova live-migration --block-migrate <instance uuid> [<target compute host>]
```

Note: The [<target compute host>] option is optional. If you do not specify a target host then the nova scheduler will choose a node for you.

OR

Stop the instances on the node by running the following command for each of the instances:

```
nova stop <instance-uuid>
```

4. Stop all services on the Compute node:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-stop.yml --limit <compute node>
```

5. SSH to your Compute nodes and reboot them:

```
sudo reboot
```

The operating system cleanly shuts down services and then automatically reboots. If you want to be very thorough, run your backup jobs just before you reboot.

6. Run the ardana-start.yml playbook from the Cloud Lifecycle Manager. If needed, use the bm-power-up.yml playbook to restart the node. Specify just the node(s) you want to start in the 'nodelist' parameter arguments, that is, nodelist = <node1>[,<node2>] [,<node3>].

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-power-up.yml -e nodelist=<compute node>
```

7. Execute the **ardana-start.yml** playbook. Specifying the node(s) you want to start in the 'limit' parameter arguments. This parameter accepts wildcard arguments and also '@<filename>' to process all hosts listed in the file.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit <compute node>
```

8. Re-enable provisioning on the node:

```
nova service-enable <node-name> nova-compute
```

9. Restart any instances you stopped.

```
nova start <instance-uuid>
```

13.1.1.2.4   Rebooting Swift nodes

If your Swift services are on controller node, please follow the controller node reboot instructions above.

For a dedicated Swift PAC cluster or Swift Object resource node:

For each Swift host

1. Stop all services on the Swift node:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-stop.yml --limit <Swift node>
```

2. Reboot the Swift node by running the following command on the Swift node itself:

```
sudo reboot
```

3. Wait for the node to become ssh-able and then start all services on the Swift node:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit <swift node>
```

13.1.1.2.5   Get list of status playbooks

Running the following command will yield a list of status playbooks:

```
cd ~/scratch/ansible/next/ardana/ansible
ls *status*
```

Here is the list:

```
ls *status*
bm-power-status.yml          heat-status.yml      logging-producer-status.yml
ceilometer-status.yml        FND-AP2-status.yml   ardana-status.yml
FND-CLU-status.yml           horizon-status.yml   logging-status.yml
cinder-status.yml            freezer-status.yml   ironic-status.yml
cmc-status.yml               glance-status.yml    keystone-status.yml
galera-status.yml            memcached-status.yml nova-status.yml
logging-server-status.yml    monasca-status.yml   ops-console-status.yml
monasca-agent-status.yml     neutron-status.yml   rabbitmq-status.yml
swift-status.yml             zookeeper-status.yml
```

13.1.2   Planned Control Plane Maintenance

Planned maintenance tasks for controller nodes such as full cloud reboots and replacing controller nodes.

13.1.2.1   Replacing a Controller Node

This section outlines steps for replacing a controller node in your environment.

For HPE Helion OpenStack, you must have three controller nodes. Therefore, adding or removing nodes is not an option. However, if you need to repair or replace a controller node, you may do so by following the steps outlined here. Note that to run any playbooks whatsoever for cloud maintenance, you will always run the steps from the Cloud Lifecycle Manager.

These steps will depend on whether you need to replace a shared lifecycle manager/controller node or whether this is a standalone controller node.

Keep in mind while performing the following tasks:

- Do not add entries for a new server. Instead, update the entries for the broken one.

- Be aware that all management commands are run on the node where the Cloud Lifecycle Manager is running.

13.1.2.1.1 Replacing a Shared Cloud Lifecycle Manager/Controller Node

If the controller node you need to replace was also being used as your Cloud Lifecycle Manager then use these steps below. If this is not a shared controller then skip to the next section.

1. To ensure that you use the same version of HPE Helion OpenStack that you previously had loaded on your Cloud Lifecycle Manager, you will need to download and install the lifecycle management software using the instructions from the installation guide:

    1. *Book "*Installing with Cloud Lifecycle Manager*", Chapter 3 "Installing the Cloud Lifecycle Manager server", Section 3.5.2 "Installing the HPE Helion OpenStack Extension"*

    2. To restore your data, see *Section 13.2.2.2.3, "Point-in-time Cloud Lifecycle Manager Recovery"*

2. On the new node, update your cloud model with the new `mac-addr`, `ilo-ip`, `ilo-password`, and `ilo-user` fields to reflect the attributes of the node. Do not change the `id`, `ip-addr`, `role`, or `server-group` settings.

   > ☞ **Note**
   >
   > When imaging servers with your own tooling, it is still necessary to have ILO/IPMI settings for all nodes. Even if you are not using Cobbler, the username and password fields in `servers.yml` need to be filled in with dummy settings. For example, add the following to `servers.yml`:
   >
   > ```
   > ilo-user: manual
   > ilo-password: deployment
   > ```

3. Get the **servers.yml** file stored in git:

   ```
   cd ~/openstack/my_cloud/definition/data
   git checkout site
   ```

   then change, as necessary, the `mac-addr`, `ilo-ip`, `ilo-password`, and `ilo-user` fields of this existing controller node. Save and commit the change

   ```
   git commit -a -m "repaired node X"
   ```

4. Run the configuration processor as follows:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

Then run ready-deployment:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

5. Deploy Cobbler:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

🎓 Note

After this step you may see failures because MariaDB has not finished syncing. If so, rerun this step.

6. Delete the haproxy user:

```
sudo userdel haproxy
```

7. Install the software on your new Cloud Lifecycle Manager/controller node with these three playbooks:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-rebuild-pretasks.yml
ansible-playbook -i hosts/verb_hosts osconfig-run.yml -e rebuild=True --
limit=<controller-hostname>
ansible-playbook -i hosts/verb_hosts ardana-deploy.yml -e rebuild=True --
limit=<controller-hostname>,<first-proxy-hostname>
```

8. During the replacement of the node there will be alarms that show up during the process. If those do not clear after the node is back up and healthy, restart the threshold engine by running the following playbooks:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-stop.yml --tags thresh
ansible-playbook -i hosts/verb_hosts monasca-start.yml --tags thresh
```

13.1.2.1.2 Replacing a Standalone Controller Node

If the controller node you need to replace is not also being used as the Cloud Lifecycle Manager, follow the steps below.

1. Log in to the Cloud Lifecycle Manager.

2. Update your cloud model, specifically the `servers.yml` file, with the new `mac-addr`, `ilo-ip`, `ilo-password`, and `ilo-user` fields where these have changed. Do not change the `id`, `ip-addr`, `role`, or `server-group` settings.

3. Commit your configuration to the *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*, as follows:

   ```
   cd ~/openstack/ardana/ansible
   git add -A
   git commit -m "My config or other commit message"
   ```

4. Run the configuration processor:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

5. Update your deployment directory:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

6. Remove the old controller node(s) from Cobbler. You can list out the systems in Cobbler currently with this command:

   ```
   sudo cobbler system list
   ```

   and then remove the old controller nodes with this command:

   ```
   sudo cobbler system remove --name <node>
   ```

7. Remove the SSH key of the old controller node from the known hosts file. You will specify the `ip-addr` value:

   ```
   ssh-keygen -f "~/.ssh/known_hosts" -R <ip_addr>
   ```

You should see a response similar to this one:

```
ardana@ardana-cp1-c1-m1-mgmt:~/openstack/ardana/ansible$ ssh-keygen -f "~/.ssh/
known_hosts" -R 10.13.111.135
# Host 10.13.111.135 found: line 6 type ECDSA
~/.ssh/known_hosts updated.
Original contents retained as ~/.ssh/known_hosts.old
```

8. Run the cobbler-deploy playbook to add the new controller node:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

9. Image the new node(s) by using the bm-reimage playbook. You will specify the name for the node in Cobbler in the command:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<node-name>
```

🔔 Important

You must ensure that the old controller node is powered off before completing this step. This is because the new controller node will re-use the original IP address.

10. Configure the necessary keys used for the database etc:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-rebuild-pretasks.yml
```

11. Run osconfig on the replacement controller node. For example:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts osconfig-run.yml -e rebuild=True --
limit=<controller-hostname>
```

12. If the controller being replaced is the Swift ring builder (see *Section 15.6.2.4, "Identifying the Swift Ring Building Server"*) you need to restore the Swift ring builder files to the `/etc/swiftlm/CLOUD-NAME/CONTROL-PLANE-NAME/builder_dir` directory. See *Section 15.6.2.7, "Recovering Swift Builder Files"* for details.

13. Run the ardana-deploy playbook on the replacement controller.

If the node being replaced is the Swift ring builder server then you only need to use the `--limit` switch for that node, otherwise you need to specify the hostname of your Swift ringer builder server and the hostname of the node being replaced.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-deploy.yml -e rebuild=True
--limit=<controller-hostname>,<swift-ring-builder-hostname>
```

🔔 Important

If you receive a Keystone failure when running this playbook, it is likely due to Fernet keys being out of sync. This problem can be corrected by running the `keystone-reconfigure.yml` playbook to re-sync the Fernet keys.

In this situation, do not use the `--limit` option when running `keystone-reconfigure.yml`. In order to re-sync Fernet keys, all the controller nodes must be in the play.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

🔔 Important

If you receive a RabbitMQ failure when running this playbook, review *Section 15.2.1, "Understanding and Recovering RabbitMQ after Failure"* for how to resolve the issue and then re-run the ardana-deploy playbook.

14. During the replacement of the node there will be alarms that show up during the process. If those do not clear after the node is back up and healthy, restart the threshold engine by running the following playbooks:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-stop.yml --tags thresh
ansible-playbook -i hosts/verb_hosts monasca-start.yml --tags thresh
```

13.1.3 Planned Compute Maintenance

Planned maintenance tasks for compute nodes.

13.1.3.1 Planned Maintenance for a Compute Node

If one or more of your compute nodes needs hardware maintenance and you can schedule a planned maintenance then this procedure should be followed.

13.1.3.1.1 Performing planned maintenance on a compute node

If you have planned maintenance to perform on a compute node, you have to take it offline, repair it, and restart it. To do so, follow these steps:

1. Log in to the Cloud Lifecycle Manager.

2. Source the administrator credentials:

   ```
   source ~/service.osrc
   ```

3. Obtain the hostname for your compute node, which you will use in subsequent commands when `<hostname>` is requested:

   ```
   nova host-list | grep compute
   ```

   The following example shows two compute nodes:

   ```
   $ nova host-list | grep compute
   | ardana-cp1-comp0001-mgmt | compute      | AZ1      |
   | ardana-cp1-comp0002-mgmt | compute      | AZ2      |
   ```

4. Disable provisioning on the compute node, which will prevent additional instances from being spawned on it:

   ```
   nova service-disable --reason "Maintenance mode" <hostname> nova-compute
   ```

   > **Note**
   >
   > Make sure you re-enable provisioning after the maintenance is complete if you want to continue to be able to spawn instances on the node. You can do this with the command:
   >
   > ```
   > nova service-enable <hostname> nova-compute
   > ```

5. At this point you have two choices:

   1. **Live migration**: This option enables you to migrate the instances off the compute node with minimal downtime so you can perform the maintenance without risk of losing data.

   2. **Stop/start the instances**: Issuing `nova stop` commands to each of the instances will halt them. This option lets you do maintenance and then start the instances back up, as long as no disk failures occur on the compute node data disks. This method involves downtime for the length of the maintenance.

   If you choose the live migration route, See *Section 13.1.3.3, "Live Migration of Instances"* for more details. Skip to step #6 after you finish live migration.
   If you choose the stop start method, continue on.

   1. List all of the instances on the node so you can issue stop commands to them:

      ```
      nova list --host <hostname> --all-tenants
      ```

   2. Issue the `nova stop` command against each of the instances:

      ```
      nova stop <instance uuid>
      ```

   3. Confirm that the instances are stopped. If stoppage was successful you should see the instances in a `SHUTOFF` state, as shown here:

      ```
      $ nova list --host ardana-cp1-comp0002-mgmt --all-tenants
      +------------------------------------+-----------
      +--------------------------------+---------+------------+-------------
      +---------------------+
      | ID                                 | Name      | Tenant ID
              | Status  | Task State | Power State | Networks            |
      +------------------------------------+-----------
      +--------------------------------+---------+------------+-------------
      +---------------------+
      | ef31c453-f046-4355-9bd3-11e774b1772f | instance1 |
       4365472e025c407c8d751fc578b7e368 | SHUTOFF | -          | Shutdown    |
       demo_network=10.0.0.5 |
      +------------------------------------+-----------
      +--------------------------------+---------+------------+-------------
      +---------------------+
      ```

4. Do your required maintenance. If this maintenance does not take down the disks completely then you should be able to list the instances again after the repair and confirm that they are still in their `SHUTOFF` state:

```
nova list --host <hostname> --all-tenants
```

5. Start the instances back up using this command:

```
nova start <instance uuid>
```

Example:

```
$ nova start ef31c453-f046-4355-9bd3-11e774b1772f
Request to start server ef31c453-f046-4355-9bd3-11e774b1772f has been
 accepted.
```

6. Confirm that the instances started back up. If restarting is successful you should see the instances in an `ACTIVE` state, as shown here:

```
$ nova list --host ardana-cp1-comp0002-mgmt --all-tenants
+-------------------------------------+-----------
+--------------------------------+--------+------------+-------------
+----------------------+
| ID                                  | Name      | Tenant ID
       | Status | Task State | Power State | Networks              |
+-------------------------------------+-----------
+--------------------------------+--------+------------+-------------
+----------------------+
| ef31c453-f046-4355-9bd3-11e774b1772f | instance1 |
 4365472e025c407c8d751fc578b7e368 | ACTIVE | -           | Running      |
 demo_network=10.0.0.5 |
+-------------------------------------+-----------
+--------------------------------+--------+------------+-------------
+----------------------+
```

7. If the `nova start` fails, you can try doing a hard reboot:

```
nova reboot --hard <instance uuid>
```

If this does not resolve the issue you may want to contact support.

6. Reenable provisioning when the node is fixed:

```
nova service-enable <hostname> nova-compute
```

13.1.3.2   Rebooting a Compute Node

If all you need to do is reboot a Compute node, the following steps can be used.

You can choose to live migrate all Compute instances off the node prior to the reboot. Any instances that remain will be restarted when the node is rebooted. This playbook will ensure that all services on the Compute node are restarted properly.

1. Log in to the Cloud Lifecycle Manager.

2. Reboot the Compute node(s) with the following playbook.
   You can specify either single or multiple Compute nodes using the `--limit` switch.
   An optional reboot wait time can also be specified. If no reboot wait time is specified it will default to 300 seconds.

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts nova-compute-reboot.yml --limit
     [compute_node_or_list] [-e nova_reboot_wait_timeout=(seconds)]
   ```

☞ Note

If the Compute node fails to reboot, you should troubleshoot this issue separately as this playbook will not attempt to recover after a failed reboot.

13.1.3.3   Live Migration of Instances

Live migration allows you to move active compute instances between compute nodes, allowing for less downtime during maintenance.

HPE Helion OpenStack Nova offers a set of commands that allow you to move compute instances between compute hosts. Which command you use will depend on the state of the host, what operating system is on the host, what type of storage the instances are using, and whether you want to migrate a single instance or all of the instances off of the host. We will describe these options on this page as well as give you step-by-step instructions for performing them.

13.1.3.3.1   Migration Options

If your compute node has failed

A compute host failure could be caused by hardware failure, such as the data disk needing to be replaced, power has been lost, or any other type of failure which requires that you replace the baremetal host. In this scenario, the instances on the compute node are unrecoverable and any data on the local ephemeral storage is lost. If you are utilizing block storage volumes, either as a boot device or as additional storage, they should be unaffected.

In these cases you will want to use one of the Nova evacuate commands, which will cause Nova to rebuild the instances on other hosts.

This table describes each of the evacuate options for failed compute nodes:

Command	Description
`nova evacuate <instance> <hostname>`	This command is used to evacuate a single instance from a failed host. You specify the compute instance UUID and the target host you want to evacuate it to. If no host is specified then the Nova scheduler will choose one for you.
	See `nova help evacuate` for more information and syntax. Further details can also be seen in the OpenStack documentation at http://docs.openstack.org/admin-guide/cli_nova_evacuate.html ↗.
`nova host-evacuate <hostname> --target_host <target_hostname>`	This command is used to evacuate all instances from a failed host. You specify the hostname of the compute host you want to evacuate. Optionally you can specify a target host. If no target host is specified then the Nova scheduler will choose a target host for each instance.
	See `nova help host-evacuate` for more information and syntax.

If your compute host is active, powered on and the data disks are in working order you can utilize the migration commands to migrate your compute instances. There are two migration features, "cold" migration (also referred to simply as "migration") and live migration. Migration and live migration are two different functions.

Cold migration is used to copy an instances data in a `SHUTOFF` status from one compute host to another. It does this using passwordless SSH access which has security concerns associated with it. For this reason, the `nova migrate` function has been disabled by default but you have the ability to enable this feature if you would like. Details on how to do this can be found in *Section 5.4, "Enabling the Nova Resize and Migrate Features"*.

Live migration can be performed on instances in either an `ACTIVE` or `PAUSED` state and uses the QEMU hypervisor to manage the copy of the running processes and associated resources to the destination compute host using the hypervisors own protocol and thus is a more secure method and allows for less downtime. There may be a short network outage, usually a few milliseconds but could be up to a few seconds if your compute instances are busy, during a live migration. Also there may be some performance degredation during the process.

The compute host must remain powered on during the migration process.

Both the cold migration and live migration options will honor Nova group policies, which includes affinity settings. There is a limitation to keep in mind if you use group policies and that is discussed in the *Section 13.1.3.3, "Live Migration of Instances"* section.

This table describes each of the migration options for active compute nodes:

Command	Description	SLES
`nova migrate <instance_uuid>`	Used to cold migrate a single instance from a compute host. The `nova-scheduler` will choose the new host. This command will work against instances in an `ACTIVE` or `SHUTOFF` state. The instances, if active, will be shutdown and restarted. Instances in a `PAUSED` state cannot be cold migrated. See the difference between cold migration and live migration at the start of this section.	

Command	Description	SLES
`nova host-servers-migrate <hostname>`	Used to cold migrate all instances off a specified host to other available hosts, chosen by the `nova-scheduler`. This command will work against instances in an `ACTIVE` or `SHUTOFF` state. The instances, if active, will be shutdown and restarted. Instances in a `PAUSED` state cannot be cold migrated. See the difference between cold migration and live migration at the start of this section.	
`nova live-migration <instance_uuid> [<target host>]`	Used to migrate a single instance between two compute hosts. You can optionally specify a target host or you can allow the nova scheduler to choose a host for you. If you choose to specify a target host, ensure that the target host has enough resources to host the instance prior to live migration. Works for instances that are booted from a block storage volume or that have any number of block storage volumes attached. This command works against instances in `ACTIVE` or `PAUSED` states only.	X
`nova live-migration --block-migrate <instance_uuid> [<target host>]`	Used to migrate a single instance between two compute hosts. You can optionally specify a target host or you can allow the nova scheduler to choose a host for you. If you choose to specify	X

Command	Description	SLES
	a target host, ensure that the target host has enough resources to host the instance prior to live migration. Works for instances that are booted from local (ephemeral) disk(s) only or if your instance has a mix of ephemeral disk(s) and block storage volume(s) but are not booted from a block storage volume. This command works against instances in `ACTIVE` or `PAUSED` states only.	
`nova host-evacuate-live <hostname> [--target-host <target_hostname>]`	Used to live migrate all instances off of a compute host. You can optionally specify a target host or you can allow the nova scheduler to choose a host for you. If you choose to specify a target host, ensure that the target host has enough resources to host the instance prior to live migration. Works for instances that are booted from a block storage volume or that have any number of block storage volumes attached. This command works against instances in `ACTIVE` or `PAUSED` states only.	X
`nova host-evacuate-live --block-migrate <hostname> [--target-host <target_hostname>]`	Used to live migrate all instances off of a compute host. You can optionally specify a target host or you can allow the nova scheduler to choose a host for you. If you choose to specify a target	X

Command	Description	SLES
	host, ensure that the target host has enough resources to host the instance prior to live migration. Works for instances that are booted from local (ephemeral) disk(s) only or if your instance has a mix of ephemeral disk(s) and block storage volume(s) but are not booted from a block storage volume. This command works against instances in `ACTIVE` or `PAUSED` states only.	

13.1.3.3.2 Limitations of these Features

There are limitations that may impact your use of this feature:

- To use live migration, your compute instances must be in either an `ACTIVE` or `PAUSED` state on the compute host. If you have instances in a `SHUTOFF` state then cold migration should be used.

- Instances in a `Paused` state cannot be live migrated using the Horizon dashboard. You will need to utilize the NovaClient CLI to perform these.

- Both cold migration and live migration honor an instance's group policies. If you are utilizing an affinity policy and are migrating multiple instances you may run into an error stating no hosts are available to migrate to. To work around this issue you should specify a target host when migrating these instances, which will bypass the `nova-scheduler`. You should ensure that the target host you choose has the resources available to host the instances.

- The `nova host-evacuate-live` command will produce an error if you have a compute host that has a mix of instances that use local ephemeral storage and instances that are booted from a block storage volume or have any number of block storage volumes attached. If you have a mix of these instance types, you may need to run the command twice, utilizing the `--block-migrate` option. This is described in further detail in *Section 13.1.3.3, "Live Migration of Instances"*.

- Instances on KVM hosts can only be live migrated to other KVM hosts.

- If you are using both Linux for HPE Helion (KVM) and SLES compute hosts, you cannot live migrate instances between them. Instances on KVM hosts can only be live migrated to other KVM hosts. Instances on SLES hosts can only be live migrated to other SLES hosts.

- The migration options described in this document are not available on ESX compute hosts.

- Ensure that you read and take into account any other limitations that exist in the release notes. See the release notes for more details.

13.1.3.3.3 Performing a Live Migration

Cloud administrators can perform a migration on an instance using either the Horizon dashboard, API, or CLI. Instances in a `Paused` state cannot be live migrated using the Horizon GUI. You will need to utilize the CLI to perform these.

We have documented different scenarios:

13.1.3.3.4 Migrating instances off of a failed compute host

1. Log in to the Cloud Lifecycle Manager.

2. If the compute node is not already powered off, do so with this playbook:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-power-down.yml -e nodelist=<node_name>
```

> 🎓 **Note**
>
> The value for `<node_name>` will be the name that Cobbler has when you run **sudo cobbler system list** from the Cloud Lifecycle Manager.

3. Source the admin credentials necessary to run administrative commands against the Nova API:

```
source ~/service.osrc
```

4. Force the `nova-compute` service to go down on the compute node:

```
nova service-force-down HOSTNAME nova-compute
```

> **☞ Note**
>
> The value for *HOSTNAME* can be obtained by using **nova host-list** from the Cloud Lifecycle Manager.

5. Evacuate the instances off of the failed compute node. This will cause the nova-scheduler to rebuild the instances on other valid hosts. Any local ephemeral data on the instances is lost.

   For single instances on a failed host:

   ```
   nova evacuate <instance_uuid> <target_hostname>
   ```

   For all instances on a failed host:

   ```
   nova host-evacuate <hostname> [--target_host <target_hostname>]
   ```

6. When you have repaired the failed node and start it back up again, when the **nova-compute** process starts again, it will clean up the evacuated instances.

13.1.3.3.5   Migrating instances off of an active compute host

Migrating instances using the Horizon dashboard

The Horizon dashboard offers a GUI method for performing live migrations. Instances in a `Paused` state will not provide you the live migration option in Horizon so you will need to use the CLI instructions in the next section to perform these.

1. Log into the Horizon dashboard with admin credentials.

2. Navigate to the menu *Admin › Compute › Instances*.

3. Next to the instance you want to migrate, select the drop down menu and choose the *Live Migrate Instance* option.

4. In the Live Migrate wizard you will see the compute host the instance currently resides on and then a drop down menu that allows you to choose the compute host you want to migrate the instance to. Select a destination host from that menu. You also have two checkboxes for additional options, which are described below:

   Disk Over Commit - If this is not checked then the value will be `False`. If you check this box then it will allow you to override the check that occurs to ensure the destination host has the available disk space to host the instance.

   Block Migration - If this is not checked then the value will be `False`. If you check this box then it will migrate the local disks by using block migration. Use this option if you are only using ephemeral storage on your instances. If you are using block storage for your instance then ensure this box is not checked.

5. To begin the live migration, click *Submit*.

Migrating instances using the NovaClient CLI

To perform migrations from the command-line, use the NovaClient. The Cloud Lifecycle Manager node in your cloud environment should have the NovaClient already installed. If you will be accessing your environment through a different method, ensure that the NovaClient is installed. You can do so using Python's `pip` package manager.

To run the commands in the steps below, you need administrator credentials. From the Cloud Lifecycle Manager, you can source the `service.osrc` file which is provided that has the necessary credentials:

```
source ~/service.osrc
```

Here are the steps to perform:

1. Log in to the Cloud Lifecycle Manager.

2. Identify the instances on the compute node you wish to migrate:

   ```
   nova list --all-tenants --host <hostname>
   ```

   Example showing a host with a single compute instance on it:

   ```
   ardana >  nova list --host ardana-cp1-comp0001-mgmt --all-tenants
   +-----------------------------------+------+--------------------------------
   +--------+-----------+------------+----------------------+
   | ID                                | Name | Tenant ID                     |
    Status | Task State | Power State | Networks             |
   ```

```
+------------------------------------+------+--------------------------------
+--------+-----------+-------------+----------------------+
| 553ba508-2d75-4513-b69a-f6a2a08d04e3 | test | 193548a949c146dfa1f051088e141f0b |
 ACTIVE | -          | Running     | adminnetwork=10.0.0.5 |
+------------------------------------+------+--------------------------------
+--------+-----------+-------------+----------------------+
```

3. When using live migration you can either specify a target host that the instance will be migrated to or you can omit the target to allow the nova-scheduler to choose a node for you. If you want to get a list of available hosts you can use this command:

```
nova host-list
```

4. Migrate the instance(s) on the compute node using the notes below.
   If your instance is booted from a block storage volume or has any number of block storage volumes attached, use the `nova live-migration` command with this syntax:

```
nova live-migration <instance uuid> [<target compute host>]
```

If your instance has local (ephemeral) disk(s) only or if your instance has a mix of ephemeral disk(s) and block storage volume(s), you should use the `--block-migrate` option:

```
nova live-migration --block-migrate <instance uuid> [<target compute host>]
```

🎓 Note

The `[<target compute host>]` option is optional. If you do not specify a target host then the nova scheduler will choose a node for you.

Multiple instances

If you want to live migrate all of the instances off a single compute host you can utilize the `nova host-evacuate-live` command.
Issue the host-evacuate-live command, which will begin the live migration process.
If all of the instances on the host are using at least one local (ephemeral) disk, you should use this syntax:

```
nova host-evacuate-live --block-migrate <hostname>
```

Alternatively, if all of the instances are only using block storage volumes then omit the `--block-migrate` option:

```
nova host-evacuate-live <hostname>
```

> ☞ **Note**
>
> You can either let the nova-scheduler choose a suitable target host or you can specify one using the `--target-host <hostname>` switch. See **nova help host-evacuate-live** for details.

13.1.3.3.6 Troubleshooting migration or host evacuate issues

Issue: When attempting to use `nova host-evacuate-live` against a node, you receive the error below:

```
$ nova host-evacuate-live ardana-cp1-comp0001-mgmt --target-host ardana-cp1-comp0003-mgmt
+------------------------------------+------------------------
+----------------------------------------------------------------------------------------------------
+
| Server UUID                        | Live Migration Accepted | Error Message

                                                             |
+------------------------------------+------------------------
+----------------------------------------------------------------------------------------------------
+
| 95a7ded8-ebfc-4848-9090-2df378c88a4c | False                | Error while live
 migrating instance: ardana-cp1-comp0001-mgmt is not on shared storage: Live migration
 can not be used without shared storage except a booted from volume VM which does not
 have a local disk. (HTTP 400) (Request-ID: req-9fd79670-a780-40ed-a515-c14e28e0a0a7)
 |
| 13ab4ef7-0623-4d00-bb5a-5bb2f1214be4 | False                | Error while live
 migrating instance: ardana-cp1-comp0001-mgmt is not on shared storage: Live migration
 cannot be used without shared storage except a booted from volume VM which does not have
 a local disk. (HTTP 400) (Request-ID: req-26834267-c3ec-4f8b-83cc-5193d6a394d6)     |
+------------------------------------+------------------------
+----------------------------------------------------------------------------------------------------
+
```

Fix: This occurs when you are attempting to live evacuate a host that contains instances booted from local storage and you are not specifying `--block-migrate` in your command. Re-attempt the live evacuation with this syntax:

```
nova host-evacuate-live --block-migrate <hostname> [--target-host <target_hostname>]
```

Issue: When attempting to use `nova host-evacuate-live` against a node, you receive the error below:

```
$ nova host-evacuate-live --block-migrate ardana-cp1-comp0001-mgmt --target-host ardana-
cp1-comp0003-mgmt
+-----------------------------------+-----------------------
+----------------------------------------------------------------------
+
| Server UUID                       | Live Migration Accepted | Error Message



        |
+-----------------------------------+-----------------------
+----------------------------------------------------------------------
+
| e9874122-c5dc-406f-9039-217d9258c020 | False                   | Error while
 live migrating instance: ardana-cp1-comp0001-mgmt is not on local storage:
 Block migration can not be used with shared storage. (HTTP 400) (Request-ID:
  req-60b1196e-84a0-4b71-9e49-96d6f1358e1a)      |
| 84a02b42-9527-47ac-bed9-8fde1f98e3fe | False                   | Error while
 live migrating instance: ardana-cp1-comp0001-mgmt is not on local storage:
 Block migration can not be used with shared storage. (HTTP 400) (Request-ID:
  req-0cdf1198-5dbd-40f4-9e0c-e94aa1065112)      |
+-----------------------------------+-----------------------
+----------------------------------------------------------------------
+
```

Fix: This occurs when you are attempting to live evacuate a host that contains instances booted from a block storage volume and you are specifying `--block-migrate` in your command. Re-attempt the live evacuation with this syntax:

```
nova host-evacuate-live <hostname> [--target-host <target_hostname>]
```

Issue: When attempting to use `nova live-migration` against an instance, you receive the error below:

```
$ nova live-migration 2a13ffe6-e269-4d75-8e46-624fec7a5da0 ardana-cp1-comp0002-mgmt
ERROR (BadRequest): ardana-cp1-comp0001-mgmt is not on shared storage: Live migration can
 not be used without shared storage except a booted from volume VM which does not have a
 local disk. (HTTP 400) (Request-ID: req-158dd415-0bb7-4613-8529-6689265387e7)
```

Fix: This occurs when you are attempting to live migrate an instance that was booted from local storage and you are not specifying `--block-migrate` in your command. Re-attempt the live migration with this syntax:

```
nova live-migration --block-migrate <instance_uuid> <target_hostname>
```

Issue: When attempting to use `nova live-migration` against an instance, you receive the error below:

```
$ nova live-migration --block-migrate 84a02b42-9527-47ac-bed9-8fde1f98e3fe ardana-cp1-
comp0001-mgmt
ERROR (BadRequest): ardana-cp1-comp0002-mgmt is not on local storage: Block migration
 can not be used with shared storage. (HTTP 400) (Request-ID: req-51fee8d6-6561-4afc-
b0c9-7afa7dc43a5b)
```

Fix: This occurs when you are attempting to live migrate an instance that was booted from a block storage volume and you are specifying `--block-migrate` in your command. Re-attempt the live migration with this syntax:

```
nova live-migration <instance_uuid> <target_hostname>
```

13.1.3.4 Adding Compute Node

Adding a Compute Node allows you to add capacity.

13.1.3.4.1 Adding a SLES Compute Node

Adding a SLES compute node allows you to add additional capacity for more virtual machines.

You may have a need to add additional SLES compute hosts for more virtual machine capacity or another purpose and these steps will help you achieve this.

There are two methods you can use to add SLES compute hosts to your environment:

1. Adding SLES pre-installed compute hosts. This method does not require the SLES ISO be on the Cloud Lifecycle Manager to complete.

2. Using the provided Ansible playbooks and Cobbler, SLES will be installed on your new compute hosts. This method requires that you provided a SUSE Linux Enterprise Server 12 SP3 ISO during the initial installation of your cloud, following the instructions at *Book "Installing with Cloud Lifecycle Manager", Chapter 19 "Installing SLES Compute", Section 19.1 "SLES Compute Node Installation Overview"*.

If you want to use the provided Ansible playbooks and Cobbler to setup and configure your SLES hosts and you did not have the SUSE Linux Enterprise Server 12 SP3 ISO on your Cloud Lifecycle Manager during your initial installation then ensure you look at the note at the top of that section before proceeding.

13.1.3.4.1.1 Prerequisites

You need to ensure your input model files are properly setup for SLES compute host clusters. This must be done during the installation process of your cloud and is discussed further at *Book "Installing with Cloud Lifecycle Manager", Chapter 19 "Installing SLES Compute", Section 19.3 "Using the Cloud Lifecycle Manager to Deploy SLES Compute Nodes"* and *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 10 "Modifying Example Configurations for Compute Nodes", Section 10.1 "SLES Compute Nodes".*

13.1.3.4.1.2 Adding a SLES compute node

Adding pre-installed SLES compute hosts

This method requires that you have SUSE Linux Enterprise Server 12 SP3 pre-installed on the baremetal host prior to beginning these steps.

1. Ensure you have SUSE Linux Enterprise Server 12 SP3 pre-installed on your baremetal host.

2. Log in to the Cloud Lifecycle Manager.

3. Edit your `~/openstack/my_cloud/definition/data/servers.yml` file to include the details about your new compute host(s).
   For example, if you already had a cluster of three SLES compute hosts using the `SLES-COMPUTE-ROLE` role and needed to add a fourth one you would add your details to the bottom of the file in the format. Note that we left out the IPMI details because they will not be needed since you pre-installed the SLES OS on your host(s).

   ```
   - id: compute4
     ip-addr: 192.168.102.70
     role: SLES-COMPUTE-ROLE
     server-group: RACK1
   ```

You can find detailed descriptions of these fields in *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 6 "Configuration Objects", Section 6.5 "Servers"*. Ensure that you use the same role for any new SLES hosts you are adding as you specified on your existing SLES hosts.

⛨ Important

You will need to verify that the `ip-addr` value you choose for this host does not conflict with any other IP address in your cloud environment. You can confirm this by checking the `~/openstack/my_cloud/info/address_info.yml` file on your Cloud Lifecycle Manager.

4. In your `~/openstack/my_cloud/definition/data/control_plane.yml` file you will need to check the values for `member-count`, `min-count`, and `max-count`. If you specified them, ensure that they match up with your new total node count. For example, if you had previously specified `member-count: 3` and are adding a fourth compute node, you will need to change that value to `member-count: 4`.

   See for *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 6 "Configuration Objects", Section 6.2 "Control Plane"* more details.

5. Commit the changes to git:

```
git add -A
git commit -a -m "Add node <name>"
```

6. Run the configuration processor and resolve any errors that are indicated:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

Before proceeding, you may want to take a look at **info/server_info.yml** to see if the assignment of the node you have added is what you expect. It may not be, as nodes will not be numbered consecutively if any have previously been removed. This is to prevent loss of data; the config processor retains data about removed nodes and keeps their ID numbers

from being reallocated. See *Book "Planning an Installation with Cloud Lifecycle Manager",* *Chapter 7 "Other Topics", Section 7.3 "Persisted Data", Section 7.3.1 "Persisted Server Allocations"* for information on how this works.

8. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your non-OS partitions on your nodes are completely wiped prior to continuing with the installation.

> ### 🎓 Note
>
> The `wipe_disks.yml` playbook is only meant to be run on systems immediately after running `bm-reimage.yml`. If used for any other case, it may not wipe all of the expected partitions.

The location of `hostname` is `~/scratch/ansible/next/ardana/ansible/hosts`.

```
cd ~/scratch/ansible/next/ardana/ansible/
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --limit <hostname>
```

9. Complete the compute host deployment with this playbook:

```
cd ~/scratch/ansible/next/ardana/ansible/
ansible-playbook -i hosts/verb_hosts site.yml --tag "generate_hosts_file"
ansible-playbook -i hosts/verb_hosts site.yml --limit <hostname>
```

Adding SLES compute hosts with Ansible playbooks and Cobbler

These steps will show you how to add the new SLES compute host to your `servers.yml` file and then run the playbooks that update your cloud configuration. You will run these playbooks from the lifecycle manager.

If you did not have the SUSE Linux Enterprise Server 12 SP3 ISO available on your Cloud Lifecycle Manager during your initial installation, it must be installed before proceeding further. Instructions can be found in *Book "Installing with Cloud Lifecycle Manager", Chapter 19 "Installing SLES Compute".*

When you are prepared to continue, use these steps:

1. Log in to your Cloud Lifecycle Manager.

2. Checkout the `site` branch of your local git so you can begin to make the necessary edits:

```
cd ~/openstack/my_cloud/definition/data
git checkout site
```

3. Edit your `~/openstack/my_cloud/definition/data/servers.yml` file to include the details about your new compute host(s).

   For example, if you already had a cluster of three SLES compute hosts using the `SLES-COMPUTE-ROLE` role and needed to add a fourth one you would add your details to the bottom of the file in this format:

```
- id: compute4
  ip-addr: 192.168.102.70
  role: SLES-COMPUTE-ROLE
  server-group: RACK1
  mac-addr: e8:39:35:21:32:4e
  ilo-ip: 10.1.192.36
  ilo-password: password
  ilo-user: admin
  distro-id: sles12sp3-x86_64
```

   You can find detailed descriptions of these fields in *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 6 "Configuration Objects", Section 6.5 "Servers"*. Ensure that you use the same role for any new SLES hosts you are adding as you specified on your existing SLES hosts.

   ## 🔔 Important

   > You will need to verify that the `ip-addr` value you choose for this host does not conflict with any other IP address in your cloud environment. You can confirm this by checking the `~/openstack/my_cloud/info/address_info.yml` file on your Cloud Lifecycle Manager.

4. In your `~/openstack/my_cloud/definition/data/control_plane.yml` file you will need to check the values for `member-count`, `min-count`, and `max-count`. If you specified them, ensure that they match up with your new total node count. For example, if you had previously specified `member-count: 3` and are adding a fourth compute node, you will need to change that value to `member-count: 4`.

   See *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 6 "Configuration Objects", Section 6.2 "Control Plane"* for more details.

5. Commit the changes to git:

```
git add -A
git commit -a -m "Add node <name>"
```

6. Run the configuration processor and resolve any errors that are indicated:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. The following playbook confirms that your servers are accessible over their IPMI ports.

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-power-status.yml -e nodelist=compute4
```

8. Add the new node into Cobbler:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

9. Run the following playbook, ensuring that you specify only your UEFI SLES nodes using the nodelist. This playbook will reconfigure Cobbler for the nodes listed.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook prepare-sles-grub2.yml -e nodelist=node1[,node2,node3]
```

10. Then you can image the node:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<node name>
```

Note

If you do not know the `<node name>`, you can get it by using **`sudo cobbler system list`**.

Before proceeding, you may want to take a look at **info/server_info.yml** to see if the assignment of the node you have added is what you expect. It may not be, as nodes will not be numbered consecutively if any have previously been removed. This is to prevent loss of data; the config processor retains data about removed nodes and keeps their ID numbers from being reallocated. See *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 7 "Other Topics", Section 7.3 "Persisted Data", Section 7.3.1 "Persisted Server Allocations"* for information on how this works.

11. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
```

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

12. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your non-OS partitions on your hosts are completely wiped prior to continuing with the installation. The `wipe_disks.yml` playbook is only meant to be run on systems immediately after running `bm-reimage.yml`. If used for any other case, it may not wipe all of the expected partitions.

```
cd ~/scratch/ansible/next/ardana/ansible/
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --limit <hostname>
```

> ☞ **Note**
>
> You can obtain the `<hostname>` from the file `~/scratch/ansible/next/ardana/ansible/hosts/verb_hosts`.

13. You should verify that the netmask, bootproto, and other necessary settings are correct and if they are not then re-do them. See *Book "Installing with Cloud Lifecycle Manager", Chapter 19 "Installing SLES Compute"* for details.

14. Complete the compute host deployment with these playbooks. For the last one, ensure you specify the compute hosts you are added with the `--limit` switch:

```
cd ~/scratch/ansible/next/ardana/ansible/
ansible-playbook -i hosts/verb_hosts site.yml --tag "generate_hosts_file"
ansible-playbook -i hosts/verb_hosts site.yml --limit <hostname>
```

13.1.3.4.1.3 Adding a new SLES compute node to monitoring

If you want to add a new Compute node to the monitoring service checks, there is an additional playbook that must be run to ensure this happens:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-deploy.yml --tags "active_ping_checks"
```

13.1.3.4.2 Adding a RHEL Compute Node

Adding a RHEL compute node allows you to increase cloud capacity for more virtual machines. These steps will help you add new RHEL compute hosts for this purpose.

13.1.3.4.2.1 Prerequisites

You need to ensure your input model files are properly setup for RHEL compute host clusters. This must be done during the installation process of your cloud and is discussed further at *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 10 "Modifying Example Configurations for Compute Nodes", Section 10.2 "RHEL Compute Nodes"*.

13.1.3.4.2.2 Adding a RHEL compute node

You must have RHEL 7.5 pre-installed on the baremetal host prior to beginning these steps.

1. Ensure you have RHEL 7.5 pre-installed on your baremetal host.

2. Log in to the Cloud Lifecycle Manager.

3. Edit your `~/openstack/my_cloud/definition/data/servers.yml` file to include the details about your new compute host(s).
   For example, if you already had a cluster of three RHEL compute hosts using the `RHEL-COMPUTE-ROLE` role and needed to add a fourth one you would add your details to the bottom of the file in the format. Note that we left out the IPMI details because they will not be needed since you pre-installed the RHEL OS on your host(s).

   ```
   - id: compute4
     ip-addr: 192.168.102.70
     role: RHEL-COMPUTE-ROLE
     server-group: RACK1
   ```

   You can find detailed descriptions of these fields in *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 6 "Configuration Objects", Section 6.5 "Servers"*. Ensure that you use the same role for any new RHEL hosts you are adding as you specified on your existing RHEL hosts.

   > **Important**
   >
   > Verify that the `ip-addr` value you choose for this host does not conflict with any other IP address in your cloud environment. You can confirm this by checking the `~/openstack/my_cloud/info/address_info.yml` file on your Cloud Lifecycle Manager.

4. In your `~/openstack/my_cloud/definition/data/control_plane.yml` file you will need to check the values for `member-count`, `min-count`, and `max-count`. If you specified them, ensure that they match up with your new total node count. For example, if you had previously specified `member-count: 3` and are adding a fourth compute node, you will need to change that value to `member-count: 4`.

   See *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 6 "Configuration Objects", Section 6.2 "Control Plane"* for more details.

5. Commit the changes to git:

   ```
   git add -A
   git commit -a -m "Add node <name>"
   ```

6. Run the configuration processor and resolve any errors that are indicated:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

7. Update your deployment directory:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

   Before proceeding, you may want to take a look at **info/server_info.yml** to see if the assignment of the node you have added is what you expect. It may not be, as nodes will not be numbered consecutively if any have previously been removed. This is to prevent loss of data; the config processor retains data about removed nodes and keeps their ID numbers from being reallocated. See *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 7 "Other Topics", Section 7.3 "Persisted Data", Section 7.3.1 "Persisted Server Allocations"* for information on how this works.

8. Look up the value for the new compute node's `hostname` in `~/scratch/ansible/next/ardana/ansible/hosts/verb_hosts`.

   Then, complete the compute host deployment with this playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible/
   ansible-playbook -i hosts/verb_hosts site.yml --tag "generate_hosts_file"
   ansible-playbook -i hosts/verb_hosts site.yml --limit <hostname>
   ```

13.1.3.4.2.3 Adding a new RHEL compute node to monitoring

If you want to add a new Compute node to the monitoring service checks, there is an additional playbook that must be run to ensure this happens:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-deploy.yml --tags "active_ping_checks"
```

13.1.3.5 Removing a Compute Node

Removing a Compute node allows you to remove capacity.

You may have a need to remove a Compute node and these steps will help you achieve this.

13.1.3.5.1 Disable Provisioning on the Compute Host

1. Get a list of the Nova services running which will provide us with the details we need to disable the provisioning on the Compute host you are wanting to remove:

   ```
   nova service-list
   ```

   Here is an example below. I've highlighted the Compute node we are going to remove in the examples:

   ```
   $ nova service-list
   +----+-----------------+-------------------------+----------+---------+-------
   +--------------------------+-----------------+
   | Id | Binary          | Host                    | Zone     | Status  | State |
    Updated_at                | Disabled Reason |
   +----+-----------------+-------------------------+----------+---------+-------
   +--------------------------+-----------------+
   | 1  | nova-conductor  | ardana-cp1-c1-m1-mgmt   | internal | enabled | up    |
    2015-11-22T22:50:43.000000 | -               |
   | 10 | nova-scheduler  | ardana-cp1-c1-m1-mgmt   | internal | enabled | up    |
    2015-11-22T22:50:34.000000 | -               |
   | 13 | nova-conductor  | ardana-cp1-c1-m3-mgmt   | internal | enabled | up    |
    2015-11-22T22:50:43.000000 | -               |
   | 16 | nova-conductor  | ardana-cp1-c1-m2-mgmt   | internal | enabled | up    |
    2015-11-22T22:50:43.000000 | -               |
   | 25 | nova-consoleauth | ardana-cp1-c1-m1-mgmt  | internal | enabled | up    |
    2015-11-22T22:50:38.000000 | -               |
   | 28 | nova-scheduler  | ardana-cp1-c1-m2-mgmt   | internal | enabled | up    |
    2015-11-22T22:50:38.000000 | -               |
   ```

```
| 31 | nova-scheduler  | ardana-cp1-c1-m3-mgmt   | internal | enabled | up    |
  2015-11-22T22:50:42.000000 | -             |
| 34 | nova-compute    | ardana-cp1-comp0001-mgmt | AZ1      | enabled | up    |
  2015-11-22T22:50:35.000000 | -             |
| 37 | nova-compute    | ardana-cp1-comp0002-mgmt | AZ2      | enabled | up    |
  2015-11-22T22:50:44.000000 | -             |
+----+-----------------+-------------------------+---------+--------+-------
+-------------------------+----------------+
```

2. Disable the Nova service on the Compute node you are wanting to remove which will ensure it is taken out of the scheduling rotation:

```
nova service-disable --reason "<enter reason here>" <node hostname> nova-compute
```

Here is an example if I wanted to remove the `ardana-cp1-comp0002-mgmt` in the output above:

```
$ nova service-disable --reason "hardware reallocation" ardana-cp1-comp0002-mgmt
 nova-compute
+-------------------------+--------------+----------+----------------------+
| Host                    | Binary       | Status   | Disabled Reason      |
+-------------------------+--------------+----------+----------------------+
| ardana-cp1-comp0002-mgmt | nova-compute | disabled | hardware reallocation |
+-------------------------+--------------+----------+----------------------+
```

13.1.3.5.2   Remove the Compute Host from its Availability Zone

If you configured the Compute host to be part of an availability zone, these steps will show you how to remove it.

1. Get a list of the Nova services running which will provide us with the details we need to remove a Compute node:

```
nova service-list
```

Here is an example below. I've highlighted the Compute node we are going to remove in the examples:

```
$ nova service-list
+----+-----------------+-------------------------+---------+--------+-------
+-------------------------+----------------------+
| Id | Binary          | Host                    | Zone    | Status | State |
  Updated_at               | Disabled Reason      |
```

```
+----+-----------------+-------------------------+----------+---------+-------
+-------------------------+----------------------+
| 1  | nova-conductor  | ardana-cp1-c1-m1-mgmt   | internal | enabled | up    |
 2015-11-22T22:50:43.000000 | -                   |
| 10 | nova-scheduler  | ardana-cp1-c1-m1-mgmt   | internal | enabled | up    |
 2015-11-22T22:50:34.000000 | -                   |
| 13 | nova-conductor  | ardana-cp1-c1-m3-mgmt   | internal | enabled | up    |
 2015-11-22T22:50:43.000000 | -                   |
| 16 | nova-conductor  | ardana-cp1-c1-m2-mgmt   | internal | enabled | up    |
 2015-11-22T22:50:43.000000 | -                   |
| 25 | nova-consoleauth | ardana-cp1-c1-m1-mgmt  | internal | enabled | up    |
 2015-11-22T22:50:38.000000 | -                   |
| 28 | nova-scheduler  | ardana-cp1-c1-m2-mgmt   | internal | enabled | up    |
 2015-11-22T22:50:38.000000 | -                   |
| 31 | nova-scheduler  | ardana-cp1-c1-m3-mgmt   | internal | enabled | up    |
 2015-11-22T22:50:42.000000 | -                   |
| 34 | nova-compute    | ardana-cp1-comp0001-mgmt | AZ1     | enabled | up    |
 2015-11-22T22:50:35.000000 | -                   |
| 37 | nova-compute    | ardana-cp1-comp0002-mgmt | AZ2     | enabled | up    |
 2015-11-22T22:50:44.000000 | hardware reallocation |
+----+-----------------+-------------------------+----------+---------+-------
+-------------------------+----------------------+
```

2. You can remove the Compute host from the availability zone it was a part of with this command:

```
nova aggregate-remove-host <availability zone> <nova hostname>
```

So for the same example as the previous step, the `ardana-cp1-comp0002-mgmt` host was in the `AZ2` availability zone so I would use this command to remove it:

```
$ nova aggregate-remove-host AZ2 ardana-cp1-comp0002-mgmt
Host ardana-cp1-comp0002-mgmt has been successfully removed from aggregate 4
+----+------+-------------------+-------+-----------------------+
| Id | Name | Availability Zone | Hosts | Metadata              |
+----+------+-------------------+-------+-----------------------+
| 4  | AZ2  | AZ2               |       | 'availability_zone=AZ2' |
+----+------+-------------------+-------+-----------------------+
```

3. You can confirm the last two steps completed successfully by running another `nova service-list`.

Here is an example which confirms that the node has been disabled and that it has been removed from the availability zone. I have highlighted these:

```
$ nova service-list
```

```
+----+-----------------+-------------------------+----------+----------+-------
+-------------------------+----------------------+
| Id | Binary          | Host                    | Zone     | Status   | State |
 Updated_at              | Disabled Reason      |
+----+-----------------+-------------------------+----------+----------+-------
+-------------------------+----------------------+
| 1  | nova-conductor  | ardana-cp1-c1-m1-mgmt   | internal | enabled  | up    |
 2015-11-22T23:04:33.000000 | -                 |
| 10 | nova-scheduler  | ardana-cp1-c1-m1-mgmt   | internal | enabled  | up    |
 2015-11-22T23:04:34.000000 | -                 |
| 13 | nova-conductor  | ardana-cp1-c1-m3-mgmt   | internal | enabled  | up    |
 2015-11-22T23:04:33.000000 | -                 |
| 16 | nova-conductor  | ardana-cp1-c1-m2-mgmt   | internal | enabled  | up    |
 2015-11-22T23:04:33.000000 | -                 |
| 25 | nova-consoleauth| ardana-cp1-c1-m1-mgmt   | internal | enabled  | up    |
 2015-11-22T23:04:28.000000 | -                 |
| 28 | nova-scheduler  | ardana-cp1-c1-m2-mgmt   | internal | enabled  | up    |
 2015-11-22T23:04:28.000000 | -                 |
| 31 | nova-scheduler  | ardana-cp1-c1-m3-mgmt   | internal | enabled  | up    |
 2015-11-22T23:04:32.000000 | -                 |
| 34 | nova-compute    | ardana-cp1-comp0001-mgmt | AZ1     | enabled  | up    |
 2015-11-22T23:04:25.000000 | -                 |
| 37 | nova-compute    | ardana-cp1-comp0002-mgmt | nova     | disabled | up    |
 2015-11-22T23:04:34.000000 | hardware reallocation |
+----+-----------------+-------------------------+----------+----------+-------
+-------------------------+----------------------+
```

13.1.3.5.3 Use Live Migration to Move Any Instances on this Host to Other Hosts

1. You will need to verify if the Compute node is currently hosting any instances on it. You
   can do this with the command below:

   ```
   nova list --host=<nova hostname> --all_tenants=1
   ```

   Here is an example below which shows that we have a single running instance on this
   node currently:

   ```
   $ nova list --host=ardana-cp1-comp0002-mgmt --all_tenants=1
   +--------------------------------------+--------+--------------------------------
   +--------+------------+-------------+-----------------+
   | ID                                   | Name   | Tenant ID                      |
    Status | Task State | Power State | Networks        |
   +--------------------------------------+--------+--------------------------------
   +--------+------------+-------------+-----------------+
   ```

```
| 78fdb938-a89c-4a0c-a0d4-b88f1555c3b9 | paul4d | 5e9998f1b1824ea9a3b06ad142f09ca5 |
 ACTIVE | -          | Running   | paul=10.10.10.7 |
+------------------------------------+--------+--------------------------------
+--------+-----------+------------+----------------+
```

2. You will likely want to migrate this instance off of this node before removing it. You can do this with the live migration functionality within Nova. The command will look like this:

```
nova live-migration --block-migrate <nova instance ID>
```

Here is an example using the instance in the previous step:

```
$ nova live-migration --block-migrate 78fdb938-a89c-4a0c-a0d4-b88f1555c3b9
```

You can check the status of the migration using the same command from the previous step:

```
$ nova list --host=ardana-cp1-comp0002-mgmt --all_tenants=1
+------------------------------------+--------+--------------------------------
+-----------+-----------+------------+----------------+
| ID                                 | Name   | Tenant ID                      |
 Status    | Task State | Power State | Networks      |
+------------------------------------+--------+--------------------------------
+-----------+-----------+------------+----------------+
| 78fdb938-a89c-4a0c-a0d4-b88f1555c3b9 | paul4d | 5e9998f1b1824ea9a3b06ad142f09ca5 |
 MIGRATING | migrating  | Running    | paul=10.10.10.7 |
+------------------------------------+--------+--------------------------------
+-----------+-----------+------------+----------------+
```

3. Run nova list again

```
$ nova list --host=ardana-cp1-comp0002-mgmt --all_tenants=1
```

to see that the running instance has been migrated:

```
+----+------+-----------+--------+-----------+------------+----------+
| ID | Name | Tenant ID | Status | Task State | Power State | Networks |
+----+------+-----------+--------+-----------+------------+----------+
+----+------+-----------+--------+-----------+------------+----------+
```

13.1.3.5.4 Disable Neutron Agents on Node to be Removed

You should also locate and disable or remove neutron agents. To see the neutron agents running:

```
$ neutron agent-list | grep NODE_NAME
```

```
+---------------------------------------+--------------------+------------------------
+-------+---------------+-------------------------+
| id                                    | agent_type         | host
 | alive | admin_state_up | binary                 |
+---------------------------------------+--------------------+------------------------
+-------+---------------+-------------------------+
| 08f16dbc-4ba2-4c1d-a4a3-a2ff2526ebe4 | L3 agent           | ardana-cp1-comp0002-mgmt
 | :-)   | True          | neutron-l3-agent        |
| dbe4fe11-8f08-4306-8244-cc68e98bb770 | Metadata agent     | ardana-cp1-comp0002-mgmt
 | :-)   | True          | neutron-metadata-agent  |
| f0d262d1-7139-40c7-bdc2-f227c6dee5c8 | Open vSwitch agent  | ardana-cp1-comp0002-mgmt
 | :-)   | True          | neutron-openvswitch-agent |
+---------------------------------------+--------------------+------------------------
+-------+---------------+-------------------------+

$ neutron agent-update --admin-state-down 08f16dbc-4ba2-4c1d-a4a3-a2ff2526ebe4
$ neutron agent-update --admin-state-down dbe4fe11-8f08-4306-8244-cc68e98bb770
$ neutron agent-update --admin-state-down f0d262d1-7139-40c7-bdc2-f227c6dee5c8
```

13.1.3.5.5 Shut down or Stop the Nova and Neutron Services on the Compute Host

To perform this step you have a few options. You can SSH into the Compute host and run the following commands:

```
sudo systemctl stop nova-compute
```

```
sudo systemctl stop neutron-*
```

Because the Neutron agent self-registers against Neutron server, you may want to prevent the following services from coming back online. Here is how you can get the list:

```
sudo systemctl list-units neutron-* --all
```

Here are the results:

```
UNIT                                LOAD        ACTIVE      SUB       DESCRIPTION
neutron-common-rundir.service       loaded      inactive    dead      Create /var/run/
neutron
•neutron-dhcp-agent.service         not-found   inactive    dead      neutron-dhcp-
agent.service
neutron-l3-agent.service            loaded      inactive    dead      neutron-l3-agent
 Service
neutron-lbaasv2-agent.service       loaded      inactive    dead      neutron-lbaasv2-
agent Service
neutron-metadata-agent.service      loaded      inactive    dead      neutron-metadata-
agent Service
```

```
•neutron-openvswitch-agent.service      loaded      failed     failed    neutron-
openvswitch-agent Service
neutron-ovs-cleanup.service             loaded      inactive   dead      Neutron OVS Cleanup
 Service


       LOAD  = Reflects whether the unit definition was properly loaded.
       ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
       SUB   = The low-level unit activation state, values depend on unit type.


       7 loaded units listed.
       To show all installed unit files use 'systemctl list-unit-files'.
```

For each loaded service issue the command

```
sudo systemctl disable <service-name>
```

In the above example that would be each service, *except neutron-dhcp-agent.service*

For example:

```
sudo systemctl disable neutron-common-rundir neutron-l3-agent neutron-lbaasv2-agent
 neutron-metadata-agent neutron-openvswitch-agent
```

Now you can shut down the node:

```
sudo shutdown now
```

OR

From the Cloud Lifecycle Manager you can use the `bm-power-down.yml` playbook to shut down the node:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-power-down.yml -e nodelist=<node name>
```

🎓 Note

The `<node name>` value will be the value corresponding to this node in Cobbler. You can run **`sudo cobbler system list`** to retrieve these names.

13.1.3.5.6   Delete the Compute Host from Nova

Retrieve the list of Nova services:

```
nova service-list
```

Here is an example highlighting the Compute host we're going to remove:

```
$ nova service-list
+----+-----------------+------------------------+----------+----------+-------
+--------------------------+----------------------+
| Id | Binary          | Host                   | Zone     | Status   | State |
 Updated_at               | Disabled Reason      |
+----+-----------------+------------------------+----------+----------+-------
+--------------------------+----------------------+
| 1  | nova-conductor  | ardana-cp1-c1-m1-mgmt  | internal | enabled  | up    |
 2015-11-22T23:04:33.000000 | -                    |
| 10 | nova-scheduler  | ardana-cp1-c1-m1-mgmt  | internal | enabled  | up    |
 2015-11-22T23:04:34.000000 | -                    |
| 13 | nova-conductor  | ardana-cp1-c1-m3-mgmt  | internal | enabled  | up    |
 2015-11-22T23:04:33.000000 | -                    |
| 16 | nova-conductor  | ardana-cp1-c1-m2-mgmt  | internal | enabled  | up    |
 2015-11-22T23:04:33.000000 | -                    |
| 25 | nova-consoleauth| ardana-cp1-c1-m1-mgmt  | internal | enabled  | up    |
 2015-11-22T23:04:28.000000 | -                    |
| 28 | nova-scheduler  | ardana-cp1-c1-m2-mgmt  | internal | enabled  | up    |
 2015-11-22T23:04:28.000000 | -                    |
| 31 | nova-scheduler  | ardana-cp1-c1-m3-mgmt  | internal | enabled  | up    |
 2015-11-22T23:04:32.000000 | -                    |
| 34 | nova-compute    | ardana-cp1-comp0001-mgmt | AZ1    | enabled  | up    |
 2015-11-22T23:04:25.000000 | -                    |
| 37 | nova-compute    | ardana-cp1-comp0002-mgmt | nova   | disabled | up    |
 2015-11-22T23:04:34.000000 | hardware reallocation |
+----+-----------------+------------------------+----------+----------+-------
+--------------------------+----------------------+
```

Delete the host from Nova using the command below:

```
nova service-delete <service ID>
```

Following our example above, you would use:

```
nova service-delete 37
```

Use the command below to confirm that the Compute host has been completely removed from Nova:

```
nova hypervisor-list
```

13.1.3.5.7   Delete the Compute Host from Neutron

Multiple Neutron agents are running on the compute node. You have to remove all of the agents running on the node using the "neutron agent-delete" command. In the example below, the l3-agent, openvswitch-agent and metadata-agent are running:

```
$ neutron agent-list | grep NODE_NAME
+-------------------------------------+---------------------+------------------------
+-------+---------------+-------------------------+
| id                                  | agent_type          | host
 | alive | admin_state_up | binary                 |
+-------------------------------------+---------------------+------------------------
+-------+---------------+-------------------------+
| 08f16dbc-4ba2-4c1d-a4a3-a2ff2526ebe4 | L3 agent            | ardana-cp1-comp0002-mgmt
 | :-)   | False          | neutron-l3-agent        |
| dbe4fe11-8f08-4306-8244-cc68e98bb770 | Metadata agent      | ardana-cp1-comp0002-mgmt
 | :-)   | False          | neutron-metadata-agent  |
| f0d262d1-7139-40c7-bdc2-f227c6dee5c8 | Open vSwitch agent  | ardana-cp1-comp0002-mgmt
 | :-)   | False          | neutron-openvswitch-agent |
+-------------------------------------+---------------------+------------------------
+-------+---------------+-------------------------+

$ neutron agent-delete AGENT_ID

$ neutron agent-delete 08f16dbc-4ba2-4c1d-a4a3-a2ff2526ebe4
$ neutron agent-delete dbe4fe11-8f08-4306-8244-cc68e98bb770
$ neutron agent-delete f0d262d1-7139-40c7-bdc2-f227c6dee5c8
```

13.1.3.5.8   Remove the Compute Host from the servers.yml File and Run the Configuration Processor

Complete these steps from the Cloud Lifecycle Manager to remove the Compute node:

1. Log in to the Cloud Lifecycle Manager

2. Edit your `servers.yml` file in the location below to remove references to the Compute node(s) you want to remove:

   ```
   ~/openstack/my_cloud/definition/data/servers.yml
   ```

3. You may also need to edit your `control_plane.yml` file to update the values for `member-count`, `min-count`, and `max-count` if you used those to ensure they reflect the proper number of nodes you are using.

See *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 6 "Configuration Objects", Section 6.2 "Control Plane"* for more details.

4. Commit the changes to git:

```
git commit -a -m "Remove node <name>"
```

5. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

To free up the resources when running the configuration processor, use the switches `remove_deleted_servers` and `free_unused_addresses`. For more information, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 7 "Other Topics", Section 7.3 "Persisted Data"*.

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml -e
  remove_deleted_servers="y" -e free_unused_addresses="y"
```

6. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

13.1.3.5.9 Remove the Compute Host from Cobbler

Complete these steps to remove the node from Cobbler:

1. Confirm the system name in Cobbler with this command:

```
sudo cobbler system list
```

2. Remove the system from Cobbler using this command:

```
sudo cobbler system remove --name=<node>
```

3. Run the `cobbler-deploy.yml` playbook to complete the process:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

13.1.3.5.10 Remove the Compute Host from Monitoring

Once you have removed the Compute nodes, the alarms against them will trigger so there are additional steps to take to resolve this issue.

To find all Monasca API servers

```
tux > sudo cat /etc/haproxy/haproxy.cfg | grep MON
listen ardana-cp1-vip-public-MON-API-extapi-8070
    bind ardana-cp1-vip-public-MON-API-extapi:8070  ssl crt /etc/ssl/private//my-public-
cert-entry-scale
    server ardana-cp1-c1-m1-mgmt-MON_API-8070 ardana-cp1-c1-m1-mgmt:8070 check inter 5000
 rise 2 fall 5
    server ardana-cp1-c1-m2-mgmt-MON_API-8070 ardana-cp1-c1-m2-mgmt:8070 check inter 5000
 rise 2 fall 5
    server ardana-cp1-c1-m3-mgmt-MON_API-8070 ardana-cp1-c1-m3-mgmt:8070 check inter 5000
 rise 2 fall 5
listen ardana-cp1-vip-MON-API-mgmt-8070
    bind ardana-cp1-vip-MON-API-mgmt:8070  ssl crt /etc/ssl/private//ardana-internal-cert

    server ardana-cp1-c1-m1-mgmt-MON_API-8070 ardana-cp1-c1-m1-mgmt:8070 check inter 5000
 rise 2 fall 5
    server ardana-cp1-c1-m2-mgmt-MON_API-8070 ardana-cp1-c1-m2-mgmt:8070 check inter 5000
 rise 2 fall 5
    server ardana-cp1-c1-m3-mgmt-MON_API-8070 ardana-cp1-c1-m3-mgmt:8070 check inter 5000
 rise 2 fall 5
```

In above example `ardana-cp1-c1-m1-mgmt`, `ardana-cp1-c1-m2-mgmt`, `ardana-cp1-c1-m3-mgmt` are Monasa API servers

You will want to SSH to each of the Monasca API servers and edit the `/etc/monasca/agent/conf.d/host_alive.yaml` file to remove references to the Compute node you removed. This will require `sudo` access. The entries will look similar to the one below:

```
- alive_test: ping
  built_by: HostAlive
  host_name: ardana-cp1-comp0001-mgmt
  name: ardana-cp1-comp0001-mgmt ping
```

Once you have removed the references on each of your Monasca API servers you then need to restart the monasca-agent on each of those servers with this command:

```
tux > sudo service openstack-monasca-agent restart
```

With the Compute node references removed and the monasca-agent restarted, you can then delete the corresponding alarm to finish this process. To do so we recommend using the Monasca CLI which should be installed on each of your Monasca API servers by default:

```
monasca alarm-list --metric-dimensions hostname=<compute node deleted>
```

For example, if your Compute node looked like the example above then you would use this command to get the alarm ID:

```
monasca alarm-list --metric-dimensions hostname=ardana-cp1-comp0001-mgmt
```

You can then delete the alarm with this command:

```
monasca alarm-delete <alarm ID>
```

13.1.4   Planned Network Maintenance

Planned maintenance task for networking nodes.

13.1.4.1   Adding a Neutron Network Node

Adding an additional Neutron networking node allows you to increase the performance of your cloud.

You may have a need to add an additional Neutron network node for increased performance or another purpose and these steps will help you achieve this.

13.1.4.1.1   Prerequisites

If you are using the mid-scale model then your networking nodes are already separate and the roles are defined. If you are not already using this model and wish to add separate networking nodes then you need to ensure that those roles are defined. You can look in the `~/openstack/examples` folder on your Cloud Lifecycle Manager for the mid-scale example model files which show how to do this. We have also added the basic edits that need to be made below:

1. In your `server_roles.yml` file, ensure you have the `NEUTRON-ROLE` defined.
   Path to file:

   ```
   ~/openstack/my_cloud/definition/data/server_roles.yml
   ```

Example snippet:

```
- name: NEUTRON-ROLE
  interface-model: NEUTRON-INTERFACES
  disk-model: NEUTRON-DISKS
```

2. In your net_interfaces.yml file, ensure you have the NEUTRON-INTERFACES defined.
   Path to file:

```
~/openstack/my_cloud/definition/data/net_interfaces.yml
```

Example snippet:

```
- name: NEUTRON-INTERFACES
  network-interfaces:
  - device:
      name: hed3
    name: hed3
    network-groups:
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT
```

3. Create a disks_neutron.yml file, ensure you have the NEUTRON-DISKS defined in it.
   Path to file:

```
~/openstack/my_cloud/definition/data/disks_neutron.yml
```

Example snippet:

```
product:
  version: 2

disk-models:
- name: NEUTRON-DISKS
  volume-groups:
    - name: ardana-vg
      physical-volumes:
       - /dev/sda_root

      logical-volumes:
      # The policy is not to consume 100% of the space of each volume group.
      # 5% should be left free for snapshots and to allow for some flexibility.
        - name: root
          size: 35%
```

```
                  fstype: ext4
                  mount: /
              - name: log
                  size: 50%
                  mount: /var/log
                  fstype: ext4
                  mkfs-opts: -O large_file
              - name: crash
                  size: 10%
                  mount: /var/crash
                  fstype: ext4
                  mkfs-opts: -O large_file
```

4. Modify your `control_plane.yml` file, ensure you have the `NEUTRON-ROLE` defined as well as the Neutron services added.

   Path to file:

   ```
   ~/openstack/my_cloud/definition/data/control_plane.yml
   ```

   Example snippet:

   ```
    - allocation-policy: strict
      cluster-prefix: neut
      member-count: 1
      name: neut
      server-role: NEUTRON-ROLE
      service-components:
      - ntp-client
      - neutron-vpn-agent
      - neutron-dhcp-agent
      - neutron-metadata-agent
      - neutron-openvswitch-agent
   ```

You should also have one or more baremetal servers that meet the minimum hardware requirements for a network node which are documented in the *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 2 "Hardware and Software Support Matrix"*.

13.1.4.1.2   Adding a network node

These steps will show you how to add the new network node to your `servers.yml` file and then run the playbooks that update your cloud configuration. You will run these playbooks from the lifecycle manager.

1. Log in to your Cloud Lifecycle Manager.

2. Checkout the `site` branch of your local git so you can begin to make the necessary edits:

```
ardana > cd ~/openstack/my_cloud/definition/data
ardana > git checkout site
```

3. In the same directory, edit your `servers.yml` file to include the details about your new network node(s).

   For example, if you already had a cluster of three network nodes and needed to add a fourth one you would add your details to the bottom of the file in this format:

```
# network nodes
- id: neut3
  ip-addr: 10.13.111.137
  role: NEUTRON-ROLE
  server-group: RACK2
  mac-addr: "5c:b9:01:89:b6:18"
  nic-mapping: HP-DL360-6PORT
  ip-addr: 10.243.140.22
  ilo-ip: 10.1.12.91
  ilo-password: password
  ilo-user: admin
```

> **🔔 Important**
>
> You will need to verify that the `ip-addr` value you choose for this node does not conflict with any other IP address in your cloud environment. You can confirm this by checking the `~/openstack/my_cloud/info/address_info.yml` file on your Cloud Lifecycle Manager.

4. In your `control_plane.yml` file you will need to check the values for `member-count`, `min-count`, and `max-count`, if you specified them, to ensure that they match up with your new total node count. So for example, if you had previously specified `member-count: 3` and are adding a fourth network node, you will need to change that value to `member-count: 4`.

5. Commit the changes to git:

```
ardana > git commit -a -m "Add new networking node <name>"
```

6. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. Add the new node into Cobbler:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

9. Then you can image the node:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<hostname>
```

🎓 Note

If you do not know the `<hostname>`, you can get it by using **`sudo cobbler system list`**.

10. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your non-OS partitions on your nodes are completely wiped prior to continuing with the installation. The `wipe_disks.yml` playbook is only meant to be run on systems immediately after running `bm-reimage.yml`. If used for any other case, it may not wipe all of the expected partitions.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts wipe_disks.yml --limit <hostname>
```

11. Configure the operating system on the new networking node with this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit <hostname>
```

12. Complete the networking node deployment with this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/verb_hosts ardana-deploy.yml --limit <hostname>
```

13. Run the `site.yml` playbook with the required tag so that all other services become aware
of the new node:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts site.yml --tags "generate_hosts_file"
```

13.1.4.1.3 Adding a New Network Node to Monitoring

If you want to add a new networking node to the monitoring service checks, there is an additional
playbook that must be run to ensure this happens:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-deploy.yml --tags
 "active_ping_checks"
```

13.1.5 Planned Storage Maintenance

Planned maintenance procedures for Swift storage nodes.

13.1.5.1 Planned Maintenance Tasks for Swift Nodes

Planned maintenance tasks including recovering, adding, and removing Swift nodes.

13.1.5.1.1 Adding a Swift Object Node

Adding additional object nodes allows you to increase capacity.

This topic describes how to add additional Swift object server nodes to an existing system.

13.1.5.1.1.1 To add a new node

To add a new node to your cloud, you will need to add it to `servers.yml`, and then run the scripts that update your cloud configuration. To begin, access the `servers.yml file` by checking out the Git branch where you are required to make the changes:

Then, perform the following steps to add a new node:

1. Log in to the Cloud Lifecycle Manager node.

2. Get the `servers.yml` file stored in Git:

   ```
   cd ~/openstack/my_cloud/definition/data
   git checkout site
   ```

3. If not already done, set the `weight-step` attribute. For instructions, see *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"*.

4. Add the details of new nodes to the `servers.yml` file. In the following example only one new server **swobj4** is added. However, you can add multiple servers by providing the server details in the `servers.yml` file:

   ```
   servers:
   ...
   - id: swobj4
     role: SWOBJ_ROLE
     server-group: <server-group-name>
     mac-addr: <mac-address>
     nic-mapping: <nic-mapping-name>
     ip-addr: <ip-address>
     ilo-ip: <ilo-ip-address>
     ilo-user: <ilo-username>
     ilo-password: <ilo-password>
   ```

5. Commit your changes:

   ```
   git add -A
   git commit -m "Add Node <name>"
   ```

Note

Before you run any playbooks, remember that you need to export the encryption key in the following environment variable:

```
export HOS_USER_PASSWORD_ENCRYPT_KEY=ENCRYPTION_KEY
```

For instructions, see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 18 "Installation for HPE Helion OpenStack Entry-scale Cloud with Swift Only"*.

6. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. Create a deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. Configure Cobbler to include the new node, and then reimage the node (if you are adding several nodes, use a comma-separated list with the `nodelist` argument):

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<server-id>
```

In the following example, the server id is **swobj4** (mentioned in step 3):

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=swobj4
```

Note

You must use the server id as it appears in the file `servers.yml` in the field `server-id.`

9. Configure the operating system:

```
cd ~/scratch/ansible/next/ardana/ansible
```

```
ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit <hostname>
```

The hostname of the newly added server can be found in the list generated from the output of the following command:

```
grep hostname ~/openstack/my_cloud/info/server_info.yml
```

For example, for **swobj4**, the hostname is **ardana-cp1-swobj0004-mgmt**.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit ardana-cp1-swobj0004-
mgmt
```

10. Validate that the disk drives of the new node are compatible with the disk model used by the node:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml --limit SWF*
```

If any errors occur, correct them. For instructions, see *Section 15.6.2.3, "Interpreting Swift Input Model Validation Errors"*.

11. Run the following playbook to ensure that all other server's host file are updated with the new server:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts site.yml --tags "generate_hosts_file"
```

12. Run the `ardana-deploy.yml` playbook to rebalance the rings to include the node, deploy the rings, and configure the new node. Do not limit this to just the node (**swobj4**) that you are adding:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-deploy.yml
```

13. You may need to perform further rebalances of the rings. For instructions, see the "Weight Change Phase of Ring Rebalance" and the "Final Rebalance Phase" sections of *Section 8.5.5, "Applying Input Model Changes to Existing Rings"*.
For example:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-update-from-model-rebalance-rings.yml
```

13.1.5.1.2   Adding a Swift Proxy, Account, Container (PAC) Node

Steps for adding additional PAC nodes to your Swift system.

This topic describes how to add additional Swift proxy, account, and container (PAC) servers to an existing system.

13.1.5.1.2.1   Adding a new node

To add a new node to your cloud, you will need to add it to `servers.yml`, and then run the scripts that update your cloud configuration. To begin, access the `servers.yml file` by checking out the Git branch where you are required to make the changes:

Then, perform the following steps to add a new node:

1. Log in to the Cloud Lifecycle Manager.

2. Get the `servers.yml` file stored in Git:

    ```
    cd ~/openstack/my_cloud/definition/data
    git checkout site
    ```

3. If not already done, set the weight-step attribute. For instructions, see *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"*.

4. Add details of new nodes to the `servers.yml` file:

    ```
    servers:
    ...
    - id: swpac6
      role: SWPAC-ROLE
      server-group: <server-group-name>
      mac-addr: <mac-address>
      nic-mapping: <nic-mapping-name>
      ip-addr: <ip-address>
      ilo-ip: <ilo-ip-address>
      ilo-user: <ilo-username>
      ilo-password: <ilo-password>
    ```

    In the above example, only one new server **swpac6** is added. However, you can add multiple servers by providing the server details in the `servers.yml` file.

    In the entry-scale configurations there is no dedicated Swift PAC cluster. Instead, there is a cluster using servers that have a role of `CONTROLLER-ROLE`. You cannot add `swpac4` to this cluster because that would change the `member-count`. If your system does not already

have a dedicated Swift PAC cluster you will need to add it to the configuration files. For details on how to do this, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 11 "Modifying Example Configurations for Object Storage using Swift", Section 11.7 "Creating a Swift Proxy, Account, and Container (PAC) Cluster"*.

If using a new PAC nodes you must add the PAC node's configuration details in the following yaml files:

```
control_plane.yml
disks_pac.yml
net_interfaces.yml
servers.yml
server_roles.yml
```

You can see a good example of this in the example configurations for the mid-scale model in the `~/openstack/examples/mid-scale-kvm` directory.

The following steps assume that you have already created a dedicated Swift PAC cluster and that it has two members (**swpac4** and **swpac5**).

5. Increase the member count of the Swift PAC cluster, as appropriate. For example, if you are adding **swpac6** and you previously had two Swift PAC nodes, the increased member count should be 3 as shown in the following example:

```
control-planes:
    - name: control-plane-1
      control-plane-prefix: cp1

  . . .
  clusters:
  . . .
    - name: ....
      cluster-prefix: c2
      server-role: SWPAC-ROLE
      member-count: 3
  . . .
```

6. Commit your changes:

```
git add -A
git commit -m "Add Node <name>"
```

> **☞ Note**
>
> Before you run any playbooks, remember that you need to export the encryption key in the following environment variable:
>
> ```
> export HOS_USER_PASSWORD_ENCRYPT_KEY=ENCRYPTION_KEY
> ```
>
> For instructions, see *Book "Installing with Cloud Lifecycle Manager", Chapter 18 "Installation for HPE Helion OpenStack Entry-scale Cloud with Swift Only"*.

7. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

8. Create a deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

9. Configure Cobbler to include the new node and reimage the node (if you are adding several nodes, use a comma-separated list for the `nodelist` argument):

```
ansible-playbook -i hosts/localhost cobbler-deploy.yml
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<server-id>
```

In the following example, the server id is **swpac6** (mentioned in step 3):

```
ansible-playbook -i hosts/localhost cobbler-deploy.yml
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=swpac6
```

> **☞ Note**
>
> You must use the server id as it appears in the file `servers.yml` in the field `server-id`.

10. Review the `cloudConfig.yml` and `data/control_plane.yml` files to get the host prefix (for example, openstack) and the control plane name (for example, cp1). This gives you the hostname of the node. Configure the operating system:

```
cd ~/scratch/ansible/next/ardana/ansible
```

```
ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit <hostname>
```

For example, for **swpac6**, the hostname is **ardana-cp1-c2-m3-mgmt**:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit ardana-cp1-c2-m3-mgmt
```

11. Validate that the disk drives of the new node are compatible with the disk model used by the node:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml
```

If any errors occur, correct them. For instructions, see *Section 15.6.2.3, "Interpreting Swift Input Model Validation Errors"*.

12. Run the following playbook to ensure that all other server's host file are updated with the new server:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts site.yml --tags "generate_hosts_file"
```

13. Run the `ardana-deploy.yml` playbook to rebalance the rings to include the node, deploy the rings, and configure the new node. Do not limit this to just the node (**swpac6**) that you are adding:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-deploy.yml
```

14. You may need to perform further rebalances of the rings. For instructions, see the "Weight Change Phase of Ring Rebalance" and the "Final Rebalance Phase" sections of *Section 8.5.5, "Applying Input Model Changes to Existing Rings"*.

13.1.5.1.3   Adding Additional Disks to a Swift Node

Steps for adding additional disks to any nodes hosting Swift services.

You may have a need to add additional disks to a node for Swift usage and we can show you how. These steps work for adding additional disks to Swift object or proxy, account, container (PAC) nodes. It can also apply to adding additional disks to a controller node that is hosting the Swift service, like you would see if you are using one of the entry-scale example models.

Read through the notes below before beginning the process.

You can add multiple disks at the same time, there is no need to do it one at a time.

🔔 Important: Add the Same Number of Disks

You must add the *same* number of disks to each server that the disk model applies to. For example, if you have a single cluster of three Swift servers and you want to increase capacity and decide to add two additional disks, you must add two to each of your three Swift servers.

13.1.5.1.3.1 Adding additional disks to your Swift servers

1. Verify the general health of the Swift system and that it is safe to rebalance your rings. See *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"* for details on how to do this.

2. Perform the disk maintenance.

   a. Shut down the first Swift server you wish to add disks to.

   b. Add the additional disks to the physical server. The disk drives that are added should be clean. They should either contain no partitions or a single partition the size of the entire disk. It should not contain a file system or any volume groups. Failure to comply will cause errors and the disk will not be added.
   For more details, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 11 "Modifying Example Configurations for Object Storage using Swift", Section 11.6 "Swift Requirements for Device Group Drives"*.

   c. Power the server on.

   d. While the server was shutdown, data that normally would have been placed on the server is placed elsewhere. When the server is rebooted, the Swift replication process will move that data back onto the server. Monitor the replication process to determine when it is complete. See *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"* for details on how to do this.

   e. Repeat the steps from *Step 2.a* for each of the Swift servers you are adding the disks to, one at a time.

> **☞ Note**
>
> If the additional disks can be added to the Swift servers online (for example, via hotplugging) then there is no need to perform the last two steps.

3. On the Cloud Lifecycle Manager, update your cloud configuration with the details of your additional disks.

   a. Edit the disk configuration file that correlates to the type of server you are adding your new disks to.

   Path to the typical disk configuration files:

   ```
   ~/openstack/my_cloud/definition/data/disks_swobj.yml
   ~/openstack/my_cloud/definition/data/disks_swpac.yml
   ~/openstack/my_cloud/definition/data/disks_controller_*.yml
   ```

   Example showing the addition of a single new disk, indicated by the `/dev/sdd`, in bold:

   ```
   device-groups:
     - name: SwiftObject
       devices:
         - name: "/dev/sdb"
         - name: "/dev/sdc"
         - name: "/dev/sdd"
       consumer:
         name: swift
         ...
   ```

   > **☞ Note**
   >
   > For more details on how the disk model works, see *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 6 "Configuration Objects"*.

   b. Configure the Swift weight-step value in the `~/openstack/my_cloud/definition/data/swift/rings.yml` file. See *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"* for details on how to do this.

    c. Commit the changes to Git:

```
cd ~/openstack
git commit -a -m "adding additional Swift disks"
```

    d. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

    e. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

4. Run the `osconfig-run.yml` playbook against the Swift nodes you have added disks to. Use the `--limit` switch to target the specific nodes:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit <hostnames>
```

You can use a wildcard when specifying the hostnames with the `--limit` switch. If you added disks to all of the Swift servers in your environment and they all have the same prefix (for example, `ardana-cp1-swobj...`) then you can use a wildcard like `ardana-cp1-swobj*`. If you only added disks to a set of nodes but not all of them, you can use a comma deliminated list and enter the hostnames of each of the nodes you added disks to.

5. Validate your Swift configuration with this playbook which will also provide details of each drive being added:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml --extra-vars
  "drive_detail=yes"
```

6. Verify that Swift services are running on all of your servers:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-status.yml
```

7. If everything looks okay with the Swift status, then apply the changes to your Swift rings with this playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
```

```
ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

8. At this point your Swift rings will begin rebalancing. You should wait until replication has completed or min-part-hours has elapsed (whichever is longer), as described in *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"* and then follow the "Weight Change Phase of Ring Rebalance" process as described in *Section 8.5.5, "Applying Input Model Changes to Existing Rings"*.

13.1.5.1.4 Removing a Swift Node

Removal process for both Swift Object and PAC nodes.

You can use this process when you want to remove one or more Swift nodes permanently. This process applies to both Swift Proxy, Account, Container (PAC) nodes and Swift Object nodes.

13.1.5.1.4.1 Setting the Pass-through Attributes

This process will remove the Swift node's drives from the rings and move it to the remaining nodes in your cluster.

1. Log in to the Cloud Lifecycle Manager.

2. Ensure that the weight-step attribute is set. See *Section 8.5.2, "Using the Weight-Step Attributes to Prepare for Ring Changes"* for more details.

3. Add the pass-through definition to your input model, specifying the server ID (as opposed to the server name). It is easiest to include in your `~/openstack/my_cloud/definition/data/servers.yml` file since your server IDs are already listed in that file. For more information about pass-through, see *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 6 "Configuration Objects", Section 6.17 "Pass Through"*.
   Here is the general format:

```
pass-through:
  servers:
    - id: <server-id>
      data:
         <subsystem>:
             <subsystem-attributes>
```

Here is an example:

```
---
  product:
    version: 2

  pass-through:
    servers:
      - id: ccn-0001
        data:
          swift:
            drain: yes
```

By setting this pass-through attribute, you indicate that the system should reduce the weight of the server's drives. The weight reduction is determined by the weight-step attribute as described in the previous step. This process is known as "draining", where you remove the Swift data from the node in preparation for removing the node.

4. Commit your configuration to the local Git repository (see *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

5. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Use the playbook to create a deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Swift deploy playbook to perform the first ring rebuild. This will remove some of the partitions from all drives on the node you are removing:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

8. Wait until the replication has completed. For further details, see *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"*

9. Determine whether all of the partitions have been removed from all drives on the Swift node you are removing. You can do this by SSH'ing into the first account server node and using these commands:

```
cd /etc/swiftlm/cloud1/cp1/builder_dir/
sudo swift-ring-builder <ring_name>.builder
```

For example, if the node you are removing was part of the object-o ring the command would be:

```
sudo swift-ring-builder object-0.builder
```

Check the output. You will need to know the IP address of the server being drained. In the example below, the number of partitions of the drives on 192.168.245.3 has reached zero for the object-0 ring:

```
$ cd /etc/swiftlm/cloud1/cp1/builder_dir/
$ sudo swift-ring-builder object-0.builder
account.builder, build version 6
4096 partitions, 3.000000 replicas, 1 regions, 1 zones, 6 devices, 0.00 balance,
 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 16
The overload factor is 0.00% (0.000000)
Devices:    id  region  zone      ip address   port  replication ip   replication port
      name weight partitions balance meta
            0      1     1   192.168.245.3  6002   192.168.245.3              6002
     disk0  0.00            0   -0.00 padawan-ccp-c1-m1:disk0:/dev/sdc
            1      1     1   192.168.245.3  6002   192.168.245.3              6002
     disk1  0.00            0   -0.00 padawan-ccp-c1-m1:disk1:/dev/sdd
            2      1     1   192.168.245.4  6002   192.168.245.4              6002
     disk0 18.63         2048   -0.00 padawan-ccp-c1-m2:disk0:/dev/sdc
            3      1     1   192.168.245.4  6002   192.168.245.4              6002
     disk1 18.63         2048   -0.00 padawan-ccp-c1-m2:disk1:/dev/sdd
            4      1     1   192.168.245.5  6002   192.168.245.5              6002
     disk0 18.63         2048   -0.00 padawan-ccp-c1-m3:disk0:/dev/sdc
            5      1     1   192.168.245.5  6002   192.168.245.5              6002
     disk1 18.63         2048   -0.00 padawan-ccp-c1-m3:disk1:/dev/sdd
```

10. If the number of partitions is zero for the server on all rings, you can move to the next step, otherwise continue the ring rebalance cycle by repeating steps 7-9 until the weight has reached zero.

11. If the number of partitions is zero for the server on all rings, you can remove the Swift nodes' drives from all rings. Edit the pass-through data you created in step #3 and set the `remove` attribute as shown in this example:

```
---
  product:
    version: 2

  pass-through:
    servers:
      - id: ccn-0001
        data:
          swift:
            remove: yes
```

12. Commit your configuration to the local Git repository (see *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

13. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

14. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

15. Run the Swift deploy playbook to rebuild the rings by removing the server:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

16. At this stage, the server has been removed from the rings and the data that was originally stored on the server has been replicated in a balanced way to the other servers in the system. You can proceed to the next phase.

13.1.5.1.4.2 To Disable Swift on a Node

The next phase in this process will disable the Swift service on the node. In this example, **swobj4** is the node being removed from Swift.

1. Log in to the Cloud Lifecycle Manager.

2. Stop Swift services on the node using the `swift-stop.yml` playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-stop.yml --limit <hostname>
```

> **Note**
>
> When using the `--limit` argument, you must specify the full hostname (for example: *ardana-cp1-swobj0004*) or use the wild card `*` (for example, **swobj4**).

The following example uses the `swift-stop.yml` playbook to stop Swift services on **ardana-cp1-swobj0004**:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-stop.yml --limit ardana-cp1-swobj0004
```

3. Remove the configuration files.

```
ssh ardana-cp1-swobj4-mgmt sudo rm -R /etc/swift
```

> **Note**
>
> Do not run any other playbooks until you have finished the process described in *Section 13.1.5.1.4.3, "To Remove a Node from the Input Model"*. Otherwise, these playbooks may recreate `/etc/swift` and restart Swift on *swobj4*. If you accidentally run a playbook, repeat the process in *Section 13.1.5.1.4.2, "To Disable Swift on a Node"*.

13.1.5.1.4.3 To Remove a Node from the Input Model

Use the following steps to finish the process of removing the Swift node.

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/definition/data/servers.yml` file and remove the entry for the node (**swobj4** in this example).

3. If this was a SWPAC node, reduce the member-count attribute by 1 in the `~/openstack/my_cloud/definition/data/control_plane.yml` file. For SWOBJ nodes, no such action is needed.

4. Commit your configuration to the local Git repository (see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 10 "Using Git for Configuration Management"*), as follows:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

5. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

You may want to use the `remove_deleted_servers` and `free_unused_addresses` switches to free up the resources when running the configuration processor. For more information, see *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 7 "Other Topics", Section 7.3 "Persisted Data"*.

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e
  remove_deleted_servers="y" -e free_unused_addresses="y"
```

6. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Validate the changes you have made to the configuration files using the playbook below before proceeding further:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml --limit SWF*
```

If any errors occur, correct them in your configuration files and repeat steps 3-5 again until no more errors occur before going to the next step.

For more details on how to interpret and resolve errors, see *Section 15.6.2.3, "Interpreting Swift Input Model Validation Errors"*

8. Remove the node from Cobbler:

```
sudo cobbler system remove --name=swobj4
```

9. Run the Cobbler deploy playbook:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

10. The final step will depend on what type of Swift node you are removing.

    If the node was a SWPAC node, run the `ardana-deploy.yml` playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-deploy.yml
```

    If the node was a SWOBJ node, run the `swift-deploy.yml` playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-deploy.yml
```

11. Wait until replication has finished. For more details, see *Section 8.5.4, "Determining When to Rebalance and Deploy a New Ring"*.

12. You may need to continue to rebalance the rings. For instructions, see **Final Rebalance Phase** at *Section 8.5.5, "Applying Input Model Changes to Existing Rings"*.

13.1.5.1.4.4 Remove the Swift Node from Monitoring

Once you have removed the Swift node(s), the alarms against them will trigger so there are additional steps to take to resolve this issue.

You will want to SSH to each of the Monasca API servers and edit the `/etc/monasca/agent/conf.d/host_alive.yaml` file to remove references to the Swift node(s) you removed. This will require `sudo` access.

Once you have removed the references on each of your Monasca API servers you then need to restart the monasca-agent on each of those servers with this command:

```
tux > sudo service openstack-monasca-agent restart
```

With the Swift node references removed and the monasca-agent restarted, you can then delete the corresponding alarm to finish this process. To do so we recommend using the Monasca CLI which should be installed on each of your Monasca API servers by default:

```
monasca alarm-list --metric-dimensions hostname=<swift node deleted>
```

You can then delete the alarm with this command:

```
monasca alarm-delete <alarm ID>
```

13.1.5.1.5   Replacing a Swift Node

Maintenance steps for replacing a failed Swift node in your environment.

This process is used when you want to replace a failed Swift node in your cloud.

> ⛨ **Warning**
>
> If it applies to the server, do not skip step 10. If you do, the system will overwrite the existing rings with new rings. This will not cause data loss, but, potentially, will move most objects in your system to new locations and may make data unavailable until the replication process has completed.

13.1.5.1.5.1   How to replace a Swift node in your environment

1. Log in to the Cloud Lifecycle Manager.

2. Update your cloud configuration with the details of your replacement Swift node.

    a. Edit your `servers.yml` file to include the details (MAC address, IPMI user, password, and IP address (IPME) if these have changed) about your replacement Swift node.

    > ☞ **Note**
    >
    > Do not change the server's IP address (that is, `ip-addr`).

    Path to file:

    ```
    ~/openstack/my_cloud/definition/data/servers.yml
    ```

Example showing the fields to edit, in bold:

```
- id: swobj5
  role: SWOBJ-ROLE
  server-group: rack2
  mac-addr: 8c:dc:d4:b5:cb:bd
  nic-mapping: HP-DL360-6PORT
  ip-addr: 10.243.131.10
  ilo-ip: 10.1.12.88
  ilo-user: iLOuser
  ilo-password: iLOpass
  ...
```

b. Commit the changes to Git:

```
cd ~/openstack
git commit -a -m "replacing a Swift node"
```

c. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

d. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Update Cobbler and reimage your replacement Swift node:

a. Obtain the name in Cobbler for your node you wish to remove. You will use this value to replace <node name> in future steps.

```
sudo cobbler system list
```

b. Remove the replaced Swift node from Cobbler:

```
sudo cobbler system remove --name <node name>
```

c. Re-run the cobbler-deploy.yml playbook to add the replaced node:

```
cd ~/scratch/ansible/next/ardana/ansible
```

```
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

d. Reimage the node using this playbook:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<node name>
```

4. Complete the deployment of your replacement Swift node.

a. Obtain the hostname for your new Swift node. You will use this value to replace `<hostname>` in future steps.

```
cat ~/openstack/my_cloud/info/server_info.yml
```

b. Configure the operating system on your replacement Swift node:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts osconfig-run.yml -limit <hostname>
```

c. If this is the Swift ring builder server, restore the Swift ring builder files to the `/etc/swiftlm/CLOUD-NAME/CONTROL-PLANE-NAME/builder_dir` directory. For more information and instructions, see *Section 15.6.2.4, "Identifying the Swift Ring Building Server"* and *Section 15.6.2.7, "Recovering Swift Builder Files"*.

d. Configure services on the node using the `ardana-deploy.yml` playbook. If you have used an encryption password when running the configuration processor, include the `--ask-vault-pass` argument.

```
ansible-playbook -i hosts/verb_hosts ardana-deploy.yml --ask-vault-pass --limit
  <hostname>
```

13.1.5.1.6   Replacing Drives in a Swift Node

Maintenance steps for replacing drives in a Swift node.

This process is used when you want to remove a failed hard drive from Swift node and replace it with a new one.

There are two different classes of drives in a Swift node that needs to be replaced; the operating system disk drive (generally **/dev/sda**) and storage disk drives. There are different procedures for the replacement of each class of drive to bring the node back to normal.

13.1.5.1.6.1 To Replace the Operating System Disk Drive

After the operating system disk drive is replaced, the node must be reimaged.

1. Log in to the Cloud Lifecycle Manager.

2. Update your Cobbler profile:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/localhost cobbler-deploy.yml
   ```

3. Reimage the node using this playbook:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<server name>
   ```

   In the example below **swobj2** server is reimaged:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=swobj2
   ```

4. Review the `cloudConfig.yml` and `data/control_plane.yml` files to get the host prefix
   (for example, openstack) and the control plane name (for example, cp1). This gives you
   the hostname of the node. Configure the operating system:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit <hostname>
   ```

   In the following example, for **swobj2**, the hostname is **ardana-cp1-swobj0002**:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts osconfig-run.yml -limit ardana-cp1-swobj0002*
   ```

5. If this is the first server running the swift-proxy service, restore the Swift Ring Builder
   files to the `/etc/swiftlm/`*CLOUD-NAME/CONTROL-PLANE-NAME*`/builder_dir` directory.
   For more information and instructions, see *Section 15.6.2.4, "Identifying the Swift Ring Building
   Server"* and *Section 15.6.2.7, "Recovering Swift Builder Files"*.

6. Configure services on the node using the `ardana-deploy.yml` playbook. If you have used
   an encryption password when running the configuration processor include the `--ask-vault-pass` argument.

   ```
   ansible-playbook -i hosts/verb_hosts ardana-deploy.yml --ask-vault-pass \
   ```

```
  --limit <hostname>
```

For example:

```
ansible-playbook -i hosts/verb_hosts ardana-deploy.yml --ask-vault-pass --limit
 ardana-cp1-swobj0002*
```

13.1.5.1.6.2   To Replace a Storage Disk Drive

After a storage drive is replaced, there is no need to reimage the server. Instead, run the `swift-reconfigure.yml` playbook.

1. Log onto the Cloud Lifecycle Manager.

2. Run the following commands:

    ```
    cd ~/scratch/ansible/next/ardana/ansible
    ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml --limit <hostname>
    ```

    In following example, the server used is **swobj2**:

    ```
    cd ~/scratch/ansible/next/ardana/ansible
    ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml --limit ardana-cp1-
    swobj0002-mgmt
    ```

13.1.6   Updating MariaDB with Galera

Updating MariaDB with Galera must be done manually. Updates are not installed automatically. In particular, this situation applies to upgrades to MariaDB 10.2.17 or higher from MariaDB 10.2.16 or earlier. See MariaDB 10.2.22 Release Notes - Notable Changes (https://mariadb.com/kb/en/library/mariadb-10222-release-notes/) ↗ .

Using the CLI, update MariaDB with the following procedure:

1. Mark Galera as unmanaged:

    ```
    crm resource unmanage galera
    ```

    Or put the whole cluster into maintenance mode:

    ```
    crm configure property maintenance-mode=true
    ```

2. Pick a node other than the one currently targeted by the loadbalancer and stop MariaDB on that node:

```
crm_resource --wait --force-demote -r galera -V
```

3. Perform updates:

   a. Uninstall the old versions of MariaDB and the Galera wsrep provider.

   b. Install the new versions of MariaDB and the Galera wsrep provider. Select the appropriate instructions at Installing MariaDB with zypper (https://mariadb.com/kb/en/library/installing-mariadb-with-zypper/) ⌝.

   c. Change configuration options if necessary.

4. Start MariaDB on the node.

```
crm_resource --wait --force-promote -r galera -V
```

5. Run **mysql_upgrade** with the ‑‑skip‑write‑binlog option.

6. On the other nodes, repeat the process detailed above: stop MariaDB, perform updates, start MariaDB, run **mysql_upgrade**.

7. Mark Galera as managed:

```
crm resource manage galera
```

Or take the cluster out of maintenance mode.

13.2 Unplanned System Maintenance

Unplanned maintenance tasks for your cloud.

13.2.1 Whole Cloud Recovery Procedures

Unplanned maintenance procedures for your whole cloud.

13.2.1.1 Full Disaster Recovery

In this disaster scenario, you have lost everything in the cloud, including Swift.

13.2.1.1.1 Restore from a Swift backup:

Restoring from a Swift backup is not possible because Swift is gone.

13.2.1.1.2 Restore from an SSH backup:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the following file so it contains the same information as it had previously:

   ```
   ~/openstack/my_cloud/config/freezer/ssh_credentials.yml
   ```

3. On the Cloud Lifecycle Manager copy the following files:

   ```
   cp -r ~/hp-ci/openstack/* ~/openstack/my_cloud/definition/
   ```

4. Run this playbook to restore the Cloud Lifecycle Manager helper:

   ```
   cd ~/openstack/ardana/ansible/
   ansible-playbook -i hosts/localhost _deployer_restore_helper.yml
   ```

5. Run as root, and change directories:

   ```
   sudo su
   cd /root/deployer_restore_helper/
   ```

6. Execute the restore:

   ```
   ./deployer_restore_script.sh
   ```

7. Run this playbook to deploy your cloud:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ```

```
ansible-playbook -i hosts/verb_hosts site.yml -e '{ "freezer_backup_jobs_upload":
  false }'
```

8. You can now perform the procedures to restore MySQL and Swift. Once everything is restored, re-enable the backups from the Cloud Lifecycle Manager:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
```

13.2.1.2   Full Disaster Recovery Test

Full Disaster Recovery Test

13.2.1.2.1   Prerequisites

HPE Helion OpenStack platform

An external server to store backups to via SSH

13.2.1.2.2   Goals

Here is a high level view of how we expect to test the disaster recovery of the platform.

1. Backup the control plane using Freezer to an SSH target

2. Backup the Cassandra Database

3. Re-install Controller 1 with the HPE Helion OpenStack ISO

4. Use Freezer to recover deployment data (model …)

5. Re-install HPE Helion OpenStack on Controller 1, 2, 3

6. Recover the Cassandra Database

7. Recover the backup of the MariaDB database

13.2.1.2.3   Description of the testing environment

The testing environment is very similar to the Entry Scale model.

It used 5 servers: 3 Controllers and 2 computes.

The controller node have three disks. The first one is reserved for the system, while others are used for swift.

☞ Note

During this Disaster Recovery exercise, we have saved the data on disk 2 and 3 of the swift controllers.

This allow to restore the swift objects after the recovery.

If these disks were to be wiped as well, swift data would be lost but the procedure would not change.

The only difference is that Glance images would be lost and they will have to be re-uploaded.

13.2.1.2.4 Disaster recovery test note

If it is not specified otherwise, all the commands should be executed on controller 1, which is also the deployer node.

13.2.1.2.5 Pre-Disaster testing

In order to validate the procedure after recovery, we need to create some workloads.

1. Source the service credential file

   ```
   ardana > source ~/service.osrc
   ```

2. Copy an image to the platform and create a Glance image with it. In this example, Cirros is used

   ```
   ardana > openstack image create --disk-format raw --container-format bare --public
    --file ~/cirros-0.3.5-x86_64-disk.img cirros
   ```

3. Create a network

   ```
   ardana > openstack network create test_net
   ```

4. Create a subnet

```
ardana > neutron subnet-create 07c35d11-13f9-41d4-8289-fa92147b1d44 192.168.42.0/24
  --name test_subnet
```

5. Create some instances

```
ardana > openstack server create server_1 --image 411a0363-7f4b-4bbc-889c-
b9614e2da52e --flavor m1.small --nic net-id=07c35d11-13f9-41d4-8289-fa92147b1d44
ardana > openstack server create server_2 --image 411a0363-7f4b-4bbc-889c-
b9614e2da52e --flavor m1.small --nic net-id=07c35d11-13f9-41d4-8289-fa92147b1d44
ardana > openstack server create server_3 --image 411a0363-7f4b-4bbc-889c-
b9614e2da52e --flavor m1.small --nic net-id=07c35d11-13f9-41d4-8289-fa92147b1d44
ardana > openstack server create server_4 --image 411a0363-7f4b-4bbc-889c-
b9614e2da52e --flavor m1.small --nic net-id=07c35d11-13f9-41d4-8289-fa92147b1d44
ardana > openstack server create server_5 --image 411a0363-7f4b-4bbc-889c-
b9614e2da52e --flavor m1.small --nic net-id=07c35d11-13f9-41d4-8289-fa92147b1d44
ardana > openstack server list
```

6. Create containers and objects

```
ardana > swift upload container_1 ~/service.osrc
var/lib/ardana/service.osrc

ardana > swift upload container_1 ~/backup.osrc
swift upload container_1 ~/backup.osrc

ardana > swift list container_1
var/lib/ardana/backup.osrc
var/lib/ardana/service.osrc
```

13.2.1.2.6   Preparation of the backup server

Preparation of the backup server

13.2.1.2.6.1   Preparation to store Freezer backups

In this example, we want to store the backups on the server 192.168.69.132

Freezer will connect with the user backupuser on port 22 and store the backups in the `/mnt/ backups/` directory.

1. Connect to the backup server

2. Create the user

```
root # useradd backupuser --create-home --home-dir /mnt/backups/
```

3. Switch to that user

```
root # su backupuser
```

4. Create the SSH keypair

```
backupuser > ssh-keygen -t rsa
> # Just leave the default for the first question and do not set any passphrase
> Generating public/private rsa key pair.
> Enter file in which to save the key (/mnt/backups//.ssh/id_rsa):
> Created directory '/mnt/backups//.ssh'.
> Enter passphrase (empty for no passphrase):
> Enter same passphrase again:
> Your identification has been saved in /mnt/backups//.ssh/id_rsa
> Your public key has been saved in /mnt/backups//.ssh/id_rsa.pub
> The key fingerprint is:
> a9:08:ae:ee:3c:57:62:31:d2:52:77:a7:4e:37:d1:28 backupuser@padawan-ccp-c0-m1-mgmt
> The key's randomart image is:
> +---[RSA 2048]----+
> |          o      |
> |    . . E + .    |
> |   o . . + .     |
> | o +   o +       |
> |  + o o S .      |
> | . + o o         |
> |   o + .         |
> |.o .             |
> |++o              |
> +-----------------+
```

5. Add the public key to the list of the keys authorized to connect to that user on this server

```
backupuser > cat /mnt/backups/.ssh/id_rsa.pub >> /mnt/backups/.ssh/authorized_keys
```

6. Print the private key. This is what we will use for the backup configuration (ssh_credentials.yml file)

```
backupuser > cat /mnt/backups/.ssh/id_rsa

> -----BEGIN RSA PRIVATE KEY-----
> MIIEogIBAAKCAQEAvjwKu6f940IVGHpUj3ffl3eKXACgVr3L5s9UJnb15+zV3K5L
> BZuor8MLvwtskSkgdXNrpPZhNCsWSkryJff5I335Jhr/e5o03Yy+RqIMrJAIa0X5
> ...
```

```
> ...
> ...
> iBKVKGPhOnn4ve3dDqy3q7fS5sivTqCrpaYtByJmPrcJNjb2K7VMLNvgLamK/AbL
> qpSTZjicKZCCl+J2+8lrKAaDWqWtIjSUs29kCL78QmaPOgEvfsw=
> -----END RSA PRIVATE KEY-----
```

13.2.1.2.6.2 Preparation to store Cassandra backups

In this example, we want to store the backups on the server 192.168.69.132. We will store the backups in the `/mnt/backups/cassandra_backups/` directory.

1. Create a directory on the backup server to store cassandra backups

```
backupuser > mkdir /mnt/backups/cassandra_backups
```

2. Copy private ssh key from backupserver to all controller nodes

```
backupuser > scp /mnt/backups/.ssh/id_rsa ardana@CONTROLLER:~/.ssh/id_rsa_backup
        Password:
        id_rsa     100% 1675     1.6KB/s   00:00
```

Replace CONTROLLER with each control node e.g. doc-cp1-c1-m1-mgmt, doc-cp1-c1-m2-mgmt etc

3. Login to each controller node and copy private ssh key to the root user's .ssh directory

```
tux > sudo cp /var/lib/ardana/.ssh/id_rsa_backup /root/.ssh/
```

4. Verify that you can ssh to backup server as backup user using the private key

```
root # ssh -i ~/.ssh/id_rsa_backup backupuser@doc-cp1-comp0001-mgmt
```

13.2.1.2.7 Perform Backups for disaster recovery test

Perform Backups for disaster recovery

13.2.1.2.7.1 Execute backup of Cassandra

Execute backup of Cassandra

Create cassandra-backup-extserver.sh script on all controller nodes where Cassandra runs, which can be determined by running this command on deployer

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible FND-CDB --list-hosts
```

```
root # cat > ~/cassandra-backup-extserver.sh << EOF
#!/bin/sh

# backup user
BACKUP_USER=backupuser
# backup server
BACKUP_SERVER=192.168.69.132
# backup directory
BACKUP_DIR=/mnt/backups/cassandra_backups/

# Setup variables
DATA_DIR=/var/cassandra/data/data
NODETOOL=/usr/bin/nodetool

# e.g. cassandra-snp-2018-06-26-1003
SNAPSHOT_NAME=cassandra-snp-\$(date +%F-%H%M)
HOST_NAME=\$(/bin/hostname)_

# Take a snapshot of cassandra database
\$NODETOOL snapshot -t \$SNAPSHOT_NAME monasca

# Collect a list of directories that make up the snapshot
SNAPSHOT_DIR_LIST=\$(find \$DATA_DIR -type d -name \$SNAPSHOT_NAME)
for d in \$SNAPSHOT_DIR_LIST
  do
    # copy snapshot directories to external server
    rsync -avR -e "ssh -i /root/.ssh/id_rsa_backup" \$d \$BACKUP_USER@\$BACKUP_SERVER:\
$BACKUP_DIR/\$HOST_NAME\$SNAPSHOT_NAME
  done

\$NODETOOL clearsnapshot monasca
EOF
```

```
root # chmod +x ~/cassandra-backup-extserver.sh
```

Execute following steps on all the controller nodes

☞ Note

/usr/local/sbin/cassandra-backup-extserver.sh should be executed on all the three controller nodes at the same time (within seconds of each other) for a successful backup

1. Edit /usr/local/sbin/cassandra-backup-extserver.sh script

   Set `BACKUP_USER` and `BACKUP_SERVER` to the desired backup user (for example, `backupuser`) and desired backup server (for example, `192.168.68.132`), respectively.

   ```
   BACKUP_USER=backupuser
   BACKUP_SERVER=192.168.69.132
   BACKUP_DIR=/mnt/backups/cassandra_backups/
   ```

2. Execute ~/cassandra-backup-extserver.sh

   ```
   root # ~/cassandra-backup-extserver.sh (on all controller nodes which are also
    cassandra nodes)

   Requested creating snapshot(s) for [monasca] with snapshot name [cassandra-
   snp-2018-06-28-0251] and options {skipFlush=false}
   Snapshot directory: cassandra-snp-2018-06-28-0251
   sending incremental file list
   created directory /mnt/backups/cassandra_backups//doc-cp1-c1-m1-mgmt_cassandra-
   snp-2018-06-28-0251
   /var/
   /var/cassandra/
   /var/cassandra/data/
   /var/cassandra/data/data/
   /var/cassandra/data/data/monasca/

   ...
   ...
   ...

   /var/cassandra/data/data/monasca/measurements-e29033d0488d11e8bdabc32666406af1/
   snapshots/cassandra-snp-2018-06-28-0306/mc-72-big-Summary.db
   /var/cassandra/data/data/monasca/measurements-e29033d0488d11e8bdabc32666406af1/
   snapshots/cassandra-snp-2018-06-28-0306/mc-72-big-TOC.txt
   /var/cassandra/data/data/monasca/measurements-e29033d0488d11e8bdabc32666406af1/
   snapshots/cassandra-snp-2018-06-28-0306/schema.cql
   sent 173,691 bytes  received 531 bytes  116,148.00 bytes/sec
   total size is 171,378  speedup is 0.98
   Requested clearing snapshot(s) for [monasca]
   ```

3. Verify cassandra backup directory on backup server

   ```
   backupuser > ls -alt /mnt/backups/cassandra_backups
   total 16
   drwxr-xr-x 4 backupuser users 4096 Jun 28 03:06 .
   drwxr-xr-x 3 backupuser users 4096 Jun 28 03:06 doc-cp1-c1-m2-mgmt_cassandra-
   snp-2018-06-28-0306
   ```

```
drwxr-xr-x 3 backupuser users 4096 Jun 28 02:51 doc-cp1-c1-m1-mgmt_cassandra-
snp-2018-06-28-0251
drwxr-xr-x 8 backupuser users 4096 Jun 27 20:56 ..

$backupuser@backupserver> du -shx /mnt/backups/cassandra_backups/*
6.2G    /mnt/backups/cassandra_backups/doc-cp1-c1-m1-mgmt_cassandra-
snp-2018-06-28-0251
6.3G    /mnt/backups/cassandra_backups/doc-cp1-c1-m2-mgmt_cassandra-
snp-2018-06-28-0306
```

13.2.1.2.7.2   Execute backup of HPE Helion OpenStack

Execute backup of HPE Helion OpenStack

1. Edit the configuration file for SSH backups (be careful to format the private key as requested: pipe on the first line and two spaces indentation). The private key is the key we created on the backup server earlier.

```
ardana > vi ~/openstack/my_cloud/config/freezer/ssh_credentials.yml

$ cat ~/openstack/my_cloud/config/freezer/ssh_credentials.yml
freezer_ssh_host: 192.168.69.132
freezer_ssh_port: 22
freezer_ssh_username: backupuser
freezer_ssh_base_dir: /mnt/backups
freezer_ssh_private_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEowIBAAKCAQEAyzhZ+F+sXQp70N8zCDDb6ORKAxreT/qD4zAetjOTuBoFlGb8
  pRBY79t9vNp7qvrKaXHBfb1OkKzhqyUwEqNcC9bdngABbb8KkCq+OkfDSAZRrmja
  wa5PzgtSaZcSJm9jQcF04Fq19mZY2BLK3OJL4qISp1DmN3ZthgJcpksYid2G3YG+
  bY/EogrQrdgHfcyLaoEkiBWQSBTEENKTKFBB2jFQYdmif3KaeJySv9cJqihmyotB
  s5YTdvB5Zn/fFCKG66THhKnIm19NftbJcKc+Y3Z/ZX4W9SpMSj5dL2YW0Y176mLy
  gMLyZK9u5k+fVjYLqY7XlVAFalv9+HZsvQ3OQQIDAQABAoIBACfUkqXAsrrFrEDj
  DlCDqwZ5gBwdrwcD9ceYjdxuPXyu9PsCOHBtxNC2N23FcMmxP+zs09y+NuDaUZzG
  vCZbCFZ1tZgbLiyBbiOVjRVFLXw3aNkDSiT98jxTMcLqTi9kU5L2xN6YSOPTaYRo
  IoSqge8YjwlmLMkgGBVU7y3UuCmE/Rylclb1EI9mMPElTF+87tYK9IyA2QbIJm/w
  4aZugSZa3PwUvKGG/TCJVD+JfrZ1kCz6MFnNS1jYT/cQ6nzLsQx7UuYLgpvTMDK6
  Fjq63TmVg9Z1urTB4dqhxzpDbTNfJrV55MuA/z9/qFHs649tFB1/hCsG3EqWcDnP
  mcv79nECgYEA9WdOsDnnCI1bamKA0XZxovb2rpYZyRakv3GujjqDrYTI97zoG+Gh
  gLcD1EMLnLLQWAkDTITIf8eurkVLKzhb1xlN0Z4xCLs7ukgMetlVWfNrcYEkzGa8
  wec7n1LfHcH5BNjjancRH0Q1Xcc2K7UgGe2iw/Iw67wlJ8i5j2Wq3sUCgYEA0/6/
  irdJzFB/9aTC8SFWbqj1DdyrpjJPm4yZeXkRAdn2GeLU2jefqPtxYwMCB1goeORc
  gQLspQpxeDvLdiQod1Y1aTAGYOcZOyAatIlOqiI40y3Mmj8YU/KnL7NMkaYBCrJh
  aW//xo+l20dz52pONzLFjw1tW9vhCsG1QlrCaU0CgYB03qUn4ft4JDHUAWNN3fWS
  YcDrNkrDbIg7MD2sOIu7WFCJQyrbFGJgtUgaj295SeNU+b3bdCU0TXmQPynkRGvg
```

```
jYl0+bxqZxizx1pCKzytoPKbVKCcw5TDV4caglIFjvoz58KuUlQSKt6rcZMHz7Oh
BX4NiUrpCWo8fyh39Tgh7QKBgEUajm92Tc0XFI8LNSyK9HTACJmLLDzRu5d13nV1
XHDhDtLjWQUFCrt3sz9WNKwWNaMqtWisfl1SKSjLPQh2wuYbqO9v4zRlQJlAXtQo
yga1fxZ/oGlLVe/PcmYfKT91AHPvL8fB5XthSexPv11ZDsP5feKiutots47hE+fc
U/ElAoGBAItNX4jpUfnaOj0mR0L+2R2XNmC5b4PrMhH/+XRRdSr1t76+RJ23MDwf
SV3u3/30eS7Ch2OV9o9lr0sjMKRgBsLZcaSmKp9K0j/sotwBl0+C4nauZMUKDXqg
uGCyWeTQdAOD9QblzGoWy6g3ZI+XZWQIMt0pH38d/ZRbuSUk5o5v
-----END RSA PRIVATE KEY-----
```

2. Save the modifications in the GIT repository

```
ardana > cd ~/openstack/
ardana > git add -A
ardana > git commit -a -m "SSH backup configuration"
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Create the Freezer jobs

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
```

4. Wait until all the SSH backup jobs have finished running

   Freezer backup jobs are scheduled at interval specified in job specification

   You will have to wait for the scheduled time interval for the backup job to run

   To find the interval:

```
ardana > freezer job-list | grep SSH

| 34c1364692f64a328c38d54b95753844 | Ardana Default: deployer backup to SSH      |
      7 | success | scheduled |        |           |
| 944154642f624bb7b9ff12c573a70577 | Ardana Default: swift backup to SSH         |
      1 | success | scheduled |        |           |
| 22c6bab7ac4d43debcd4f5a9c4c4bb19 | Ardana Default: mysql backup to SSH         |
      1 | success | scheduled |        |           |

ardana > freezer job-show 944154642f624bb7b9ff12c573a70577
+-------------
+-----------------------------------------------------------------------------+
| Field       | Value
           |
+-------------
+-----------------------------------------------------------------------------+
| Job ID      | 944154642f624bb7b9ff12c573a70577
           |
```

```
| Client ID   | ardana-qe201-cp1-c1-m1-mgmt
             |
| User ID     | 33a6a77adc4b4799a79a4c3bd40f680d
             |
| Session ID  |
             |
| Description | Ardana Default: swift backup to SSH
             |
| Actions     | [{u'action_id': u'e8373b03ca4b41fdafd83f9ba7734bfa',
             |
|             |    u'freezer_action': {u'action': u'backup',
             |
|             |                         u'backup_name':
 u'freezer_swift_builder_dir_backup',        |
|             |                         u'container': u'/mnt/backups/
freezer_rings_backups',        |
|             |                         u'log_config_append': u'/etc/freezer/agent-
logging.conf', |
|             |                         u'max_level': 14,
             |
|             |                         u'path_to_backup': u'/etc/swiftlm/',
             |
|             |                         u'remove_older_than': 90,
             |
|             |                         u'snapshot': True,
             |
|             |                         u'ssh_host': u'192.168.69.132',
             |
|             |                         u'ssh_key': u'/etc/freezer/ssh_key',
             |
|             |                         u'ssh_port': u'22',
             |
|             |                         u'ssh_username': u'backupuser',
             |
|             |                         u'storage': u'ssh'},
             |
|             |    u'max_retries': 5,
             |
|             |    u'max_retries_interval': 60,
             |
|             |    u'user_id': u'33a6a77adc4b4799a79a4c3bd40f680d'}]
             |
| Start Date  |
             |
| End Date    |
             |
```

```
| Interval    | 24 hours
             |
+-------------
  +--------------------------------------------------------------------------+
```

Swift SSH backup job has Interval of 24 hours, so the next backup would run after 24 hours. In the default installation Interval for various backup jobs are:

TABLE 13.1: DEFAULT INTERVAL FOR FREEZER BACKUP JOBS

Job Name	Interval
Ardana Default: deployer backup to SSH	48 hours
Ardana Default: mysql backup to SSH	12 hours
Ardana Default: swift backup to SSH	24 hours

You will have to wait for as long as 48 hours for all the backup jobs to run

5. On the backup server, you can verify that the backup files are present

```
backupuser > ls -lah  /mnt/backups/
total 16
drwxr-xr-x 2 backupuser users 4096 Jun 27  2017 bin
drwxr-xr-x 2 backupuser users 4096 Jun 29 14:04 freezer_database_backups
drwxr-xr-x 2 backupuser users 4096 Jun 29 14:05 freezer_lifecycle_manager_backups
drwxr-xr-x 2 backupuser users 4096 Jun 29 14:05 freezer_rings_backups
```

```
backupuser > du -shx *
4.0K    bin
509M    freezer_audit_logs_backups
2.8G    freezer_database_backups
24G     freezer_lifecycle_manager_backups
160K    freezer_rings_backups
```

13.2.1.2.8   Restore of the first controller

Restore of the first controller

1. Edit the SSH backup configuration (re-enter the same information as earlier)

```
ardana > vi ~/openstack/my_cloud/config/freezer/ssh_credentials.yml
```

2. Execute the restore helper. When prompted, enter the hostname the first controller had.
   In this example: `doc-cp1-c1-m1-mgmt`

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost _deployer_restore_helper.yml
```

3. Execute the restore. When prompted, leave the first value empty (none) and validate the
   restore by typing 'yes'.

```
ardana > sudo su
cd /root/deployer_restore_helper/
./deployer_restore_script.sh
```

4. Create a restore file for Swift rings

```
ardana > nano swift_rings_restore.ini
ardana > cat swift_rings_restore.ini
```

Help:

```
[default]
action = restore
storage = ssh
# backup server ip
ssh_host = 192.168.69.132
# username to connect to the backup server
ssh_username = backupuser
ssh_key = /etc/freezer/ssh_key
# base directory for backups on the backup server
container = /mnt/backups/freezer_ring_backups
backup_name = freezer_swift_builder_dir_backup
restore_abs_path = /etc/swiftlm
log_file = /var/log/freezer-agent/freezer-agent.log
# hostname that the controller
hostname = doc-cp1-c1-m1-mgmt
overwrite = True
```

5. Execute the restore of the swift rings

```
ardana > freezer-agent --config ./swift_rings_restore.ini
```

13.2.1.2.9   Re-deployment of controllers 1, 2 and 3

Re-deployment of controllers 1, 2 and 3

1. Change back to the default ardana user

2. Deactivate the freezer backup jobs (otherwise empty backups would be added on top of the current good backups)

```
ardana > nano ~/openstack/my_cloud/config/freezer/activate_jobs.yml
ardana > cat ~/openstack/my_cloud/config/freezer/activate_jobs.yml

# If set to false, We wont create backups jobs.
freezer_create_backup_jobs: false

# If set to false, We wont create restore jobs.
freezer_create_restore_jobs: true
```

3. Save the modification in the GIT repository

```
ardana > cd ~/openstack/
ardana > git add -A
ardana > git commit -a -m "De-Activate SSH backup jobs during re-deployment"
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

4. Run the cobbler-deploy.yml playbook

```
ardana > ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost cobbler-deploy.xml
```

5. Run the bm-reimage.yml playbook limited to the second and third controller

```
ardana > ansible-playbook -i hosts/localhost bm-reimage.yml -e
 nodelist=controller2,controller3
```

controller2 and controller3 names can vary. You can use the bm-power-status.yml playbook in order to check the cobbler names of these nodes.

6. Run the site.yml playbook limited to the three controllers and localhost. In this example, this means: doc-cp1-c1-m1-mgmt, doc-cp1-c1-m2-mgmt, doc-cp1-c1-m3-mgmt and localhost

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

```
ansible-playbook -i hosts/verb_hosts site.yml --limit doc-cp1-c1-m1-mgmt,doc-cp1-c1-
m2-mgmt,doc-cp1-c1-m3-mgmt,localhost
```

13.2.1.2.10 Cassandra database restore

Cassandra database restore

Create a script cassandra-restore-extserver.sh on all controller nodes

```
root # cat > ~/cassandra-restore-extserver.sh << EOF
#!/bin/sh

# backup user
BACKUP_USER=backupuser
# backup server
BACKUP_SERVER=192.168.69.132
# backup directory
BACKUP_DIR=/mnt/backups/cassandra_backups/

# Setup variables
DATA_DIR=/var/cassandra
NODETOOL=/usr/bin/nodetool

HOST_NAME=\$(/bin/hostname)_

#Get snapshot name from command line.
if [ -z "\$*"  ]
then
  echo "usage \$0 <snapshot to restore>"
  exit 1
fi
SNAPSHOT_NAME=\$1

# restore
rsync -av -e "ssh -i /root/.ssh/id_rsa_backup" \$BACKUP_USER@\$BACKUP_SERVER:\
$BACKUP_DIR/\$HOST_NAME\$SNAPSHOT_NAME/ /

# set ownership of newley restored files
chown -R cassandra:cassandra \$DATA_DIR

# Get a list of snapshot directories that have files to be restored.
RESTORE_LIST=\$(find \$DATA_DIR -type d -name \$SNAPSHOT_NAME)

# use RESTORE_LIST to move snapshot files back into place of database.
for d in \$RESTORE_LIST
do
```

```
  cd \$d
  mv * ../..
  KEYSPACE=\$(pwd | rev | cut -d '/' -f4 | rev)
  TABLE_NAME=\$(pwd | rev | cut -d '/' -f3 |rev | cut -d '-' -f1)
  \$NODETOOL refresh \$KEYSPACE \$TABLE_NAME
done
cd
# Cleanup snapshot directories
\$NODETOOL clearsnapshot \$KEYSPACE
EOF
```

```
root # chmod +x ~/cassandra-restore-extserver.sh
```

Execute following steps on all the controller nodes

1. Edit ~/cassandra-restore-extserver.sh script

   Set *BACKUP_USER*, *BACKUP_SERVER* to the desired backup user (for example, `backupuser`)
   and the desired backup server (for example, `192.168.68.132`), respectively.

   ```
   BACKUP_USER=backupuser
   BACKUP_SERVER=192.168.69.132
   BACKUP_DIR=/mnt/backups/cassandra_backups/
   ```

2. Execute ~/cassandra-restore-extserver.sh *SNAPSHOT_NAME*

   You will have to find out *SNAPSHOT_NAME* from listing of /mnt/backups/
   cassandra_backups. All the directories are of format *HOST_SNAPSHOT_NAME*

   ```
   ls -alt /mnt/backups/cassandra_backups
   total 16
   drwxr-xr-x 4 backupuser users 4096 Jun 28 03:06 .
   drwxr-xr-x 3 backupuser users 4096 Jun 28 03:06 doc-cp1-c1-m2-mgmt_cassandra-
   snp-2018-06-28-0306
   ```

   ```
   root # ~/cassandra-restore-extserver.sh cassandra-snp-2018-06-28-0306

   receiving incremental file list
   ./
   var/
   var/cassandra/
   var/cassandra/data/
   var/cassandra/data/data/
   var/cassandra/data/data/monasca/
   var/cassandra/data/data/monasca/alarm_state_history-
   e6bbdc20488d11e8bdabc32666406af1/
   var/cassandra/data/data/monasca/alarm_state_history-
   e6bbdc20488d11e8bdabc32666406af1/snapshots/
   ```

```
var/cassandra/data/data/monasca/alarm_state_history-
e6bbdc20488d11e8bdabc32666406af1/snapshots/cassandra-snp-2018-06-28-0306/
var/cassandra/data/data/monasca/alarm_state_history-
e6bbdc20488d11e8bdabc32666406af1/snapshots/cassandra-snp-2018-06-28-0306/
manifest.json
var/cassandra/data/data/monasca/alarm_state_history-
e6bbdc20488d11e8bdabc32666406af1/snapshots/cassandra-snp-2018-06-28-0306/mc-37-big-
CompressionInfo.db
var/cassandra/data/data/monasca/alarm_state_history-
e6bbdc20488d11e8bdabc32666406af1/snapshots/cassandra-snp-2018-06-28-0306/mc-37-big-
Data.db
...
...
...
/usr/bin/nodetool clearsnapshot monasca
```

13.2.1.2.11 Databases restore

Databases restore

13.2.1.2.11.1 MariaDB database restore

MariaDB database restore

1. Source the backup credentials file

```
ardana > source ~/backup.osrc
```

2. List Freezer jobs
   Gather the id of the job corresponding to the first controller and with the description. For
   example:

```
ardana > freezer job-list | grep "mysql restore from SSH"
+--------------------------------+-------------------------------------------
+----------+---------+----------+-------+-----------+
| Job ID                         | Description                               | #
 Actions | Result  | Status    | Event | Session ID |
+--------------------------------+-------------------------------------------
+----------+---------+----------+-------+-----------+
| 64715c6ce8ed40e1b346136083923260 | Ardana Default: mysql restore from SSH    |
      1 |         | stop      |       |           |

ardana > freezer job-show 64715c6ce8ed40e1b346136083923260
```

```
+-------------
+----------------------------------------------------------------------------+
| Field       | Value
          |
+-------------
+----------------------------------------------------------------------------+
| Job ID      | 64715c6ce8ed40e1b346136083923260
          |
| Client ID   | doc-cp1-c1-m1-mgmt
   |
| User ID     | 33a6a77adc4b4799a79a4c3bd40f680d
          |
| Session ID  |
          |
| Description | Ardana Default: mysql restore from SSH
          |
| Actions     | [{u'action_id': u'19dfb0b1851e41c682716ecc6990b25b',
          |
|             |   u'freezer_action': {u'action': u'restore',
          |
|             |                       u'backup_name': u'freezer_mysql_backup',
          |
|             |                       u'container': u'/mnt/backups/
freezer_database_backups',   |
|             |                       u'hostname': u'doc-cp1-c1-m1-mgmt',
   |
|             |                       u'log_config_append': u'/etc/freezer/agent-
logging.conf', |
|             |                       u'restore_abs_path': u'/tmp/mysql_restore/',
          |
|             |                       u'ssh_host': u'192.168.69.132',
          |
|             |                       u'ssh_key': u'/etc/freezer/ssh_key',
          |
|             |                       u'ssh_port': u'22',
          |
|             |                       u'ssh_username': u'backupuser',
          |
|             |                       u'storage': u'ssh'},
          |
|             |   u'max_retries': 5,
          |
|             |   u'max_retries_interval': 60,
          |
|             |   u'user_id': u'33a6a77adc4b4799a79a4c3bd40f680d'}]
          |
```

```
| Start Date  |
             |
| End Date    |
             |
| Interval    |
             |
+-------------
+------------------------------------------------------------------------------------+
```

3. Start the job using its id

```
ardana > freezer job-start 64715c6ce8ed40e1b346136083923260
Start request sent for job 64715c6ce8ed40e1b346136083923260
```

4. Wait for the job result to be success

```
ardana > freezer job-list | grep "mysql restore from SSH"
+---------------------------------+---------------------------------------------
+----------+--------+-----------+-------+-----------+
| Job ID                          | Description                                 | #
 Actions | Result  | Status    | Event | Session ID |
+---------------------------------+---------------------------------------------
+----------+--------+-----------+-------+-----------+
| 64715c6ce8ed40e1b346136083923260 | Ardana Default: mysql restore from SSH      |
      1 |         | running   |       |           |
```

```
ardana > freezer job-list | grep "mysql restore from SSH"
+---------------------------------+---------------------------------------------
+----------+--------+-----------+-------+-----------+
| Job ID                          | Description                                 | #
 Actions | Result  | Status    | Event | Session ID |
+---------------------------------+---------------------------------------------
+----------+--------+-----------+-------+-----------+
| 64715c6ce8ed40e1b346136083923260 | Ardana Default: mysql restore from SSH      |
      1 | success | completed |       |           |
```

5. Verify that the files have been restored on the controller

```
ardana > sudo du -shx /tmp/mysql_restore/*

16K     /tmp/mysql_restore/aria_log.00000001
4.0K    /tmp/mysql_restore/aria_log_control
3.4M    /tmp/mysql_restore/barbican
8.0K    /tmp/mysql_restore/ceilometer
4.2M    /tmp/mysql_restore/cinder
2.9M    /tmp/mysql_restore/designate
```

```
129M     /tmp/mysql_restore/galera.cache
2.1M     /tmp/mysql_restore/glance
4.0K     /tmp/mysql_restore/grastate.dat
4.0K     /tmp/mysql_restore/gvwstate.dat
2.6M     /tmp/mysql_restore/heat
752K     /tmp/mysql_restore/horizon
4.0K     /tmp/mysql_restore/ib_buffer_pool
76M      /tmp/mysql_restore/ibdata1
128M     /tmp/mysql_restore/ib_logfile0
128M     /tmp/mysql_restore/ib_logfile1
12M      /tmp/mysql_restore/ibtmp1
16K      /tmp/mysql_restore/innobackup.backup.log
313M     /tmp/mysql_restore/keystone
716K     /tmp/mysql_restore/magnum
12M      /tmp/mysql_restore/mon
8.3M     /tmp/mysql_restore/monasca_transform
0        /tmp/mysql_restore/multi-master.info
11M      /tmp/mysql_restore/mysql
4.0K     /tmp/mysql_restore/mysql_upgrade_info
14M      /tmp/mysql_restore/nova
4.4M     /tmp/mysql_restore/nova_api
14M      /tmp/mysql_restore/nova_cell0
3.6M     /tmp/mysql_restore/octavia
208K     /tmp/mysql_restore/opsconsole
38M      /tmp/mysql_restore/ovs_neutron
8.0K     /tmp/mysql_restore/performance_schema
24K      /tmp/mysql_restore/tc.log
4.0K     /tmp/mysql_restore/test
8.0K     /tmp/mysql_restore/winchester
4.0K     /tmp/mysql_restore/xtrabackup_galera_info
```

6. Repeat steps 2-5 on the other two controllers where the MariaDB/Galera database is running, which can be determined by running below command on deployer

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible FND-MDB --list-hosts
```

7. Stop HPE Helion OpenStack services on the three controllers (replace the hostnames of the controllers in the command)

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-stop.yml --limit doc-cp1-c1-m1-
mgmt,doc-cp1-c1-m2-mgmt,doc-cp1-c1-m3-mgmt,localhost
```

8. Clean the mysql directory and copy the restored backup on all three controllers where MariaDB/Galera database is running

```
root # cd /var/lib/mysql/
root # rm -rf ./*
root # cp -pr /tmp/mysql_restore/* ./
```

Switch back to the ardana user once the copy is finished

13.2.1.2.11.2 Restart HPE Helion OpenStack services

Restart HPE Helion OpenStack services

1. Restart the MariaDB Database

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts galera-bootstrap.yml
```

On the deployer node, execute the `galera-bootstrap.yml` playbook which will automatically determine the log sequence number, bootstrap the main node, and start the database cluster.

If this process fails to recover the database cluster, please refer to *Section 13.2.2.1.2, "Recovering the MariaDB Database"*. There Scenario 3 covers the process of manually starting the database.

2. Restart HPE Helion OpenStack services limited to the three controllers (replace the the hostnames of the controllers in the command).

```
ansible-playbook -i hosts/verb_hosts ardana-start.yml \
  --limit doc-cp1-c1-m1-mgmt,doc-cp1-c1-m2-mgmt,doc-cp1-c1-m3-mgmt,localhost
```

3. Re-configure HPE Helion OpenStack

```
ansible-playbook -i hosts/verb_hosts ardana-reconfigure.yml
```

13.2.1.2.11.3 Re-enable SSH backups

Re-enable SSH backups

1. Re-activate Freezer backup jobs

```
ardana > vi ~/openstack/my_cloud/config/freezer/activate_jobs.yml
ardana > cat ~/openstack/my_cloud/config/freezer/activate_jobs.yml
```

```
# If set to false, We wont create backups jobs.
freezer_create_backup_jobs: true

# If set to false, We wont create restore jobs.
freezer_create_restore_jobs: true
```

2. Save the modifications in the GIT repository

```
cd ~/openstack/ardana/ansible/
git add -A
git commit -a -m "Re-Activate SSH backup jobs"
ansible-playbook -i hosts/localhost config-processor-run.yml
ansible-playbook -i hosts/localhost ready-deployment.yml
```

3. Create Freezer jobs

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
```

13.2.1.2.12   Post restore testing

Post restore testing

1. Source the service credential file

```
ardana > source ~/service.osrc
```

2. Swift

```
ardana > swift list
container_1
volumebackups

ardana > swift list container_1
var/lib/ardana/backup.osrc
var/lib/ardana/service.osrc

ardana > swift download container_1 /tmp/backup.osrc
```

3. Neutron

```
ardana > openstack network list
+-------------------------------------+---------------------
+-------------------------------------+
```

```
| ID                                   | Name                 | Subnets
             |
+--------------------------------------+--------------------
+------------------------------------+
| 07c35d11-13f9-41d4-8289-fa92147b1d44 | test-net             | 02d5ca3b-1133-4a74-
a9ab-1f1dc2853ec8|
+--------------------------------------+--------------------
+------------------------------------+
```

4. Glance

```
ardana > openstack image list
+--------------------------------------+---------------------+--------+
| ID                                   | Name                | Status |
+--------------------------------------+---------------------+--------+
| 411a0363-7f4b-4bbc-889c-b9614e2da52e | cirros-0.4.0-x86_64 | active |
+--------------------------------------+---------------------+--------+
ardana > openstack image save --file /tmp/cirros f751c39b-
f1e3-4f02-8332-3886826889ba
ardana > ls -lah /tmp/cirros
-rw-r--r-- 1 ardana ardana 12716032 Jul  2 20:52 /tmp/cirros
```

5. Nova

```
ardana > openstack server list

ardana > openstack server list

ardana > openstack server create server_6 --image 411a0363-7f4b-4bbc-889c-
b9614e2da52e  --flavor m1.small --nic net-id=07c35d11-13f9-41d4-8289-fa92147b1d44
+------------------------------------
+-----------------------------------------------------------+
| Field                               | Value
             |
+------------------------------------
+-----------------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL
             |
| OS-EXT-AZ:availability_zone         |
             |
| OS-EXT-SRV-ATTR:host                | None
             |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
             |
| OS-EXT-SRV-ATTR:instance_name       |
             |
```

```
| OS-EXT-STS:power_state        | NOSTATE
             |
| OS-EXT-STS:task_state         | scheduling
             |
| OS-EXT-STS:vm_state           | building
             |
| OS-SRV-USG:launched_at        | None
             |
| OS-SRV-USG:terminated_at      | None
             |
| accessIPv4                    |
             |
| accessIPv6                    |
             |
| addresses                     |
             |
| adminPass                     | iJBoBaj53oUd
             |
| config_drive                  |
             |
| created                       | 2018-07-02T21:02:01Z
             |
| flavor                        | m1.small (2)
             |
| hostId                        |
             |
| id                            | ce7689ff-23bf-4fe9-b2a9-922d4aa9412c
             |
| image                         | cirros-0.4.0-x86_64 (f751c39b-
f1e3-4f02-8332-3886826889ba) |
| key_name                      | None
             |
| name                          | server_6
             |
| progress                      | 0
             |
| project_id                    | cca416004124432592b2949a5c5d9949
             |
| properties                    |
             |
| security_groups               | name='default'
             |
| status                        | BUILD
             |
| updated                       | 2018-07-02T21:02:01Z
             |
```

```
| user_id                              | 8cb1168776d24390b44c3aaa0720b532
                |
| volumes_attached                     |
                |
+------------------------------------
+--------------------------------------------------------------+

ardana > openstack server list
+--------------------------------------+----------+--------
+-------------------------------+--------------------+-----------+
| ID                                   | Name     | Status | Networks
        | Image                | Flavor    |
+--------------------------------------+----------+--------
+-------------------------------+--------------------+-----------+
| ce7689ff-23bf-4fe9-b2a9-922d4aa9412c | server_6 | ACTIVE | n1=1.1.1.8
        | cirros-0.4.0-x86_64 | m1.small  |

ardana > openstack server delete ce7689ff-23bf-4fe9-b2a9-922d4aa9412c
```

13.2.2 Unplanned Control Plane Maintenance

Unplanned maintenance tasks for controller nodes such as recovery from power failure.

13.2.2.1 Restarting Controller Nodes After a Reboot

Steps to follow if one or more of your controller nodes lose network connectivity or power, which includes if the node is either rebooted or needs hardware maintenance.

When a controller node is rebooted, needs hardware maintenance, loses network connectivity or loses power, these steps will help you recover the node.

These steps may also be used if the Host Status (ping) alarm is triggered for one or more of your controller nodes.

13.2.2.1.1 Prerequisites

The following conditions must be true in order to perform these steps successfully:

- Each of your controller nodes should be powered on.

- Each of your controller nodes should have network connectivity, verified by SSH connectivity from the Cloud Lifecycle Manager to them.

- The operator who performs these steps will need access to the lifecycle manager.

13.2.2.1.2 Recovering the MariaDB Database

The recovery process for your MariaDB database cluster will depend on how many of your controller nodes need to be recovered. We will cover two scenarios:

Scenario 1: Recovering one or two of your controller nodes but not the entire cluster

Follow these steps to recover one or two of your controller nodes but not the entire cluster, then use these steps:

1. Ensure the controller nodes have power and are booted to the command prompt.

2. If the MariaDB service is not started, start it with this command:

   ```
   sudo service mysql start
   ```

3. If MariaDB fails to start, proceed to the next section which covers the bootstrap process.

Scenario 2: Recovering the entire controller cluster with the bootstrap playbook

If the scenario above failed or if you need to recover your entire control plane cluster, use the process below to recover the MariaDB database.

1. Make sure no `mysqld` daemon is running on any node in the cluster before you continue with the steps in this procedure. If there is a `mysqld` daemon running, then use the command below to shut down the daemon.

   ```
   sudo systemctl stop mysql
   ```

   If the mysqld daemon does not go down following the service stop, then kill the daemon using `kill -9` before continuing.

2. On the deployer node, execute the `galera-bootstrap.yml` playbook which will automatically determine the log sequence number, bootstrap the main node, and start the database cluster.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts galera-bootstrap.yml
```

13.2.2.1.3 Restarting Services on the Controller Nodes

From the Cloud Lifecycle Manager you should execute the `ardana-start.yml` playbook for each node that was brought down so the services can be started back up.

If you have a dedicated (separate) Cloud Lifecycle Manager node you can use this syntax:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit=<hostname_of_node>
```

If you have a shared Cloud Lifecycle Manager/controller setup and need to restart services on this shared node, you can use `localhost` to indicate the shared node, like this:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-start.yml --
limit=<hostname_of_node>,localhost
```

🎓 Note

If you leave off the `--limit` switch, the playbook will be run against all nodes.

13.2.2.1.4 Restart the Monitoring Agents

As part of the recovery process, you should also restart the `monasca-agent` and these steps will show you how:

1. Log in to the Cloud Lifecycle Manager.

2. Stop the `monasca-agent`:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-agent-stop.yml
```

3. Restart the `monasca-agent`:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-agent-start.yml
```

4. You can then confirm the status of the `monasca-agent` with this playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts monasca-agent-status.yml
```

13.2.2.2   Recovering the Control Plane

If one or more of your controller nodes has experienced data or disk corruption due to power loss or hardware failure and you need perform disaster recovery then we provide different scenarios for how to resolve them to get your cloud recovered.

If one or more of your controller nodes has experienced data or disk corruption due to power-loss or hardware failure and you need perform disaster recovery then we provide different scenarios for how to resolve them to get your cloud recovered.

> ⌖ **Note**
>
> You should have backed up `/etc/group` of the Cloud Lifecycle Manager manually after installation. While recovering a Cloud Lifecycle Manager node, manually copy the `/etc/group` file from a backup of the old Cloud Lifecycle Manager.

13.2.2.2.1   Point-in-Time MariaDB Database Recovery

In this scenario, everything is still running (Cloud Lifecycle Manager, cloud controller nodes, and compute nodes) but you want to restore the MariaDB database to a previous state.

13.2.2.2.1.1   Restore from a Swift backup

1. Log in to the Cloud Lifecycle Manager.

2. Determine which node is the first host member in the `FND-MDB` group, which will be the first node hosting the MariaDB service in your cloud. You can do this by using these commands:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > grep -A1 FND-MDB--first-member hosts/verb_hosts
```

The result will be similar to the following example:

```
[FND-MDB--first-member:children]
ardana002-cp1-c1-m1
```

In this example, the host name of the node is `ardana002-cp1-c1-m1`

3. Find the host IP address which will be used to log in.

```
ardana > cat /etc/hosts | grep ardana002-cp1-c1-m1
10.84.43.82      ardana002-cp1-c1-m1-extapi ardana002-cp1-c1-m1-extapi
192.168.24.21    ardana002-cp1-c1-m1-mgmt ardana002-cp1-c1-m1-mgmt
10.1.2.1         ardana002-cp1-c1-m1-guest ardana002-cp1-c1-m1-guest
10.84.65.3       ardana002-cp1-c1-m1-EXTERNAL-VM ardana002-cp1-c1-m1-external-vm
```

In this example, `192.168.24.21` is the IP address for the host.

4. SSH into the host.

```
ardana > ssh ardana@192.168.24.21
```

5. Source the backup file.

```
ardana > source /var/lib/ardana/backup.osrc
```

6. Find the `Client ID` for the host name from the beginning of this procedure (`ardana002-cp1-c1-m1`) in this example.

```
ardana > freezer client-list
+----------------------------+----------------------------------+----------------------------
+-------------+
| Client ID                  | uuid                             | hostname                   |
 description |
+----------------------------+----------------------------------+----------------------------
+-------------+
| ardana002-cp1-comp0001-mgmt | f4d9cfe0725145fb91aaf95c80831dd6 | ardana002-cp1-comp0001-mgmt |
         |
| ardana002-cp1-comp0002-mgmt | 55c93eb7d609467a8287f175a2275219 | ardana002-cp1-comp0002-mgmt |
         |
```

```
| ardana002-cp1-c0-m1-mgmt   | 50d26318e81a408e97d1b6639b9404b2 | ardana002-cp1-c0-m1-mgmt   |
        |
| ardana002-cp1-c1-m1-mgmt   | 78fe921473914bf6a802ad360c09d35b | ardana002-cp1-c1-m1-mgmt   |
        |
| ardana002-cp1-c1-m2-mgmt   | b2e9a4305c4b4272acf044e3f89d327f | ardana002-cp1-c1-m2-mgmt   |
        |
| ardana002-cp1-c1-m3-mgmt   | a3ceb80b8212425687dd11a92c8bc48e | ardana002-cp1-c1-m3-mgmt   |
        |
+----------------------------+----------------------------------+----------------------------
+------------+
```

In this example, the `hostname` and the `Client ID` are the same: `ardana002-cp1-c1-m1-mgmt`.

7. List the jobs

   ```
   ardana > freezer job-list -C CLIENT ID
   ```

   Using the example in the previous step:

   ```
   ardana > freezer job-list -C ardana002-cp1-c1-m1-mgmt
   ```

8. Get the corresponding job id for `Ardana Default: mysql restore from Swift`.

9. Launch the restore process with:

   ```
   ardana > freezer job-start JOB-ID
   ```

10. This will take some time. You can follow the progress by running **tail -f /var/log/freezer/freezer-scheduler.log**. Wait until the restore job is finished before doing the next step.

11. Log in to the Cloud Lifecycle Manager.

12. Stop the MariaDB service.

    ```
    ardana > cd ~/scratch/ansible/next/ardana/ansible
    ardana > ansible-playbook -i hosts/verb_hosts percona-stop.yml
    ```

13. Log back in to the first node running the MariaDB service, the same node as in *Step 3*.

14. Clean the MariaDB directory using this command:

    ```
    tux > sudo rm -r /var/lib/mysql/*
    ```

15. Copy the restored files back to the MariaDB directory:

```
tux > sudo cp -pr /tmp/mysql_restore/* /var/lib/mysql
```

16. Log in to each of the other nodes in your MariaDB cluster, which were determined in *Step 3*. Remove the `grastate.dat` file from each of them.

```
tux > sudo rm /var/lib/mysql/grastate.dat
```

> ⛨ # Warning
>
> Do not remove this file from the first node in your MariaDB cluster. Ensure you only do this from the other cluster nodes.

17. Log back in to the Cloud Lifecycle Manager.

18. Start the MariaDB service.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts galera-bootstrap.yml
```

13.2.2.2.1.2   Restore from an SSH backup

Follow the same procedure as the one for Swift but select the job `Ardana Default: mysql restore from SSH`.

13.2.2.2.1.3   Restore MariaDB manually

If restoring MariaDB fails during the procedure outlined above, you can follow this procedure to manually restore MariaDB:

1. Log in to the Cloud Lifecycle Manager.

2. Stop the MariaDB cluster:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts percona-stop.yml
```

3. On all of the nodes running the MariaDB service, which should be all of your controller nodes, run the following command to purge the old database:

```
tux > sudo rm -r /var/lib/mysql/*
```

4. On the first node running the MariaDB service restore the backup with the command below. If you have already restored to a temporary directory, copy the files again.

```
tux > sudo cp -pr /tmp/mysql_restore/* /var/lib/mysql
```

5. If you need to restore the files manually from SSH, follow these steps:

   a. Create the `/root/mysql_restore.ini` file with the contents below. Be careful to substitute the `{{ values }}`. Note that the SSH information refers to the SSH server you configured for backup before installing.

   ```
   [default]
   action = restore
   storage = ssh
   ssh_host = {{ freezer_ssh_host }}
   ssh_username = {{ freezer_ssh_username }}
   container = {{ freezer_ssh_base_dir }}/freezer_mysql_backup
   ssh_key = /etc/freezer/ssh_key
   backup_name = freezer_mysql_backup
   restore_abs_path = /var/lib/mysql/
   log_file = /var/log/freezer-agent/freezer-agent.log
   hostname = {{ hostname of the first MariaDB node }}
   ```

   b. Execute the restore job:

   ```
   ardana > freezer-agent --config /root/mysql_restore.ini
   ```

6. Log back in to the Cloud Lifecycle Manager.

7. Start the MariaDB service.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts galera-bootstrap.yml
```

8. After approximately 10-15 minutes, the output of the `percona-status.yml` playbook should show all the MariaDB nodes in sync. MariaDB cluster status can be checked using this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/verb_hosts percona-status.yml
```

An example output is as follows:

```
TASK: [FND-MDB | status | Report status of "{{ mysql_service }}"] *************
  ok: [ardana-cp1-c1-m1-mgmt] => {
  "msg": "mysql is synced."
  }
  ok: [ardana-cp1-c1-m2-mgmt] => {
  "msg": "mysql is synced."
  }
  ok: [ardana-cp1-c1-m3-mgmt] => {
  "msg": "mysql is synced."
  }
```

13.2.2.2.1.4 Point-in-Time Cassandra Recovery

A node may have been removed either due to an intentional action in the Cloud Lifecycle Manager Admin UI or as a result of a fatal hardware event that requires a server to be replaced. In either case, the entry for the failed or deleted node should be removed from Cassandra before a new node is brought up.

The following steps should be taken before enabling and deploying the replacement node.

1. Determine the IP address of the node that was removed or is being replaced.

2. On one of the functional Cassandra control plane nodes, log in as the `ardana` user.

3. Run the command `nodetool status` to display a list of Cassandra nodes.

4. If the node that has been removed (no IP address matches that of the removed node) is not in the list, skip the next step.

5. If the node that was removed is still in the list, copy its node `ID`.

6. Run the command `nodetool removenode ID`.

After any obsolete node entries have been removed, the replacement node can be deployed as usual (for more information, see *Section 13.1.2, "Planned Control Plane Maintenance"*). The new Cassandra node will be able to join the cluster and replicate data.

For more information, please consult the Cassandra documentation (http://cassandra.apache.org/doc/latest/operating/topo_changes.html) .

13.2.2.2.2 Point-in-Time Swift Rings Recovery

In this situation, everything is still running (Cloud Lifecycle Manager, control plane nodes, and compute nodes) but you want to restore your Swift rings to a previous state.

🎓 Note

Freezer backs up and restores Swift rings only, not Swift data.

13.2.2.2.2.1 Restore from a Swift backup

1. Log in to the first Swift Proxy (`SWF-PRX[0]`) node.
   To find the first Swift Proxy node:

   a. On the Cloud Lifecycle Manager

   ```
   ardana > cd  ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts swift-status.yml \
   --limit SWF-PRX[0]
   ```

   At the end of the output, you will see something like the following example:

   ```
   ...
   Jun 18 20:01:49 ardana-qe102-cp1-c1-m1-mgmt swiftlm-uptime-mon[3985]: 'uptime-mon - INFO :
    Metric:keystone-get-token:max-latency: 0.679254770279 (at 1529352109.66)'
   Jun 18 20:01:49 ardana-qe102-cp1-c1-m1-mgmt swiftlm-uptime-mon[3985]: 'uptime-mon - INFO :
    Metric:keystone-get-token:avg-latency: 0.679254770279 (at 1529352109.66)'

   PLAY RECAP *********************************************************************
   ardana-qe102-cp1-c1-m1 : ok=12 changed=0 unreachable=0 failed=0```
   ```

   b. Find the first node name and its IP address. For example:

   ```
   ardana > cat /etc/hosts | grep ardana-qe102-cp1-c1-m1
   ```

2. Source the backup environment file:

   ```
   ardana > source /var/lib/ardana/backup.osrc
   ```

3. Find the client id.

   ```
   ardana > freezer client-list
   +----------------------------+----------------------------------+----------------------------
   +------------+
   ```

```
| Client ID                 | uuid                             | hostname                   |
 description |
+---------------------------+----------------------------------+---------------------------
+-------------+
| ardana002-cp1-comp0001-mgmt | f4d9cfe0725145fb91aaf95c80831dd6 | ardana002-cp1-comp0001-mgmt |
          |
| ardana002-cp1-comp0002-mgmt | 55c93eb7d609467a8287f175a2275219 | ardana002-cp1-comp0002-mgmt |
          |
| ardana002-cp1-c0-m1-mgmt   | 50d26318e81a408e97d1b6639b9404b2 | ardana002-cp1-c0-m1-mgmt   |
          |
| ardana002-cp1-c1-m1-mgmt   | 78fe921473914bf6a802ad360c09d35b | ardana002-cp1-c1-m1-mgmt   |
          |
| ardana002-cp1-c1-m2-mgmt   | b2e9a4305c4b4272acf044e3f89d327f | ardana002-cp1-c1-m2-mgmt   |
          |
| ardana002-cp1-c1-m3-mgmt   | a3ceb80b8212425687dd11a92c8bc48e | ardana002-cp1-c1-m3-mgmt   |
          |
+---------------------------+----------------------------------+---------------------------
+-------------+
```

In this example, the `hostname` and the `Client ID` are the same: `ardana002-cp1-c1-m1-mgmt`.

4. List the jobs

   ```
   ardana > freezer job-list -C CLIENT ID
   ```

   Using the example in the previous step:

   ```
   ardana > freezer job-list -C ardana002-cp1-c1-m1-mgmt
   ```

5. Get the corresponding job id for `Ardana Default: swift restore from Swift` in the `Description` column.

6. Launch the restore job:

   ```
   ardana > freezer job-start JOB-ID
   ```

7. This will take some time. You can follow the progress by running **tail -f /var/log/ freezer/freezer-scheduler.log** Wait until the restore job is finished before doing the next step.

8. Log in to the Cloud Lifecycle Manager.

9. Stop the Swift service:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts swift-stop.yml
   ```

10. Log back in to the first Swift Proxy (`SWF-PRX[0]`) node, which was determined in *Step 1*.

11. Copy the restored files.

    ```
    tux > sudo cp -pr /tmp/swift_builder_dir_restore/CLOUD_NAME/CONTROL_PLANE_NAME/
    builder_dir/* \
        /etc/swiftlm/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/
    ```

    For example

    ```
    tux > sudo cp -pr /tmp/swift_builder_dir_restore/entry-scale-kvm/control-plane-1/
    builder_dir/* \
        /etc/swiftlm/entry-scale-kvm/control-plane-1/builder_dir/
    ```

12. Log back in to the Cloud Lifecycle Manager.

13. Reconfigure the Swift service:\

    ```
    ardana > cd ~/scratch/ansible/next/ardana/ansible
    ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
    ```

13.2.2.2.2.2   Restore from an SSH backup

Follow almost the same procedure as for Swift in the section immediately preceding this one: *Section 13.2.2.2.2.1, "Restore from a Swift backup"*. The only change is that the restore job uses a different job id. Get the corresponding job id for `Ardana Default: Swift restore from SSH` in the `Description` column.

13.2.2.2.3   Point-in-time Cloud Lifecycle Manager Recovery

In this scenario, everything is still running (Cloud Lifecycle Manager, controller nodes, and compute nodes) but you want to restore the Cloud Lifecycle Manager to a previous state.

PROCEDURE 13.1: RESTORING FROM A SWIFT OR SSH BACKUP

1. Log in to the Cloud Lifecycle Manager.

2. Source the backup environment file:

    ```
    tux > source /var/lib/ardana/backup.osrc
    ```

3. Find the `Client ID`.

```
tux > freezer client-list
+----------------------------+---------------------------------+---------------------------
+-------------+
| Client ID                  | uuid                            | hostname                   |
 description |
+----------------------------+---------------------------------+---------------------------
+-------------+
| ardana002-cp1-comp0001-mgmt | f4d9cfe0725145fb91aaf95c80831dd6 | ardana002-cp1-comp0001-mgmt |
            |
| ardana002-cp1-comp0002-mgmt | 55c93eb7d609467a8287f175a2275219 | ardana002-cp1-comp0002-mgmt |
            |
| ardana002-cp1-c0-m1-mgmt    | 50d26318e81a408e97d1b6639b9404b2 | ardana002-cp1-c0-m1-mgmt    |
            |
| ardana002-cp1-c1-m1-mgmt    | 78fe921473914bf6a802ad360c09d35b | ardana002-cp1-c1-m1-mgmt    |
            |
| ardana002-cp1-c1-m2-mgmt    | b2e9a4305c4b4272acf044e3f89d327f | ardana002-cp1-c1-m2-mgmt    |
            |
| ardana002-cp1-c1-m3-mgmt    | a3ceb80b8212425687dd11a92c8bc48e | ardana002-cp1-c1-m3-mgmt    |
            |
+----------------------------+---------------------------------+---------------------------
+-------------+
```

In this example, the `hostname` and the `Client ID` are the same: `ardana002-cp1-c1-m1-mgmt`.

4. List the jobs

```
tux > freezer job-list -C CLIENT ID
```

Using the example in the previous step:

```
tux > freezer job-list -C ardana002-cp1-c1-m1-mgmt
```

5. Find the correct job ID:

SSH Backups: Get the id corresponding to the job id for `Ardana Default: deployer restore from SSH`.
or

Swift Backups. Get the id corresponding to the job id for `Ardana Default: deployer restore from Swift`.

6. Stop the Dayzero UI:

```
tux > sudo systemctl stop dayzero
```

7. Launch the restore job:

```
tux > freezer job-start JOB ID
```

8. This will take some time. You can follow the progress by running **tail -f /var/log/ freezer/freezer-scheduler.log**. Wait until the restore job is finished before doing the next step.

9. Start the Dayzero UI:

```
tux > sudo systemctl start dayzero
```

13.2.2.2.4 Cloud Lifecycle Manager Disaster Recovery

In this scenario everything is still running (controller nodes and compute nodes) but you have lost either a dedicated Cloud Lifecycle Manager or a shared Cloud Lifecycle Manager/controller node.

To ensure that you use the same version of HPE Helion OpenStack that you previously had loaded on your Cloud Lifecycle Manager, you will need to download and install the lifecycle management software using the instructions from the *Book "Installing with Cloud Lifecycle Manager", Chapter 3 "Installing the Cloud Lifecycle Manager server", Section 3.5.2 "Installing the HPE Helion OpenStack Extension"* before proceeding further.

13.2.2.2.4.1 Restore from a Swift backup

1. Log in to the Cloud Lifecycle Manager.

2. Install the freezer-agent using the following playbook:

```
ardana > cd ~/openstack/ardana/ansible/
ardana > ansible-playbook -i hosts/localhost _deployer_restore_helper.yml
```

3. Access one of the other controller or compute nodes in your environment to perform the following steps:

   a. Retrieve the `/var/lib/ardana/backup.osrc` file and copy it to the `/var/lib/ ardana/` directory on the Cloud Lifecycle Manager.

   b. Copy all the files in the `/opt/stack/service/freezer-api/etc/` directory to the same directory on the Cloud Lifecycle Manager.

c. Copy all the files in the `/var/lib/ca-certificates` directory to the same directory on the Cloud Lifecycle Manager.

d. Retrieve the `/etc/hosts` file and replace the one found on the Cloud Lifecycle Manager.

4. Log back in to the Cloud Lifecycle Manager.

5. Edit the value for `client_id` in the following file to contain the hostname of your Cloud Lifecycle Manager:

```
/opt/stack/service/freezer-api/etc/freezer-api.conf
```

6. Update your ca-certificates:

```
sudo update-ca-certificates
```

7. Edit the `/etc/hosts` file, ensuring you edit the 127.0.0.1 line so it points to `ardana`:

```
127.0.0.1       localhost ardana
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

8. On the Cloud Lifecycle Manager, source the backup user credentials:

```
ardana > source ~/backup.osrc
```

9. Find the `Client ID` (`ardana002-cp1-c0-m1-mgmt`) for the host name as done in previous procedures (see *Procedure 13.1, "Restoring from a Swift or SSH Backup"*).

```
ardana > freezer client-list
+----------------------------+----------------------------------+----------------------------
+------------+
| Client ID                  | uuid                             | hostname                   |
 description |
+----------------------------+----------------------------------+----------------------------
+------------+
| ardana002-cp1-comp0001-mgmt | f4d9cfe0725145fb91aaf95c80831dd6 | ardana002-cp1-comp0001-mgmt |
          |
| ardana002-cp1-comp0002-mgmt | 55c93eb7d609467a8287f175a2275219 | ardana002-cp1-comp0002-mgmt |
          |
| ardana002-cp1-c0-m1-mgmt   | 50d26318e81a408e97d1b6639b9404b2 | ardana002-cp1-c0-m1-mgmt   |
          |
| ardana002-cp1-c1-m1-mgmt   | 78fe921473914bf6a802ad360c09d35b | ardana002-cp1-c1-m1-mgmt   |
          |
```

```
| ardana002-cp1-c1-m2-mgmt    | b2e9a4305c4b4272acf044e3f89d327f | ardana002-cp1-c1-m2-mgmt    |
          |
| ardana002-cp1-c1-m3-mgmt    | a3ceb80b8212425687dd11a92c8bc48e | ardana002-cp1-c1-m3-mgmt    |
          |
+----------------------------+----------------------------------+---------------------------
+------------+
```

In this example, the `hostname` and the `Client ID` are the same: `ardana002-cp1-c0-m1-mgmt`.

10. List the Freezer jobs

    ```
    ardana > freezer job-list -C CLIENT ID
    ```

    Using the example in the previous step:

    ```
    ardana > freezer job-list -C ardana002-cp1-c0-m1-mgmt
    ```

11. Get the id of the job corresponding to `Ardana Default: deployer backup to Swift`. Stop that job so the freezer scheduler does not begin making backups when started.

    ```
    ardana > freezer job-stop JOB-ID
    ```

    If it is present, also stop the Cloud Lifecycle Manager's SSH backup.

12. Start the freezer scheduler:

    ```
    sudo systemctl start openstack-freezer-scheduler
    ```

13. Get the id of the job corresponding to `Ardana Default: deployer restore from Swift` and launch that job:

    ```
    ardana > freezer job-start JOB-ID
    ```

    This will take some time. You can follow the progress by running **`tail -f /var/log/freezer/freezer-scheduler.log`**. Wait until the restore job is finished before doing the next step.

14. When the job completes, the previous Cloud Lifecycle Manager contents should be restored to your home directory:

    ```
    ardana > cd ~
    ardana > ls
    ```

15. If you are using Cobbler, restore your Cobbler configuration with these steps:

    a. Remove the following files:

    ```
    sudo rm -rf /var/lib/cobbler
    sudo rm -rf /srv/www/cobbler
    ```

    b. Deploy Cobbler:

    ```
    ardana > cd ~/openstack/ardana/ansible
    ardana > ansible-playbook -i hosts/localhost cobbler-deploy.yml
    ```

    c. Set the `netboot-enabled` flag for each of your nodes with this command:

    ```
    for h in $(sudo cobbler system list)
    do
      sudo cobbler system edit --name=$h --netboot-enabled=0
    done
    ```

16. Update your deployment directory:

    ```
    ardana > cd ~/openstack/ardana/ansible
    ardana > ansible-playbook -i hosts/localhost ready_deployment.yml
    ```

17. If you are using a dedicated Cloud Lifecycle Manager, follow these steps:

    - re-run the deployment to ensure the Cloud Lifecycle Manager is in the correct state:

    ```
    ardana > cd ~/scratch/ansible/next/ardana/ansible
    ardana > ansible-playbook -i hosts/verb_hosts site.yml --limit localhost
    ```

18. If you are using a shared Cloud Lifecycle Manager/controller, follow these steps:

    a. If the node is also a Cloud Lifecycle Manager hypervisor, run the following commands to recreate the virtual machines that were lost:

    ```
    ardana > cd ~/scratch/ansible/next/ardana/ansible
    ardana > ansible-playbook -i hosts/verb_hosts ardana-hypervisor-setup.yml --
    limit <this node>
    ```

b. If the node that was lost (or one of the VMs that it hosts) was a member of the RabbitMQ cluster then you need to remove the record of the old node, by running the following command **on any one of the other cluster members**. In this example the nodes are called `cloud-cp1-rmq-mysql-m*-mgmt` but you need to use the correct names for your system, which you can find in `/etc/hosts`:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ssh cloud-cp1-rmq-mysql-m3-mgmt sudo rabbitmqctl forget_cluster_node
 \
rabbit@cloud-cp1-rmq-mysql-m1-mgmt
```

c. Run the `site.yml` against the complete cloud to reinstall and rebuild the services that were lost. If you replaced one of the RabbitMQ cluster members then you will need to add the `-e` flag shown below, to nominate a new master node for the cluster, otherwise you can omit it.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts site.yml -e \
rabbit_primary_hostname=cloud-cp1-rmq-mysql-m3
```

13.2.2.2.4.2   Restore from an SSH backup

1. On the Cloud Lifecycle Manager, edit the following file so it contains the same information as it did previously:

```
ardana > ~/openstack/my_cloud/config/freezer/ssh_credentials.yml
```

2. On the Cloud Lifecycle Manager, copy the following files, change directories, and run the playbook _deployer_restore_helper.yml:

```
ardana > cp -r ~/hp-ci/openstack/* ~/openstack/my_cloud/definition/
ardana > cd ~/openstack/ardana/ansible/
ardana > ansible-playbook -i hosts/localhost _deployer_restore_helper.yml
```

3. Perform the restore. First become root and change directories:

```
sudo su
root # cd /root/deployer_restore_helper/
```

4. Execute the restore job:

```
ardana > ./deployer_restore_script.sh
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready_deployment.yml
```

6. When the Cloud Lifecycle Manager is restored, re-run the deployment to ensure the Cloud Lifecycle Manager is in the correct state:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts site.yml --limit localhost
```

13.2.2.2.5 One or Two Controller Node Disaster Recovery

This scenario makes the following assumptions:

- Your Cloud Lifecycle Manager is still intact and working.

- One or two of your controller nodes went down, but not the entire cluster.

- The node needs to be rebuilt from scratch, not simply rebooted.

13.2.2.2.5.1 Steps to recovering one or two controller nodes

1. Ensure that your node has power and all of the hardware is functioning.

2. Log in to the Cloud Lifecycle Manager.

3. Verify that all of the information in your `~/openstack/my_cloud/definition/data/servers.yml` file is correct for your controller node. You may need to replace the existing information if you had to either replacement your entire controller node or just pieces of it.

4. If you made changes to your `servers.yml` file then commit those changes to your local git:

```
ardana > git add -A
ardana > git commit -a -m "editing controller information"
```

5. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
```

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Ensure that Cobbler has the correct system information:

   a. If you replaced your controller node with a completely new machine, you need to verify that Cobbler has the correct list of controller nodes:

   ```
   ardana > sudo cobbler system list
   ```

   b. Remove any controller nodes from Cobbler that no longer exist:

   ```
   ardana > sudo cobbler system remove --name=<node>
   ```

   c. Add the new node into Cobbler:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost cobbler-deploy.yml
   ```

8. Then you can image the node:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<node_name>
```

🎓 **Note**

> If you do not know the `<node name>` already, you can get it by using **`sudo cobbler system list`**.

Before proceeding, you may want to take a look at **info/server_info.yml** to see if the assignment of the node you have added is what you expect. It may not be, as nodes will not be numbered consecutively if any have previously been removed. This is to prevent loss of data; the config processor retains data about removed nodes and keeps their ID numbers from being reallocated. See the Persisted Server Allocations section in for information on how this works.

9. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your non-OS partitions on your nodes are completely wiped prior to continuing with the installation. The `wipe_disks.yml` playbook is only meant to be run on systems immediately after running `bm-reimage.yml`. If used for any other case, it may not wipe all of the expected partitions.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts wipe_disks.yml --limit
 <controller_node_hostname>
```

10. Complete the rebuilding of your controller node with the two playbooks below:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts osconfig-run.yml -e rebuild=True --
limit=<controller_node_hostname>
ardana > ansible-playbook -i hosts/verb_hosts ardana-deploy.yml -e rebuild=True --
limit=<controller_node_hostname>
```

13.2.2.2.6 Three Control Plane Node Disaster Recovery

In this scenario, all control plane nodes are destroyed which need to be rebuilt or replaced.

13.2.2.2.6.1 Restore from a Swift backup:

Restoring from a Swift backup is not possible because Swift is gone.

13.2.2.2.6.2 Restore from an SSH backup

1. Log in to the Cloud Lifecycle Manager.

2. Disable the default backup job(s) by editing the following file:

```
ardana > ~/scratch/ansible/next/ardana/ansible/roles/freezer-jobs/defaults/
activate.yml
```

Set the value for `freezer_create_backup_jobs` to `false`:

```
# If set to false, We won't create backups jobs.
freezer_create_backup_jobs: false
```

3. Deploy the control plane nodes, using the values for your control plane node hostnames:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts site.yml --limit \
  CONTROL_PLANE_HOSTNAME1,CONTROL_PLANE_HOSTNAME2, \
  CONTROL_PLANE_HOSTNAME3 -e rebuild=True
```

For example, if you were using the default values from the example model files your command would look like this:

```
ardana > ansible-playbook -i hosts/verb_hosts site.yml \
    --limit ardana-ccp-c1-m1-mgmt,ardana-ccp-c1-m2-mgmt,ardana-ccp-c1-m3-mgmt \
    -e rebuild=True
```

🎓 Note

The `-e rebuild=True` is only used on a single control plane node when there are other controllers available to pull configuration data from. This will cause the MariaDB database to be reinitialized, which is the only choice if there are no additional control nodes.

4. Restore the MariaDB backup on the first controller node.

   a. List the Freezer jobs:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > freezer job-list -C FIRST_CONTROLLER_NODE
   ```

   b. Run the `Ardana Default: mysql restore from SSH` job for your first controller node, replacing the `JOB_ID` for that job:

   ```
   ardana > freezer job-start JOB_ID
   ```

5. You can monitor the restore job by connecting to your first controller node via SSH and running the following commands:

```
ardana > ssh FIRST_CONTROLLER_NODE
ardana > sudo su
root # tail -n 100 /var/log/freezer/freezer-scheduler.log
```

6. Log back in to the Cloud Lifecycle Manager.

7. Stop MySQL:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts percona-stop.yml
```

8. Log back in to the first controller node and move the following files:

```
ardana > ssh FIRST_CONTROLLER_NODE
ardana > sudo su
root # rm -rf /var/lib/mysql/*
root # cp -pr /tmp/mysql_restore/* /var/lib/mysql/
```

9. Log back in to the Cloud Lifecycle Manager and bootstrap MySQL:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts galera-bootstrap.yml
```

10. Verify the status of MySQL:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts percona-status.yml
```

11. Re-enable the default backup job(s) by editing the following file:

```
~/scratch/ansible/next/ardana/ansible/roles/freezer-jobs/defaults/activate.yml
```

Set the value for `freezer_create_backup_jobs` to `true`:

```
# If set to false, We won't create backups jobs.
freezer_create_backup_jobs: true
```

12. Run this playbook to deploy the backup jobs:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
```

13.2.2.2.7 Swift Rings Recovery

To recover your Swift rings in the event of a disaster, follow the procedure that applies to your situation: either recover the rings from one Swift node if possible, or use the SSH backup that you have set up.

To recover your Swift rings in the event of a disaster, follow the procedure that applies to your situation: either recover the rings from one Swift node if possible, or use the SSH backup that you have set up.

13.2.2.2.7.1 Restore from the Swift deployment backup

See *Section 15.6.2.7, "Recovering Swift Builder Files"*.

13.2.2.2.7.2 Restore from the SSH Freezer backup

In the very specific use case where you lost all system disks of all object nodes, and Swift proxy nodes are corrupted, you can recover the rings because a copy of the Swift rings is stored in Freezer. This means that Swift data is still there (the disks used by Swift needs to be still accessible).

Recover the rings with these steps.

1. Log in to a node that has the freezer-agent installed.

2. Become root:

   ```
   ardana > sudo su
   ```

3. Create the temporary directory to restore your files to:

   ```
   root # mkdir /tmp/swift_builder_dir_restore/
   ```

4. Create a restore file with the following content:

   ```
   root # cat << EOF > ./restore_config.ini
   [default]
   action = restore
   storage = ssh
   compression = bzip2
   restore_abs_path = /tmp/swift_builder_dir_restore/
   ssh_key = /etc/freezer/ssh_key
   ssh_host = <freezer_ssh_host>
   ssh_port = <freezer_ssh_port>
   ssh_user name = <freezer_ssh_user name>
   container = <freezer_ssh_base_rid>/freezer_swift_backup_name =
    freezer_swift_builder_backup
   hostname = <hostname of the old first Swift-Proxy (SWF-PRX[0])>
   ```

```
EOF
```

5. Edit the file and replace all <tags> with the right information.

```
vim ./restore_config.ini
```

You will also need to put the SSH key used to do the backups in /etc/freezer/ssh_key and remember to set the right permissions: 600.

6. Execute the restore job:

```
root # freezer-agent --config ./restore_config.ini
```

You now have the Swift rings in `/tmp/swift_builder_dir_restore/`

7. If the SWF-PRX[0] is already deployed, copy the contents of the restored directory (`/tmp/swift_builder_dir_restore/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/`) to `/etc/swiftlm/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/` on the SWF-PRX[0] Then from the Cloud Lifecycle Manager run:

```
ardana > sudo cp -pr /tmp/swift_builder_dir_restore/CLOUD_NAME/CONTROL_PLANE_NAME/
builder_dir/* \
    /etc/swiftlm/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/
```

For example

```
ardana > sudo cp -pr /tmp/swift_builder_dir_restore/entry-scale-kvm/control-plane-1/
builder_dir/* \
    /etc/swiftlm/entry-scale-kvm/control-plane-1/builder_dir/
```

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
```

8. If the SWF-ACC[0] is **not** deployed, from the Cloud Lifecycle Manager run these playbooks:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts guard-deployment.yml
ardana > ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit <SWF-ACC[0]-
hostname>
```

9. Copy the contents of the restored directory (`/tmp/swift_builder_dir_restore/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/`) to `/etc/swiftlm/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/` on the SWF-ACC[0] You will have to create the directories : `/etc/swiftlm/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/`

```
ardana > sudo cp -pr /tmp/swift_builder_dir_restore/CLOUD_NAME/CONTROL_PLANE_NAME/
builder_dir/* \
    /etc/swiftlm/CLOUD_NAME/CONTROL_PLANE_NAME/builder_dir/
```

For example

```
ardana > sudo cp -pr /tmp/swift_builder_dir_restore/entry-scale-kvm/control-plane-1/
builder_dir/* \
    /etc/swiftlm/entry-scale-kvm/control-plane-1/builder_dir/
```

10. From the Cloud Lifecycle Manager, run the `ardana-deploy.yml` playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-deploy.yml
```

13.2.3   Unplanned Compute Maintenance

Unplanned maintenance tasks including recovering compute nodes.

13.2.3.1   Recovering a Compute Node

If one or more of your compute nodes has experienced an issue such as power loss or hardware failure, then you need to perform disaster recovery. Here we provide different scenarios and how to resolve them to get your cloud repaired.

Typical scenarios in which you will need to recover a compute node include the following:

- The node has failed, either because it has shut down has a hardware failure, or for another reason.

- The node is working but the `nova-compute` process is not responding, thus instances are working but you cannot manage them (for example to delete, reboot, and attach/detach volumes).

- The node is fully operational but monitoring indicates a potential issue (such as disk errors) that require down time to fix.

13.2.3.1.1   What to do if your compute node is down

Compute node has power but is not powered on

If your compute node has power but is not powered on, use these steps to restore the node:

1. Log in to the Cloud Lifecycle Manager.

2. Obtain the name for your compute node in Cobbler:

   ```
   sudo cobbler system list
   ```

3. Power the node back up with this playbook, specifying the node name from Cobbler:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-power-up.yml -e nodelist=<node name>
   ```

Compute node is powered on but services are not running on it

If your compute node is powered on but you are unsure if services are running, you can use these steps to ensure that they are running:

1. Log in to the Cloud Lifecycle Manager.

2. Confirm the status of the compute service on the node with this playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts nova-status.yml --limit <hostname>
   ```

3. You can start the compute service on the node with this playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts nova-start.yml --limit <hostname>
   ```

13.2.3.1.2   Scenarios involving disk failures on your compute nodes

Your compute nodes should have a minimum of two disks, one that is used for the operating system and one that is used as the data disk. These are defined during the installation of your cloud, in the `~/openstack/my_cloud/definition/data/disks_compute.yml` file on the Cloud Lifecycle Manager. The data disk(s) are where the `nova-compute` service lives. Recovery scenarios will depend on whether one or the other, or both, of these disks experienced failures.

If your operating system disk failed but the data disk(s) are okay

If you have had issues with the physical volume that nodes your operating system you need to ensure that your physical volume is restored and then you can use the following steps to restore the operating system:

1. Log in to the Cloud Lifecycle Manager.

2. Source the administrator credentials:

   ```
   source ~/service.osrc
   ```

3. Obtain the hostname for your compute node, which you will use in subsequent commands when `<hostname>` is requested:

   ```
   nova host-list | grep compute
   ```

4. Obtain the status of the `nova-compute` service on that node:

   ```
   nova service-list --host <hostname>
   ```

5. You will likely want to disable provisioning on that node to ensure that `nova-scheduler` does not attempt to place any additional instances on the node while you are repairing it:

   ```
   nova service-disable --reason "node is being rebuilt" <hostname> nova-compute
   ```

6. Obtain the status of the instances on the compute node:

   ```
   nova list --host <hostname> --all-tenants
   ```

7. Before continuing, you should either evacuate all of the instances off your compute node or shut them down. If the instances are booted from volumes, then you can use the `nova evacuate` or `nova host-evacuate` commands to do this. See *Section 13.1.3.3, "Live Migration of Instances"* for more details on how to do this.

   If your instances are not booted from volumes, you will need to stop the instances using the `nova stop` command. Because the `nova-compute` service is not running on the node you will not see the instance status change, but the `Task State` for the instance should change to `powering-off`.

   ```
   nova stop <instance_uuid>
   ```

Verify the status of each of the instances using these commands, verifying the `Task State` states `powering-off`:

```
nova list --host <hostname> --all-tenants
nova show <instance_uuid>
```

8. At this point you should be ready with a functioning hard disk in the node that you can use for the operating system. Follow these steps:

   a. Obtain the name for your compute node in Cobbler, which you will use in subsequent commands when `<node_name>` is requested:

   ```
   sudo cobbler system list
   ```

   b. Reimage the compute node with this playbook:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=<node name>
   ```

9. Once reimaging is complete, use the following playbook to configure the operating system and start up services:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts site.yml --limit <hostname>
```

10. You should then ensure any instances on the recovered node are in an `ACTIVE` state. If they are not then use the `nova start` command to bring them to the `ACTIVE` state:

```
nova list --host <hostname> --all-tenants
nova start <instance_uuid>
```

11. Reenable provisioning:

```
nova service-enable <hostname> nova-compute
```

12. Start any instances that you had stopped previously:

```
nova list --host <hostname> --all-tenants
nova start <instance_uuid>
```

If your data disk(s) failed but the operating system disk is okay OR if all drives failed

In this scenario your instances on the node are lost. First, follow steps 1 to 5 and 8 to 9 in the previous scenario.

After that is complete, use the `nova rebuild` command to respawn your instances, which will also ensure that they receive the same IP address:

```
nova list --host <hostname> --all-tenants
nova rebuild <instance_uuid>
```

13.2.4 Unplanned Storage Maintenance

Unplanned maintenance tasks for storage nodes.

13.2.4.1 Unplanned Swift Storage Maintenance

Unplanned maintenance tasks for Swift storage nodes.

13.2.4.1.1 Recovering a Swift Node

If one or more of your Swift Object or PAC nodes has experienced an issue, such as power loss or hardware failure, and you need to perform disaster recovery then we provide different scenarios and how to resolve them to get your cloud repaired.

Typical scenarios in which you will need to repair a Swift object or PAC node include:

- The node has either shut down or been rebooted.

- The entire node has failed and needs to be replaced.

- A disk drive has failed and must be replaced.

13.2.4.1.1.1 What to do if your Swift host has shut down or rebooted

If your Swift host has power but is not powered on, from the lifecycle manager you can run this playbook:

1. Log in to the Cloud Lifecycle Manager.

2. Obtain the name for your Swift host in Cobbler:

   ```
   sudo cobbler system list
   ```

3. Power the node back up with this playbook, specifying the node name from Cobbler:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-power-up.yml -e nodelist=<node name>
   ```

Once the node is booted up, Swift should start automatically. You can verify this with this playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-status.yml
```

Any alarms that have triggered due to the host going down should clear within 10 minutes. See *Section 15.1.1, "Alarm Resolution Procedures"* if further assistance is needed with the alarms.

13.2.4.1.1.2 How to replace your Swift node

If your Swift node has irreparable damage and you need to replace the entire node in your environment, see *Section 13.1.5.1.5, "Replacing a Swift Node"* for details on how to do this.

13.2.4.1.1.3 How to replace a hard disk in your Swift node

If you need to do a hard drive replacement in your Swift node, see *Section 13.1.5.1.6, "Replacing Drives in a Swift Node"* for details on how to do this.

13.3 Cloud Lifecycle Manager Maintenance Update Procedure

PROCEDURE 13.2: PREPARING FOR UPDATE

1. Ensure that the update repositories have been properly set up on all nodes. The easiest way to provide the required repositories on the Cloud Lifecycle Manager Server is to set up an SMT server as described in *Book "Installing with Cloud Lifecycle Manager", Chapter 4 "Installing and Setting Up an SMT Server on the Cloud Lifecycle Manager server (Optional)"*. Alternatives to setting up an SMT server are described in *Book "Installing with Cloud Lifecycle Manager", Chapter 5 "Software Repository Setup"*.

2. Read the Release Notes for the security and maintenance updates that will be installed.

3. Have a backup strategy in place. For further information, see *Chapter 14, Backup and Restore*.

4. Ensure that you have a known starting state by resolving any unexpected alarms.

5. Determine if you need to reboot your cloud after updating the software. Rebooting is highly recommended to ensure that all affected services are restarted. Reboot may be required after installing Linux kernel updates, but it can be skipped if the impact on running services is non-existent or well understood.

6. Review steps in *Section 13.1.4.1, "Adding a Neutron Network Node"* and *Section 13.1.1.2, "Rolling Reboot of the Cloud"* to minimize the impact on existing workloads. These steps are critical when the Neutron services are not provided via external SDN controllers.

7. Before the update, prepare your working loads by consolidating all of your instances to one or more Compute Nodes. After the update is complete on the 324 evacuated Compute Nodes, reboot them and move the images from the remaining Compute Nodes to the newly booted ones. Then, update the remaining Compute Nodes.

13.3.1 Performing the Update

Before you proceed, get the status of all your services:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-status.yml
```

If status check returns an error for a specific service, run the `SERVICE-reconfigure.yml` playbook. Then run the `SERVICE-status.yml` playbook to check that the issue has been resolved.

Update and reboot all nodes in the cloud one by one. Start with the deployer node, then follow the order recommended in *Section 13.1.1.2, "Rolling Reboot of the Cloud"*.

> 🎓 **Note**
>
> The described workflow also covers cases in which the deployer node is also provisioned as an active cloud node.

To minimize the impact on the existing workloads, the node should first be prepared for an update and a subsequent reboot by following the steps leading up to stopping services listed in *Section 13.1.1.2, "Rolling Reboot of the Cloud"*, such as migrating singleton agents on Control Nodes and evacuating Compute Nodes. Do not stop services running on the node, as they need to be running during the update.

PROCEDURE 13.3: UPDATE INSTRUCTIONS

1. Install all available security and maintenance updates on the deployer using the **zypper patch** command.

2. Initialize the Cloud Lifecycle Manager and prepare the update playbooks.

   a. Run the `ardana-init` initialization script to update the deployer.

   b. Redeploy cobbler:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost cobbler-deploy.yml
   ```

   c. Run the configuration processor:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

   d. Update your deployment directory:

   ```
   ardana > cd ~/openstack/ardana/ansible
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

3. Installation and management of updates can be automated with the following playbooks:

- `ardana-update-pkgs.yml`

- `ardana-update.yml`

- `ardana-update-status.yml`

> **⛴ Important**
>
> Some playbooks are being deprecated. To determine how your system is affected, run:
>
> ```
> ardana > rpm -qa ardana-ansible
> ```
>
> The result will be `ardana-ansible-8.0+git.` followed by a version number string.
>
> - If the first part of the version number string is greater than or equal to 1553878455 (for example, ardana-ansible-8.0 + git.**1553878455**.7439e04), use the newly introduced parameters:
>
>   - `pending_clm_update`
>
>   - `pending_service_update`
>
>   - `pending_system_reboot`
>
> - If the first part of the version number string is less than 1553878455 (for example, ardana-ansible-8.0 + git.**1552032267**.5298d45), use the following parameters:
>
>   - `update_status_var`
>
>   - `update_status_set`
>
>   - `update_status_reset`

- `ardana-reboot.yml`

4. Confirm version changes by running `hostnamectl` before and after running the `ardana-update-pkgs` playbook on each node.

```
ardana > hostnamectl
```

Notice that the `Boot ID:` and `Kernel:` information has changed.

5. By default, the `ardana-update-pkgs.yml` playbook will install patches and updates that do not require a system reboot. Patches and updates that **do** require a system reboot will be installed later in this process.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-update-pkgs.yml \
--limit TARGET_NODE_NAME
```

There may be a delay in the playbook output at the following task while updates are pulled from the deployer.

```
TASK: [ardana-upgrade-tools | pkg-update | Download and install
package updates] ***
```

6. After running the `ardana-update-pkgs.yml` playbook to install patches and updates not requiring reboot, check the status of remaining tasks.

```
ardana > ansible-playbook -i hosts/verb_hosts ardana-update-status.yml \
--limit TARGET_NODE_NAME
```

7. To install patches that require reboot, run the `ardana-update-pkgs.yml` playbook with the parameter `-e zypper_update_include_reboot_patches=true`.

```
ardana > ansible-playbook -i hosts/verb_hosts ardana-update-pkgs.yml \
--limit  TARGET_NODE_NAME \
-e zypper_update_include_reboot_patches=true
```

If the output of `ardana-update-pkgs.yml` indicates that a reboot is required, run `ardana-reboot.yml` *after* completing the `ardana-update.yml` step below. Running `ardana-reboot.yml` will cause cloud service interruption.

🎓 Note
To update a single package (for example, apply a PTF on a single node or on all nodes), run **zypper update** *PACKAGE* .

To install all package updates using **zypper update**.

8. Update services:

```
ardana > ansible-playbook -i hosts/verb_hosts ardana-update.yml \
--limit TARGET_NODE_NAME
```

9. If indicated by the `ardana-update-status.yml` playbook, reboot the node.
   There may also be a warning to reboot after running the `ardana-update-pkgs.yml`.
   This check can be overridden by setting the `SKIP_UPDATE_REBOOT_CHECKS` environment
   variable or the `skip_update_reboot_checks` Ansible variable.

```
ansible-playbook -i hosts/verb_hosts ardana-reboot.yml \
--limit TARGET_NODE_NAME
```

10. To recheck pending system reboot status at a later time, run the following commands:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-update-status.yml \
--limit ardana-cp1-c1-m2
```

11. The pending system reboot status can be reset by running:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-update-status.yml \
--limit ardana-cp1-c1-m2 \
-e pending_system_reboot=off
```

12. Multiple servers can be patched at the same time with `ardana-update-pkgs.yml` by
    setting the option `-e skip_single_host_checks=true`.

> ⛨ Warning
>
> When patching multiple servers at the same time, take care not to compromise HA
> capability by updating an entire cluster (controller, database, monitor, logging) at
> the same time.

If multiple nodes are specified on the command line (with `--limit`), services on those servers will experience outages as the packages are shutdown and updated. On Compute Nodes (or group of Compute Nodes) migrate the workload off if you plan to update it. The same applies to Control Nodes: move singleton services off of the control plane node that will be updated.

> ## 🔔 Important
>
> Do not reboot all of your controllers at the same time.

13. When the node comes up after the reboot, run the `spark-start.yml` file:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts spark-start.yml
```

14. Verify that Spark is running on all Control Nodes:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts spark-status.yml
```

15. After all nodes have been updated, check the status of all services:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-status.yml
```

13.3.2   Summary of the Update Playbooks

ardana-update-pkgs.yml

Top-level playbook automates the installation of package updates on a single node. It also works for multiple nodes, if the single-node restriction is overridden by setting the SKIP_SINGLE_HOST_CHECKS environment variable `ardana-update-pkgs.yml -e skip_single_host_checks=true`.

Provide the following `-e` options to modify default behavior:

- `zypper_update_method` (default: patch)

- `patch` will install all patches for the system. Patches are intended for specific bug and security fixes.

- `update` will install all packages that have a higher version number than the installed packages.

- `dist-upgrade` replaces each package installed with the version from the repository and deletes packages not available in the repositories.

- `zypper_update_repositories` (default: all) restricts the list of repositories used

- `zypper_update_gpg_checks` (default: true) enables GPG checks. If set to `true`, checks if packages are correctly signed.

- `zypper_update_licenses_agree` (default: false) automatically agrees with licenses. If set to `true`, zypper automatically accepts third party licenses.

- `zypper_update_include_reboot_patches` (default: false) includes patches that require reboot. Setting this to `true` installs patches that require a reboot (such as kernel or glibc updates).

ardana-update.yml

Top level playbook that automates the update of all the services. Runs on all nodes by default, or can be limited to a single node by adding `--limit nodename`.

ardana-reboot.yml

Top-level playbook that automates the steps required to reboot a node. It includes pre-boot and post-boot phases, which can be extended to include additional checks.

ardana-update-status.yml

This playbook can be used to check or reset the update-related status variables maintained by the update playbooks. The main reason for having this mechanism is to allow the update status to be checked at any point during the update procedure. It is also used heavily by the automation scripts to orchestrate installing maintenance updates on multiple nodes.

13.4 Cloud Lifecycle Manager Program Temporary Fix (PTF) Deployment

Occasionally, in order to fix a given issue, SUSE will provide a set of packages known as a Program Temporary Fix (PTF). Such a PTF is fully supported by SUSE until the Maintenance Update containing a permanent fix has been released via the regular Update repositories. Customers running PTF fixes will be notified through the related Service Request when a permanent patch for a PTF has been released.

Use the following steps to deploy a PTF:

1. When SUSE has developed a PTF, you will receive a URL for that PTF. You should download the packages from the location provided by SUSE Support to a temporary location on the Cloud Lifecycle Manager. For example:

   ```
   ardana > tmpdir=`mktemp -d`
   ardana > cd $tmpdir
   ardana > sudo wget --no-directories --recursive --reject "index.html*"\
   --user=USER_NAME \
   --password=PASSWORD \
   --no-parent https://ptf.suse.com/54321aaaa...dddd12345/cloud8/042171/x86_64/20181030
   ```

2. Remove any old data from the PTF repository, such as a listing for a PTF repository from a migration or when previous product patches were installed.

   ```
   ardana > sudo rm -rf /srv/www/suse-12.3/x86_64/repos/PTF/*
   ```

3. Move packages from the temporary download location to the PTF repository directory on the CLM Server. This example is for a Neutron PTF.

   ```
   ardana > sudo mkdir -p /srv/www/suse-12.3/x86_64/repos/PTF/
   ardana > sudo mv $tmpdir/*
      /srv/www/suse-12.3/x86_64/repos/PTF/
   ardana > sudo chown --recursive root:root /srv/www/suse-12.3/x86_64/repos/PTF/*
   ardana > rmdir $tmpdir
   ```

4. Create or update the repository metadata:

   ```
   ardana > sudo /usr/local/sbin/createrepo-cloud-ptf
   Spawning worker 0 with 2 pkgs
   Workers Finished
   Saving Primary metadata
   Saving file lists metadata
   ```

```
Saving other metadata
```

5. Refresh the PTF repository before installing package updates on the Cloud Lifecycle
Manager

```
ardana > sudo zypper refresh --force --repo PTF
Forcing raw metadata refresh
Retrieving repository 'PTF' metadata
.......................................[d
one]
Forcing building of repository cache
Building repository 'PTF' cache .......................................[done]
Specified repositories have been refreshed.
```

6. The PTF shows as available on the deployer.

```
ardana > sudo zypper se --repo PTF
Loading repository data...
Reading installed packages...

S | Name                         | Summary                                 | Type
--+------------------------------+-----------------------------------------
+--------
  | python-neutronclient         | Python API and CLI for OpenStack Neutron |
 package
i | venv-openstack-neutron-x86_64 | Python virtualenv for OpenStack Neutron |
 package
```

7. Install the PTF venv packages on the Cloud Lifecycle Manager

```
ardana > sudo zypper dup  --from PTF
Refreshing service
Loading repository data...
Reading installed packages...
Computing distribution upgrade...

The following package is going to be upgraded:
  venv-openstack-neutron-x86_64

The following package has no support information from its vendor:
  venv-openstack-neutron-x86_64

1 package to upgrade.
Overall download size: 64.2 MiB. Already cached: 0 B. After the operation,
 additional 6.9 KiB will be used.
Continue? [y/n/...? shows all options] (y): y
```

```
Retrieving package venv-openstack-neutron-
x86_64-11.0.2-13.8.1.042171.0.PTF.102473.noarch ... (1/1),  64.2 MiB ( 64.6 MiB
 unpacked)
Retrieving: venv-openstack-neutron-
x86_64-11.0.2-13.8.1.042171.0.PTF.102473.noarch.rpm ....[done]
Checking for file
 conflicts: ......................................................[done]
(1/1) Installing: venv-openstack-neutron-
x86_64-11.0.2-13.8.1.042171.0.PTF.102473.noarch ....[done]
Additional rpm output:
warning
warning: /var/cache/zypp/packages/PTF/noarch/venv-openstack-neutron-
x86_64-11.0.2-13.8.1.042171.0.PTF.102473.noarch.rpm: Header V3 DSA/SHA1 Signature,
 key ID b37b98a9: NOKEY
```

8. Validate the venv tarball has been installed into the deployment directory:(note:the packages file under that dir shows the registered tarballs that will be used for the services, which should align with the installed venv RPM)

```
ardana > ls -la /opt/ardana_packager/ardana-8/sles_venv/x86_64
total 898952
drwxr-xr-x 2 root root     4096 Oct 30 16:10 .
...
-rw-r--r-- 1 root root 67688160 Oct 30 12:44 neutron-20181030T124310Z.tgz <<<
-rw-r--r-- 1 root root 64674087 Aug 14 16:14 nova-20180814T161306Z.tgz
-rw-r--r-- 1 root root 45378897 Aug 14 16:09 octavia-20180814T160839Z.tgz
-rw-r--r-- 1 root root     1879 Oct 30 16:10 packages
-rw-r--r-- 1 root root 27186008 Apr 26  2018 swift-20180426T230541Z.tgz
```

9. Install the non-venv PTF packages on the Compute Node

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-update-pkgs.yml --extra-vars
 '{"zypper_update_method": "update", "zypper_update_repositories": ["PTF"]}' --limit
 comp0001-mgmt
```

When it has finished, you can see that the upgraded package has been installed on comp0001-mgmt .

```
ardana > sudo zypper se --detail python-neutronclient
Loading repository data...
Reading installed packages...

S | Name                 | Type    | Version                            | Arch    |
 Repository
```

```
--+---------------------+----------+--------------------------------+--------
+-----------------------------------
i | python-neutronclient | package  | 6.5.1-4.361.042171.0.PTF.102473 | noarch | PTF
  | python-neutronclient | package  | 6.5.0-4.361                     | noarch |
  SUSE-OPENSTACK-CLOUD-x86_64-GM-DVD1
```

10. Running the ardana update playbook will distribute the PTF venv packages to the cloud server. Then you can find them loaded in the virtual environment directory with the other venvs.

    The Compute Node before running the update playbook:

    ```
    ardana > ls -la /opt/stack/venv
    total 24
    drwxr-xr-x  9 root root 4096 Jul 18 15:47 neutron-20180718T154642Z
    drwxr-xr-x  9 root root 4096 Aug 14 16:13 neutron-20180814T161306Z
    drwxr-xr-x 10 root root 4096 May 28 09:30 nova-20180528T092954Z
    drwxr-xr-x 10 root root 4096 Aug 14 16:13 nova-20180814T161306Z
    ```

11. Run the update.

    ```
    ardana > cd ~/scratch/ansible/next/ardana/ansible
    ardana > ansible-playbook -i hosts/verb_hosts ardana-update.yml --limit comp0001-
    mgmt
    ```

    When it has finished, you can see that an additional virtual environment has been installed.

    ```
    ardana > ls -la /opt/stack/venv
    total 28
    drwxr-xr-x  9 root root 4096 Jul 18 15:47 neutron-20180718T154642Z
    drwxr-xr-x  9 root root 4096 Aug 14 16:13 neutron-20180814T161306Z
    drwxr-xr-x  9 root root 4096 Oct 30 12:43 neutron-20181030T124310Z <<< New venv
     installed
    drwxr-xr-x 10 root root 4096 May 28 09:30 nova-20180528T092954Z
    drwxr-xr-x 10 root root 4096 Aug 14 16:13 nova-20180814T161306Z
    ```

12. The PTF may also have RPM package updates in addition to venv updates. To complete the update, follow the instructions at *Section 13.3.1, "Performing the Update"*.

13.5 Periodic OpenStack Maintenance Tasks

Heat-manage helps manage Heat specific database operations. The associated database should be periodically purged to save space. The following should be setup as a cron job on the servers where the heat service is running at `/etc/cron.weekly/local-cleanup-heat` with the following content:

```
#!/bin/bash
su heat -s /bin/bash -c "/usr/bin/heat-manage purge_deleted -g days 14" || :
```

nova-manage db archive_deleted_rows command will move deleted rows from production tables to shadow tables. Including `--until-complete` will make the command run continuously until all deleted rows are archived. It is recommended to setup this task as `/etc/cron.weekly/local-cleanup-nova` on the servers where the nova service is running, with the following content:

```
#!/bin/bash
su nova -s /bin/bash -c "/usr/bin/nova-manage db archive_deleted_rows --until-complete"
|| :
```

14 Backup and Restore

Information about how to back up and restore your cloud.

Freezer is a Backup and Restore as a Service platform that helps you automate the backup and restore process for your data. This backup and restore component (Freezer) executes backups and restores as jobs, and executes these jobs independently and/or as managed sessions (multiple jobs in multiple machines sharing a state).

There are a number of things you must do before installing your cloud so that you achieve the backup plan you need.

First, you should consider *Section 14.4, "Enabling Default Backups of the Control Plane to an SSH Target"* in case you lose cloud servers that the Freezer backup and restore service uses by default.

Second, you can prevent the Freezer backup and restore service from being installed completely, or designate which services should be backed up by default. *Section 14.11, "Disabling Backup/Restore before Deployment"*.

HPE Helion OpenStack 8 supports backup and restore of control plane services. It comes with playbooks and procedures to recover the control plane from various disaster scenarios.

The following features are supported:

- Backup of your file system using a point-in-time snapshot.

- Strong encryption: AES-256-CFB.

- Backup of your MySQL database with LVM snapshot.

- Restoring your data from a specific date automatically to your file system.

- Low storage consumption: the backups are uploaded as a stream.

- Flexible backup policy (both incremental and differential).

- Data archived in GNU Tar format for file-based incremental.

- Multiple compression algorithm support (zlib, bzip2, xz).

- Removal old backup automatically according the provided parameters.

- Multiple storage media support (Swift, local file system, SSH).

- Management of multiple jobs (multiple backups on the same node).

- Synchronization of backup and restore on multiple nodes.

- Execution of scripts/commands before or after a job execution.

14.1 Architecture

The backup and restore service/Freezer uses GNU Tar under the hood to execute incremental backup and restore. When a key is provided, it uses Open SSL to encrypt data (AES-256-CFB).

The architecture consists of the following components:

Component	Description
Freezer Scheduler	A client-side component running on the node from where the data backup is executed. It consists of a daemon that retrieves the data from the freezer API and executes jobs (that is, backups, restore, admin actions, info actions, and pre- and/or post- job scripts) by running the Freezer Agent. The metrics and exit codes returned by the Freezer Agent are captured and sent to the Freezer API.
	The scheduler manages the execution and synchronization of multiple jobs executed on a single node or multiple nodes. The status of the execution of all the nodes is saved through the API.
	The Freezer scheduler takes care of uploading jobs to the API by reading job files on the file system. It also has its own configuration file where job sessions or other settings such as the Freezer API polling interval can be configured.
Freezer Agent	Multiprocessing Python software that runs on the client side where the data backup is executed. It can be executed as a standalone or by the Freezer Scheduler. The freezer-agent provides a flexible way to execute backup, restore, and perform other actions on a running system.
	To provide flexibility in terms of data integrity, speed, performance, resource usage, and so on, the Freezer Agent offers a wide range of options to execute optimized backup according the available resources, such as:
	• Segments size (the amount of memory used)
	• Queues size (optimize backups where I/O, bandwidth, memory, or CPU is a constraint)
	• I/O affinity and process priority (can be used with real time I/O and maximum user-level process priority)

Component	Description
	- Bandwidth limitation - Client-side Encryption (AES-256-CFB) - Compression (multiple algorithms supported as zlib, bzip2, xz/lzma) - Parallel upload to pluggable storage media (that is, upload backup to Swift and to a remote node by SSH, or upload to two or more independent Swift instances with different credentials, and so on) - Execute file-based incremental (such as tar), block-based incremental (such as rsync algorithm), and differential-based backup and restore - Multi platform: you can run it on SUSE Linux, Windows, *BSD, and OSX - Automatic removal of old backups
Freezer API	Stores and provides metadata to the Freezer Scheduler. Also stores session information for multi node backup synchronization. Workload data is not stored in the API .
DB Elasticsearch	API uses the backend to store and retrieve metrics metadata sessions information job status, and so on.

14.2 Architecture of the Backup/Restore Service

Backup/recovery architecture

Component	Description	Runs on
API	API service to add / fetch Freezer jobs	Controller nodes with Elasticsearch
Scheduler	Daemon that stores and retrieves backup/restore jobs and executes them	Nodes needing backup/restore (controllers, Cloud Lifecycle Manager)
Agent	The agent that backs up and restores to and from targets. Invoked from scheduler or manually.	Nodes needing backup/restore (controllers, Cloud Lifecycle Manager)

14.3 Default Automatic Backup Jobs

By default, the following are **automatically** backed up. You do not have to do anything for these backup jobs to run. However if you want to back up to somewhere outside the cluster, you do need to *Section 14.4, "Enabling Default Backups of the Control Plane to an SSH Target"*.

- **Cloud Lifecycle Manager Data.** All important information on the Cloud Lifecycle Manager

- **MariaDB Database.** The MariaDB database contains most of the data needed to restore services. While the MariaDB database only allows for an incomplete recovery of ESX data, for other services it allows full recovery. Logging data in Elasticsearch is not backed up. Swift objects are not backed up because of the redundant nature of Swift.

- **Swift Rings.** Swift rings are backed up so that you can recover more quickly even though Swift can rebuild the rings without this data. However automatically rebuilding the rings is slower than restoring via a backup.

The following services will be effectively backed up. In other words, the data needed to restore the services is backed up. The critical data that will be backed up are the databases and the configuration-related files. Note the data that is not backed up per service:

- Ceilometer. However, there is no backup of metrics data

- Cinder. However, there is no backup of the volumes

- Glance. However, there is no backup of the images

- Heat

- Horizon

- Keystone

- Neutron

- Nova. However, there is no backup of the images

- Swift. However, there is no backup of the objects. Swift has its own high availability/ redundancy. Swift rings are backed up. Although Swift will rebuild the rings itself, restoring from backup is faster.

- Operations Console

- Monasca. However, there is no backup of the metrics

14.3.1 Limitations

The following limitations apply to backups created by the Freezer backup and restore service in HPE Helion OpenStack:

- Recovery of the following services (or cloud topologies) will be partially backed up. They will need additional data (other than the data stored in MariaDB) to return to fully functional.

    - ESX Cloud

    - Network services - LBaaS and VPNaaS

- Logging data (that is, log files).

- VMs and volumes are not currently backed up.

14.4 Enabling Default Backups of the Control Plane to an SSH Target

This topic describes how you can set up an external server as a backup server in case you lose access to your cloud servers that store the default backups.

14.4.1 Default Backup and Restore

As part of the installation procedure in HPE Helion OpenStack, automatic backup/restore jobs are set up to back up to Swift via the Freezer scheduler component of the backup and restore service. The backup jobs perform scheduled backups of HPE Helion OpenStack control plane data (files/directories/db). The restore jobs can be used to restore appropriate control plane data. Additional automatic jobs can be added to backup/restore from the secure shell (SSH) server that you set up/designate. It is recommended that you set up SSH backups so that in the event that you lose all of your control plane nodes at once, you have a backup on remote servers that you can use to restore the control plane nodes. Note that you do not have to restore from the SSH location if only one or two control plane nodes are lost. In that case, they can be recovered from the data on the remaining control plane node following the restore procedures in *Section 13.2.2.2, "Recovering the Control Plane"*. That document also explains how to recover using your remote server SSH backups.

While control plane backups to Swift are set up automatically, you must use the following procedure to set up SSH backups.

14.4.2   Setting up SSH backups

By default, during HPE Helion OpenStack 8 deployment, backup jobs are automatically deployed to Swift, the MySQL database, the Cloud Lifecycle Manager, and Swift rings. Restore jobs are also deployed for convenience. It is more secure to store those backups also outside of the HPE Helion OpenStack infrastructure. If you provide all the values required in the following file, jobs will also be deployed to backup and restore to/from an SSH server of your choice:

```
~/openstack/my_cloud/config/freezer/ssh_credentials.yml
```

14.4.3   Backing up your HPE Helion OpenStack control plane to an SSH server

You must provide the following connection information to this server:

- The SSH server's IP address

- The SSH server's port to connect to (usually port 22). You may want to confirm how to open the port on the HPE Helion OpenStack firewall.

- The user to connect to the SSH server as

- The SSH private key authorized to connect to that user (see below for details of how to set up one if it is not already done)

- The directory where you wish to store the backup on that server

14.4.4   Setting up SSH for backups before deployment

Before running the configuration processor, edit the following file:

```
~/openstack/my_cloud/config/freezer/ssh_credentials.yml
```

All parameters are mandatory. Take care in providing the SSH private key.

14.4.5 Preparing the server that will store the backup

In this example, the information is as follows:

- IP: 192.168.100.42

- Port: 22

- User: backupuser

- Target directory for backups: /mnt/backups/

Please replace these values to meet your own requirements, as appropriate.

1. Connect to the server:

   ```
   ardana > ssh -p 22 root@192.168.100.42
   ```

2. Create the user:

   ```
   tux > sudo useradd backupuser --create-home --home-dir /mnt/backups/
   ```

3. Switch to that user:

   ```
   su backupuser
   ```

4. Create the SSH keypair:

   ```
   backupuser > ssh-keygen -t rsa
   # Just leave the default for the first question and do not set any passphrase
   > Generating public/private rsa key pair.
   > Enter file in which to save the key (/mnt/backups//.ssh/id_rsa):
   > Created directory '/mnt/backups//.ssh'.
   > Enter passphrase (empty for no passphrase):
   > Enter same passphrase again:
   > Your identification has been saved in /mnt/backups//.ssh/id_rsa
   > Your public key has been saved in /mnt/backups//.ssh/id_rsa.pub
   > The key fingerprint is:
   > a9:08:ae:ee:3c:57:62:31:d2:52:77:a7:4e:37:d1:28 backupuser@padawan-ccp-c0-m1-mgmt
   > The key's randomart image is:
   > +---[RSA 2048]----+
   > |          o      |
   > |    . . E + .    |
   > |   o . . + .     |
   > | o +   o +       |
   > |  + o o S .      |
   ```

```
> |  . + o o        |
> |   o + .          |
> |.o .              |
> |++o               |
> +----------------+
```

5. Add the public key to the list of the keys authorized to connect to that user on this server:

```
backupuser > cat /mnt/backups/.ssh/id_rsa.pub >> /mnt/backups/.ssh/authorized_keys
```

6. View the private key. This is what you will use for the backup configuration:

```
backupuser > cat /mnt/backups/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvjwKu6f940IVGHpUj3ffl3eKXACgVr3L5s9UJnb15+zV3K5L
BZuor8MLvwtskSkgdXNrpPZhNCsWSkryJff5I335Jhr/e5o03Yy+RqIMrJAIa0X5
...
iBKVKGPhOnn4ve3dDqy3q7fS5sivTqCrpaYtByJmPrcJNjb2K7VMLNvgLamK/AbL
qpSTZjicKZCCl+J2+8lrKAaDWqWtIjSUs29kCL78QmaPOgEvfsw=
-----END RSA PRIVATE KEY-----
```

Your server is now ready to receive backups. If you wish, you can check our advice on how to secure it in *Section 14.4.8, "Securing your SSH backup server"*.

14.4.6   Setting up SSH for backups after deployment

If you already deployed your cloud and forgot to configure SSH backups, or if you wish to modify the settings for where the backups are stored, follow the following instructions:

1. Edit the following file:

```
~/openstack/my_cloud/config/freezer/ssh_credentials.yml
```

Please be advised that all parameters are mandatory, and take care in providing the SSH private key.

2. Run the following commands:

```
ardana > cd ~/openstack
ardana > git add -A
ardana > git commit -m "My config"
ardana > cd ~/openstack/ardana/ansible/
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

```
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

This will deploy the SSH key and configure SSH backup and restore jobs for you. It may take some time before the backups occur.

14.4.7 Opening ports in the cloud firewall

There is a strict policy of firewalling deployed with HPE Helion OpenStack. If you use a non-standard SSH port, you may need to specifically open it by using the following process:

1. When creating your model, edit the following file:

```
~/openstack/my_cloud/definition/data/firewall_rules.yml
```

2. You must add a new element in the firewall-rules list, such as:

```
- name: BACKUP
  # network-groups is a list of all the network group names
  # that the rules apply to
  network-groups:
  - MANAGEMENT
  rules:
  - type: allow
    # range of remote addresses in CIDR format that this
    # rule applies to
    remote-ip-prefix:
    port-range-min:
    port-range-max:
    # protocol must be one of: null, tcp, udp or icmp
    protocol: tcp
```

14.4.8 Securing your SSH backup server

You can do the following to harden an SSH server (these techniques are well documented on the internet):

- Disable root login

- Move SSH to a non-default port (that is, something other than 22)

- Disable password login (only allow RSA keys)

- Disable SSH v1

- Authorize Secure File Transfer Protocol (SFTP) only for that user (disable SSH shell)

- Firewall SSH traffic to ensure it comes from the HPE Helion OpenStack address range

- Install a Fail2Ban solution

- Restrict users that are allowed to SSH

Remove the key pair generated earlier on the backup server: the only thing needed is the .ssh/ authorized_keys. You can remove the .ssh/id_rsa and .ssh/id_rsa.pub files. Be sure to save a backup of them somewhere.

14.4.9 Finish Firewall Configuration

Run the following commands to finish configuring the firewall.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
```

14.4.10 General tips

- Take care when sizing the directory that will receive the backup.

- Monitor the space left on that directory.

- Keep the system up to date on that server.

14.5 Changing Default Jobs

The procedure to make changes to jobs created by default in HPE Helion OpenStack is to edit the model file, `my_cloud/config/freezer/jobs.yml` and then re-run the `_freezer-manage-jobs.yml` playbook. (Note that the backup/restore component is called "Freezer" so you may see commands by that name.)

1. Open jobs.yml in an editor, then change and save the file:

   ```
   ardana > cd ~/openstack
   ```

```
ardana > nano my_cloud/config/freezer/jobs.yml
```

2. Commit the file to the local git repository:

```
ardana > git add -A
ardana > git commit -m "Backup job changes"
```

3. Next, run the configuration processor followed by the ready-deployment playbooks:

```
ardana > cd ~/openstack/ardana/ansible/
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

4. Run _freezer_manage_jobs.yml:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
```

14.6 Backup/Restore Via the Horizon UI

A number of backup and restore tasks can be performed using the Horizon UI. This topic lists the available tasks, the access requirements, and limitations of using Horizon for backup and restore.

14.6.1 Accessing the UI

User name

> The only supported user in this version is "ardana_backup". The login credentials are available in `backup.osrc` located at `~/backup.osrc/`

> ### ☞ Note
>
>> The location of backup.osrc has now changed from the old location of `/opt/stack/service/freezer-agent/etc/backup.osrc`.

UI access

> To access the Horizon UI, follow the instructions shown in *Book "User Guide", Chapter 3 "Cloud Admin Actions with the Dashboard"*. Once logged in as "ardana_backup", navigate to "Disaster Recovery" panel located in the left-hand menu where you should see "Backup and Restore."

14.6.2 Backup and Restore Operations Supported in the UI

The following Operations are supported via the UI

- Ability to create new jobs to Backup/Restore files

- List the freezer jobs that have completed

- Create sessions to link multiple jobs

- List the various nodes (hosts/servers) on which the freezer scheduler and freezer agent are installed

14.6.3 Limitations

The following limitations apply to Freezer backups in HPE Helion OpenStack:

- The UI for backup and restore is supported only if you log in as "ardana_backup". All other users will see the UI panel but the UI will not work.

- If Backup/Restore action fails via the UI, you must check the Freezer logs for details of the failure.

- Job Status and Job Result on the UI and backend (CLI) are not in sync.

- For a given "Action" the following modes are not supported from the UI:

  - Microsoft SQL Server

  - Cinder

  - Nova

- There is a known issue which will be fixed in future releases while using Start and End dates and times in creating a job. Please refrain from using those fields.

14.7 Restore from a Specific Backup

This topic describes how you can get a list of previous backups and how to restore from them.

> # ⛨ Warning
>
> Note that the existing contents of the directory to which you will restore your data (and its children) will be completely overwritten. You must take that into account if there is data in that directory that you want to survive the restore by either copying that data somewhere else or changing the directory to which you will restore.

By default, freezer-agent restores only the latest (most recent) backup. Here is a manual procedure to restore from a list of backups

1. Obtain the list of backups:

   ```
   ardana > freezer backup-list
     [--limit]
     [--offset]
   ```

   `--limit` limit results to this limit.

   `--offset` return results from this offset.

   ```
   ardana > freezer backup-list
   +--------------------------------+-------------+----------------------------
   +-----------------------------------------------------------------------
   +--------------------+-------+
   | Backup ID                      | Backup UUID | Hostname                   |
    Path                                                           | Created
    at           | Level |
   +--------------------------------+-------------+----------------------------
   +-----------------------------------------------------------------------
   +--------------------+-------+
   | 75f8312788fa4e95bf975807905287f8 |           | ardana-qe202-cp1-c1-m3-mgmt | /
   var/lib/freezer/mount_94e03f120c9e4ae78ad50328d782cea6/.          | 2018-07-06
    08:26:00 |     0 |
   | 4229d71c840e4ee1b78680131695a330 |           | ardana-qe202-cp1-c1-m2-mgmt | /
   var/lib/freezer/mount_77d3c7a76b16435181bcaf41837cc7fe/.          | 2018-07-06
    08:26:01 |     0 |
   | 6fe59b58924e43f88729dc0a1fe1290b |           | ardana-qe202-cp1-c1-m1-mgmt | /
   var/lib/freezer/mount_4705ac61026c4e77b6bf59b7bcfc286a/.          | 2018-07-06
    16:38:50 |     0 |
   ```

2. Use the "restore-from-date" option to restore a backup based on data/timestamp. The restore-from-data is an option available in freezer-agent. When using the parameter `--restore-from-date`, Freezer searches the available backups and selects the nearest older backup relative to the provided date. To use this option, the following parameters of the

backup must be provided - storage target details (example, `target-name`, `container-name`), backup_name, hostname. Usually these parameters can be obtained from the backup_job.

For example, take the following simple backup job:

```
[default]
 action = backup
 backup_name = mystuffbackup
 storage = local
 container = /home/me/mystorage
 max_level = 7
     path_to_backup = ~/mydata
```

Suppose you schedule that every day and you end up with backups that happened at:

```
1) 2015-12-10T02:00:00
     2) 2015-12-11T02:00:00
     3) 2015-12-12T02:00:00
     3) 2015-12-13T02:00:00
```

Now, if you restore using the following parameters:

```
[default]
    action = restore
    backup_name = mystuffbackup
    storage = local
    container = /home/me/mystorage
    restore_abs_path = ~/mydata_restore_dir
    restore_from_date = 2015-12-11T23:00:00
```

The nearest oldest backup will be number 2, taken at 2015-12-11T02:00:00.

14.8 Backup/Restore Scheduler

14.8.1 Freezer (backup/restore service) Scheduler Overview

This document explains, through examples, how to set up backup and restore jobs using the backup/restore service scheduler (referred to as Freezer Scheduler).

The scheduler is a long running process that executes the following:

- Interact with the Freezer API

- Generate a client_id to register the client on to the API (to identify the node during the next executions)

- Execute the freezer-agent according the jobs information retrieved from the API

- Write to the freezer API the outcome of the freezer-agent execution

☞ Note

Freezer API maintains information about jobs in the Elasticsearch Database.

☞ Note

You must run as root to perform any tasks using the Freezer backup/restore service.

14.8.2 Freezer (backup/restore service) Scheduler Client-ID

In HPE Helion OpenStack 8, Freezer Scheduler is automatically installed on the Cloud Lifecycle Manager and controller nodes.

There is a client_id for each node and its corresponds to the hostname. The client_id is created at registration time. The registration is done automatically when the scheduler executes any request to the API.

The following command lists all the freezer scheduler clients:

```
ardana > freezer client-list
```

Here is an example:

```
ardana > freezer client-list
+--------------------------------+---------------------------------
+--------------------------------+------------+
| Client ID                      | uuid                            | hostname
         | description |
+--------------------------------+---------------------------------
+--------------------------------+------------+
| ardana-qe202-cp1-c1-m3-mgmt    | 7869340f2efc4fb9b29e94397385ac39 | ardana-qe202-cp1-
c1-m3-mgmt     |              |
```

```
| ardana-qe202-cp1-c1-m2-mgmt   | 18041c2b12054802bdaf8cc458abc35d | ardana-qe202-cp1-
c1-m2-mgmt    |                 |
| ardana-qe202-cp1-c1-m1-mgmt   | 884045a72026425dbcea754806d1022d | ardana-qe202-cp1-
c1-m1-mgmt    |                 |
| ardana-qe202-cp1-comp0001-mgmt | e404b34e5f7844ed957ca5dd90e6446f | ardana-qe202-cp1-
comp0001-mgmt |                 |
+-------------------------------+----------------------------------
+-------------------------------+------------+
```

14.8.3   Creating a Scheduler Job

1. Log in to a controller node and create the job.

2. Source the operating system variables and use the correct client_id. (The client-id corresponds to the node where the backup files/directory/database resides.) In HPE Helion OpenStack the sourcing of the variable should be done like this when you need to use ardana_backup user and backup tenant (used for infrastructure backup): Note that when you perform these actions you must be **running as root**. The following command will provide the necessary credentials to run the job.

> 🎓 Note
>
> The location of backup.osrc has now changed from the old location of `/opt/stack/service/freezer-agent/etc/backup.osrc`.

```
ardana > source ~/backup.osrc
```

And with the following when you need to use admin user and admin tenant. The following file will contain the admin user credentials. These are not for jobs that were created automatically; they are only used for jobs created manually to be created/executed under the admin account. Jobs created automatically use the credentials stored in the backup.osrc file noted above.

```
source ~/service.osrc
```

```
{
            "job_actions": [
            {
            "freezer_action": {
```

```
                    "action": "backup",
                    "mode": "fs",
                    "backup_name": "backup1",
                    "path_to_backup": "/home/user/tmp",
                    "container": "tmp_backups"
                    },
                    "max_retries": 3,
                    "max_retries_interval": 60
                    }
                    ],
                    "job_schedule": {
                    "schedule_interval": "24 hours"
                    },
                    "description": "backup for tmp dir"
                    }
```

3. Upload it into the api using the correct client_id:

```
ardana > freezer job-create -C CLIENT-ID --file FREEZER-FILE
```

> ### 🎓 Note
>
> Freezer file examples can be found in *Section 14.8.6, "Example Backup Job File"*.

4. The status of the jobs can be checked with:

```
ardana > freezer job-list -C CLIENT-ID
```

5. If no scheduling information is provided, the job will be executed as soon as possible so its status will go into a "running" state, then "completed".

You can find information about the scheduling and backup-execution in /var/log/freezer/ freezer-scheduler.log and /var/log/freezer-api/freezer-api.log, respectively.

> ### 🎓 Note
>
> Recurring jobs never go into a "completed" state, as they go back into "scheduled" state.

14.8.4 Restore from a Different Node

The scheduler can be used to restore from a different node using the hostname parameter that you see in the JSON below. Here is an example conf file.

```
{
                "job_actions": [
                {
                "freezer_action": {
                "action": "restore",
                "restore_abs_path": "/var/lib/mysql",
                "hostname": "test_machine_1",
                "backup_name": "freezer-db-mysql",
                "container": "freezer_backup_devstack_1"
                },
                "max_retries": 5,
                "max_retries_interval": 60,
                "mandatory": true
                }
                ],
                "description": "mysql test restore"
                }
```

Create the job like so:

```
ardana > freezer job-create -C CLIENT-ID --file job-restore-mysql.conf
```

14.8.5 Differential Backup and Restore

The difference is in the use of the parameter `always_level: 1`. We also specify a different
container, so it is easier to spot the files created in the Swift container:

```
ardana > swift list freezer_backup_devstack_1_alwayslevel
```

14.8.6 Example Backup Job File

Here is a sample backup file:

```
{
            "job_actions": [
            {
            "freezer_action": {
            "mode" : "mysql",
            "mysql_conf" : "/etc/mysql/debian.cnf",
            "path_to_backup": "/var/lib/mysql/",
            "backup_name": "freezer-db-mysql",
            "snapshot": true,
```

```
            "always_level": 1,
            "max_priority": true,
            "remove_older_than": 90,
            "container": "freezer_backup_devstack_1_alwayslevel"
            },
            "max_retries": 5,
            "max_retries_interval": 60,
            "mandatory": true
            }
            ],
            "job_schedule" : {
            },
            "description": "mysql backup"
            }
```

To create the job:

```
ardana > freezer job-create -C client_node_1 --file job-backup.conf
```

14.8.7   Example Restore Job File

Here is an example of `job-restore.conf`

```
{
            "job_actions": [
            {
            "freezer_action": {
            "action": "restore",
            "restore_abs_path": "/var/lib/mysql",
            "hostname": "test_machine_1",
            "backup_name": "freezer-db-mysql",
            "container": "freezer_backup_devstack_1_alwayslevel"
            },
            "max_retries": 5,
            "max_retries_interval": 60,
            "mandatory": true
            }
            ],
            "description": "mysql test restore"
            }
```

To create the job:

```
ardana > freezer job-create -C client_node_1 --file job-restore.conf
```

14.9 Backup/Restore Agent

This topic describes how to configure backup jobs and restore jobs.

14.9.1 Introduction

The backup/restore service agent (Freezer Agent) is a tool that is used to manually back up and restore your data. It can be run from any place you want to take a backup (or do a restore) because all HPE Helion OpenStack nodes have the freezer-agent installed on them. To use it, you should run as root. The agent runs in conjunction with the *Section 14.8, "Backup/Restore Scheduler"*. The following explains their relationship:

- The backup/restore scheduler (openstack-freezer-scheduler, also see *Section 14.8, "Backup/Restore Scheduler"*) takes JSON-style config files, and can run them automatically according to a schedule in the job_schedule field of the scheduler's JSON config file. It takes anything you pass in via the job_actions field and translates those requirements into an INI-style config file. Then it runs freezer-agent. As a user, you could also run the freezer agent using `freezer-agent --config file.ini`, which is exactly how the scheduler runs it.

- The agent (freezer-agent) actually performs the jobs. Whenever any backup or restore action happens, the agent is the one doing the actual work. It can be run directly by the user, as noted above, or by the scheduler. It accepts either command-line flags (such as `--action backup`) or INI-style config files.

> ☞ Note
>
> You can run `freezer-agent --help` to view a definitive list of all possible flags that can be used (with the transform rules mentioned) in these configuration files.

For HPE Helion OpenStack 8, you must follow these steps to perform backups:

1. Define what you want to back up.

2. Define a mode for that backup. The following modes are available:

   - fs (filesystem) (default)

   - mysql

   - sqlserver

> **⯈ Note**
> It is recommended that you use snapshots if the mode is mysql or sqlserver.

3. Define whether to use a snapshot in the file system for the backup:

   - In Unix systems LVM is used (when available).

   - In Windows systems virtual shadow copies are used.

4. Define a storage media in a job from the following list:

   - Swift (requires OpenStack credentials)(default)

   - Local (no credentials required)

   - SSH (no credentials required) (not implemented on Windows)

14.9.2   Basic Configuration for Backups

There are several mandatory parameters you need to specify in order to execute a backup. Note storage is optional:

- action (backup by default)

- mode (fs by default)

- path-to-backup

- backup-name

- container (Swift container or local path)

- storage is not mandatory. It is Swift by default.

For HPE Helion OpenStack 8, you can create a backup using only mandatory values, as in the following example:

```
ardana > freezer-agent --action backup --mode fs --storage swift --path-to-backup /home/
user/tmp --container tmp_backups --backup-name backup1
```

Running the above command from the command line will cause this backup to execute once. To create a configuration file for this same backup, in case you want to run it manually another time, create a configuration file like the one below. Note that in the config file, the parameter

names such as backup-name will use underscores instead of dashes. Thus backup-name as used in the CLI will be backup_name when used in the config file. Note also that where you use -- in the CLI, such as --mode, you do not use the -- in the config file.

```
[default]
              action = backup
              mode = fs
              backup_name = backup1
              path_to_backup = /home/user/tmp
              container = tmp_backups
```

A configuration file similar to the one above will be generated if you create a JSON configuration file for automated jobs to be run by the scheduler. Instructions on how to do that are found on the *Section 14.8, "Backup/Restore Scheduler"* page.

14.9.3   Restoring your Data

For HPE Helion OpenStack 8, you must do the following in order to restore data after a backup:

1. Select a backup to restore.

2. Define a mode for the restore: The following modes are available:

    - fs (filesystem) (default)

    - mysql

    - sqlserver

3. If the restore involves an application (such as MariaDB) remember to shut down the application or service and start it again after the restore.

14.9.4   Basic Configuration for Restoring

To restore from a backup, note that in some cases you must stop the service (for instance, MariaDB) before the restore.

There are several parameters that are required and there are some optional parameters used to execute a restore:

- action (backup by default)

- mode (fs by default)

- restore-abs-path

- backup-name

- container (Swift container or local path)

- restore-from-host

- restore-from-date (optional)

- storage is not mandatory. It is Swift by default

You can create a restore using mandatory values, as in the following example:

```
ardana > freezer-agent --action restore --mode fs --storage swift --restore-abs-path /
home/user/tmp --container tmp_backups --backup-name backup1 --restore-from-host ubuntu
```

To create a configuration file for this same restore, the file would look like the one below. Note that in the config file, the parameter names such as backup-name will use underscores instead of dashes. Thus backup-name as used in the CLI will be backup_name when used in the config file. Note also that where you use -- in the CLI, such as --mode, you do not use the -- in the config file. This is the same format as used above for backup configuration.

```
{
                "job_actions": [
                {
                "freezer_action": {
                "action": "restore",
                "mode": "fs",
                "backup_name": "backup1",
                "restore_abs_path": "/home/user/tmp",
                "container": "tmp_backups",
                "hostname": "ubuntu"
                },
                "max_retries": 3,
                "max_retries_interval": 60
                }
                ],
                "description": "backup for tmp dir"
                }
```

14.10   Backup and Restore Limitations

The following limitations apply to backups created by the Freezer backup and restore service in HPE Helion OpenStack:

- Recovery of the following services (or cloud topologies) will be partially supported as they need additional data (other than MariaDB) to return to fully functional.

  - ESX Cloud

  - Network services - LBaaS and VPNaaS

- Logging data (that is, log files)

14.11   Disabling Backup/Restore before Deployment

Backups are enabled by default. Therefore, you must take action if you want backups to be disabled for any reason. This topic explains how to disable default backup jobs before completing the installation of your cloud.

⛨ Warning

You should make modifications in the `~/openstack/my_cloud/` directory before running the configuration processor and ready-deployment steps.

14.11.1   Disable backups before installation:

To disable deployment of the Freezer backup and restore service, remove the following lines in `control_plane.yml`:

- freezer-agent

- freezer-api

🎓 Note

This action is required even if you already removed Freezer lines from your model (`control_plane.yml`).

14.11.2 Deploy Freezer but disable backup/restore job creation:

It is also possible to allow Freezer deployment yet prevent the lifecycle manager from creating automatic backup jobs. By default, the lifecycle manager deployment automatically creates jobs for the backup and restore of the following:

- Lifecycle-manager node

- MySQL database

- Swift rings

Before running the configuration processor, you can prevent Freezer from automatically creating backup and restore jobs by changing the variables `freezer_create_backup_jobs` and `freezer_create_restore_jobs` to `false` in:

```
~/openstack/my_cloud/config/freezer/activate_jobs.yml
```

Alternatively, you can disable the creation of those jobs while launching the deployment process, as follows:

```
ardana > ansible-playbook -i hosts/verb_hosts site.yml -e
 '{ "freezer_create_backup_jobs": false }' -e '{ "freezer_create_restore_jobs": false }'
```

When using these options, the Freezer infrastructure will still be deployed but will not execute any backups.

14.11.3 Disable backup and restore jobs for a specific service

To manage which jobs will be enabled, set the appropriate paramters in the `jobs.yml` freezer jobs configuration file:

```
~/openstack/my_cloud/config/freezer/jobs.yml
```

You can completely disable the backup of a component by changing the `enabled` field that corresponds to that service to false in `jobs.yml`.

You can specify where a job will store its backup by setting `store_in_swift, store_in_ssh, store_in_local` to true or false. Note that these are not mutually exclusive. You can set true for all of these backup targets. Setting SSH, Swift, and local to true will cause one backup job (and one restore job) per storage target to be created.

Note also that even if `store_in_ssh` is set to true, the SSH backup job will not be created unless SSH credentials are provided in `/openstack/my_cloud/config/freezer/ssh_credentials.yml`.

When setting `store_in_local` to `true`, the backup job will store backups on the server executing the backup. This option is useful, for example, if you plan to mount an NFS share and want your backup stored on it. You need to provide the path where the backup will be stored by setting the `local_storage_base_dir` parameter.

By default, one backup job per storage medium per component will be created. A corresponding restore job for each of those backup jobs will also be created by default. These jobs can be used to quickly restore the corresponding backup. To disable the creation of these restore jobs, change `also_create_restore_job` to `false`.

14.11.4 Activating and deactivating jobs after cloud deployment

1. Make modifications similar to those discussed above in `/openstack/my_cloud/config/freezer/jobs.yml`.

2. Commit modifications to the git repo

   ```
   ardana > git add -A
   ardana > git commit -m "A message that explains what modifications have been made"
   ```

3. Run the configuration processor

   ```
   ardana > cd ~/openstack/ardana/ansible/
   ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

4. Run the ready deployment playbook. (This will update the scratch/... directories with all of the above modifications).

   ```
   ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

5. Change directories to scratch

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ```

6. Run _freezer_manage_jobs.yml

   ```
   ardana > ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
   ```

14.12 Enabling, Disabling and Restoring Backup/ Restore Services

14.12.1 Stop, Start and Restart the Backup Services

To stop the Freezer backup and restore service globally, launch the following playbook from the Cloud Lifecycle Manager (this will stop all freezer-api and all freezer-agent running on your clusters):

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts freezer-stop.yml
```

To start the Freezer backup and restore service globally, launch the following playbook from the Cloud Lifecycle Manager:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts freezer-start.yml
```

To restart the Freezer backup and restore services use the ansible playbooks from above.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts freezer-stop.yml
ardana > ansible-playbook -i hosts/verb_hosts freezer-start.yml
```

It is possible to target only specific nodes using the ansible --limit parameter.

14.12.2 Manually

For the freezer-agent:

1. Connect to the concerned host.

2. Run the following command to stop the freezer agent:

   ```
   tux > sudo systemctl stop openstack-freezer-scheduler
   ```

   or run the following command to start the freezer-agent:

   ```
   tux > sudo systemctl start openstack-freezer-scheduler
   ```

or run the following command to restart the freezer-agent:

```
tux > sudo systemctl restart openstack-freezer-scheduler
```

For the freezer-api:

1. Connect to the concerned host.

2. Run the following commands to stop the freezer-api:

```
tux > sudo rm /etc/apache2/vhosts.d/freezer-modwsgi.conf
```

```
tux > sudo systemctl reload apache2
```

or run the following commands to start the freezer-api:

```
tux > sudo ln -s /etc/apache2/vhosts.d/freezer-modwsgi.vhost /etc/apache2/vhosts.d/
freezer-modwsgi.conf
```

```
tux > sudo systemctl reload apache2
```

14.13   Backing up and Restoring Audit Logs

To enable backup of the audit log directory, follow these steps. Before performing the following steps, run through *Section 12.2.7.2, "Enable Audit Logging"* .

- First, from the Cloud Lifecycle Manager node, run the following playbook:

```
tux > sudo ansible-playbook -i hosts/verb_hosts _freezer_manage_jobs.yml
```

  This will create a job to back up the audit log directory on any node where that directory exists. In order to limit this to only specific nodes, use the --limit option of Ansible

In order to restore the logs, follow one of the following procedures:

To restore from the node that made the backup to the same node directly in the audit log directory (for example, the folder has been deleted):

1. Connect to the node

2. Source OpenStack credentials

```
ardana > source ~/backup.osrc
```

3. List pre-configured jobs

```
ardana > freezer job-list -C `hostname`
```

4. Note the id corresponding to the job: "Ardana Default: Audit log restore from ..."

5. Schedule the restore

```
ardana > freezer job-start JOB-ID
```

To restore the backup in another directory, or from another host,

1. Connect to the node

2. Source the OpenStack credentials

```
ardana > source ~/backup.osrc
```

3. Choose from where you will restore (from Swift or from and SSH backup)

   a. Swift: Create a restore config file (for example, `restore.ini`) with the following content to restore from a swift backup (make sure to fill in < value >)

   ```
   [default]
   action = restore
   backup_name = freezer_audit_log_backup
   container = freezer_audit_backup
   log_file = /freezer-agent/freezer-agent.log
   restore_abs_path = PATH TO THE DIRECTORY WHERE YOU WANT TO RESTORE
   hostname = HOSTNAME OF THE HOST YOU WANT TO RESTORE THE BACKUP FROM
   ```

   b. Or: Create a restore configuration file (for example, `restore.ini`) with the following content to restore from an SSH backup (make sure to fill in *VALUE*) SSH information is available in `openstack/my_cloud/config/freezer/ssh_credentials.yml`

   ```
   [default]
   action = restore
   storage = ssh
   backup_name = freezer_audit_log_backup
   log_file = /freezer-agent/freezer-agent.log
   ```

```
ssh_key = /etc/freezer/ssh_key
restore_abs_path = PATH TO THE DIRECTORY WHERE YOU WANT TO RESTORE
hostname = HOSTNAME OF THE HOST YOU WANT TO RESTORE THE BACKUP FROM
ssh_host = YOUR SSH BACKUP HOST
ssh_port = YOUR SSH BACKUP PORT
ssh_username = YOUR SSH BACKUP USERNAME
container = YOUR SSH BACKUP BASEDIR/freezer_audit_backup
```

4. Run the freezer-agent to restore

```
freezer-agent --config restore.ini
```

15 Troubleshooting Issues

Troubleshooting and support processes for solving issues in your environment.

This section contains troubleshooting tasks for your HPE Helion OpenStack cloud.

15.1 General Troubleshooting

General troubleshooting procedures for resolving your cloud issues including steps for resolving service alarms and support contact information.

Before contacting support to help you with a problem on SUSE OpenStack Cloud, we recommend gathering as much information as possible about your system and the problem. For this purpose, HPE Helion OpenStack ships with a tool called `supportconfig`. It gathers system information such as the current kernel version being used, the hardware, RPM database, partitions, and other items. `supportconfig` also collects the most important log files. This information assists support staff to identify and solve your problem.

Always run `supportconfig` on the Cloud Lifecycle Manager and on the Control Node(s). If a Compute Node or a Storage Node is part of the problem, run `supportconfig` on the affected node as well. For details on how to run `supportconfig`, see https://documentation.suse.com/sles/12-SP5/single-html/SLES-admin/#cha-adm-support ↗.

15.1.1 Alarm Resolution Procedures

HPE Helion OpenStack provides a monitoring solution based on OpenStack's Monasca service. This service provides monitoring and metrics for all OpenStack components, as well as much of the underlying system. By default, HPE Helion OpenStack comes with a set of alarms that provide coverage of the primary systems. In addition, you can define alarms based on threshold values for any metrics defined in the system. You can view alarm information in the Operations Console. You can also receive or deliver this information to others by configuring email or other mechanisms. Alarms provide information about whether a component failed and is affecting the system, and also what condition triggered the alarm.

Here is a list of the included service-specific alarms and the recommended troubleshooting steps. We have organized these alarms by the section of the HPE Helion OpenStack Operations Console, they are organized in as well as the `service` dimension defined.

15.1.1.1 Compute Alarms

These alarms show under the Compute section of the HPE Helion OpenStack Operations Console.

15.1.1.1.1 SERVICE: COMPUTE

Alarm Information	Mitigation Tasks
Name: HTTP Status **Description:** This is a `nova-api` health check. **Likely cause:** Process crashed.	Restart the `nova-api` process on the affected node. Review the `nova-api.log` files. Try to connect locally to the http port that is found in the dimension field of the alarm to see if the connection is accepted.
Name: Host Status **Description::** Alarms when the specified host is down or not reachable. **Likely cause:** The host is down, has been rebooted, or has network connectivity issues.	If it is a single host, attempt to restart the system. If it is multiple hosts, investigate networking issues.
Name: Process Bound Check **Description::** `process_name=nova-api` This alarm checks that the number of processes found is in a predefined range. **Likely cause:** Process crashed or too many processes running	Stop all the processes and restart the nova-api process on the affected host. Review the system and nova-api logs.

Alarm Information	Mitigation Tasks
Name: Process Check **Description::** Separate alarms for each of these Nova services, specified by the `component` dimension: • nova-api • nova-cert • nova-compute • nova-consoleauth • nova-conductor • nova-scheduler • nova-novncproxy **Likely cause:** Process specified by the `component` dimension has crashed on the host specified by the `hostname` dimension.	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the Nova start playbook against the affected node: ``` cd ~/scratch/ansible/next/ardana/ansible ansible-playbook -i hosts/verb_hosts nova-start.yml \ --limit <hostname> ``` Review the associated logs. The logs will be in the format of `<service>.log`, such as `nova-compute.log` or `nova-scheduler.log`.
Name: nova.heartbeat **Description::** Check that all services are sending heartbeats. **Likely cause:** Process for service specified in the alarm has crashed or is hung and not reporting its status to the database.	Restart the affected service. If the service is reporting OK the issue may be with RabbitMQ or MySQL. In that case, check the alarms for those services.

Alarm Information	Mitigation Tasks
Alternatively it may be the service is fine but an issue with messaging or the database which means the status is not being updated correctly.	
Name: Service Log Directory Size **Description::** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.1.2 SERVICE: IMAGE-SERVICE in Compute section

Alarm Information	Mitigation Tasks
Name: HTTP Status	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager.

Alarm Information	Mitigation Tasks

Description:: Separate alarms for each of these Glance services, specified by the `component` dimension:
 	
  • glance-api
 	
  • glance-registry
 	
Likely cause: API is unresponsive.
	2. Use the Glance start playbook against the affected node:
	
	``
	ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts glance-start.yml \
	--limit <hostname>
	``
	
	Review the associated logs.

Name: Service Log Directory Size **Description::** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.1.3   SERVICE: BAREMETAL in Compute section

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = ironic-api` **Likely cause:** The Ironic API is unresponsive.	Restart the `ironic-api` process with these steps: 1. Log in to the affected host via SSH. 2. Restart the `ironic-api` process with this command: `sudo service ironic-api restart`

Name: Process Check
 	
Description: Alarms when the specified process is not running:
`process_name = ironic-conductor`
 	
Likely cause: The `ironic-conductor` process has crashed.
	Restart the `ironic-conductor` process with these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Source your `admin` user credentials:
	
	`source ~/service.osrc`
	
	3. Locate the `messaging_deployer` VM:
	
	`openstack server list --all-tenants | grep mess`
	
	4. SSH to the `messaging_deployer` VM:
	
	`sudo -u ardana ssh <IP_ADDRESS>`
	
	5. Stop the `ironic-conductor` process by using this playbook:
	
	`cd ~/scratch/ansible/next/ardana/ansible`
	`ansible-playbook -i hosts/verb_hosts ironic-stop.yml`
	
	6. Start the process back up again, effectively restarting it, by using this playbook:
	
	`cd ~/scratch/ansible/next/ardana/ansible`
	`ansible-playbook -i hosts/verb_hosts ironic-start.yml`

Alarm Information	Mitigation Tasks
Name: HTTP Status **Description:** Alarms when the specified HTTP endpoint is down or not reachable. **Likely cause:** The API is unresponsive.	1. Log in to the Cloud Lifecycle Manager. 2. Source your `admin` user credentials: ``` source ~/service.osrc ``` 3. Locate the `messaging_deployer` VM: ``` openstack server list --all-tenants | grep mess ``` 4. SSH to the `messaging_deployer` VM: ``` sudo -u ardana ssh <IP_ADDRESS> ``` 5. Stop the `ironic-api` process by using this playbook: ``` cd ~/scratch/ansible/next/ardana/ansible ansible-playbook -i hosts/verb_hosts ironic-stop.yml ``` 6. Start the process back up again, effectively restarting it, by using this playbook: ``` cd ~/scratch/ansible/next/ardana/ansible ansible-playbook -i hosts/verb_hosts ironic-start.yml ```
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
log rotate not configured properly so old log files are not being deleted properly.	

15.1.1.2 Storage Alarms

These alarms show under the Storage section of the HPE Helion OpenStack Operations Console.

15.1.1.2.1 SERVICE: OBJECT-STORAGE

Alarm Information	Mitigation Tasks
Name: swiftlm-scan monitor **Description:** Alarms if `swiftlm-scan` cannot execute a monitoring task. **Likely cause:** The `swiftlm-scan` program is used to monitor and measure a number of metrics. If it is unable to monitor or measure something, it raises this alarm.	Click on the alarm to examine the `Details` field and look for a `msg` field. The text may explain the error problem. To view/confirm this, you can also log into the host specified by the `hostname` dimension, and then run this command: ``` sudo swiftlm-scan | python -mjson.tool ``` The `msg` field is contained in the `value_meta` item.
Name: Swift account replicator last completed in 12 hours **Description:** Alarms if an `account-replicator` process did not complete a replication cycle within the last 12 hours.	Another cause of this problem may be that a file system may be corrupt. Look for sign of this in these logs on the affected node: ``` /var/log/swift/swift.log /var/log/kern.log ```

Alarm Information	Mitigation Tasks
| **Likely cause:** This can indicate that the `account-replication` process is stuck. | The file system may need to be wiped, contact Professional Services for advice on the best way to do that if needed. You can then reformat the file system with these steps:<br><br>1. Log in to the Cloud Lifecycle Manager.<br><br>2. Run the Swift deploy playbook against the affected node, which will format the wiped file system:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts
 swift-deploy.yml
--limit <hostname>
``` |
| **Name: Swift container replicator last** completed in 12 hours<br><br>**Description:** Alarms if a container-replicator process did not complete a replication cycle within the last 12 hours<br><br>**Likely cause:** This can indicate that the container-replication process is stuck. | SSH to the affected host and restart the process with this command:<br><br>```
sudo systemctl restart swift-container-replicator
```<br><br>Another cause of this problem may be that a file system may be corrupt. Look for sign of this in these logs on the affected node:<br><br>```
/var/log/swift/swift.log
/var/log/kern.log
```<br><br>The file system may need to be wiped, contact Professional Services for advice on the best way to do that if needed. You can then reformat the file system with these steps:<br><br>1. Log in to the Cloud Lifecycle Manager.<br><br>2. Run the Swift deploy playbook against the affected node, which will format the wiped file system:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts
 swift-deploy.yml \
--limit <hostname>
``` |

Alarm Information	Mitigation Tasks
Name: Swift object replicator last completed in 24 hours **Description:** Alarms if an object-replicator process did not complete a replication cycle within the last 24 hours **Likely cause:** This can indicate that the object-replication process is stuck.	SSH to the affected host and restart the process with this command: ``` sudo systemctl restart swift-account-replicator ``` Another cause of this problem may be that a file system may be corrupt. Look for sign of this in these logs on the affected node: ``` /var/log/swift/swift.log /var/log/kern.log ``` The file system may need to be wiped, contact Professional Services for advice on the best way to do that if needed. You can then reformat the file system with these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Run the Swift deploy playbook against the affected node, which will format the wiped file system: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-deploy.yml \ --limit <hostname> ```
Name: Swift configuration file ownership **Description:** Alarms if files/directories in `/etc/swift` are not owned by Swift. **Likely cause:** For files in `/etc/swift`, somebody may have manually edited or created a file.	For files in `/etc/swift`, use this command to change the file ownership: ``` ardana > sudo chown swift.swift /etc/swift/, /etc/swift/* ```

Alarm Information	Mitigation Tasks
Name: Swift data filesystem ownership **Description:** Alarms if files or directories in `/srv/node` are not owned by Swift. **Likely cause:** For directories in `/srv/node/*`, it may happen that the root partition was reimaged or reinstalled and the UID assigned to the Swift user change. The directories and files would then not be owned by the UID assigned to the Swift user.	For directories and files in `/srv/node/*`, compare the swift UID of this system and other systems and the UID of the owner of `/srv/node/*`. If possible, make the UID of the Swift user match the directories or files. Otherwise, change the ownership of all files and directories under the `/srv/node` path using a similar **chown swift.swift** command as above.
Name: Drive URE errors detected **Description:** Alarms if `swift-drive-audit` reports an unrecoverable read error on a drive used by the Swift service. **Likely cause:** An unrecoverable read error occurred when Swift attempted to access a directory.	The UREs reported only apply to file system metadata (that is, directory structures). For UREs in object files, the Swift system automatically deletes the file and replicates a fresh copy from one of the other replicas. UREs are a normal feature of large disk drives. It does not mean that the drive has failed. However, if you get regular UREs on a specific drive, then this may indicate that the drive has indeed failed and should be replaced. You can use standard XFS repair actions to correct the UREs in the file system. If the XFS repair fails, you should wipe the GPT table as follows (where <drive_name> is replaced by the actual drive name):

```
ardana > sudo dd if=/dev/zero of=/dev/sd<drive_name> \
bs=$((1024*1024)) count=1
```

Alarm Information	Mitigation Tasks
	Then follow the steps below which will reformat the drive, remount it, and restart Swift services on the affected node. 1. Log in to the Cloud Lifecycle Manager. 2. Run the Swift reconfigure playbook, specifying the affected node: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts _swift-configure.yml \ --limit <hostname> ``` It is safe to reformat drives containing Swift data because Swift maintains other copies of the data (usually, Swift is configured to have three replicas of all data).
Name: Swift service **Description:** Alarms if a Swift process, specified by the `component` field, is not running. **Likely cause:** A daemon specified by the `component` dimension on the host specified by the `hostname` dimension has stopped running.	Examine the `/var/log/swift/swift.log` file for possible error messages related the Swift process. The process in question is listed in the alarm dimensions in the `component` dimension. Restart Swift processes by running the `swift-start.yml` playbook, with these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Run the Swift start playbook against the affected host: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-start.yml \ --limit <hostname> ```
Name: Swift filesystem mount point status **Description:** Alarms if a file system/drive used by Swift is not correctly mounted.	Reboot the node and see if the file system remains unmounted. If the file system is corrupt, see the process used for the "Drive URE errors" alarm to wipe and reformat the drive.

Alarm Information	Mitigation Tasks
Likely cause: The device specified by the `device` dimension is not correctly mounted at the mountpoint specified by the `mount` dimension. The most probable cause is that the drive has failed or that it had a temporary failure during the boot process and remained unmounted. Other possible causes are a file system corruption that prevents the device from being mounted.	

Name: Swift uptime-monitor status
 	
Description: Alarms if the swiftlm-uptime-monitor has errors using Keystone (`keystone-get-token`), Swift (`rest-api`) or Swift's healthcheck.
 	
Likely cause: The swiftlm-uptime-monitor cannot get a token from Keystone or cannot get a successful response from the Swift Object-Storage API.
	Check that the Keystone service is running:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Check the status of the Keystone service:
	
	```\nardana > cd ~/scratch/ansible/next/ardana/ansible/\nardana > ansible-playbook -i hosts/verb_hosts\n keystone-status.yml\n```
	
	3. If it is not running, start the service:
	
	```\nardana > cd ~/scratch/ansible/next/ardana/ansible/\nardana > ansible-playbook -i hosts/verb_hosts\n keystone-start.yml\n```
	
	4. Contact the support team if further assistance troubleshooting the Keystone service is needed.

Alarm Information	Mitigation Tasks
	Check that Swift is running: 1. Log in to the Cloud Lifecycle Manager. 2. Check the status of the Keystone service: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-status.yml ``` 3. If it is not running, start the service: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-start.yml ``` Restart the swiftlm-uptime-monitor as follows: 1. Log into the first server running the swift-proxy-server service. Use this playbook below to determine whcih host this is: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-status.yml --limit SWF-PRX[0] ``` 2. Restart the swiftlm-uptime-monitor with this command: ``` ardana > sudo systemctl restart swiftlm-uptime-monitor ```
Name: Swift Keystone server connect **Description:** Alarms if a socket cannot be opened to the Keystone service (used for token validation)	The `URL` dimension contains the name of the virtual IP address. Use cURL or a similar program to confirm that a connection can or cannot be made to the virtual IP address. Check that `haproxy` is running. Check that the Keystone service is working.

Alarm Information	Mitigation Tasks
Likely cause: The Identity service (Keystone) server may be down. Another possible cause is that the network between the host reporting the problem and the Keystone server or the `haproxy` process is not forwarding requests to Keystone.	
Name: Swift service listening on ip and port **Description:** Alarms when a Swift service is not listening on the correct port or ip. **Likely cause:** The Swift service may be down.	Verify the status of the Swift service on the affected host, as specified by the `hostname` dimension. 1. Log in to the Cloud Lifecycle Manager. 2. Run the Swift status playbook to confirm status: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-status.yml \ --limit <hostname> ``` If an issue is determined, you can stop and restart the Swift service with these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Stop the Swift service on the affected host: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts Swift-stop.yml \ --limit <hostname> ``` 3. Restart the Swift service on the affected host: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-start.yml \ ```

Alarm Information	Mitigation Tasks
	``` --limit <hostname> ```
Name: Swift rings checksum **Description:** Alarms if the Swift rings checksums do not match on all hosts. **Likely cause:** The Swift ring files must be the same on every node. The files are located in `/etc/swift/ *.ring.gz`. If you have just changed any of the rings and you are still deploying the change, it is normal for this alarm to trigger.	If you have just changed any of your Swift rings, if you wait until the changes complete then this alarm will likely clear on its own. If it does not, then continue with these steps. Use **`sudo swift-recon --md5`** to find which node has outdated rings. Run the `swift-reconfigure.yml` playbook, using the steps below. This deploys the same set of rings to every node. 1. Log in to the Cloud Lifecycle Manager. 2. Run the Swift start playbook against the affected host: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift- reconfigure.yml ```
Name: Swift memcached server connect **Description:** Alarms if a socket cannot be opened to the specified memcached server. **Likely cause:** The server may be down. The memcached daemon running the server may have stopped.	If the server is down, restart it. If memcached has stopped, you can restart it by using the `memcached-start.yml` playbook, using the steps below. If this fails, rebooting the node will restart the process. 1. Log in to the Cloud Lifecycle Manager. 2. Run the memcached start playbook against the affected host: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts memcached-start.yml \ --limit <hostname> ``` If the server is running and memcached is running, there may be a network problem blocking port 11211.

Alarm Information	Mitigation Tasks
	If you see sporadic alarms on different servers, the system may be running out of resources. Contact Professional Services for advice.
Name: Swift individual disk usage exceeds 80% **Description:** Alarms when a disk drive used by Swift exceeds 80% utilization. **Likely cause:** Generally all disk drives will fill roughly at the same rate. If an individual disk drive becomes filled faster than other drives it can indicate a problem with the replication process.	If many or most of your disk drives are 80% full, you need to add more nodes to your system or delete existing objects. If one disk drive is noticeably (more than 30%) more utilized than the average of other disk drives, check that Swift processes are working on the server (use the steps below) and also look for alarms related to the host. Otherwise continue to monitor the situation. 1. Log in to the Cloud Lifecycle Manager. 2. Run the Swift status: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-status.yml ```
Name: Swift individual disk usage exceeds 90% **Description:** Alarms when a disk drive used by Swift exceeds 90% utilization. **Likely cause:** Generally all disk drives will fill roughly at the same rate. If an individual disk drive becomes filled faster than other drives it can indicate a problem with the replication process.	If one disk drive is noticeably (more than 30%) more utilized than the average of other disk drives, check that Swift processes are working on the server, using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Run the Swift status: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-status.yml ``` Also look for alarms related to the host. An individual disk drive filling can indicate a problem with the replication process.

Alarm Information	Mitigation Tasks
	Restart Swift on that host using the `--limit` argument to target the host: 1. Log in to the Cloud Lifecycle Manager. 2. Stop the Swift service: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-stop.yml \ --limit <hostname> ``` 3. Start the Swift service back up: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible/ ardana > ansible-playbook -i hosts/verb_hosts swift-start.yml \ --limit <hostname> ``` If the utilization does not return to similar values as other disk drives, you can reformat the disk drive. You should only do this if the average utilization of all disk drives is less than 80%. To format a disk drive contact Professional Services for instructions.
Name: Swift total disk usage exceeds 80% **Description:** Alarms when the average disk utilization of Swift disk drives exceeds 80% utilization. **Likely cause:** The number and size of objects in your system is beginning to fill the available disk space. Account and container storage is included in disk utilization. However, this	You need to add more nodes to your system or delete existing objects to remain under 80% utilization. If you delete a project/account, the objects in that account are not removed until a week later by the `account-reaper` process, so this is not a good way of quickly freeing up space.

Alarm Resolution Procedures   HPE Helion OpenStack 8

Alarm Information	Mitigation Tasks
generally consumes 1-2% of space compared to objects, so object storage is the dominate consumer of disk space.	
Name: Swift total disk usage exceeds 90% **Description:** Alarms when the average disk utilization of Swift disk drives exceeds 90% utilization. **Likely cause:** The number and size of objects in your system is beginning to fill the available disk space. Account and container storage is included in disk utilization. However, this generally consumes 1-2% of space compared to objects, so object storage is the dominate consumer of disk space.	If your disk drives are 90% full, you must immediately stop all applications that put new objects into the system. At that point you can either delete objects or add more servers. Using the steps below, set the `fallocate_reserve` value to a value higher than the currently available space on disk drives. This will prevent more objects being created. 1. Log in to the Cloud Lifecycle Manager. 2. Edit the configuration files below and change the value for `fallocate_reserve` to a value higher than the currently available space on the disk drives: ``` ~/openstack/my_cloud/config/swift/account- server.conf.j2 ~/openstack/my_cloud/config/swift/container- server.conf.j2 ~/openstack/my_cloud/config/swift/object- server.conf.j2 ``` 3. Commit the changes to git: ``` ardana > git add -A ardana > git commit -a -m "changing Swift fallocate_reserve value" ``` 4. Run the configuration processor: ``` ardana > cd ~/openstack/ardana/ansible ardana > ansible-playbook -i hosts/localhost config- processor-run.yml ``` 5. Update your deployment directory: ``` ardana > cd ~/openstack/ardana/ansible ```

Alarm Resolution Procedures   HPE Helion OpenStack 8

Alarm Information	Mitigation Tasks
| | ```
ardana > ansible-playbook -i hosts/localhost ready-
deployment.yml
```<br><br>6. Run the Swift reconfigure playbook to deploy the change:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts swift-
reconfigure.yml
```<br><br>If you allow your file systems to become full, you will be unable to delete objects or add more nodes to the system. This is because the system needs some free space to handle the replication process when adding nodes. With no free space, the replication process cannot work. |
| **Name: Swift service per-minute availability**

Description: Alarms if the Swift service reports unavailable for the previous minute.

Likely cause: The `swiftlm-uptime-monitor` service runs on the first proxy server. It monitors the Swift endpoint and reports latency data. If the endpoint stops reporting, it generates this alarm. | There are many reasons why the endpoint may stop running. Check:

• Is `haproxy` running on the control nodes?

• Is `swift-proxy-server` running on the Swift proxy servers? |
| **Name: Swift rsync connect**<br><br>**Description:** Alarms if a socket cannot be opened to the specified rsync server | Reboot the server if it is down.<br><br>Attempt to restart rsync with this command:<br><br>```
systemctl restart rsync.service
``` |

Alarm Information	Mitigation Tasks
Likely cause: The rsync daemon on the specified node cannot be contacted. The most probable cause is that the node is down. The rsync service might also have been stopped on the node.	
Name: Swift smart array controller status **Description:** Alarms if there is a failure in the Smart Array. **Likely cause:** The Smart Array or Smart HBA controller has a fault or a component of the controller (such as a battery) is failed or caching is disabled. The HPE Smart Storage Administrator (HPE SSA) CLI component will have to be installed for SSACLI status to be reported. HPE-specific binaries that are not based on open source are distributed directly from and supported by HPE. To download and install the SSACLI utility, please refer to: https://support.hpe.com/hpsc/swd/public/detail?swItemId=MTX_3d16386b418a443388c18da82f ⬈	Log in to the reported host and run these commands to find out the status of the controllers: ``` sudo hpssacli => controller show all detail ``` For hardware failures (such as failed battery), replace the failed component. If the cache is disabled, reenable the cache.

Alarm Information	Mitigation Tasks
Name: Swift physical drive status **Description:** Alarms if there is a failure in the Physical Drive. **Likely cause:** A disk drive on the server has failed or has warnings.	Log in to the reported and run these commands to find out the status of the drive: ``` sudo hpssacli => ctrl slot=1 pd all show ``` Replace any broken drives.
Name: Swift logical drive status **Description:** Alarms if there is a failure in the Logical Drive. **Likely cause:** A LUN on the server is degraded or has failed.	Log in to the reported host and run these commands to find out the status of the LUN: ``` sudo hpssacli => ctrl slot=1 ld all show => ctrl slot=1 pd all show ``` Replace any broken drives.
Name: Process Check **Description:** Alarms when the specified process is not running. **Likely cause:** If the `service` dimension is `object-store`, see the description of the "Swift Service" alarm for possible causes.	If the `service` dimension is `object-storage`, see the description of the "Swift Service" alarm for possible mitigation tasks.
Name: HTTP Status **Description:** Alarms when the specified HTTP endpoint is down or not reachable.	If the `service` dimension is `object-storage`, see the description of the "Swift host socket connect" alarm for possible mitigation tasks.

Alarm Information	Mitigation Tasks
Likely cause: If the `service` dimension is `object-store`, see the description of the "Swift host socket connect" alarm for possible causes.	
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.2.2   SERVICE: BLOCK-STORAGE in Storage section

Alarm Information	Mitigation Tasks
Name: Process Check	Restart the process on the affected node. Review the associated logs. 1. Log in to the Cloud Lifecycle Manager.

Alarm Information	Mitigation Tasks

Description: Separate alarms for each of these Cinder services, specified by the `component` dimension:
 	
  • cinder-api
 	
  • cinder-backup
 	
  • cinder-scheduler
 	
  • cinder-volume
 	
Likely cause: Process crashed.
	2. Run the `cinder-start.yml` playbook to start the process back up:
	
	``
	ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts cinder-start.yml
	--limit <hostname>
	``
	
	☞ **Note**
	
	The `--limit <hostname>` switch is optional. If it is included, then the `<hostname>` you should use is the host where the alarm was raised.

Name: Process Check **Description:** Alarms when the specified process is not running: `process_name=cinder-backup` **Likely cause:** Process crashed.	Alert may be incorrect if the service has migrated. Validate that the service is intended to be running on this node before restarting the service. Review the associated logs.
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name=cinder-scheduler` **Likely cause:** Process crashed.	Restart the process on the affected node. Review the associated logs. 1. Log in to the Cloud Lifecycle Manager. 2. Run the `cinder-start.yml` playbook to start the process back up: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts cinder-start.yml \ --limit <hostname> ```

Alarm Information	Mitigation Tasks
	🎓 **Note** The `--limit <hostname>` switch is optional. If it is included, then the `<hostname>` you should use is the host where the alarm was raised.
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name=cinder-volume` **Likely cause:** Process crashed.	Alert may be incorrect if the service has migrated. Validate that the service is intended to be running on this node before restarting the service. Review the associated logs.
Name: Cinder backup running <**hostname**> **check** **Description:** Cinder backup singleton check. **Likely cause:** Backup process is one of the following: • It is running on a node it should not be on • It is not running on a node it should be on	Run the `cinder-migrate-volume.yml` playbook to migrate the volume and back up to the correct node: 1. Log in to the Cloud Lifecycle Manager. 2. Run this playbook to migrate the service: ```\nardana > cd ~/scratch/ansible/next/ardana/ansible/\nardana > ansible-playbook -i hosts/verb_hosts cinder-\nmigrate-volume.yml\n```

Alarm Information	Mitigation Tasks

Name: Cinder volume running <**hostname**> **check**
 	
Description: Cinder volume singleton check.
 	
Likely cause: The `cinder-volume` process is either:
 	
• running on a node it should not be on, or
 	
• not running on a node it should be on
	Run the `cinder-migrate-volume.yml` playbook to migrate the volume and backup to correct node:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Run this playbook to migrate the service:
	
	<pre>ardana > cd ~/scratch/ansible/next/ardana/ansible/
	ardana > ansible-playbook -i hosts/verb_hosts cinder-
	migrate-volume.yml</pre>

Name: Storage faulty lun check
 	
Description: Alarms if local LUNs on your HPE servers using smartarray are not OK.
 	
Likely cause: A LUN on the server is degraded or has failed.
	Log in to the reported host and run these commands to find out the status of the LUN:
	
	<pre>sudo hpssacli
	=> ctrl slot=1 ld all show
	=> ctrl slot=1 pd all show</pre>
	
	Replace any broken drives.

Name: Storage faulty drive check
 	
Description: Alarms if the local disk drives on your HPE servers using smartarray are not OK.
 	
Likely cause: A disk drive on the server has failed or has warnings.
	Log in to the reported and run these commands to find out the status of the drive:
	
	<pre>sudo hpssacli
	=> ctrl slot=1 pd all show</pre>
	
	Replace any broken drives.

Alarm Information	Mitigation Tasks
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.3   Networking Alarms

These alarms show under the Networking section of the HPE Helion OpenStack Operations Console.

15.1.1.3.1   SERVICE: NETWORKING

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** Alarms when the specified process is not running. Separate alarms for	Restart the process on the affected node: 1. Log in to the Cloud Lifecycle Manager. 2. Check the status of the networking status: `ardana > cd ~/scratch/ansible/next/ardana/ansible`

Alarm Information	Mitigation Tasks

each of these Neutron services, specified by the `component` dimension:
 	
• ipsec/charon
 	
• neutron-openvswitch-agent
 	
• neutron-l3-agent
 	
• neutron-dhcp-agent
 	
• neutron-metadata-agent
 	
• neutron-server
 	
• neutron-vpn-agent
 	
Likely cause: Process crashed.
	```ansible-playbook -i hosts/verb_hosts neutron-status.yml```
	
	3. Make note of the failed service names and the affected hosts which you will use to review the logs later.
	
	4. Using the affected hostname(s) from the previous output, run the Neutron start playbook to restart the services:
	
	```ardana > cd ~/scratch/ansible/next/ardana/ansible```
	```ardana > ansible-playbook -i hosts/verb_hosts neutron-start.yml \```
	```--limit <hostname>```
	
	🎓 **Note**
	<blockquote>You can pass multiple hostnames with `--limit` option by separating them with a colon `:`.</blockquote>
	5. Check the status of the networking service again:
	
	```ardana > cd ~/scratch/ansible/next/ardana/ansible```
	```ardana > ansible-playbook -i hosts/verb_hosts neutron-status.yml```
	
	6. Once all services are back up, you can SSH to the affected host(s) and review the logs in the location below for any errors around the time that the alarm triggered:
	
	```/var/log/neutron/<service_name>```

Name: Process Check **Description:** Alarms when the specified process is not running: ```process_name = neutron-rootwrap```	Currently `neutron-rootwrap` is only used to run `ovsdb-client`. To restart this process, use these steps: 1. SSH to the affected host(s). 2. Restart the process: ```sudo systemctl restart neutron-openvswitch-agent```

Alarm Information	Mitigation Tasks
Likely cause: Process crashed.	3. Review the logs at the location below for errors: `/var/log/neutron/neutron-openvswitch-agent.log`

Name: HTTP Status
 	
Description: neutron api health check
 	
Likely cause: Process is stuck if the `neutron-server` Process Check is not OK.
	1. SSH to the affected host(s).
	
	2. Run this command to restart the `neutron-server` process:
	
	`sudo systemctl restart neutron-server`
	
	3. Review the logs at the location below for errors:
	
	`/var/log/neutron/neutron-server.log`

Name: HTTP Status **Description:** neutron api health check **Likely cause:** The node crashed. Alternatively, only connectivity might have been lost if the local node HTTP Status is OK or UNKNOWN.	Reboot the node if it crashed or diagnose the networking connectivity failures between the local and remote nodes. Review the logs.
Name: Service Directory Log Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	

15.1.1.3.2 SERVICE: DNS in Networking section

Alarm Information	Mitigation Tasks

Name: Process Check
 	
Description: Alarms when the specified process is not running:
 	
`process_name = designate-zone-manager`
 	
Likely cause: Process crashed.
	Restart the process on the affected node using these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Use the Designate start playbook against the affected node:
	
	```ardana > cd ~/scratch/ansible/next/ardana/ansible\nardana > ansible-playbook -i hosts/verb_hosts\n designate-start.yml \\\n--limit 'DES-ZMG'```
	
	Review the log located at:
	
	`/var/log/designate/designate-zone-manager.log`

Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = designate-pool-manager` **Likely cause:** Process crashed.	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the Designate start playbook against the affected node: ```ardana > cd ~/scratch/ansible/next/ardana/ansible\nardana > ansible-playbook -i hosts/verb_hosts\n designate-start.yml \\\n--limit 'DES-PMG'```

Alarm Information	Mitigation Tasks
	Review the log located at: `/var/log/designate/designate-pool-manager.log`

Name: Process Check
 	
Description: Alarms when the specified process is not running:
 	
`process_name = designate-central`
 	
Likely cause: Process crashed.
	Restart the process on the affected node using these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Use the Designate start playbook against the affected node:
	
	```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts designate-start.yml \ --limit 'DES-CEN'```
	
	Review the log located at:
	
	`/var/log/designate/designate-central.log`

Name: Process Check
 	
Description: Alarms when the specified process is not running:
 	
`process_name = designate-api`
 	
Likely cause: Process crashed.
	Restart the process on the affected node using these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Use the Designate start playbook against the affected node:
	
	```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts designate-start.yml \ --limit 'DES-API'```
	
	Review the log located at:
	
	`/var/log/designate/designate-api.log`

Name: Process Check	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager.

Alarm Information	Mitigation Tasks

Description: Alarms when the specified process is not running:
 	
``
process_name =
  designate-mdns
``
 	
Likely cause: Process crashed.
	2. Use the Designate start playbook against the affected node:
	
	``
	        ardana > cd ~/scratch/ansible/next/ardana/
	ansible
	ardana > ansible-playbook -i hosts/verb_hosts
	 designate-start.yml \
	--limit 'DES-MDN'
	``
	
	Review the log located at:
	
	``
	/var/log/designate/designate-mdns.log
	``

Name: HTTP Status **Description:** `component = designate-api` This alarm will also have the `api_endpoint` and `monitored_host_types` dimensions defined. The likely cause and mitigation steps are the same for both. **Likely cause:** The API is unresponsive.	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the Designate start playbook against the affected node: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts designate-start.yml \ --limit 'DES-API,DES-CEN' ``` Review the logs located at: ``` /var/log/designate/designate-api.log /var/log/designate/designate-central.log ```
Name: Service Directory Log Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	

15.1.1.3.3 SERVICE: BIND in Networking section

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = pdns_server` **Likely cause:** Process crashed.	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the PowerDNS start playbook against the affected node: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts bind-start.yml ``` Review the log located at, querying against `process = pdns_server`: ``` /var/log/syslog ```
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = named` **Likely cause:** Process crashed.	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the Bind start playbook against the affected node: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts bind-start.yml ```

Alarm Information	Mitigation Tasks
	Review the log located at, querying against `process = named`: `/var/log/syslog`

15.1.1.4 Identity Alarms

These alarms show under the Identity section of the HPE Helion OpenStack Operations Console.

15.1.1.4.1 SERVICE: IDENTITY-SERVICE

Alarm Information	Mitigation Tasks

Name: HTTP Status
 	
Description: This check is contacting the Keystone public endpoint directly.
 	
`component=keystone-api`
`api_endpoint=public`
 	
Likely cause: The Keystone service is down on the affected node.
	Restart the Keystone service on the affected node:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Use the Keystone start playbook against the affected node:
	
	`cd ~/scratch/ansible/next/ardana/ansible`
	`ansible-playbook -i hosts/verb_hosts keystone-start.yml \`
	`--limit <hostname>`

Name: HTTP Status
 	
Description: This check is contacting the Keystone admin endpoint directly
 	
`component=keystone-api`
`api_endpoint=admin`
 	
Likely cause: The Keystone service is down on the affected node.
	Restart the Keystone service on the affected node:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Use the Keystone start playbook against the affected node:
	
	`cd ~/scratch/ansible/next/ardana/ansible`
	`ansible-playbook -i hosts/verb_hosts keystone-start.yml \`
	`--limit <hostname>`

Alarm Information	Mitigation Tasks

Name: HTTP Status
 	
Description: This check is contacting the Keystone admin endpoint via the virtual IP address (HAProxy)
 	
`component=keystone-api`
`monitored_host_type=vip`
 	
Likely cause: The Keystone service is unreachable via the virtual IP address.
	If neither the `api_endpoint=public` or `api_endpoint=admin` alarms are triggering at the same time then there is likely a problem with haproxy.
	
	You can restart the haproxy service with these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Use this playbook against the affected node:
	
	``
	cd ~/scratch/ansible/next/ardana/ansible
	ansible-playbook -i hosts/verb_hosts FND-CLU-start.yml \
	--limit <hostname>
	``

Name: Process Check **Description:** Separate alarms for each of these Glance services, specified by the `component` dimension: • keystone-main • keystone admin **Likely cause:** Process crashed.	You can restart the Keystone service with these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use this playbook against the affected node: ``` cd ~/scratch/ansible/next/ardana/ansible ansible-playbook -i hosts/verb_hosts keystone-start.yml \ --limit <hostname> ``` Review the logs in `/var/log/keystone` on the affected node.
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
Likely cause: This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	

15.1.1.5 Telemetry Alarms

These alarms show under the Telemetry section of the HPE Helion OpenStack Operations Console.

15.1.1.5.1 SERVICE: TELEMETRY

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** Alarms when the `ceilometer-agent-notification` process is not running. **Likely cause:** Process has crashed.	Review the logs on the alarming host in the following location for the cause: `/var/log/ceilometer/ceilometer-agent-notification-json.log` Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the Ceilometer start playbook against the affected node: `ardana > cd ~/scratch/ansible/next/ardana/ansible`

Alarm Information	Mitigation Tasks
| | ```
ardana > ansible-playbook -i hosts/verb_hosts
 ceilometer-start.yml \
--limit <hostname>
``` |
| **Name: Process Check**<br><br>**Description:** Alarms when the `ceilometer-polling` process is not running.<br><br>**Likely cause:** Process has crashed. | Review the logs on the alarming host in the following location for the cause:<br><br>```
/var/log/ceilometer/ceilometer-polling-json.log
```<br><br>Restart the process on the affected node using these steps:<br><br>1. Log in to the Cloud Lifecycle Manager.<br><br>2. Use the Ceilometer start playbook against the affected node:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts
 ceilometer-start.yml \
--limit <hostname>
``` |

15.1.1.5.2 SERVICE: METERING in Telemetry section

Alarm Information	Mitigation Tasks
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	

15.1.1.5.3 SERVICE: KAFKA in Telemetry section

Alarm Information	Mitigation Tasks
Name: Kafka Persister Metric Consumer Lag **Description:** Alarms when the Persister consumer group is not keeping up with the incoming messages on the metric topic. **Likely cause:** There is a slow down in the system or heavy load.	Verify that all of the monasca-persister services are up with these steps: 1. Log in to the Cloud Lifecycle Manager 2. Verify that all of the `monasca-persister` services are up with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-persister ``` Look for high load in the various systems. This alert can fire for multiple topics or on multiple hosts. Determining which alarms are firing can help diagnose likely causes. For example, if the alarm is alerting all on one machine it could be the machine. If one topic across multiple machines it is likely the consumers of that topic, etc.
Name: Kafka Alarm Transition Consumer Lag	Check that monasca-thresh and monasca-notification are up.

Alarm Information	Mitigation Tasks
Description: Alarms when the specified consumer group is not keeping up with the incoming messages on the alarm state transition topic. **Likely cause:** There is a slow down in the system or heavy load.	Look for high load in the various systems. This alert can fire for multiple topics or on multiple hosts. Which alarms are firing can help diagnose likely causes. For example: • If all alarms are on the same machine, the machine could be at fault. • If one topic is shared across multiple machines, the consumers of that topic are likely at fault.
Name: Kafka Kronos Consumer Lag **Description:** Alarms when the Kronos consumer group is not keeping up with the incoming messages on the metric topic. **Likely cause:** There is a slow down in the system or heavy load.	Look for high load in the various systems. This alert can fire for multiple topics or on multiple hosts. Which alarms are firing can help diagnose likely causes. For example: • If all alarms are on the same machine, the machine could be at fault. • If one topic is shared across multiple machines, the consumers of that topic are likely at fault.
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = kafka.Kafka` **Likely cause:**	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Stop the kafka service with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --tags kafka ``` 3. Start the kafka service back up with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ```

Alarm Information	Mitigation Tasks
| | ```
ardana > ansible-playbook -i hosts/verb_hosts monasca-
status.yml \
--tags kafka
``` |
| | Review the logs in `/var/log/kafka/server.log` |

15.1.1.5.4 SERVICE: LOGGING in Telemetry section

Alarm Information	Mitigation Tasks
Name: Beaver Memory Usage **Description:** Beaver is using more memory than expected. This may indicate that it cannot forward messages and its queue is filling up. If you continue to see this, see the troubleshooting guide. **Likely cause:** Overloaded system or services with memory leaks.	Log on to the reporting host to investigate high memory users.
Name: Audit Log Partition Low Watermark **Description:** The `/var/audit` disk space usage has crossed low watermark. If the high watermark is reached,	Find the service that is consuming too much disk space. Look at the logs. If DEBUG log entries exist, set the logging level to INFO. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
logrotate will be run to free up disk space. If needed, adjust: `var_audit_low_watermark_percent` **Likely cause:** This could be due to a service set to DEBUG instead of INFO level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	
Name: Audit Log Partition High Watermark **Description:** The `/var/audit` volume is running low on disk space. Logrotate will be run now to free up space. If needed, adjust: `var_audit_high_watermark_percent` **Likely cause:** This could be due to a service set to DEBUG instead of INFO level. Another reason could be due to a	Find the service that is consuming too much disk space. Look at the logs. If DEBUG log entries exist, set the logging level to INFO. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	
Name: Elasticsearch Unassigned Shards **Description:** component = elasticsearch; Elasticsearch unassigned shards count is greater than 0. **Likely cause:** Environment could be misconfigured.	To find the unassigned shards, run the following command on the Cloud Lifecycle Manager from the `~/scratch/ansible/next/ardana/ansible` directory: ```ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible -i hosts/verb_hosts LOG-SVR[0] -m shell -a \
"curl localhost:9200/_cat/shards?pretty -s"	grep

UNASSIGNED```
 	
 This shows which shards are unassigned, like this:
 	
 ```logstash-2015.10.21 4 p UNASSIGNED ... 10.240.75.10 NodeName```
 	
 The last column shows the name that Elasticsearch uses for the node that the unassigned shards are on. To find the actual host name, run:
 	
 ``ardana > cd ~/scratch/ansible/next/ardana/ansible

ardana > ansible -i hosts/verb_hosts LOG-SVR[0] -m shell -a \
"curl localhost:9200/_nodes/_all/name?pretty -s"```   When you find the host name, take the following steps: 1. Make sure the node is not out of disk space, and free up space if needed.   2. Restart the node (use caution, as this may affect other services as well).

Alarm Information	Mitigation Tasks
	3. Make sure all versions of Elasticsearch are the same: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible -i hosts/verb_hosts LOG-SVR -m shell -a \ "curl localhost:9200/_nodes/_local/name?pretty -s"	grep version ``` 4. Contact customer support.
Name: Elasticsearch Number of Log Entries **Description:** Elasticsearch Number of Log Entries: `component = elasticsearch;` **Likely cause:** The number of log entries may get too large.	Older versions of Kibana (version 3 and earlier) may hang if the number of log entries is too large (for example, above 40,000), and the page size would need to be small enough (about 20,000 results), because if it is larger (for example, 200,000), it may hang the browser, but Kibana 4 should not have this issue.
Name: Elasticsearch Field Data Evictions **Description:** Elasticsearch Field Data Evictions count is greater than 0: `component = elasticsearch` **Likely cause:** Field Data Evictions may be found even though it is nowhere near the limit set.	The `elasticsearch_indices_fielddata_cache_size` is set to `unbounded` by default. If this is set by the user to a value that is insufficient, you may need to increase this configuration parameter or set it to `unbounded` and run a reconfigure using the steps below: 1. Log in to the Cloud Lifecycle Manager. 2. Edit the configuration file below and change the value for `elasticsearch_indices_fielddata_cache_size` to your desired value: ``` ~/openstack/my_cloud/config/logging/main.yml ``` 3. Commit the changes to git: ``` ardana > git add -A ```

Alarm Information	Mitigation Tasks
| | ```
ardana > git commit -a -m "Elasticsearch fielddata
 cache size"
``` |

4. Run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-
processor-run.yml
```

5. Update your deployment directory:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost ready-
deployment.yml
```

6. Run the Logging reconfigure playbook to deploy the change:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible/
ardana > ansible-playbook -i hosts/verb_hosts kronos-
reconfigure.yml
```

Alarm Information	Mitigation Tasks
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
properly so old log files are not being deleted properly.	

Name: Process Check
 	
Description: Separate alarms for each of these logging services, specified by the `process_name` dimension:
 	
   • elasticsearch
 	
   • logstash
 	
   • beaver
 	
   • apache2
 	
   • kibana
 	
Likely cause: Process has crashed.
	On the affected node, attempt to restart the process.
	
	If the `elasticsearch` process has crashed, use:
	
	``
	ardana > sudo systemctl restart elasticsearch
	``
	
	If the logstash process has crashed, use:
	
	``
	ardana > sudo systemctl restart logstash
	``
	
	The rest of the processes can be restarted using similar commands, listed here:
	
	``
	ardana > sudo systemctl restart beaver
	ardana > sudo systemctl restart apache2
	ardana > sudo systemctl restart kibana
	``

15.1.1.5.5 SERVICE: MONASCA-TRANSFORM in Telemetry section

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** `process_name = pyspark` **Likely cause:** Service process has crashed.	Restart process on affected node. Review logs. Child process of `spark-worker` but created once the `monasca-transform` process begins processing streams. If the process fails on one node only, along with the pyspark process, it is likely that the `spark-worker` has failed to connect to the elected leader of the `spark-master` service. In this case the `spark-worker` service should be started on the affected node. If on multiple nodes check the `spark-worker`, `spark-master` and `monasca-transform` services and logs. If the `monasca-`

Alarm Information	Mitigation Tasks
	`transform` or `spark` services have been interrupted this process may not re-appear for up to ten minutes (the stream processing interval).
Name: Process Check **Description:** `process_name = org.apache.spark.executor.CoarseGrainedExecutorBackend` **Likely cause:** Service process has crashed.	Restart process on affected node. Review logs. Child process of `spark-worker` but created once the `monasca-transform` process begins processing streams. If the process fails on one node only, along with the pyspark process, it is likely that the `spark-worker` has failed to connect to the elected leader of the `spark-master` service. In this case the `spark-worker` service should be started on the affected node. If on multiple nodes check the `spark-worker`, `spark-master` and `monasca-transform` services and logs. If the `monasca-transform` or `spark` services have been interrupted this process may not re-appear for up to ten minutes (the stream processing interval).
Name: Process Check **Description:** `process_name = monasca-transform` **Likely cause:** Service process has crashed.	Restart the service on affected node. Review logs.

15.1.1.5.6   SERVICE: MONITORING in Telemetery section

Alarm Information	Mitigation Tasks
Name: HTTP Status **Description:** Persister Health Check `component = monasca-persister`	If the process has crashed, restart it using the steps below. If a dependent service is down, address that issue. Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager.

Alarm Information	Mitigation Tasks
Likely cause: The process has crashed or a dependency is out.	2. Check if `monasca-api` is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-persister ``` 3. Use the Monasca start playbook against the affected node to restart it: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --tags persister ``` 4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-persister ``` Review the associated logs.
Name: HTTP Status **Description:** API Health Check `component = monasca-api` **Likely cause:** The process has crashed or a dependency is out.	If the process has crashed, restart it using the steps below. If a dependent service is down, address that issue. Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Check if `monasca-api` is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-api ```

Alarm Information	Mitigation Tasks
	3. Use the Monasca start playbook against the affected node to restart it: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --tags monasca-api ``` 4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-api ``` Review the associated logs.
Name: Monasca Agent Collection Time **Description:** Alarms when the elapsed time the `monasca-agent` takes to collect metrics is high. **Likely cause:** Heavy load on the box or a stuck agent plug-in.	Address the load issue on the machine. If needed, restart the agent using the steps below: Restart the agent on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Check if `monasca-agent` is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-agent-status.yml ``` 3. Use the Monasca start playbook against the affected node to restart it: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --limit <hostname> ``` 4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ```

Alarm Information	Mitigation Tasks
| | ```
ardana > ansible-playbook -i hosts/verb_hosts monasca-
agent-status.yml
```<br><br>Review the associated logs. |
| **Name: Process Check**<br><br>**Description:** Alarms when the specified process is not running:<br><br>`component = kafka`<br><br>**Likely cause:** Process crashed. | Restart the process on the affected node using these steps:<br><br>1. Log in to the Cloud Lifecycle Manager.<br><br>2. Check if Kafka is running on all nodes with this playbook:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-
status.yml \
--tags kafka
```<br><br>3. Use the Monasca start playbook against the affected node to restart it:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-
start.yml \
--tags kafka
```<br><br>4. Verify that Kafka is running on all nodes with this playbook:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-
status.yml \
--tags kafka
```<br><br>Review the associated logs. |
| **Name: Process Check**<br><br>**Description:** Alarms when the specified process is not running:<br><br>`process_name = monasca-notification` | Restart the process on the affected node using these steps:<br><br>1. Log in to the Cloud Lifecycle Manager.<br><br>2. Check if `monasca-api` is running on all nodes with this playbook:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
``` |

Alarm Information	Mitigation Tasks
Likely cause: Process crashed.	```ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags notification``` 3. Use the Monasca start playbook against the affected node to restart it: ```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --tags notification``` 4. Verify that it is running on all nodes with this playbook: ```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags notification``` Review the associated logs.
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = monasca-agent` **Likely cause:** Process crashed.	Restart the agent on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Check if `monasca-agent` is running on all nodes with this playbook: ```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-agent-status.yml``` 3. Use the Monasca start playbook against the affected node to restart it: ```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --limit <hostname>```

Alarm Information	Mitigation Tasks
	4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-agent-status.yml ``` Review the associated logs.
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = monasca-api` **Likely cause:** Process crashed.	> Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Check if `monasca-api` is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-api ``` 3. Use the Monasca start playbook against the affected node to restart it: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --tags monasca-api ``` 4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-api ``` Review the associated logs.
Name: Process Check	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager.

Alarm Information	Mitigation Tasks
Description: Alarms when the specified process is not running: `process_name = monasca-persister` **Likely cause:** Process crashed.	2. Check if `monasca-api` is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-persister ``` 3. Use the Monasca start playbook against the affected node to restart it: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --tags persister ``` 4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags monasca-persister ``` Review the associated logs.
Name: Process Check **Description:** Alarms when the specified process is not running: ``` process_name = backtype.storm.daemon.nimbus component = apache- storm ``` **Likely cause:** Process crashed.	Review the logs in the `/var/log/storm` directory on all storm hosts to find the root cause. 🎓 Note The logs containing threshold engine logging are on the 2nd and 3rd controller nodes. Restart `monasca-thresh`, if necessary, with these steps: 1. Log in to the Cloud Lifecycle Manager.

Alarm Information	Mitigation Tasks
| | 2. Check if `monasca-thresh` is running on all nodes with this playbook:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \
--tags thresh
```<br><br>3. Use the Monasca start playbook against the affected node to restart it:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \
--tags thresh
```<br><br>4. Verify that it is running on all nodes with this playbook:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \
--tags thresh
``` |
| **Name: Process Check**<br><br>**Description:** Alarms when the specified process is not running:<br><br>```
process_name =
 backtype.storm.daemon.supervisor
component = apache-
storm
```<br><br>**Likely cause:** Process crashed. | Review the logs in the `/var/log/storm` directory on all storm hosts to find the root cause.<br><br>### 🎓 Note<br><br>The logs containing threshold engine logging are on the 2nd and 3rd controller nodes.<br><br>Restart monasca-thresh with these steps:<br><br>1. Log in to the Cloud Lifecycle Manager.<br><br>2. Stop the monasca-thresh service:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-stop.yml \
``` |

Alarm Information	Mitigation Tasks
| | ```
--tags thresh
```<br><br>3. Start the monasca-thresh service back up:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-
start.yml \
--tags thresh
```<br><br>4. Verify that it is running on all nodes with this playbook:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-
status.yml \
--tags thresh
``` |
| **Name: Process Check**<br><br>**Description:** Alarms when the specified process is not running:<br><br>```
process_name =
 backtype.storm.daemon.worker
component = apache-
storm
```<br><br>**Likely cause:** Process crashed. | Review the logs in the `/var/log/storm` directory on all storm hosts to find the root cause.<br><br>☞ Note<br><br>   The logs containing threshold engine logging are on the 2nd and 3rd controller nodes.<br><br>Restart `monasca-thresh` with these steps:<br><br>1. Log in to the Cloud Lifecycle Manager.<br><br>2. Stop the `monasca-thresh` service:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-
stop.yml \
--tags thresh
```<br><br>3. Start the `monasca-thresh` service back up:<br><br>```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts monasca-
start.yml \
--tags thresh
``` |

Alarm Information	Mitigation Tasks
	4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags thresh ```
Name: Process Check **Description:** Alarms when the specified process is not running: ``` process_name = monasca-thresh component = apache-storm ``` **Likely cause:** Process crashed.	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Check if `monasca-thresh` is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags thresh ``` 3. Use the Monasca start playbook against the affected node to restart it: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml \ --tags thresh ``` 4. Verify that it is running on all nodes with this playbook: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts monasca-status.yml \ --tags thresh ``` Review the associated logs.
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log

Alarm Information	Mitigation Tasks
Likely cause: The service log directory, as indicated by the `path` dimension, is over the 2.5 GB quota.	rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.6   Console Alarms

These alarms show under the Console section of the HPE Helion OpenStack Operations Console.

Alarm Information	Mitigation Tasks
Name: HTTP Status **Description:** `service=ops-console` **Likely cause:** The Operations Console is unresponsive	Review logs in `/var/log/ops-console` and logs in `/var/log/apache2`. Restart ops-console by running the following commands on the Cloud Lifecycle Manager: ```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts ops-console-stop.yml ardana > ansible-playbook -i hosts/verb_hosts ops-console-start.yml```
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name=leia-leia_monitor` **Likely cause:** Process crashed or unresponsive.	Review logs in `/var/log/ops-console`. Restart ops-console by running the following commands on the Cloud Lifecycle Manager: ```ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts ops-console-stop.yml ardana > ansible-playbook -i hosts/verb_hosts ops-console-start.yml```

15.1.1.7   System Alarms

These alarms show under the System section and are set up per `hostname` and/or `mount_point`.

15.1.1.7.1 SERVICE: SYSTEM

Alarm Information	Mitigation Tasks
Name: CPU Usage **Description:** Alarms on high CPU usage. **Likely cause:** Heavy load or runaway processes.	Log onto the reporting host and diagnose the heavy CPU usage.
Name: Elasticsearch Low Watermark **Description:** `component = elasticsearch` Elasticsearch disk low watermark. Backup indices. If high watermark is reached, indices will be deleted. Adjust curator_low_watermark_percent, curator_high_watermark_percent, and elasticsearch_max_total_indices_size_in_bytes if needed. **Likely cause:** Running out of disk space for `/var/lib/ elasticsearch`.	Free up space by removing indices (backing them up first if desired). Alternatively, adjust `curator_low_watermark_percent`, `curator_high_watermark_percent`, and/or `elasticsearch_max_total_indices_size_in_bytes` if needed. For more information about how to back up your centralized logs, see *Section 12.2.5, "Configuring Centralized Logging"*.
Name: Elasticsearch High Watermark **Description:** `component = elasticsearch` Elasticsearch disk high watermark. Attempting to delete indices to	Verify that disk space was freed up by the curator. If needed, free up additional space by removing indices (backing them up first if desired). Alternatively, adjust curator_low_watermark_percent, curator_high_watermark_percent, and/or elasticsearch_max_total_indices_size_in_bytes if needed.

Alarm Information	Mitigation Tasks
free disk space. Adjust `curator_low_watermark_percent`, `curator_high_watermark_percent`, and `elasticsearch_max_total_indices_size_in_bytes` if needed. **Likely cause:** Running out of disk space for `/var/lib/elasticsearch`	For more information about how to back up your centralized logs, see *Section 12.2.5, "Configuring Centralized Logging"*.
Name: Log Partition Low Watermark **Description:** The `/var/log` disk space usage has crossed the low watermark. If the high watermark is reached, `logrotate` will be run to free up disk space. Adjust `var_log_low_watermark_percent` if needed. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

Alarm Information	Mitigation Tasks
Name: Log Partition High Watermark **Description:** The `/var/log` volume is running low on disk space. Logrotate will be run now to free up space. Adjust `var_log_high_watermark_percent` if needed. **Likely cause:** This could be due to a service set to `DEBUG` instead of `INFO` level. Another reason could be due to a repeating error message filling up the log files. Finally, it could be due to log rotate not configured properly so old log files are not being deleted properly.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.
Name: Crash Dump Count **Description:** Alarms if it receives any metrics with `crash.dump_count` > 0 **Likely cause:** When a crash dump is generated by kdump, the crash dump file is put into the `/var/crash` directory by default. Any crash	Analyze the crash dump file(s) located in `/var/crash` on the host that generated the alarm to try to determine if a service or hardware caused the crash. Move the file to a new location so that a developer can take a look at it. Make sure all of the processes are back up after the crash (run the `<service>-status.yml` playbooks). When the `/var/crash` directory is empty, the `Crash Dump Count` alarm should transition back to OK.

Alarm Information	Mitigation Tasks
dump files in this directory will cause the `crash.dump_count` metric to show a value greater than 0.	
Name: Disk Inode Usage **Description:** Nearly out of inodes for a partition, as indicated by the `mount_point` reported. **Likely cause:** Many files on the disk.	Investigate cleanup of data or migration to other partitions.
Name: Disk Usage **Description:** High disk usage, as indicated by the `mount_point` reported. **Likely cause:** Large files on the disk.	Investigate cleanup of data or migration to other partitions.
Name: Host Status **Description:** Alerts when a host is unreachable. `test_type = ping` **Likely cause:** Host or network is down.	If a single host, attempt to restart the system. If multiple hosts, investigate network issues.
Name: Memory Usage **Description:** High memory usage. **Likely cause:** Overloaded system or services with memory leaks.	Log onto the reporting host to investigate high memory users.

Alarm Information	Mitigation Tasks
Name: Network Errors **Description:** Alarms on a high network error rate. **Likely cause:** Bad network or cabling.	Take this host out of service until the network can be fixed.

Name: NTP Time Sync
 	
Description: Alarms when the NTP time offset is high.
	Log in to the reported host and check if the ntp service is running.
	
	If it is running, then use these steps:
	
	1. Stop the service:
	`service ntpd stop`
	
	2. Resynchronize the node's time:
	`/usr/sbin/ntpdate -b  <ntp-server>`
	
	3. Restart the ntp service:
	`service ntp start`
	
	4. Restart rsyslog:
	`service rsyslog restart`

15.1.1.8 Other Services Alarms

These alarms show under the Other Services section of the HPE Helion OpenStack Operations Console.

15.1.1.8.1 SERVICE: APACHE

Alarm Information	Mitigation Tasks
Name: Apache Status	

Alarm Information	Mitigation Tasks
Description: Alarms on failure to reach the Apache status endpoint.	
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = apache2`	If the Apache process goes down, connect to the affected node via SSH and restart it with this command: **`sudo systemctl restart apache2`**
Name: Apache Idle Worker Count **Description:** Alarms when there are no idle workers in the Apache server.	

15.1.1.8.2 SERVICE: BACKUP in Other Services section

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = freezer-scheduler` **Likely cause:** Process crashed.	Restart the process on the affected node. Review the associated logs.
Name: HTTP Status	see `Description`

Alarm Information	Mitigation Tasks
Description: Alarms when the specified HTTP endpoint is down or not reachable: `process_name = freezer-api` **Likely cause:** see `Description`	
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** The service log directory, as indicated by the `path` dimension, is over the 2.5 GB quota.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.8.3 SERVICE: HAPROXY in Other Services section

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = haproxy` **Likely cause:** HA Proxy is not running on this machine.	Restart the process on the affected node: 1. Log in to the Cloud Lifecycle Manager. 2. Run this playbook on the affected node: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ansible-playbook -i hosts/verb_hosts FND-CLU-start.yml \ --limit <hostname> ``` Review the associated logs.

15.1.1.8.4 SERVICE: ARDANA-UX-SERVICES in Other Services section

Alarm Information	Mitigation Tasks
Name: HTTP Status **Description:** Alarms when the specified HTTP endpoint is down or not reachable.	

15.1.1.8.5 SERVICE: KEY-MANAGER in Other Services section

Alarm Information	Mitigation Tasks
Name: Process Check **Description:** Alarms when the specified process is not running: `process_name = barbican-api` **Likely cause:** Process has crashed.	Restart the process on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the Barbican start playbook against the affected node: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts barbican-start.yml \ --limit <hostname> ```

Name: HTTP Status
 	
Description: Alarms when the specified HTTP endpoint is down or not reachable.
 	
`component = barbican-api`
`api_endpoint = public or internal`
	For the HTTP Status alarms for the public and internal endpoints, restart the process on the affected node using these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Stop the barbican service:
	
	``
	ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts
	 barbican-stop.yml \
	--limit <hostname>
	``

Alarm Information	Mitigation Tasks
Likely cause: The endpoint is not responsive, it may be down.	3. Restart the barbican service back up: ``` ardana > cd ~/scratch/ansible/next/ardana/ansible ardana > ansible-playbook -i hosts/verb_hosts barbican-start.yml \ --limit <hostname> ``` Examine the logs in `/var/log/barbican/` for possible error messages.
Name: HTTP Status **Description:** Alarms when the specified HTTP endpoint is down or not reachable. ``` component = barbican- api monitored_host_type = vip ``` **Likely cause:** The Barbican API on the admin virtual IP is down.	This alarm is verifying access to the Barbican API via the virtual IP address (HAProxy). If this check is failing but the other two HTTP Status alarms for the key-manager service are not then the issue is likely with HAProxy so you should view the alarms for that service. If the other two HTTP Status alarms are alerting as well then restart Barbican using the steps listed.
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** The service log directory, as indicated by the `path` dimension, is over the 2.5 GB quota.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.8.6　SERVICE: MYSQL in Other Services section

Alarm Information	Mitigation Tasks
Name: MySQL Slow Query Rate **Description:** Alarms when the slow query rate is high. **Likely cause:** The system load is too high.	This could be an indication of near capacity limits or an exposed bad query. First, check overall system load and then investigate MySQL details.
Name: Process Check **Description:** Alarms when the specified process is not running. **Likely cause:** MySQL crashed.	Restart MySQL on the affected node.

15.1.1.8.7　SERVICE: OCTAVIA in Other Services section

Alarm Information	Mitigation Tasks

Name: Process Check
 	
Description: Alarms when the specified process is not running. There are individual alarms for each of these processes:
 	
• octavia-worker
 	
• octavia-housekeeping
	Restart the process on the affected node using these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Use the Octavia start playbook against the affected node:
	
	<pre>ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts octavia-start.yml \
	--limit <hostname></pre>

Alarm Information	Mitigation Tasks
• octavia-api • octavia-health- manager **Likely cause:** The process has crashed.	
Name: HTTP Status **Description:** Alarms when the specified HTTP endpoint is down or not reachable. **Likely cause:** The `octavia-api` process could be down or you could be experiencing an issue with either haproxy or another network related issue.	If the `octavia-api` process is down, restart it on the affected node using these steps: 1. Log in to the Cloud Lifecycle Manager. 2. Use the Octavia start playbook against the affected node: ```\nardana > cd ~/scratch/ansible/next/ardana/ansible\nardana > ansible-playbook -i hosts/verb_hosts octavia-\nstart.yml \\\n--limit <hostname>\n``` If it is not the `octavia-process` that is the issue, then check if there is an issue with `haproxy` or possibly a network issue and troubleshoot accordingly.
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** The service log directory, as indicated by the `path` dimension, is over the 2.5 GB quota.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.8.8   SERVICE: ORCHESTRATION in Other Services section

Alarm Information	Mitigation Tasks

Name: Process Check
 	
Description: Alarms when the specified process is not running. There are individual alarms for each of these processes:
 	
• heat-api
• heat-api-cfn
• heat-api-cloudwatch
• heat-engine
 	
heat-api process check on each node
 	
Likely cause: Process crashed.
	Restart the process with these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Stop all the Heat processes:
	
	<pre>ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts heat-start.yml</pre>
	3. Start the Heat processes back up:
	
	<pre>ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts heat-start.yml</pre>
	Review the relevant log at the following locations on the affected node:
	
	<pre>/var/log/heat/heat-api.log
	/var/log/heat/heat-cfn.log
	/var/log/heat/heat-cloudwatch.log
	/var/log/heat/heat-engine.log</pre>

Name: HTTP Status
 	
Description: Alarms when the specified HTTP endpoint is down or not reachable.
 	
• heat-api
• heat-api-cfn
• heat-api-cloudwatch
	Restart the Heat service with these steps:
	
	1. Log in to the Cloud Lifecycle Manager.
	
	2. Stop all the Heat processes:
	
	<pre>ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts heat-start.yml</pre>
	3. Start the Heat processes back up:
	
	<pre>ardana > cd ~/scratch/ansible/next/ardana/ansible
	ardana > ansible-playbook -i hosts/verb_hosts heat-start.yml</pre>

Alarm Information	Mitigation Tasks
	Review the relevant log at the following locations on the affected node: ``` /var/log/heat/heat-api.log /var/log/heat/heat-cfn.log /var/log/heat/heat-cloudwatch.log ```
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** The service log directory, as indicated by the `path` dimension, is over the 2.5 GB quota.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.8.9   SERVICE: OVSVAPP-SERVICEVM in Other Services section

Alarm Information	Mitigation Tasks
Name: Process Check **Description:**Alarms when the specified process is not running: ``` process_name = ovs- vswitchd process_name = neutron- ovsvapp-agent process_name = ovsdb- server ``` **Likely cause:** Process has crashed.	Restart process on affected node. Review logs.

15.1.1.8.10 SERVICE: RABBITMQ in Other Services section

Alarm Information	Mitigation Tasks

Name: Process Check
 	
Description: Alarms when the specified process is not running:
 	
`process_name = rabbitmq`
`process_name = epmd`
 	
Likely cause: Process has crashed.
	Restart process on affected node. Review logs.

15.1.1.8.11 SERVICE: SPARK in Other Services section

Alarm Information	Mitigation Tasks

Name: Process Check
 	
Description: Alarms when the specified process is not running
 	
`process_name =`
` org.apache.spark.deploy.master.Master`
`process_name =`
` org.apache.spark.deploy.worker.Worker`
 	
Likely cause: Process has crashed.
	Restart process on affected node. Review logs.

15.1.1.8.12 SERVICE: WEB-UI in Other Services section

Alarm Information	Mitigation Tasks
Name: HTTP Status	Check that Apache is running; investigate Horizon logs.

Alarm Information	Mitigation Tasks
Description: Alarms when the specified HTTP endpoint is down or not reachable. **Likely cause:** Apache is not running or there is a misconfiguration.	
Name: Service Log Directory Size **Description:** Service log directory consuming more disk than its quota. **Likely cause:** The service log directory, as indicated by the `path` dimension, is over the 2.5 GB quota.	Find the service that is consuming too much disk space. Look at the logs. If `DEBUG` log entries exist, set the logging level to `INFO`. If the logs are repeatedly logging an error message, do what is needed to resolve the error. If old log files exist, configure log rotate to remove them. You could also choose to remove old log files by hand after backing them up if needed.

15.1.1.8.13 SERVICE: ZOOKEEPER in Other Services section

Alarm Information	Mitigation Tasks

Name: Process Check
 	
Description: Alarms when the specified process is not running:
`process_name =`
`org.apache.zookeeper.server`
 	
Likely cause: Process crashed.
	Restart the process on the affected node. Review the associated logs.

Name: ZooKeeper Latency	Check the individual system as well as activity across the entire service.

Alarm Information	Mitigation Tasks
Description: Alarms when the ZooKeeper latency is high. **Likely cause:** Heavy system load.	

15.1.1.9   ESX vCenter Plugin Alarms

These alarms relate to your ESX cluster, if you are utilizing one.

Alarm Information	Mitigation Tasks
Name: ESX cluster CPU Usage **Description:** Alarms when average of CPU usage for a particular cluster exceeds 90% continuously for 3 polling cycles. Alarm will have the following dimension: `esx_cluster_id=<domain>.<vcenter-id>` **Likely cause:** Virtual machines are consuming more than 90% of allocated vCPUs.	• Reduce the load on virtual machines with high consumption by restarting/stopping one or more services. • Add more vCPUs to the host(s) attached to the cluster.

Alarm Information	Mitigation Tasks
Name: ESX cluster Disk Usage **Description:** • Alarms when the total size of the all shared datastores attached to the cluster exceeds 90% of their total allocated capacity. • Or in the case of cluster having a single host, the size of non-shared datastore exceeds 90% of its allocated capacity. • Alarm will have the following dimension: ```esx_cluster_id=<domain>.<vcenter-id>``` **Likely cause:** • Virtual machines occupying the storage. • Large file or image being copied on the datastore(s).	• Check the virtual machines that are consuming more disk space. Delete unnecessary files. • Delete unnecessary files and images from database(s). • Add storage to the datastore(s).

Alarm Information	Mitigation Tasks
Name: ESX cluster Memory Usage **Description:** Alarms when average of RAM memory usage for a particular cluster, exceeds 90% continuously for 3 polling cycles. Alarm will have the following dimension: `esx_cluster_id=<domain>.<vcenter-id>` **Likely cause:** Virtual machines are consuming more than 90% of their total allocated memory.	• Reduce the load on virtual machines with high consumption by restarting or stopping one or more services. • Add more memory to the host(s) attached to the cluster.

15.1.2   Support Resources

To solve issues in your cloud, consult the Knowledge Base or contact Professional Services.

15.1.2.1   Using the Knowledge Base

The public knowledge base for HPE Helion OpenStack 8 can be reached at Hewlett Packard Enterprise Support Center - HPE Helion OpenStack (http://h20566.www2.hpe.com/portal/site/hpsc/public/psi/home/?sp4ts.oid=7261055#topIssuesAndSolution) ⬀.

You can look through the top issues or search for the specific issue you are experiencing. Knowledge documents listed with the content type `Knowledge Centered Support` are made by HPE TS Support cloud CoE engineers. If you do not find your issue listed then we suggest you contact support directly.

15.1.2.2 Contacting the HPE TS Support Cloud CoE

Call the HPE Support Number

Once you have registered for HPE TS Support you will be mailed a Service Agreement ID (SAID) and a phone number to call (www.hpe.com/contact). HPE customers with a Datacenter Care for Cloud contract will have to use their designated contact number. You can use that phone number to log support requests. Please have your SAID number available when calling. The support agent will log a case with the information you provide and the case will be sent to the HPE TS Support Cloud CoE Team. The response time will depend on the service level agreement (SLA) in place.

Electronic Case Logging

There is the option of visiting the HPE Support Center (http://www.hpe.com/support/hpesc) ↗ to log a support case via the website. You will be asked to setup and provide a PIN code for correct routing of your support case to the HPE TS Support Cloud CoE team. Use your HPE Passport and your Service Agreement ID (SAID) to electronically log your support case.

15.2 Control Plane Troubleshooting

Troubleshooting procedures for control plane services.

15.2.1 Understanding and Recovering RabbitMQ after Failure

RabbitMQ is the message queue service that runs on each of your controller nodes and brokers communication between multiple services in your HPE Helion OpenStack cloud environment. It is important for cloud operators to understand how different troubleshooting scenarios affect RabbitMQ so they can minimize downtime in their environments. We are going to discuss multiple scenarios and how it affects RabbitMQ. We will also explain how you can recover from them if there are issues.

15.2.1.1 How upgrades affect RabbitMQ

There are two types of upgrades within HPE Helion OpenStack -- major and minor. The effect that the upgrade process has on RabbitMQ depends on these types.

A major upgrade is defined by an erlang change or major version upgrade of RabbitMQ. A minor upgrade would be an upgrade where RabbitMQ stays within the same version, such as v3.4.3 to v.3.4.6.

During both types of upgrades there may be minor blips in the authentication process of client services as the accounts are recreated.

RabbitMQ during a major upgrade

There will be a RabbitMQ service outage while the upgrade is performed.

During the upgrade, high availability consistency is compromised -- all but the primary node will go down and will be reset, meaning their database copies are deleted. The primary node is not taken down until the last step and then it is upgrade. The database of users and permissions is maintained during this process. Then the other nodes are brought back into the cluster and resynchronized.

RabbitMQ during a minor upgrade

Minor upgrades are performed node by node. This "rolling" process means there should be no overall service outage because each node is taken out of its cluster in turn, its database is reset, and then it is added back to the cluster and resynchronized.

15.2.1.2 How RabbitMQ is affected by other operational processes

There are operational tasks, such as *Section 13.1.1.1, "Bringing Down Your Cloud: Services Down Method"*, where you use the `ardana-stop.yml` and `ardana-start.yml` playbooks to gracefully restart your cloud. If you use these playbooks, and there are no errors associated with them forcing you to troubleshoot further, then RabbitMQ is brought down gracefully and brought back up. There is nothing special to note regarding RabbitMQ in these normal operational processes.

However, there are other scenarios where an understanding of RabbitMQ is important when a graceful shutdown did not occur.

These examples that follow assume you are using one of the entry-scale models where RabbitMQ is hosted on your controller node cluster. If you are using a mid-scale model or have a dedicated cluster that RabbitMQ lives on you may need to alter the steps accordingly. To determine which nodes RabbitMQ is on you can use the `rabbit-status.yml` playbook from your Cloud Lifecycle Manager.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts rabbitmq-status.yml
```

Your entire control plane cluster goes down

If you have a scenario where all of your controller nodes went down, either manually or via another process such as a power outage, then an understanding of how RabbitMQ should be brought back up is important. Follow these steps to recover RabbitMQ on your controller node cluster in these cases:

1. The order in which the nodes went down is key here. Locate the last node to go down as this will be used as the primary node when bringing the RabbitMQ cluster back up. You can review the timestamps in the `/var/log/rabbitmq` log file to determine what the last node was.

   > **Note**
   >
   > The `primary` status of a node is transient, it only applies for the duration that this process is running. There is no long-term distinction between any of the nodes in your cluster. The primary node is simply the one that owns the RabbitMQ configuration database that will be synchronized across the cluster.

2. Run the `ardana-start.yml` playbook specifying the primary node (aka the last node down determined in the first step):

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts ardana-start.yml -e
     rabbit_primary_hostname=<hostname>
   ```

   > **Note**
   >
   > The `<hostname>` value will be the "shortname" for your node, as found in the `/etc/hosts` file.

If one of your controller nodes goes down

First step here is to determine whether the controller that went down is the primary RabbitMQ host or not. The primary host is going to be the first host member in the `FND-RMQ` group in the file below on your Cloud Lifecycle Manager:

```
ardana > ~/scratch/ansible/next/ardana/ansible/hosts/verb_hosts
```

In this example below, `ardana-cp1-c1-m1-mgmt` would be the primary:

```
[FND-RMQ-ccp-cluster1:children]
ardana-cp1-c1-m1-mgmt
ardana-cp1-c1-m2-mgmt
ardana-cp1-c1-m3-mgmt
```

If your primary RabbitMQ controller node has gone down and you need to bring it back up, you can follow these steps. In this playbook you are using the `rabbit_primary_hostname` parameter to specify the hostname for one of the other controller nodes in your environment hosting RabbitMQ, which will service as the primary node in the recovery. You will also use the `--limit` parameter to specify the controller node you are attempting to bring back up.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-start.yml
 -e rabbit_primary_hostname=<new_primary_hostname> --limit
 <hostname_of_node_you_are_bringing_up>
```

If the node you need to bring back is **not** the primary RabbitMQ node then you can just run the `ardana-start.yml` playbook with the `--limit` parameter and your node should recover:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit
 <hostname_of_node_you_are_bringing_up>
```

If you are replacing one or more of your controller nodes

The same general process noted above is used if you are removing or replacing one or more of your controller nodes.

If your node needs minor hardware repairs, but does not need to be replaced with a new node, you should use the `ardana-stop.yml` playbook with the `--limit` parameter to stop services on that node prior to removing it from the cluster.

1. Log into the Cloud Lifecycle Manager.

2. Run the `rabbitmq-stop.yml` playbook, specifying the hostname of the node you are removing, which will remove the node from the RabbitMQ cluster:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts rabbitmq-stop.yml --limit
    <hostname_of_node_you_are_removing>
   ```

3. Run the `ardana-stop.yml` playbook, again specifying the hostname of the node you are removing, which will stop the rest of the services and prepare it to be removed.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-stop.yml --limit
 <hostname_of_node_you_are_removing>
```

If your node cannot be repaired and needs to be replaced with another baremetal node, any references to the replaced node must be removed from the RabbitMQ cluster. This is because RabbitMQ associates a cookie with each node in the cluster which is derived, in part, by the specific hardware. So it is possible to replace a hard drive in a node. However changing a motherboard or replacing the node with another node entirely may cause RabbitMQ to stop working. When this happens, the running RabbitMQ cluster must be edited from a running RabbitMQ node. The following steps show how to do this.

In this example, controller 3 is the node being replaced with the following steps:

1.
```
ardana > cd ~/scratch/ansible/next/ardana/ansible
```

2. SSH to a running RabbitMQ cluster node.

```
ardana > ssh cloud-cp1-rmq-mysql-m1-mgmt
```

3. Force the cluster to forget the node you are removing (in this example, the controller 3 node).

```
ardana > sudo rabbitmqctl forget_cluster_node \
rabbit@cloud-cp1-rmq-mysql-m3-mgmt
```

4. Confirm that the node has been removed:

```
ardana > sudo rabbitmqctl cluster_status
```

5. On the replacement node, information and services related to RabbitMQ must be removed.

```
ardana > sudo systemctl stop rabbitmq-server
ardana > sudo systemctl stop epmd.socket>
```

6. Verify that the epmd service has stopped (kill it if it is still running).

```
ardana > ps -eaf | grep epmd.
```

7. Remove the Mnesia database directory.

```
ardana > sudo rm -rf /var/lib/rabbitmq/mnesia
```

8. Restart the RabbitMQ server.

```
ardana > sudo systemctl start rabbitmq-server
```

9. On the Cloud Lifecycle Manager, run the `ardana-start.yml` playbook.

If the node you are removing/replacing is your primary host then when you are adding it to your cluster then you will want to ensure that you specify a new primary host when doing so, as follows:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-start.yml -e
 rabbit_primary_hostname=<new_primary_hostname> --limit <hostname_of_node_you_are_adding>
```

If the node you are removing/replacing is not your primary host then you can add it as follows:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit
 <hostname_of_node_you_are_adding>
```

If one of your controller nodes has rebooted or temporarily lost power

After a single reboot, RabbitMQ will not automatically restart. This is by design to protect your RabbitMQ cluster. To restart RabbitMQ, you should follow the process below.

If the rebooted node was your primary RabbitMQ host, you will specify a different primary hostname using one of the other nodes in your cluster:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-start.yml -e
 rabbit_primary_hostname=<new_primary_hostname> --limit <hostname_of_node_that_rebooted>
```

If the rebooted node was not the primary RabbitMQ host then you can just start it back up with this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit
 <hostname_of_node_that_rebooted>
```

15.2.1.3 Recovering RabbitMQ

In this section we will show you how to check the status of RabbitMQ and how to do a variety of disaster recovery procedures.

Verifying the status of RabbitMQ

You can verify the status of RabbitMQ on each of your controller nodes by using the following steps:

1. Log in to the Cloud Lifecycle Manager.

2. Run the `rabbitmq-status.yml` playbook:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible
   ardana > ansible-playbook -i hosts/verb_hosts rabbitmq-status.yml
   ```

3. If all is well, you should see an output similar to the following:

   ```
   PLAY RECAP *********************************************************
   rabbitmq | status | Check RabbitMQ running hosts in cluster ------------ 2.12s
   rabbitmq | status | Check RabbitMQ service running -------------------- 1.69s
   rabbitmq | status | Report status of RabbitMQ ------------------------- 0.32s
   -----------------------------------------------------------------------
   Total: ---------------------------------------------------------- 4.36s
   ardana-cp1-c1-m1-mgmt  : ok=2    changed=0    unreachable=0    failed=0
   ardana-cp1-c1-m2-mgmt  : ok=2    changed=0    unreachable=0    failed=0
   ardana-cp1-c1-m3-mgmt  : ok=2    changed=0    unreachable=0    failed=0
   ```

If one or more of your controller nodes are having RabbitMQ issues then continue reading, looking for the scenario that best matches yours.

RabbitMQ recovery after a small network outage

In the case of a transient network outage, the version of RabbitMQ included with HPE Helion OpenStack is likely to recover automatically without any further action needed. However, if yours does not and the `rabbitmq-status.yml` playbook is reporting an issue then use the scenarios below to resolve your issues.

All of your controller nodes have gone down and using other methods have not brought RabbitMQ back up

If your RabbitMQ cluster is irrecoverable and you need rapid service recovery because other methods either cannot resolve the issue or you do not have time to investigate more nuanced approaches then we provide a disaster recovery playbook for you to use. This playbook will tear down and reset any RabbitMQ services. This does have an extreme effect on your services. The process will ensure that the RabbitMQ cluster is recreated.

1. Log in to your Cloud Lifecycle Manager.

2. Run the RabbitMQ disaster recovery playbook. This generally takes around two minutes.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts rabbitmq-disaster-recovery.yml
```

3. Run the reconfigure playbooks for both Cinder (Block Storage) and Heat (Orchestration), if those services are present in your cloud. These services are affected when the fan-out queues are not recovered correctly. The reconfigure generally takes around five minutes.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
ardana > ansible-playbook -i hosts/verb_hosts heat-reconfigure.yml
ardana > ansible-playbook -i hosts/verb_hosts kronos-server-configure.yml
```

4. If you need to do a safe recovery of all the services in your environment then you can use this playbook. This is a more lengthy process as all services are inspected.

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-reconfigure.yml
```

One of your controller nodes has gone down and using other methods have not brought RabbitMQ back up

This disaster recovery procedure has the same caveats as the preceding one, but the steps differ.

If your primary RabbitMQ controller node has gone down and you need to perform a disaster recovery, use this playbook from your Cloud Lifecycle Manager:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts rabbitmq-disaster-
recovery.yml -e rabbit_primary_hostname=<new_primary_hostname> --limit
 <hostname_of_node_that_needs_recovered>
```

If the controller node is not your primary, you can use this playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts rabbitmq-disaster-recovery.yml --limit
 <hostname_of_node_that_needs_recovered>
```

No reconfigure playbooks are needed because all of the fan-out exchanges are maintained by the running members of your RabbitMQ cluster.

15.3   Troubleshooting Compute Service

Troubleshooting scenarios with resolutions for the Nova service.

Nova offers scalable, on-demand, self-service access to compute resources. You can use this guide to help with known issues and troubleshooting of Nova services.

15.3.1   How can I reset the state of a compute instance?

If you have an instance that is stuck in a non-Active state, such as `Deleting` or `Rebooting` and you want to reset the state so you can interact with the instance again, there is a way to do this.

The Nova command-line tool (also known as the Nova CLI or python-novaclient) has a command, `nova reset-state`, that allows you to reset the state of a server.

Here is the content of the help information about the command which shows the syntax:

```
$ nova help reset-state
      usage: nova reset-state [--active] <server> [<server> ...]

      Reset the state of a server.

      Positional arguments:
      <server>  Name or ID of server(s).

      Optional arguments:
      --active  Request the server be reset to "active" state instead of "error"
      state (the default).
```

If you had an instance that was stuck in a `Rebooting` state you would use this command to reset it back to `Active`:

```
nova reset-state --active <instance_id>
```

15.3.2   Troubleshooting nova-consoleauth

The nova-consoleauth service runs by default on the first controller node, that is, the host with `consoleauth_host_index=0`. If nova-consoleauth fails on the first controller node, you can switch it to another controller node by running the ansible playbook nova-start.yml and passing it the index of the next controller node.

The command to switch nova-consoleauth to another controller node (controller 2 for instance) is:

```
ansible-playbook -i hosts/verb_hosts nova-start.yml --extra-vars
 "consoleauth_host_index=1"
```

After you run this command you may now see two instances of the `nova-consoleauth` service, which will show as being in `disabled` state, when you run the `nova service-list` command. You can then delete the service using these steps.

1. Obtain the service ID for the duplicated nova-consoleauth service:

```
nova service-list
```

Example:

```
$ nova service-list
+----+-----------------+--------------------------+----------+----------+-------
+--------------------------+-----------------+
| Id | Binary          | Host                     | Zone     | Status   | State |
 Updated_at                | Disabled Reason |
+----+-----------------+--------------------------+----------+----------+-------
+--------------------------+-----------------+
| 1  | nova-conductor  | ...a-cp1-c1-m1-mgmt      | internal | enabled  | up    |
 2016-08-25T12:11:48.000000 | -               |
| 10 | nova-conductor  | ...a-cp1-c1-m3-mgmt      | internal | enabled  | up    |
 2016-08-25T12:11:47.000000 | -               |
| 13 | nova-conductor  | ...a-cp1-c1-m2-mgmt      | internal | enabled  | up    |
 2016-08-25T12:11:48.000000 | -               |
| 16 | nova-scheduler  | ...a-cp1-c1-m1-mgmt      | internal | enabled  | up    |
 2016-08-25T12:11:39.000000 | -               |
| 19 | nova-scheduler  | ...a-cp1-c1-m2-mgmt      | internal | enabled  | up    |
 2016-08-25T12:11:41.000000 | -               |
| 22 | nova-scheduler  | ...a-cp1-c1-m3-mgmt      | internal | enabled  | up    |
 2016-08-25T12:11:44.000000 | -               |
| 25 | nova-consoleauth | ...a-cp1-c1-m1-mgmt     | internal | enabled  | up    |
 2016-08-25T12:11:45.000000 | -               |
| 49 | nova-compute    | ...a-cp1-comp0001-mgmt  | nova     | enabled  | up    |
 2016-08-25T12:11:48.000000 | -               |
| 52 | nova-compute    | ...a-cp1-comp0002-mgmt  | nova     | enabled  | up    |
 2016-08-25T12:11:41.000000 | -               |
| 55 | nova-compute    | ...a-cp1-comp0003-mgmt  | nova     | enabled  | up    |
 2016-08-25T12:11:43.000000 | -               |
| 70 | nova-consoleauth | ...a-cp1-c1-m3-mgmt     | internal | disabled | down  |
 2016-08-25T12:10:40.000000 | -               |
+----+-----------------+--------------------------+----------+----------+-------
+--------------------------+-----------------+
```

2. Delete the disabled duplicate service with this command:

```
nova service-delete <service_ID>
```

Given the example in the previous step, the command could be:

```
nova service-delete 70
```

15.3.3 Enabling the migrate or resize functions in Nova post-installation when using encryption

If you have used encryption for your data when running the configuration processor during your cloud deployment and are enabling the Nova resize and migrate functionality after the initial installation, there is an issue that arises if you have made additional configuration changes that required you to run the configuration processor before enabling these features.

You will only experience an issue if you have enabled encryption. If you haven't enabled encryption, then there is no need to follow the procedure below. If you are using encryption and you have made a configuration change and run the configuration processor after your initial install or upgrade, and you have run the `ready-deployment.yml` playbook, and you want to enable migrate or resize in Nova, then the following steps will allow you to proceed. Note that the ansible vault key referred to below is the encryption key that you have provided to the configuration processor.

1. Log in to the Cloud Lifecycle Manager.

2. Checkout the ansible branch of your local git:

    ```
    cd ~/openstack
    git checkout ansible
    ```

3. Do a git log, and pick the previous commit:

    ```
    git log
    ```

    In this example below, the commit is `ac54d619b4fd84b497c7797ec61d989b64b9edb3`:

    ```
    $ git log

            commit 69f95002f9bad0b17f48687e4d97b2a791476c6a
            Merge: 439a85e ac54d61
            Author: git user <user@company.com>
            Date:   Fri May 6 09:08:55 2016 +0000

            Merging promotion of saved output
    ```

```
                   commit 439a85e209aeeca3ab54d1a9184efb01604dbbbb
                   Author: git user <user@company.com>
                   Date:   Fri May 6 09:08:24 2016 +0000

                   Saved output from CP run on 1d3976dac4fd7e2e78afad8d23f7b64f9d138778

                   commit ac54d619b4fd84b497c7797ec61d989b64b9edb3
                   Merge: a794083 66ffe07
                   Author: git user <user@company.com>
                   Date:   Fri May 6 08:32:04 2016 +0000

                   Merging promotion of saved output
```

4. Checkout the commit:

```
git checkout <commit_ID>
```

Using the same example above, here is the command:

```
$ git checkout ac54d619b4fd84b497c7797ec61d989b64b9edb3
                   Note: checking out 'ac54d619b4fd84b497c7797ec61d989b64b9edb3'.

                   You are in 'detached HEAD' state. You can look around, make
 experimental
                   changes and commit them, and you can discard any commits you make in
 this
                   state without impacting any branches by performing another checkout.

                   If you want to create a new branch to retain commits you create, you
 may
                   do so (now or later) by using -b with the checkout command again.
 Example:

                   git checkout -b new_branch_name

                   HEAD is now at ac54d61... Merging promotion of saved output
```

5. Change to the ansible output directory:

```
cd ~/openstack/my_cloud/stage/ansible/group_vars/
```

6. View the group_vars file from the ansible vault - it will be of the form below, with your compute cluster name being the indicator:

```
<cloud name>-<control plane name>-<compute cluster name>
```

View this group_vars file from the ansible vault with this command which will prompt you for your vault password:

```
ansible-vault view <group_vars_file>
```

7. Search the contents of this file for the `nova_ssh_key` section which will contain both the private and public SSH keys which you should then save into a temporary file so you can use it in a later step.

Here is an example snippet, with the bold part being what you need to save:

```
NOV_KVM:
            vars:
                    nova_ssh_key:
            private: '-----BEGIN RSA PRIVATE KEY-----
            MIIEpAIBAAKCAQEAv/hhekzykD2K8HnVNBKZcJWYrVlUyb6gR8cvE6hbh2ISzooA
            jQc3xgglIwpt5TuwpTY3LL0C4PEHObxy9WwqXTHBZp8jg/02RzD02bEcZ1WT49x7
            Rj8f5+S1zutHlDv7PwEIMZPAHA8lihfGFG5o+QHUmsUHgjShkWPdHXw1+6mCO9V/
            eJVZb3nDbiunMOBvyyk364w+fSzes4UDkmCq8joDa5KkpTgQK6xfw5auEosyrh8D
            zocN/JSdr6xStlT6yY8naWziXr7p/QhG44RPD9SSD7dhkyJh+bdCfoFVGdjmF8yA
            h5DlcLu9QhbJ/scb7yMP84W4L5GwvuWCCFJTHQIDAQABAoIBAQCCH5O7ecMFoKG4
            JW0uMdlOJijqf93oLk2oucwgUANSvlivJX4AGj9k/YpmuSAKvS4cnqZBrhDwdpCG
            Q0XNM7d3mk1VCVPimNWc5gNiOBpftPNdBcuNryYqYq4WBwdq5EmGyGVMbbFPk7jH
            ZRwAJ2MCPoplKl7PlGtcCMwNu29AGNaxCQEZFmztXcEFdMrfpTh3kuBI536pBlEi
            Srh23mRILn0nvLXMAHwo94S6bI3JOQSK1DBCwtA52r5YgX0nkZbi2MvHISY1TXBw
            SiWgzqW8dakzVu9UNif9nTDyaJDpU0kr0/LWtBQNdcpXnDSkHGjjnIm2pJVBC+QJ
            SM9o8h1lAoGBANjGHtG762+dNPEUUkSNWVwd7tvzW9CZY35iMR0Rlux4PO+OXwNq
            agldHeUpgG1MPl1ya+rkf0GD62Uf4LHTDgaEkUfiXkYtcJwHbjOnj3EjZLXaYMX2
            LYBE0bMKUkQCBdYtCvZmo6+dfC2DBEWPEhvWi7zf7o0CJ9260aS4UHJzAoGBAOK1
            P//K7HBWXvKpY1yV2KSCEBEoiM9NA9+RYcLkNtIy/4rIk9ShLdCJQVWWgDfDTfso
            sJKc5S0OtOsRcomvv3OIQD1PvZVfZJLKpgKkt20/w7RwfJkYC/jSjQpzgDpZdKRU
            vRY8P5iryptleyImeqV+Vhf+1kcH8t5VQMUU2XAvAoGATpfeOqqIXMpBlJqKjUI2
            QNi1bleYVVQXp43QQrrK3mdlqHEU77cYRNbW7OwUHQyEm/rNN7eqj8VVhi99lttv
            fVt5FPf0uDrnVhq3kNDSh/GOJQTNC1kK/DN3WBOI6hFVrmZcUCO8ewJ9MD8NQG7z
            4NXzigIiiktayuBd+/u7ZxMCgYEAm6X7KaBlkn8KMypuyIsssU2GwHEG9OSYay9C
            Ym8S4GAZKGyrakm6zbjefWeV4jMZ3/1AtXg4tCWrutRAwh1CoYyDJlUQAXT79Phi
            39+8+6nSsJimQunKlmvgX7OK7wSp24U+SPzWYPhZYzVaQ8kNXYAOlezlquDfMxxv
            GqBE5QsCgYA8K2p/z2kGXCNjdMrEM02reeE2J1Ft8DS/iiXjg35PX7WVIZ31KCBk
            wgYTWq0Fwo2W/EoJVl2o74qQTHK0Bs+FTnR2nkVF3htEOAW2YXQTTN2rEsHmlQqE
            A9iGTNwm9hvzbvrWeXtx8Zk/6aYfsXCoxq193KglS40shOCaXzWX0w==
            -----END RSA PRIVATE KEY-----'
            public: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC/
 +GF6TPKQPYrwedU0Epl

  wlZitWVTJvqBHxy8TqFuHYhLOigCNBzfGCCUjCm3lO7ClNjcsvQLg8Qc5vHL1bCpdMc
            FmnyOD/TZHMPTZsRxnVZPj3HtGPx/n5LXO60eUO/s/AQgxk8AcDyWKF8YUbmj5Ad
```

```
                SaxQeCNKGRY90dfDX7qYI71X94lVlvecNuK6cw4G/
LKTfrjD59LN6zhQOSYKryOgNrkq
                SlOBArrF/
Dlq4SizKuHwPOhw38lJ2vrFK2VPrJjydpbOJevun9CEbjhE8P1JIPt2GTImH5t0
                J+gVUZ2OYXzICHkOVwu71CFsn+xxvvIw/zhbgvkbC+5YIIUlMd
                Generated Key for Nova User
             NTP_CLI:
```

8. Switch back to the `site` branch by checking it out:

```
cd ~/openstack
git checkout site
```

9. Navigate to your group_vars directory in this branch:

```
cd ~/scratch/ansible/next/ardana/ansible/group_vars
```

10. Edit your compute group_vars file, which will prompt you for your vault password:

```
ansible-vault edit <group_vars_file>
            Vault password:
            Decryption successful
```

11. Search the contents of this file for the `nova_ssh_key` section and replace the private and public keys with the contents that you had saved in a temporary file in step #7 earlier.

12. Remove the temporary file that you created earlier. You are now ready to run the deployment. For information about enabling Nova resizing and migration, see *Section 5.4, "Enabling the Nova Resize and Migrate Features"*.

15.3.4   Compute (ESX)

Unable to Create Instance Snapshot when Instance is Active

There is a known issue with VMWare vCenter where if you have a compute instance in `Active` state you will receive the error below when attempting to take a snapshot of it:

```
An error occurred while saving the snapshot: Failed to quiesce the virtual machine
```

The workaround for this issue is to stop the instance. Here are steps to achieve this using the command line tool:

1. Stop the instance using the NovaClient:

   ```
   nova stop <instance UUID>
   ```

2. Take the snapshot of the instance.

3. Start the instance back up:

   ```
   nova start <instance UUID>
   ```

15.4 Network Service Troubleshooting

Troubleshooting scenarios with resolutions for the Networking service.

15.4.1 Troubleshooting Network failures

CVR HA - Split-brain result of failover of L3 agent when master comes back up This situation is specific to when L3 HA is configured and a network failure occurs to the node hosting the currently active l3 agent. L3 HA is intended to provide HA in situations where the l3-agent crashes or the node hosting an l3-agent crashes/restarts. In the case of a physical networking issue which isolates the active l3 agent, the stand-by l3-agent takes over but when the physical networking issue is resolved, traffic to the VMs is disrupted due to a "split-brain" situation in which traffic is split over the two L3 agents. The solution is to restart the L3-agent that was originally the master.

OVSvApp loses connectivity with vCenter If the OVSvApp loses connectivity with the vCenter cluster, you will receive the following errors:

1. The OVSvApp VM will go into ERROR state

2. The OVSvApp VM will not get IP address

When you see these symptoms:

1. Restart the OVSvApp agent on the OVSvApp VM.

2. Execute the following command to restart the Network (Neutron) service:

```
sudo service neutron-ovsvapp-agent restart
```

Fail over a plain CVR router because the node became unavailable:

1. Get a list of l3 agent UUIDs which can be used in the commands that follow

```
neutron agent-list | grep l3
```

2. Determine the current host

```
neutron l3-agent-list-hosting-router <router uuid>
```

3. Remove the router from the current host

```
neutron l3-agent-router-remove <current l3 agent uuid> <router uuid>
```

4. Add the router to a new host

```
neutron l3-agent-router-add <new l3 agent uuid> <router uuid>
```

Trouble setting maximum transmission units (MTU) *Section 9.3.11, "Configuring Maximum Transmission Units in Neutron"*

Floating IP on allowed_address_pair port with DVR-routed networks `allowed_address_pair`

You may notice this issue: If you have an `allowed_address_pair` associated with multiple virtual machine (VM) ports, and if all the VM ports are ACTIVE, then the `allowed_address_pair` port binding will have the last ACTIVE VM's binding host as its bound host.

In addition, you may notice that if the floating IP is assigned to the `allowed_address_pair` that is bound to multiple VMs that are ACTIVE, then the floating IP will not work with DVR routers. This is different from the centralized router behavior where it can handle unbound `allowed_address_pair` ports that are associated with floating IPs.

Currently we support `allowed_address_pair` ports with DVR only if they have floating IPs enabled, and have just one ACTIVE port.

Using the CLI, you can follow these steps:

1. Create a network to add the host to:

```
$ neutron net-create vrrp-net
```

2. Attach a subnet to that network with a specified allocation-pool range:

```
$ neutron subnet-create  --name vrrp-subnet --allocation-pool
  start=10.0.0.2,end=10.0.0.200 vrrp-net 10.0.0.0/24
```

3. Create a router, uplink the vrrp-subnet to it, and attach the router to an upstream network called public:

```
$ neutron router-create router1
$ neutron router-interface-add router1 vrrp-subnet
$ neutron router-gateway-set router1 public
```

Create a security group called vrrp-sec-group and add ingress rules to allow ICMP and TCP port 80 and 22:

```
$ neutron security-group-create vrrp-sec-group
$ neutron security-group-rule-create  --protocol icmp vrrp-sec-group
$ neutron security-group-rule-create  --protocol tcp  --port-range-min80 --port-
range-max80 vrrp-sec-group
$ neutron security-group-rule-create  --protocol tcp  --port-range-min22 --port-
range-max22 vrrp-sec-group
```

4. Next, boot two instances:

```
$ nova boot --num-instances 2 --image ubuntu-12.04 --flavor 1 --nic net-
id=24e92ee1-8ae4-4c23-90af-accb3919f4d1 vrrp-node --security_groups vrrp-sec-group
```

5. When you create two instances, make sure that both the instances are not in ACTIVE state before you associate the `allowed_address_pair`. The instances:

```
$ nova list
+------------------------------------
+---------------------------------------------------+--------+------------
+-------------+---------------------------------------------------+
| ID                                      | Name
     | Status | Task State | Power State | Networks
              |
```

```
+-------------------------------------
+-------------------------------------------------+--------+-----------
+------------+--------------------------------------------------------+
| 15b70af7-2628-4906-a877-39753082f84f | vrrp-node-15b70af7-2628-4906-
a877-39753082f84f | ACTIVE  | -          | Running    | vrrp-net=10.0.0.3
                            |
| e683e9d1-7eea-48dd-9d3a-a54cf9d9b7d6 | vrrp-node-e683e9d1-7eea-48dd-9d3a-
a54cf9d9b7d6 | DOWN    | -          | Running    | vrrp-net=10.0.0.4
                      |
+------------------------------------
+-------------------------------------------------+--------+-----------
+------------+-------------------------------------------------------+
```

6. Create a port in the VRRP IP range that was left out of the ip-allocation range:

```
$ neutron port-create --fixed-ip ip_address=10.0.0.201 --security-group vrrp-sec-
group vrrp-net
Created a new port:
+----------------------
+--------------------------------------------------------------------------------
+
| Field                  | Value
                         |
+----------------------
+--------------------------------------------------------------------------------
+
| admin_state_up         | True
                         |
| allowed_address_pairs  |
                         |
| device_id              |
                         |
| device_owner           |
                         |
| fixed_ips              | {"subnet_id": "94a0c371-d37c-4796-821e-57c2a8ec65ae",
 "ip_address": "10.0.0.201"} |
| id                     | 6239f501-e902-4b02-8d5c-69062896a2dd
                         |
| mac_address            | fa:16:3e:20:67:9f
                         |
| name                   |
                         |
| network_id             | 24e92ee1-8ae4-4c23-90af-accb3919f4d1
                         |
| port_security_enabled  | True
                         |
```

```
| security_groups        | 36c8131f-d504-4bcc-b708-f330c9f6b67a
                         |
| status                 | DOWN
                         |
| tenant_id              | d4e4332d5f8c4a8eab9fcb1345406cb0
                         |
+----------------------
+------------------------------------------------------------------------------
+
```

7. Another thing to cross check after you associate the allowed_address_pair port to the VM port, is whether the `allowed_address_pair` port has inherited the VM's host binding:

```
$ neutron --os-username admin --os-password ZIy9xitH55 --os-tenant-name admin port-
show f5a252b2-701f-40e9-a314-59ef9b5ed7de
+----------------------
+--------------------------------------------------------------------------------------
+
| Field                  | Value
                         |
+----------------------
+--------------------------------------------------------------------------------------
+
| admin_state_up         | True
                         |
| allowed_address_pairs  |
                         |
| {color:red}binding:host_id{color} | ...-cp1-comp0001-mgmt
                         |
| binding:profile        | {}
                         |
| binding:vif_details    | {"port_filter": true, "ovs_hybrid_plug": true}
                         |
| binding:vif_type       | ovs
                         |
| binding:vnic_type      | normal
                         |
| device_id              |
                         |
| device_owner           | compute:None
                         |
| dns_assignment         | {"hostname": "host-10-0-0-201", "ip_address":
 "10.0.0.201", "fqdn": "host-10-0-0-201.openstacklocal."} |
| dns_name               |
                         |
| extra_dhcp_opts        |
                         |
```

```
| fixed_ips            | {"subnet_id": "94a0c371-d37c-4796-821e-57c2a8ec65ae",
  "ip_address": "10.0.0.201"}                    |
| id                   | 6239f501-e902-4b02-8d5c-69062896a2dd
                                                 |
| mac_address          | fa:16:3e:20:67:9f
                                                 |
| name                 |
                                                 |
| network_id           | 24e92ee1-8ae4-4c23-90af-accb3919f4d1
                                                 |
| port_security_enabled | True
                                                 |
| security_groups      | 36c8131f-d504-4bcc-b708-f330c9f6b67a
                                                 |
| status               | DOWN
                                                 |
| tenant_id            | d4e4332d5f8c4a8eab9fcb1345406cb0
                                                 |
+----------------------
+-------------------------------------------------------------------------------------------
+
```

8. Note that you were allocated a port with the IP address 10.0.0.201 as requested. Next,
   associate a floating IP to this port to be able to access it publicly:

```
$ neutron floatingip-create --port-id=6239f501-e902-4b02-8d5c-69062896a2dd public
Created a new floatingip:
+--------------------+--------------------------------------+
| Field              | Value                                |
+--------------------+--------------------------------------+
| fixed_ip_address   | 10.0.0.201                           |
| floating_ip_address | 10.36.12.139                        |
| floating_network_id | 3696c581-9474-4c57-aaa0-b6c70f2529b0 |
| id                 | a26931de-bc94-4fd8-a8b9-c5d4031667e9 |
| port_id            | 6239f501-e902-4b02-8d5c-69062896a2dd |
| router_id          | 178fde65-e9e7-4d84-a218-b1cc7c7b09c7 |
| tenant_id          | d4e4332d5f8c4a8eab9fcb1345406cb0     |
+--------------------+--------------------------------------+
```

9. Now update the ports attached to your VRRP instances to include this IP address as an
   allowed-address-pair so they will be able to send traffic out using this address. First find
   the ports attached to these instances:

```
$ neutron port-list -- --network_id=24e92ee1-8ae4-4c23-90af-accb3919f4d1
```

```
+-------------------------------------+------+-------------------
+--------------------------------------------------------------------
+
| id                                  | name | mac_address       | fixed_ips
                                                                  |
+-------------------------------------+------+-------------------
+--------------------------------------------------------------------
+
| 12bf9ea4-4845-4e2c-b511-3b8b1ad7291d |      | fa:16:3e:7a:7b:18 | {"subnet_id":
 "94a0c371-d37c-4796-821e-57c2a8ec65ae", "ip_address": "10.0.0.4"}    |
| 14f57a85-35af-4edb-8bec-6f81beb9db88 |      | fa:16:3e:2f:7e:ee | {"subnet_id":
 "94a0c371-d37c-4796-821e-57c2a8ec65ae", "ip_address": "10.0.0.2"}    |
| 6239f501-e902-4b02-8d5c-69062896a2dd |      | fa:16:3e:20:67:9f | {"subnet_id":
 "94a0c371-d37c-4796-821e-57c2a8ec65ae", "ip_address": "10.0.0.201"} |
| 87094048-3832-472e-a100-7f9b45829da5 |      | fa:16:3e:b3:38:30 | {"subnet_id":
 "94a0c371-d37c-4796-821e-57c2a8ec65ae", "ip_address": "10.0.0.1"}    |
| c080dbeb-491e-46e2-ab7e-192e7627d050 |      | fa:16:3e:88:2e:e2 | {"subnet_id":
 "94a0c371-d37c-4796-821e-57c2a8ec65ae", "ip_address": "10.0.0.3"}    |
+-------------------------------------+------+-------------------
+--------------------------------------------------------------------
+
```

10. Add this address to the ports c080dbeb-491e-46e2-ab7e-192e7627d050 and 12bf9ea4-4845-4e2c-b511-3b8b1ad7291d which are 10.0.0.3 and 10.0.0.4 (your vrrp-node instances):

```
$ neutron port-update  c080dbeb-491e-46e2-ab7e-192e7627d050 --allowed_address_pairs
 list=truetype=dict ip_address=10.0.0.201
$ neutron port-update  12bf9ea4-4845-4e2c-b511-3b8b1ad7291d --allowed_address_pairs
 list=truetype=dict ip_address=10.0.0.201
```

11. The allowed-address-pair 10.0.0.201 now shows up on the port:

```
$ neutron port-show12bf9ea4-4845-4e2c-b511-3b8b1ad7291d
+----------------------
+-----------------------------------------------------------------------------+
| Field                 | Value
                        |
+----------------------
+-----------------------------------------------------------------------------+
| admin_state_up        | True
                        |
| allowed_address_pairs | {"ip_address": "10.0.0.201", "mac_address":
 "fa:16:3e:7a:7b:18"}                   |
| device_id             | e683e9d1-7eea-48dd-9d3a-a54cf9d9b7d6
                        |
```

```
| device_owner          | compute:None
                        |
| fixed_ips             | {"subnet_id": "94a0c371-d37c-4796-821e-57c2a8ec65ae",
 "ip_address": "10.0.0.4"} |
| id                    | 12bf9ea4-4845-4e2c-b511-3b8b1ad7291d
                        |
| mac_address           | fa:16:3e:7a:7b:18
                        |
| name                  |
                        |
| network_id            | 24e92ee1-8ae4-4c23-90af-accb3919f4d1
                        |
| port_security_enabled | True
                        |
| security_groups       | 36c8131f-d504-4bcc-b708-f330c9f6b67a
                        |
| status                | ACTIVE
                        |
| tenant_id             | d4e4332d5f8c4a8eab9fcb1345406cb0
                        |
```

OpenStack traffic that must traverse VXLAN tunnel dropped when using HPE 5930 switch

Cause: UDP destination port 4789 is conflicting with OpenStack VXLAN traffic.

There is a configuration setting you can use in the switch to configure the port number the HPN kit will use for its own VXLAN tunnels. Setting this to a port number other than the one Neutron will use by default (4789) will keep the HPN kit from absconding with Neutron's VXLAN traffic. Specifically:

Parameters:

port-number: Specifies a UDP port number in the range of 1 to 65535. As a best practice, specify a port number in the range of 1024 to 65535 to avoid conflict with well-known ports.

Usage guidelines:

You must configure the same destination UDP port number on all VTEPs in a VXLAN.

Examples

```
# Set the destination UDP port number to 6666 for VXLAN packets.
<Sysname> system-view
[Sysname] vxlan udp-port 6666
```

Use vxlan udp-port to configure the destination UDP port number of VXLAN packets. Mandatory for all VXLAN packets to specify a UDP port Default The destination UDP port number is 4789 for VXLAN packets.

OVS can be configured to use a different port number itself:

```
# (IntOpt) The port number to utilize if tunnel_types includes 'vxlan'. By
# default, this will make use of the Open vSwitch default value of '4789' if
# not specified.
#
# vxlan_udp_port =
# Example: vxlan_udp_port = 8472
#
```

15.4.1.1   Issue: PCI-PT virtual machine gets stuck at boot

If you are using a machine that uses Intel NICs, if the PCI-PT virtual machine gets stuck at boot, the boot agent should be disabled.

When Intel cards are used for PCI-PT, sometimes the tenant virtual machine gets stuck at boot. If this happens, you should download Intel bootutils and use it to disable the bootagent.

Use the following steps:

1. Download `preebot.tar.gz` from the Intel website (https://downloadcenter.intel.com/download/19186/Intel-Ethernet-Connections-Boot-Utility-Preboot-Images-and-EFI-Drivers) ↗.

2. Untar the `preboot.tar.gz` file on the compute host where the PCI-PT virtual machine is to be hosted.

3. Go to path `~/APPS/BootUtil/Linux_x64` and then run following command:

   ```
   ./bootutil64e -BOOTENABLE disable -all
   ```

4. Now boot the PCI-PT virtual machine and it should boot without getting stuck.

15.5   Troubleshooting the Image (Glance) Service

Troubleshooting scenarios with resolutions for the Glance service. We have gathered some of the common issues and troubleshooting steps that will help when resolving issues that occur with the Glance service.

15.5.1 Images Created in Horizon UI Get Stuck in a Queued State

When creating a new image in the Horizon UI you will see the option for `Image Location` which allows you to enter a HTTP source to use when creating a new image for your cloud. However, this option is disabled by default for security reasons. This results in any new images created via this method getting stuck in a `Queued` state.

We cannot guarantee the security of any third party sites you use as image sources and the traffic goes over HTTP (non-SSL) traffic.

Resolution: You will need your cloud administrator to enable the HTTP store option in Glance for your cloud.

Here are the steps to enable this option:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the file below:

   ```
   ~/openstack/ardana/ansible/roles/GLA-API/templates/glance-api.conf.j2
   ```

3. Locate the Glance store options and add the `http` value in the `stores` field. It will look like this:

   ```
   [glance_store]
   stores = {{ glance_stores }}
   ```

   Change this to:

   ```
   [glance_store]
   stores = {{ glance_stores }},http
   ```

4. Commit your configuration to the *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*, as follows:

   ```
   cd ~/openstack/ardana/ansible
   git add -A
   git commit -m "adding HTTP option to Glance store list"
   ```

5. Run the configuration processor with this command:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

6. Use the playbook below to create a deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Glance service reconfigure playbook which will update these settings:

```
cd ~/scratch/ansible/next/ardana/ansible/
ansible-playbook -i hosts/verb_hosts glance-reconfigure.yml
```

15.6 Storage Troubleshooting

Troubleshooting scenarios with resolutions for Swift services.

15.6.1 Block Storage Troubleshooting

The block storage service utilizes OpenStack Cinder and can integrate with multiple back-ends including 3Par. Failures may exist at the Cinder API level, an operation may fail, or you may see an alarm trigger in the monitoring service. These may be caused by configuration problems, network issues, or issues with your servers or storage back-ends. The purpose of this page and section is to describe how the service works, where to find additional information, some of the common problems that come up, and how to address them.

15.6.1.1 Where to find information

When debugging block storage issues it is helpful to understand the deployment topology and know where to locate the logs with additional information.

The Cinder service consists of:

- An API service, typically deployed and active on the controller nodes.

- A scheduler service, also typically deployed and active on the controller nodes.

- A volume service, which is deployed on all of the controller nodes but only active on one of them.

- A backup service, which is deployed on the same controller node as the volume service.

OpenStack Cinder High Level Architecture

Architecture and Volume Flow

You can refer to your configuration files (usually located in `~/openstack/my_cloud/definition/` on the Cloud Lifecycle Manager) for specifics about where your services are located. They will usually be located on the controller nodes.

Cinder uses a MariaDB database and communicates between components by consuming messages from a RabbitMQ message service.

The Cinder API service is layered underneath a HAProxy service and accessed using a virtual IP address maintained using keepalived.

If any of the Cinder components is not running on its intended host then an alarm will be raised. Details on how to resolve these alarms can be found on our *Section 15.1.1, "Alarm Resolution Procedures"* page. You should check the logs for the service on the appropriate nodes. All Cinder logs are stored in `/var/log/cinder/` and all log entries above `INFO` level are also sent to the centralized logging service. For details on how to change the logging level of the Cinder service, see *Section 12.2.6, "Configuring Settings for Other Services"*.

In order to get the full context of an error you may need to examine the full log files on individual nodes. Note that if a component runs on more than one node you will need to review the logs on each of the nodes that component runs on. Also remember that as logs rotate that the time interval you are interested in may be in an older log file.

Log locations:

`/var/log/cinder/cinder-api.log` - Check this log if you have endpoint or connectivity issues

`/var/log/cinder/cinder-scheduler.log` - Check this log if the system cannot assign your volume to a back-end

`/var/log/cinder/cinder-backup.log` - Check this log if you have backup or restore issues

`/var/log/cinder-cinder-volume.log` - Check here for failures during volume creation

`/var/log/nova/nova-compute.log` - Check here for failures with attaching volumes to compute instances

You can also check the logs for the database and/or the RabbitMQ service if your cloud exhibits database or messaging errors.

If the API servers are up and running but the API is not reachable then checking the HAProxy logs on the active keepalived node would be the place to look.

If you have errors attaching volumes to compute instances using the Nova API then the logs would be on the compute node associated with the instance. You can use the following command to determine which node is hosting the instance:

```
nova show <instance_uuid>
```

Then you can check the logs located at `/var/log/nova/nova-compute.log` on that compute node.

15.6.1.2   Understanding the Cinder volume states

Once the topology is understood, if the issue with the Cinder service relates to a specific volume then you should have a good understanding of what the various states a volume can be in are. The states are:

- attaching

- available

- backing-up

- creating

- deleting

- downloading

- error

- error attaching

- error deleting

- error detaching

- error extending

- error restoring

- in-use

- extending

- restoring

- restoring backup

- retyping

- uploading

The common states are `in-use` which indicates a volume is currently attached to a compute instance and `available` means the volume is created on a back-end and is free to be attached to an instance. All `-ing` states are transient and represent a transition. If a volume stays in one of those states for too long indicating it is stuck, or if it fails and goes into an error state, you should check for failures in the logs.

15.6.1.3   Initial troubleshooting steps

These should be the initial troubleshooting steps you go through.

1. If you have noticed an issue with the service, you should check your monitoring system for any alarms that may have triggered. See *Section 15.1.1, "Alarm Resolution Procedures"* for resolution steps for those alarms.

2. Check if the Cinder API service is active by listing the available volumes from the Cloud Lifecycle Manager:

```
source ~/service.osrc
openstack volume list
```

3. Run a basic diagnostic from the Cloud Lifecycle Manager:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts _cinder_post_check.yml
```

This ansible playbook will list all volumes, create a 1 GB volume and then delete it using the v1 and v2 APIs, which will exercise basic Cinder capability.

15.6.1.4 Common failures

Alerts from the Cinder service

Check for alerts associated with the block storage service, noting that these could include alerts related to the server nodes being down, alerts related to the messaging and database services, or the HAProxy and keepalived services, as well as alerts directly attributed to the block storage service.

The Operations Console provides a web UI method for checking alarms. See *Book "User Guide", Chapter 1 "Using the Operations Console", Section 1.1 "Operations Console Overview"* for details on how to connect to the Operations Console.

Cinder volume service is down

The Cinder volume service could be down if the server hosting the volume service fails. (Running the command `cinder service-list` will show the state of the volume service.) In this case you should follow the documented procedure linked below to start the volume service on another controller node. See *Section 7.1.3, "Managing Cinder Volume and Backup Services"* for details.

Creating a Cinder bootable volume fails

When creating a bootable volume from an image, your Cinder volume must be larger than the Virtual Size (raw size) of your image or creation will fail with an error.

An error like this error would appear in `cinder-volume.log` file:

```
'2016-06-14 07:44:00.954 25834 ERROR oslo_messaging.rpc.dispatcher ImageCopyFailure:
 Failed to copy image to volume: qemu-img: /dev/disk/by-path/ip-192.168.92.5:3260-iscsi-
iqn.2003-10.com.lefthandnetworks:mg-ses:146:volume-c0e75c66-a20a-4368-b797-d70afedb45cc-
lun-0: error while converting raw: Device is too small
```

```
2016-06-14 07:44:00.954 25834 ERROR oslo_messaging.rpc.dispatcher'
```

In an example where creating a 1GB bootable volume fails, your image may look like this:

```
$ qemu-img info /tmp/image.qcow2
image: /tmp/image.qcow2
file format: qcow2
virtual size: 1.5G (1563295744 bytes)
disk size: 354M
cluster_size: 65536
...
```

In this case, note that the image format is qcow2 and hte virtual size is 1.5GB, which is greater than the size of the bootable volume. Even though the compressed image size is less than 1GB, this bootable volume creation will fail.

When creating your disk model for nodes that will have the cinder volume role make sure that there is sufficient disk space allocated for a temporary space for image conversion if you will be creating bootable volumes. You should allocate enough space to the filesystem as would be needed to cater for the raw size of images to be used for bootable volumes - for example Windows images can be quite large in raw format.

By default, Cinder uses `/var/lib/cinder` for image conversion and this will be on the root filesystem unless it is explicitly separated. You can ensure there is enough space by ensuring that the root file system is sufficiently large, or by creating a logical volume mounted at `/var/lib/cinder` in the disk model when installing the system.

If your system is already installed, use these steps to update this:

1. Edit the configuration item `image_conversion_dir` in `cinder.conf.j2` to point to another location with more disk space. Make sure that the new directory location has the same ownership and permissions as `/var/lib/cinder` (owner:cinder group:cinder. mode 0750).

2. Then run this playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
   ```

API-level failures

If the API is inaccessible, determine if the API service is running on the target node. If it is not, check to see why the API service is not running in the log files. If it is running okay, check if the HAProxy service is functioning properly.

Note

After a controller node is rebooted, you must make sure to run the `ardana-start.yml` playbook to ensure all the services are up and running. For more information, see *Section 13.2.2.1, "Restarting Controller Nodes After a Reboot"*.

If the API service is returning an error code, look for the error message in the API logs on all API nodes. Successful completions would be logged like this:

```
2016-04-25 10:09:51.107 30743 INFO eventlet.wsgi.server [req-a14cd6f3-6c7c-4076-
adc3-48f8c91448f6
dfb484eb00f94fb39b5d8f5a894cd163 7b61149483ba4eeb8a05efa92ef5b197 - - -] 192.168.186.105
 - - [25/Apr/2016
10:09:51] "GET /v2/7b61149483ba4eeb8a05efa92ef5b197/volumes/detail HTTP/1.1" 200 13915
 0.235921
```

where `200` represents HTTP status 200 for a successful completion. Look for a line with your status code and then examine all entries associated with the request id. The request ID in the successful completion is highlighted in bold above.

The request may have failed at the scheduler or at the volume or backup service and you should also check those logs at the time interval of interest, noting that the log file of interest may be on a different node.

Operations that do not complete

If you have started an operation, such as creating or deleting a volume, that does not complete, the Cinder volume may be stuck in a state. You should follow the procedures for detaling with stuck volumes.

There are six transitory states that a volume can get stuck in:

State	Description
creating	The Cinder volume manager has sent a request to a back-end driver to create a volume, but has not received confirmation that the volume is available.
attaching	Cinder has received a request from Nova to make a volume available for attaching to an instance but has not received confirmation from Nova that the attachment is complete.

State	Description
detaching	Cinder has received notification from Nova that it will detach a volume from an instance but has not received notification that the detachment is complete.
deleting	Cinder has received a request to delete a volume but has not completed the operation.
backing-up	Cinder backup manager has started to back a volume up to Swift, or some other backup target, but has not completed the operation.
restoring	Cinder backup manager has started to restore a volume from Swift, or some other backup target, but has not completed the operation.

At a high level, the steps that you would take to address any of these states are similar:

1. Confirm that the volume is actually stuck, and not just temporarily blocked.

2. Where possible, remove any resources being held by the volume. For example, if a volume is stuck detaching it may be necessary to remove associated iSCSI or DM devices on the compute node.

3. Reset the state of the volume to an appropriate state, for example to `available` or `error`.

4. Do any final cleanup. For example, if you reset the state to `error` you can then delete the volume.

The next sections will describe specific steps you can take for volumes stuck in each of the transitory states.

Volumes stuck in Creating

Broadly speaking, there are two possible scenarios where a volume would get stuck in `creating`. The `cinder-volume` service could have thrown an exception while it was attempting to create the volume, and failed to handle the exception correctly. Or the volume back-end could have failed, or gone offline, after it received the request from Cinder to create the volume.

These two cases are different in that for the second case you will need to determine the reason the back-end is offline and restart it. Often, when the back-end has been restarted, the volume will move from `creating` to `available` so your issue will be resolved.

If you can create volumes successfully on the same back-end as the volume stuck in `creating` then the back-end is not down. So you will need to reset the state for the volume and then delete it.

To reset the state of a volume you can use the `cinder reset-state` command. You can use either the UUID or the volume name of the stuck volume.

For example, here is a volume list where we have a stuck volume:

```
$ cinder list
+------------------------------------+----------+------+------+-------------
+------------+
|                 ID                 |  Status  | Name | Size | Volume Type |Attached
 to |
+------------------------------------+----------+------+------+-------------
+------------+
| 14b76133-e076-4bd3-b335-fa67e09e51f6 | creating | vol1 |  1   |      -      |
    |
+------------------------------------+----------+------+------+-------------
+------------+
```

You can reset the state by using the `cinder reset-state` command, like this:

```
cinder reset-state --state error 14b76133-e076-4bd3-b335-fa67e09e51f6
```

Confirm that with another listing:

```
$ cinder list
+------------------------------------+----------+------+------+-------------
+------------+
|                 ID                 |  Status  | Name | Size | Volume Type |Attached
 to |
+------------------------------------+----------+------+------+-------------
+------------+
| 14b76133-e076-4bd3-b335-fa67e09e51f6 | error    | vol1 |  1   |      -      |
    |
+------------------------------------+----------+------+------+-------------
+------------+
```

You can then delete the volume:

```
$ cinder delete 14b76133-e076-4bd3-b335-fa67e09e51f6
```

```
Request to delete volume 14b76133-e076-4bd3-b335-fa67e09e51f6 has been accepted.
```

Volumes stuck in Deleting

If a volume is stuck in the deleting state then the request to delete the volume may or may not have been sent to and actioned by the back-end. If you can identify volumes on the back-end then you can examine the back-end to determine whether the volume is still there or not. Then you can decide which of the following paths you can take. It may also be useful to determine whether the back-end is responding, either by checking for recent volume create attempts, or creating and deleting a test volume.

The first option is to reset the state of the volume to `available` and then attempt to delete the volume again.

The second option is to reset the state of the volume to `error` and then delete the volume.

If you have reset the volume state to `error` then the volume may still be consuming storage on the back-end. If that is the case then you will need to delete it from the back-end using your back-end's specific tool.

Volumes stuck in Attaching

The most complicated situation to deal with is where a volume is stuck either in attaching or detaching, because as well as dealing with the state of the volume in Cinder and the back-end, you have to deal with exports from the back-end, imports to the compute node, and attachments to the compute instance.

The two options you have here are to make sure that all exports and imports are deleted and to reset the state of the volume to `available` or to make sure all of the exports and imports are correct and to reset the state of the volume to `in-use`.

A volume that is in attaching state should never have been made available to a compute instance and therefore should not have any data written to it, or in any buffers between the compute instance and the volume back-end. In that situation, it is often safe to manually tear down the devices exported on the back-end and imported on the compute host and then reset the volume state to `available`.

You can use the management features of the back-end you are using to locate the compute host to where the volume is being exported.

Volumes stuck in Detaching

The steps in dealing with a volume stuck in `detaching` state are very similar to those for a volume stuck in `attaching`. However, there is the added consideration that the volume was attached to, and probably servicing, I/O from a compute instance. So you must take care to ensure that all buffers are properly flushed before detaching the volume.

When a volume is stuck in `detaching`, the output from a `cinder list` command will include the UUID for the instance to which the volume was attached. From that you can identify the compute host that is running the instance using the `nova show` command.

For example, here are some snippets:

```
$ cinder list
+--------------------------------------+-----------+------------------------
+----------------+
|                  ID                  |  Status   |        Name        |  Attached
 to   |
+--------------------------------------+-----------+------------------------
+----------------+
| 85384325-5505-419a-81bb-546c69064ec2 | detaching |        vol1        |
 4bedaa76-78ca-… |
+--------------------------------------+-----------+------------------------
+----------------+
```

```
$ nova show 4bedaa76-78ca-4fe3-806a-3ba57a9af361|grep host
| OS-EXT-SRV-ATTR:host                  | mycloud-cp1-comp0005-mgmt
| OS-EXT-SRV-ATTR:hypervisor_hostname   | mycloud-cp1-comp0005-mgmt
| hostId                                |
 61369a349bd6e17611a47adba60da317bd575be9a900ea590c1be816
```

The first thing to check in this case is whether the instance is still importing the volume. Use `virsh list` and `virsh dumpxml` as described in the section above. If the XML for the instance has a reference to the device, then you should reset the volume state to `in-use` and attempt the `cinder detach` operation again.

```
$ cinder reset-state --state in-use --attach-status attached
 85384325-5505-419a-81bb-546c69064ec2
```

If the volume gets stuck detaching again, there may be a more fundamental problem, which is outside the scope of this document and you should contact the Support team.

If the volume is not referenced in the XML for the instance then you should remove any devices on the compute node and back-end and then reset the state of the volume to `available`.

```
$ cinder reset-state --state available --attach-status detached
 85384325-5505-419a-81bb-546c69064ec2
```

You can use the management features of the back-end you are using to locate the compute host to where the volume is being exported.

Volumes stuck in restoring

Restoring a Cinder volume from backup will be as slow as backing it up. So you must confirm that the volume is actually stuck by examining the `cinder-backup.log`. For example:

```
# tail -f cinder-backup.log |grep 162de6d5-ba92-4e36-aba4-e37cac41081b
2016-04-27 12:39:14.612 6689 DEBUG swiftclient [req-0c65ec42-8f9d-430a-b0d5-05446bf17e34
 - -
2016-04-27 12:39:15.533 6689 DEBUG cinder.backup.chunkeddriver [req-0c65ec42-8f9d-430a-
b0d5-
2016-04-27 12:39:15.566 6689 DEBUG requests.packages.urllib3.connectionpool
 [req-0c65ec42-
2016-04-27 12:39:15.567 6689 DEBUG swiftclient [req-0c65ec42-8f9d-430a-b0d5-05446bf17e34
 - - -
```

If you determine that the volume is genuinely stuck in `detaching` then you must follow the procedure described in the detaching section above to remove any volumes that remain exported from the back-end and imported on the controller node. Remember that in this case the volumes will be imported and mounted on the controller node running `cinder-backup`. So you do not have to search for the correct compute host. Also remember that no instances are involved so you do not need to confirm that the volume is not imported to any instances.

15.6.1.5 Debugging volume attachment

In an error case, it is possible for a Cinder volume to fail to complete an operation and revert back to its initial state. For example, attaching a Cinder volume to a Nova instance, so you would follow the steps above to examine the Nova compute logs for the attach request.

15.6.1.6 Errors creating volumes

If you are creating a volume and it goes into the `ERROR` state, a common error to see is `No valid host was found`. This means that the scheduler could not schedule your volume to a back-end. You should check that the volume service is up and running. You can use this command:

```
$ sudo cinder-manage service list
Binary          Host                            Zone          Status        State
 Updated At
cinder-scheduler ha-volume-manager               nova          enabled    :-)
 2016-04-25 11:39:30
cinder-volume    ha-volume-manager@ses1          nova          enabled    XXX
 2016-04-25 11:27:26
cinder-backup    ha-volume-manager               nova          enabled    :-)
 2016-04-25 11:39:28
```

In this example, the state of XXX indicates that the service is down.

If the service is up, next check that the back-end has sufficient space. You can use this command to show the available and total space on each back-end:

```
cinder get-pools —detail
```

If your deployment is using volume types, verify that the volume_backend_name in your cinder.conf file matches the volume_backend_name for the volume type you selected.

You can verify the back-end name on your volume type by using this command:

```
openstack volume type list
```

Then list the details about your volume type. For example:

```
$ openstack volume type show dfa8ecbd-8b95-49eb-bde7-6520aebacde0
+-------------------------------+-------------------------------------+
| Field                         | Value                               |
+-------------------------------+-------------------------------------+
| description                   | None                                |
| id                            | dfa8ecbd-8b95-49eb-bde7-6520aebacde0 |
| is_public                     | True                                |
| name                          | my3par                              |
| os-volume-type-access:is_public | True                              |
| properties                    | volume_backend_name='3par'          |
+-------------------------------+-------------------------------------+
```

15.6.1.7 Diagnosing back-end issues

You can find further troubleshooting steps for specific back-end types by vising these pages:

15.6.2 Swift Storage Troubleshooting

Troubleshooting scenarios with resolutions for the Swift service. You can use these guides to help you identify and resolve basic problems you may experience while deploying or using the Object Storage service. It contains the following troubleshooting scenarios:

15.6.2.1 Deployment Fails With "MSDOS Disks Labels Do Not Support Partition Names"

Description

If a disk drive allocated to Swift uses the MBR partition table type, the deploy process refuses to label and format the drive. This is to prevent potential data loss. (For more information, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 11 "Modifying Example Configurations for Object Storage using Swift", Section 11.5 "Allocating Disk Drives for Object Storage"*. If you intend to use the disk drive for Swift, you must convert the MBR partition table to GPT on the drive using `/sbin/sgdisk`.

📖 **Note**

> This process only applies to Swift drives. It does not apply to the operating system or boot drive.

Resolution

You must install `gdisk`, before using `sgdisk`:

1. Run the following command to install `gdisk`:

   ```
   sudo zypper install gdisk
   ```

2. Convert to the GPT partition type. Following is an example for converting `/dev/sdd` to the GPT partition type:

   ```
   sudo sgdisk -g /dev/sdd
   ```

3. Reboot the node to take effect. You may then resume the deployment (repeat the playbook that reported the error).

15.6.2.2   Examining Planned Ring Changes

Before making major changes to your rings, you can see the planned layout of Swift rings using the following steps:

1. Log in to the Cloud Lifecycle Manager.

2. Run the `swift-compare-model-rings.yml` playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml --extra-vars
     "drive_detail=yes"
   ```

3. Validate the following in the output:

- Drives are being added to all rings in the ring specifications.

- Servers are being used as expected (for example, you may have a different set of servers for the account/container rings than the object rings.)

- The drive size is the expected size.

15.6.2.3 Interpreting Swift Input Model Validation Errors

The following examples provide an error message, description, and resolution.

🎓 Note

To resolve an error, you must first modify the input model and re-run the configuration processor. (For instructions, see *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management".*) Then, continue with the deployment.

1. **Example Message - Model Mismatch: Cannot find drive /dev/sdt on padawan-ccp-c1-m2 (192.168.245.3))**

Description	The disk model used for node **padawan-ccp-c1-m2** has drive `/dev/sdt` listed in the devices list of a device-group where Swift is the consumer. However, the `dev/sdt` device does not exist on that node.
Resolution	If a drive or controller is failed on a node, the operating system does not see the drive and so the corresponding block device may not exist. Sometimes this is transitory and a reboot may resolve the problem. The problem may not be with `/dev/sdt`, but with another drive. For example, if `/dev/sds` is failed, when you

boot the node, the drive that you expect to be called `/dev/sdt` is actually called `/dev/sds`.

Alternatively, there may not be enough drives installed in the server. You can add drives. Another option is to remove `/dev/sdt` from the appropriate disk model. However, this removes the drive for all servers using the disk model.

2. **Example Message - Model Mismatch: Cannot find drive /dev/sdd2 on padawan-ccp-c1-m2 (192.168.245.3)**

Description	The disk model used for node **padawan-ccp-c1-m2** has drive `/dev/sdt` listed in the devices list of a device-group where Swift is the consumer. However, the partition number (2) has been specified in the model. This is not supported - only specify the block device name (for example `/dev/sdd`), not partition names in disk models.
Resolution	Remove the partition number from the disk model.

3. **Example Message - Cannot find IP address of padawan-ccp-c1-m3-swift for ring: account host: padawan-ccp-c1-m3-mgmt**

Description	The service (in this example, swift-account) is running on the node **padawan-ccp-c1-m3**. However, this node does not have a connection to the network designated for the `swift-account` service (that is, the SWIFT network).

Resolution	Check the input model for which networks are configured for each node type.

4. **Example Message - Ring: object-2 has specified replication_policy and erasure_coding_policy. Only one may be specified.**

Description	Only either `replication-policy` or `erasure-coding-policy` may be used in `ring-specifications`.
Resolution	Remove one of the policy types.

5. **Example Message - Ring: object-3 is missing a policy type (replication-policy or erasure-coding-policy)**

Description	There is no `replication-policy` or `erasure-coding-policy` section in `ring-specifications` for the object-0 ring.
Resolution	Add a policy type to the input model file.

15.6.2.4   Identifying the Swift Ring Building Server

15.6.2.4.1   Identify the Swift Ring Building server

Perform the following steps to identify the Swift ring building server:

1. Log in to the Cloud Lifecycle Manager.

2. Run the following command:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-status.yml --limit SWF-ACC[0]
```

3. Examine the output of this playbook. The last line underneath the play recap will give you the server name which is your Swift ring building server.

```
PLAY RECAP ******************************************************************
_SWF_CMN | status | Check systemd service running --------------------- 1.61s
```

```
_SWF_CMN | status | Check systemd service running ---------------------- 1.16s
_SWF_CMN | status | Check systemd service running ---------------------- 1.09s
_SWF_CMN | status | Check systemd service running ---------------------- 0.32s
_SWF_CMN | status | Check systemd service running ---------------------- 0.31s
_SWF_CMN | status | Check systemd service running ---------------------- 0.26s
-----------------------------------------------------------------------------
Total: -------------------------------------------------------------------- 7.88s
ardana-cp1-c1-m1-mgmt      : ok=7    changed=0    unreachable=0    failed=0
```

In the above example, the first swift proxy server is `ardana-cp1-c1-m1-mgmt`.

🔔 Important

For the purposes of this document, any errors you see in the output of this playbook can be ignored if all you are looking for is the server name for your Swift ring builder server.

15.6.2.5 Verifying a Swift Partition Label

🛡 Warning

For a system upgrade do NOT clear the label before starting the upgrade.

This topic describes how to check whether a device has a label on a partition.

15.6.2.5.1 Check Partition Label

To check whether a device has label on a partition, perform the following step:

- Log on to the node and use the `parted` command:

```
sudo parted -l
```

The output lists all of the block devices. Following is an example output for `/dev/sdc` with a single partition and a label of **c0a8f502h000**. Because the partition has a label, if you are about to install and deploy the system, you must clear this label before starting the deployment. As part of the deployment process, the system will label the partition.

```
.
.
.
```

```
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdc: 20.0GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name           Flags
1       1049kB  20.0GB  20.0GB  xfs          c0a8f502h000


.
.
.
```

15.6.2.6   Verifying a Swift File System Label

⛨ Warning

For a system upgrade do NOT clear the label before starting the upgrade.

This topic describes how to check whether a file system in a partition has a label.

To check whether a file system in a partition has a label, perform the following step:

- Log on to the server and execute the `xfs_admin` command (where `/dev/sdc1` is the partition where the file system is located):

  ```
  sudo xfs_admin -l /dev/sdc1
  ```

  The output shows if a file system has a label. For example, this shows a label of **c0a8f502h000**:

  ```
  $ sudo xfs_admin -l /dev/sdc1
  label = "c0a8f502h000"
  ```

  If no file system exists, the result is as follows:

  ```
  $ sudo xfs_admin -l /dev/sde1
  xfs_admin: /dev/sde is not a valid XFS file system (unexpected SB magic number
   0x00000000)
  ```

  If you are about to install and deploy the system, you must delete the label before starting the deployment. As part of the deployment process, the system will label the partition.

15.6.2.7 Recovering Swift Builder Files

When you execute the deploy process for a system, a copy of the builder files are stored on the following nodes and directories:

1. On the Swift ring building node, the primary reference copy is stored in the `/etc/swiftlm/<cloud-name>/<control-plane-name>/builder_dir/` directory.

2. On the next node after the Swift ring building node, a backup copy is stored in the `/etc/swiftlm/<cloud-name>/<control-plane-name>/builder_dir/` directory.

3. In addition, in the deploy process, the builder files are also copied to the `/etc/swiftlm/deploy_dir/<cloud-name>` directory on every Swift node.

If a copy of the builder files are found in the `/etc/swiftlm/<cloud-name>/<control-plane-name>/builder_dir/` then no further recover action is needed. However, if all nodes running the Swift account (SWF-ACC) are lost, then you need to copy the files from the `/etc/swiftlm/deploy_dir/<cloud-name>` directory from an intact Swift node to the `/etc/swiftlm/<cloud-name>/<control-plane-name>/builder_dir/` directory on the primary Swift ring building node.

If you have no intact `/etc/swiftlm` directory on any Swift node, you may be able to restore from Freezer. See *Section 13.2.2.2, "Recovering the Control Plane"*.

To restore builder files from the `/etc/swiftlm/deploy_dir` directory, use the following process:

1. Log in to the Swift ring building server (To identify the Swift ring building server, see *Section 15.6.2.4, "Identifying the Swift Ring Building Server"*).

2. Create the `/etc/swiftlm/`*CLOUD_NAME*`/`*CONTROL_PLANE_NAME*`/builder_dir` directory structure with these commands:
   Replace *CLOUD_NAME* with the name of your cloud and *CONTROL_PLANE_NAME* with the name of your control plane.

   ```
   tux > sudo mkdir -p /etc/swiftlm/<cloud-name>/<control-plane-name>/builder_dir/
   tux > sudo chown -R ardana.ardana /etc/swiftlm/
   ```

3. Log in to a Swift node where an intact `/etc/swiftlm/deploy_dir` directory exists.

4. Copy the builder files to the Swift ring building node. In the example below we use scp to transfer the files, where `swpac-c1-m1-mgmt` is the ring building node, `cloud1` is the cloud, and `cp1` is the control plane name:

```
ardana > scp /etc/swiftlm//cloud1/cp1/* swpac-ccp-c1-m1-mgmt:/etc/swiftlm/cloud1/
cp1/builder_dir/
```

5. Log in to the Cloud Lifecycle Manager.

6. Run the Swift reconfigure playbook to make sure every Swift node has the same rings:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
```

15.6.2.8   Restarting the Object Storage Deployment

This page describes the various operational procedures performed by Swift.

15.6.2.8.1   Restart the Swift Object Storage Deployment

The structure of ring is built in an incremental stages. When you modify a ring, the new ring uses the state of the old ring as a basis for the new ring. Rings are stored in the builder file. The `swiftlm-ring-supervisor` stores builder files in the `/etc/swiftlm/cloud1/cp1/builder_dir/` directory on the Ring-Builder node. The builder files are named <ring-name> builder. Prior versions of the builder files are stored in the `/etc/swiftlm/cloud1/cp1/builder_dir/backups` directory.

Generally, you use an existing builder file as the basis for changes to a ring. However, at initial deployment, when you create a ring there will be no builder file. Instead, the first step in the process is to build a builder file. The deploy playbook does this as a part of the deployment process. If you have successfully deployed some of the system, the ring builder files will exist.

If you change your input model (for example, by adding servers) now, the process assumes you are *modifying* a ring and behaves differently than while creating a ring from scratch. In this case, the ring is not balanced. So, if the cloud model contains an error or you decide to make substantive changes, it is a best practice to start from scratch and build rings using the steps below.

15.6.2.8.2　Reset Builder Files

You must reset the builder files during the initial deployment process (only). This process should be used only when you want to restart a deployment from scratch. If you reset the builder files after completing your initial deployment, then you are at a risk of losing critical system data.

Delete the builder files in the `/etc/swiftlm/cloud1/cp1/builder-dir/` directory. For example, for the region0 Keystone region (the default single region designation), do the following:

```
sudo rm /etc/swiftlm/cloud1/cp1/builder_dir/*.builder
```

> **Note**
>
> If you have successfully deployed a system and accidentally delete the builder files, you can recover to the correct state. For instructions, see *Section 15.6.2.7, "Recovering Swift Builder Files"*.

15.6.2.9　Increasing the Swift Node Timeout Value

On a heavily loaded Object Storage system timeouts may occur when transferring data to or from Swift, particularly large objects.

The following is an example of a timeout message in the log (`/var/log/swift/swift.log`) on a Swift proxy server:

```
Jan 21 16:55:08 ardana-cp1-swpaco-m1-mgmt proxy-server: ERROR with Object server
 10.243.66.202:6000/disk1 re: Trying to write to
/v1/AUTH_1234/testcontainer/largeobject: ChunkWriteTimeout (10s)
```

If this occurs, it may be necessary to increase the `node_timeout` parameter in the `proxy-server.conf` configuration file.

The `node_timeout` parameter in the Swift `proxy-server.conf` file is the maximum amount of time the proxy server will wait for a response from the account, container, or object server. The default value is 10 seconds.

In order to modify the timeout you can use these steps:

1. Log in to the Cloud Lifecycle Manager.

2. Edit the `~/openstack/my_cloud/config/swift/proxy-server.conf.j2` file and add a line specifying the `node_timeout` into the `[app:proxy-server]` section of the file. Example, in bold, increasing the timeout to 30 seconds:

```
[app:proxy-server]
use = egg:swift#proxy
.
.
node_timeout = 30
```

3. Commit your configuration to the *Book "Installing with Cloud Lifecycle Manager", Chapter 10 "Using Git for Configuration Management"*, as follows:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

4. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Use the playbook below to create a deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Change to the deployment directory and run the Swift reconfigure playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-reconfigure.yml
```

15.6.2.10 Troubleshooting Swift File System Usage Issues

If you have recycled your environment to do a re-installation and you haven't run the `wipe_disks.yml` playbook in the process, you may experience an issue where your file system usage continues to grow exponentially even though you are not adding any files to your Swift system. This is likely occurring because the quarantined directory is getting filled up. You can find this directory at `/srv/node/disk0/quarantined`.

You can resolve this issue by following these steps:

1. SSH to each of your Swift nodes and stop the replication processes on each of them. The following commands must be executed on each of your Swift nodes. Make note of the time that you performed this action as you will reference it in step three.

   ```
   sudo systemctl stop swift-account-replicator
   sudo systemctl stop swift-container-replicator
   sudo systemctl stop swift-object-replicator
   ```

2. Examine the `/var/log/swift/swift.log` file for events that indicate when the auditor processes have started and completed audit cycles. For more details, see *Section 15.6.2.10, "Troubleshooting Swift File System Usage Issues"*.

3. Wait until you see that the auditor processes have finished two complete cycles since the time you stopped the replication processes (from step one). You must check every Swift node, which on a lightly loaded system that was recently installed this should take less than two hours.

4. At this point you should notice that your quarantined directory has stopped growing. You may now delete the files in that directory on each of your nodes.

5. Restart the replication processes using the Swift start playbook:

   a. Log in to the Cloud Lifecycle Manager.

   b. Run the Swift start playbook:

      ```
      cd ~/scratch/ansible/next/ardana/ansible
      ansible-playbook -i hosts/verb_hosts swift-start.yml
      ```

15.6.2.10.1   Examining the Swift Log for Audit Event Cycles

Below is an example of the `object-server` start and end cycle details. They were taken by using the following command on a Swift node:

```
sudo grep object-auditor /var/log/swift/swift.log|grep ALL
```

Example output:

```
$ sudo grep object-auditor /var/log/swift/swift.log|grep ALL
...
```

```
Apr  1 13:31:18 padawan-ccp-c1-m1-mgmt object-auditor: Begin object audit "forever" mode
 (ALL)
Apr  1 13:31:18 padawan-ccp-c1-m1-mgmt object-auditor: Object audit (ALL). Since Fri Apr
 1 13:31:18 2016: Locally: 0 passed, 0 quarantined, 0 errors files/sec: 0.00 , bytes/sec:
 0.00, Total time: 0.00, Auditing time: 0.00, Rate: 0.00
Apr  1 13:51:32 padawan-ccp-c1-m1-mgmt object-auditor: Object audit (ALL) "forever" mode
 completed: 1213.78s. Total quarantined: 0, Total errors: 0, Total files/sec: 7.02, Total
 bytes/sec: 9999722.38, Auditing time: 1213.07, Rate: 1.00
```

In this example, the auditor started at `13:31` and ended at `13:51`.

In this next example, the `account-auditor` and `container-auditor` use similar message structure, so we only show the container auditor. You can substitute `account` for `container` as well:

```
$ sudo grep container-auditor /var/log/swift/swift.log
...
Apr  1 14:07:00 padawan-ccp-c1-m1-mgmt container-auditor: Begin container audit pass.
Apr  1 14:07:00 padawan-ccp-c1-m1-mgmt container-auditor: Since Fri Apr  1 13:07:00 2016:
 Container audits: 42 passed audit, 0 failed audit
Apr  1 14:37:00 padawan-ccp-c1-m1-mgmt container-auditor: Container audit pass completed:
 0.10s
```

In the example, the container auditor started a cycle at `14:07` and the cycle finished at `14:37`.

15.7   Monitoring, Logging, and Usage Reporting Troubleshooting

Troubleshooting scenarios with resolutions for the Monitoring, Logging, and Usage Reporting services.

15.7.1   Troubleshooting Centralized Logging

This section contains the following scenarios:

- *Section 15.7.1.1, "Reviewing Log Files"*

- *Section 15.7.1.2, "Monitoring Centralized Logging"*

- *Section 15.7.1.3, "Situations In Which Logs Might Not Be Collected"*

- *Section 15.7.1.4, "Error When Creating a Kibana Visualization"*

15.7.1.1   Reviewing Log Files

You can troubleshoot service-specific issues by reviewing the logs. After logging into Kibana, follow these steps to load the logs for viewing:

1. Navigate to the **Settings** menu to configure an index pattern to search for.

2. In the **Index name or pattern** field, you can enter `logstash-*` to query all Elasticsearch indices.

3. Click the green **Create** button to create and load the index.

4. Navigate to the **Discover** menu to load the index and make it available to search.

☞ Note

If you want to search specific Elasticsearch indices, you can run the following command from the control plane to get a full list of available indices:

```
curl localhost:9200/_cat/indices?v
```

Once the logs load you can change the timeframe from the dropdown in the upper-righthand corner of the Kibana window. You have the following options to choose from:

- **Quick** - a variety of time frame choices will be available here

- **Relative** - allows you to select a start time relative to the current time to show this range

- **Absolute** - allows you to select a date range to query

When searching there are common fields you will want to use, such as:

- **type** - this will include the service name, such as `keystone` or `ceilometer`

- **host** - you can specify a specific host to search for in the logs

- **file** - you can specify a specific log file to search

For more details on using Kibana and Elasticsearch to query logs, see https://www.elastic.co/guide/en/kibana/3.0/working-with-queries-and-filters.html ↗

15.7.1.2 Monitoring Centralized Logging

To help keep ahead of potential logging issues and resolve issues before they affect logging, you may want to monitor the Centralized Logging Alarms.

To monitor logging alarms:

1. Log in to Operations Console.

2. From the menu button in the upper left corner, navigate to the **Alarm Definitions** page.

3. Find the alarm definitions that are applied to the various hosts. See the *Section 15.1.1, "Alarm Resolution Procedures"* for the Centralized Logging Alarm Definitions.

4. Navigate to the **Alarms** page

5. Find the alarm definitions applied to the various hosts. These should match the alarm definitions in the *Section 15.1.1, "Alarm Resolution Procedures"*.

6. See if the alarm is green (good) or is in a bad state. If any are in a bad state, see the possible actions to perform in the *Section 15.1.1, "Alarm Resolution Procedures"*.

You can use this filtering technique in the "Alarms" page to look for the following:

1. To look for processes that may be down, filter for **"Process"** then make sure the process are up:

   - Elasticsearch

   - Logstash

   - Beaver

   - Apache (Kafka)

   - Kibana

   - Monasca

2. To look for sufficient disk space, filter for **"Disk"**

3. To look for sufficient RAM memory, filter for **"Memory"**

15.7.1.3 Situations In Which Logs Might Not Be Collected

Centralized logging might not collect log data under the following circumstances:

- If the Beaver service is not running on one or more of the nodes (controller or compute), logs from these nodes will not be collected.

15.7.1.4 Error When Creating a Kibana Visualization

When creating a visualization in Kibana you may get an error similiar to this:

```
"logstash-*" index pattern does not contain any of the following field types: number
```

To resolve this issue:

1. Log in to Kibana.

2. Navigate to the `Settings` page.

3. In the left panel, select the `logstash-*` index.

4. Click the **Refresh** button. You may see a mapping conflict warning after refreshing the index.

5. Re-create the visualization.

15.7.1.5 After Deploying Logging-API, Logs Are Not Centrally Stored

If you are using the Logging-API and logs are not being centrally stored, use the following checklist to troubleshoot Logging-API.

#	Item
	Ensure Monasca is running.
	Check any alarms Monasca has triggered.
	Check to see if the Logging-API (monasca-log-api) process alarm has triggered.
	Run an Ansible playbook to get status of the Cloud Lifecycle Manager: `ansible-playbook -i hosts/verb_hosts ardana-status.yml`

#	Item
	Troubleshoot all specific tasks that have failed on the Cloud Lifecycle Manager.
	Ensure that the Logging-API daemon is up.
	Run an Ansible playbook to try and bring the Logging-API daemon up: `ansible-playbook —I hosts/verb_hosts logging-start.yml`
	If you get errors trying to bring up the daemon, resolve them.
	Verify the Logging-API configuration settings are correct in the configuration file: `roles/kronos-api/templates/kronos-apache2.conf.j2`

The following is a sample Logging-API configuration file:

```
{#
# (c) Copyright 2015-2016 Hewlett Packard Enterprise Development LP
# Licensed under the Apache License, Version 2.0 (the "License"); you may
# not use this file except in compliance with the License. You may obtain
# a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
# License for the specific language governing permissions and limitations
# under the License.
#
#}
Listen {{ kronos_api_host }}:{{ kronos_api_port }}
<VirtualHost *:{{ kronos_api_port }}>
    WSGIDaemonProcess log-api processes=4 threads=4 socket-timeout=300
 user={{ kronos_user }} group={{ kronos_group }} python-path=/opt/stack/service/
kronos/venv:/opt/stack/service/kronos/venv/bin/../lib/python2.7/site-packages/ display-
name=monasca-log-api
    WSGIProcessGroup log-api
    WSGIApplicationGroup log-api
    WSGIScriptAlias / {{ kronos_wsgi_dir }}/app.wsgi
    ErrorLog /var/log/kronos/wsgi.log
    LogLevel info
    CustomLog /var/log/kronos/wsgi-access.log combined
```

```
    <Directory /opt/stack/service/kronos/venv/bin/../lib/python2.7/site-packages/
monasca_log_api>
        Options Indexes FollowSymLinks MultiViews
        Require all granted
        AllowOverride None
        Order allow,deny
        allow from all
        LimitRequestBody 102400
    </Directory>

    SetEnv no-gzip 1
</VirtualHost>
```

15.7.1.6   Re-enabling Slow Logging

MariaDB slow logging was enabled by default in earlier versions. Slow logging logs slow MariaDB queries to `/var/log/mysql/mysql-slow.log` on FND-MDB hosts.

As it is possible for temporary tokens to be logged to the slow log, we have disabled slow log in this version for security reasons.

To re-enable slow logging follow the following procedure:

1. Login to the Cloud Lifecycle Manager and set a mariadb service configurable to enable slow logging.

   ```
   cd ~/openstack/my_cloud
   ```

   a. Check slow_query_log is currently disabled with a value of 0:

   ```
   grep slow ./config/percona/my.cfg.j2
   slow_query_log          = 0
   slow_query_log_file     = /var/log/mysql/mysql-slow.log
   ```

   b. Enable slow logging in the server configurable template file and confirm the new value:

   ```
   sed -e 's/slow_query_log = 0/slow_query_log = 1/' -i ./config/percona/my.cfg.j2
   grep slow ./config/percona/my.cfg.j2
   slow_query_log          = 1
   ```

```
slow_query_log_file      = /var/log/mysql/mysql-slow.log
```

c. Commit the changes:

```
git add -A
git commit -m "Enable Slow Logging"
```

2. Run the configuration procesor.

```
cd ~/openstack/ardana/ansible/
ansible-playbook -i hosts/localhost config-processor-run.yml
```

3. You will be prompted for an encryption key, and also asked if you want to change the encryption key to a new value, and it must be a different key. You can turn off encryption by typing the following:

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e
  rekey=""
```

4. Create a deployment directory.

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

5. Reconfigure Percona (note this will restart your mysqld server on your cluster hosts).

```
ansible-playbook -i hosts/verb_hosts percona-reconfigure.yml
```

15.7.2 Usage Reporting Troubleshooting

Troubleshooting scenarios with resolutions for the Ceilometer service.

This page describes troubleshooting scenarios for Ceilometer.

15.7.2.1 Logging

Logs for the various running components in the Overcloud Controllers can be found at */var/log/ceilometer.log*

The Upstart for the services also logs data at **/var/log/upstart**

15.7.2.2 Modifying

Change the level of debugging in Ceilometer by editing the **ceilometer.conf** file located at **/etc/ceilometer/ceilometer.conf**. To log the maximum amount of information, change the **level** entry to **DEBUG**.

Note: When the logging level for a service is changed, that service must be re-started before the change will take effect.

This is an excerpt of the **ceilometer.conf** configuration file showing where to make changes:

```
[loggers]
 keys: root

[handlers]
 keys: watchedfile, logstash

[formatters]
 keys: context, logstash

[logger_root]
 qualname: root
 handlers: watchedfile, logstash
 level: NOTSET
```

15.7.2.3 Messaging/Queuing Errors

Ceilometer relies on a message bus for passing data between the various components. In high-availability scenarios, RabbitMQ servers are used for this purpose. If these servers are not available, the Ceilometer log will record errors during "Connecting to AMQP" attempts.

These errors may indicate that the RabbitMQ messaging nodes are not running as expected and/or the RPC publishing pipeline is stale. When these errors occur, re-start the instances.

Example error:

```
Error: unable to connect to node 'rabbit@xxxx-rabbitmq0000': nodedown
```

Use the RabbitMQ CLI to re-start the instances and then the host.

1. Restart the downed cluster node.

   ```
   sudo invoke-rc.d rabbitmq-server start
   ```

2. Restart the RabbitMQ host

```
sudo rabbitmqctl start_app
```

15.8   Backup and Restore Troubleshooting

Troubleshooting scenarios with resolutions for the Backup and Restore service.

The following logs will help you troubleshoot Freezer functionality:

Component	Description
Freezer Client	/var/log/freezer-agent/freezer-agent.log
Freezer Scheduler	/var/log/freezer-agent/freezer-scheduler.log
Freezer API	/var/log/freezer-api/freezer-api-access.log/ var/log/freezer-api/freezer-api-modwsgi.log /var/log/freezer-api/freezer-api.log

The following issues apply to the Freezer UI and the backup and restore process:

- The UI for backup and restore is supported only if you log in as "ardana_backup". All other users will see the UI panel but the UI will not work.

- If a backup or restore action fails via the UI, you must check the Freezer logs for details of the failure.

- Job Status and Job Result on the UI and backend (CLI) are not in sync.

- For a given "Action" the following modes are not supported from the UI:

  - Microsoft SQL Server

  - Cinder

  - Nova

- Start and end dates and times available for job creation should not be used due to a known issue. Please refrain from using those fields.

- Once a backup is created. A listing of the contents is needed to verify if the backup of any single item was done.

15.9 Orchestration Troubleshooting

Troubleshooting scenarios with resolutions for the Orchestration services. Troubleshooting scenarios with resolutions for the Orchestration services.

15.9.1 Heat Troubleshooting

Troubleshooting scenarios with resolutions for the Heat service. This page describes troubleshooting scenarios for Heat.

15.9.1.1 RPC timeout on Heat stack creation

If you exerience a remote procedure call (RPC) timeout failure when attempting heat stack-create, you can work around the issue by increasing the timeout value and purging records of deleted stacks from the database. To do so, follow the steps below. An example of the error is:

```
MessagingTimeout: resources.XXX-LCP-Pair01.resources[0]: Timed out waiting for a reply to
 message ID e861c4e0d9d74f2ea77d3ec1984c5cb6
```

1. Increase the timeout value.

   ```
   cd ~/openstack/my_cloud/config/heat
   ```

2. Make changes to heat config files. In heat.conf.j2 add this timeout value:

   ```
   rpc_response_timeout=300
   ```

   Commit your changes

   ```
   git commit -a -m "some message"
   ```

3. Move to ansible directory and run the following playbooks:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

4. Change to the scratch directory and run heat-reconfigure:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ```

```
ansible-playbook -i hosts/verb_hosts heat-reconfigure.yml
```

5. Purge records of deleted stacks from the database. First delete all stacks that are in failed state. Then execute the following

```
sudo /opt/stack/venv/heat-20151116T000451Z/bin/python2
/opt/stack/service/heat-engine/venv/bin/heat-manage
--config-file /opt/stack/service/heat-engine-20151116T000451Z/etc/heat/heat.conf
--config-file /opt/stack/service/heat-engine-20151116T000451Z/etc/heat/engine.conf
 purge_deleted 0
```

15.9.1.2 General Heat stack creation errors

In Heat, in general when a timeout occurs it means that the underlying resource service such as Nova, Neutron, or Cinder, fails to complete the required action. No matter what error this underlying service reports, Heat simply reports it back. So in the case of time-out in Heat stack create, you should look at the logs of the underlying services, most importantly the Nova service, to understand the reason for the timeout.

15.9.1.3 Multiple Heat stack create failure

The Monasca AlarmDefinition resource, `OS::Monasca::AlarmDefinition` used for Heat autoscaling, consists of an optional property **name** for defining the alarm name. In case this optional property being specified in the Heat template, this name must be unique in the same project of the system. Otherwise, multiple heat stack create using this heat template will fail with the following conflict:

```
| cpu_alarm_low  | 5fe0151b-5c6a-4a54-bd64-67405336a740 | HTTPConflict:
 resources.cpu_alarm_low: An alarm definition already exists for project / tenant:
 835d6aeeb36249b88903b25ed3d2e55a named: CPU utilization less than 15 percent  |
 CREATE_FAILED | 2016-07-29T10:28:47 |
```

This is due to the fact that the Monasca registers the alarm definition name using this name property when it is defined in the Heat template. This name must be unique.

To avoid this problem, if you want to define an alarm name using this property in the template, you must be sure this name is unique within a project in the system. Otherwise, you can leave this optional property undefined in your template. In this case, the system will create an unique alarm name automatically during heat stack create.

15.9.1.4 Unable to Retrieve QOS Policies

Launching the Orchestration Template Generator may trigger the message: `Unable to retrieve resources Qos Policies`. This is a known upstream bug (https://storyboard.openstack.org/#!/story/2003523) ↗. This information message can be ignored.

15.9.2 Troubleshooting Magnum Service

Troubleshooting scenarios with resolutions for the Magnum service. Magnum Service provides container orchestration engines such as Docker Swarm, Kubernetes, and Apache Mesos available as first class resources. You can use this guide to help with known issues and troubleshooting of Magnum services.

15.9.2.1 Magnum cluster fails to create

Typically, small size clusters need about 3-5 minutes to stand up. If cluster stand up takes longer, you may proceed with troubleshooting, not waiting for status to turn to CREATE_FAILED after timing out.

1. Use `heat resource-list -n2` to identify which Heat stack resource is stuck in **CREATE_IN_PROGRESS**.

> 🎓 **Note**
>
> The main Heat stack has nested stacks, one for kubemaster(s) and one for kubeminion(s). These stacks are visible as resources of type *OS::Heat::ResourceGroup* (in parent stack) and *file:///...* in nested stack. If any resource remains in *CREATE_IN_PROGRESS* state within the nested stack, the overall state of the resource will be *CREATE_IN_PROGRESS*.

```
$ heat resource-list -n2 22385a42-9e15-49d9-a382-f28acef36810
+-------------------------------+------------------------------------
+------------------------------------+------------------+--------------------
+------------------------------------------------------------+
| resource_name                 | physical_resource_id              |
 resource_type                  | resource_status  | updated_time           |
 stack_name                                                   |
```

```
+--------------------------------+-----------------------------------
+------------------------------------+--------------------+----------------------
+--------------------------------------------------------------+
| api_address_floating_switch    | 06b2cc0d-77f9-4633-8d96-f51e2db1faf3 |
 Magnum::FloatingIPAddressSwitcher   | CREATE_COMPLETE    | 2017-04-10T21:25:10Z |
 my-cluster-z4aquda2mgpv                                       |
. . .

| fixed_subnet                   | d782bdf2-1324-49db-83a8-6a3e04f48bb9 |
 OS::Neutron::Subnet                 | CREATE_COMPLETE    | 2017-04-10T21:25:11Z |
 my-cluster-z4aquda2mgpv                                       |
| kube_masters                   | f0d000aa-d7b1-441a-a32b-17125552d3e0 |
 OS::Heat::ResourceGroup             | CREATE_IN_PROGRESS | 2017-04-10T21:25:10Z |
 my-cluster-z4aquda2mgpv                                       |
| 0                              | b1ff8e2c-23dc-490e-ac7e-14e9f419cfb6 | file:///
opt/s...ates/kubemaster.yaml | CREATE_IN_PROGRESS | 2017-04-10T21:25:41Z | my-
cluster-z4aquda2mgpv-kube_masters-utyggcbucbhb               |
| kube_master                    | 4d96510e-c202-4c62-8157-c0e3dddff6d5 |
 OS::Nova::Server                    | CREATE_IN_PROGRESS | 2017-04-10T21:25:48Z |
 my-cluster-z4aquda2mgpv-kube_masters-utyggcbucbhb-0-saafd5k7l7im |
. . .
```

2. If stack creation failed on some native OpenStack resource, like **OS::Nova::Server** or **OS::Neutron::Router**, proceed with respective service troubleshooting. This type of error usually does not cause time out, and cluster turns into status **CREATE_FAILED** quickly. The underlying reason of the failure, reported by Heat, can be checked via the `magnum cluster-show` command.

3. If stack creation stopped on resource of type OS::Heat::WaitCondition, Heat is not receiving notification from cluster VM about bootstrap sequence completion. Locate corresponding resource of type **OS::Nova::Server** and use its **physical_resource_id** to get information about the VM (which should be in status **CREATE_COMPLETE**)

```
$ nova show 4d96510e-c202-4c62-8157-c0e3dddff6d5
+----------------------------------
+-----------------------------------------------------------------------------------
+
| Property                         | Value
                                                     |
+----------------------------------
+-----------------------------------------------------------------------------------
+
| OS-DCF:diskConfig                | MANUAL
                                                     |
```

```
| OS-EXT-AZ:availability_zone        | nova
                                                                                 |
| OS-EXT-SRV-ATTR:host               | comp1
                                                                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname | comp1
                                                                                 |
| OS-EXT-SRV-ATTR:instance_name      | instance-00000025
                                                                                 |
| OS-EXT-STS:power_state             | 1
                                                                                 |
| OS-EXT-STS:task_state              | -
                                                                                 |
| OS-EXT-STS:vm_state                | active
                                                                                 |
| OS-SRV-USG:launched_at             | 2017-04-10T22:10:40.000000
                                                                                 |
| OS-SRV-USG:terminated_at           | -
                                                                                 |
| accessIPv4                         |
                                                                                 |
| accessIPv6                         |
                                                                                 |
| config_drive                       |
                                                                                 |
| created                            | 2017-04-10T22:09:53Z
                                                                                 |
| flavor                             | m1.small (2)
                                                                                 |
| hostId                             |
  eb101a0293a9c4c3a2d79cee4297ab6969e0f4ddd105f4d207df67d2
                        |
| id                                 | 4d96510e-c202-4c62-8157-c0e3dddff6d5
                                                                                 |
| image                              | fedora-atomic-26-20170723.0.x86_64
  (4277115a-f254-46c0-9fb0-fffc45d2fd38)                                         |
| key_name                           | testkey
                                                                                 |
| metadata                           | {}
                                                                                 |
| name                               | my-zaqshggwge-0-sqhpyez4dig7-kube_master-
wc4vv7ta42r6                                                                      |
| os-extended-volumes:volumes_attached | [{"id":
  "24012ce2-43dd-42b7-818f-12967cb4eb81"}]
                |
| private network                    | 10.0.0.14, 172.31.0.6
                                                                                 |
```

```
| progress                                | 0
                                             |
| security_groups                         | my-cluster-z7ttt2jvmyqf-secgroup_base-
gzcpzsiqkhxx, my-cluster-z7ttt2jvmyqf-secgroup_kube_master-27mzhmkjiv5v |
| status                                  | ACTIVE
                                             |
| tenant_id                               | 2f5b83ab49d54aaea4b39f5082301d09
                                             |
| updated                                 | 2017-04-10T22:10:40Z
                                             |
| user_id                                 | 7eba6d32db154d4790e1d3877f6056fb
                                             |
+------------------------------------
 +-----------------------------------------------------------------------------------------
 +
```

4. Use the floating IP of the master VM to log into first master node. Use the appropriate username below for your VM type. Passwords should not be required as the VMs should have public ssh key installed.

VM Type	Username
Kubernetes or Swarm on Fedora Atomic	fedora
Kubernetes on CoreOS	core
Mesos on Ubuntu	ubuntu

5. Useful dianostic commands

- Kubernetes cluster on Fedora Atomic

```
sudo journalctl --system
sudo journalctl -u cloud-init.service
sudo journalctl -u etcd.service
sudo journalctl -u docker.service
sudo journalctl -u kube-apiserver.service
sudo journalctl -u kubelet.service
sudo journalctl -u wc-notify.service
```

- Kubernetes cluster on CoreOS

```
sudo journalctl --system
sudo journalctl -u oem-cloudinit.service
sudo journalctl -u etcd2.service
sudo journalctl -u containerd.service
```

```
sudo journalctl -u flanneld.service
sudo journalctl -u docker.service
sudo journalctl -u kubelet.service
sudo journalctl -u wc-notify.service
```

- Swarm cluster on Fedora Atomic

```
sudo journalctl --system
sudo journalctl -u cloud-init.service
sudo journalctl -u docker.service
sudo journalctl -u swarm-manager.service
sudo journalctl -u wc-notify.service
```

- Mesos cluster on Ubuntu

```
sudo less /var/log/syslog
sudo less /var/log/cloud-init.log
sudo less /var/log/cloud-init-output.log
sudo less /var/log/os-collect-config.log
sudo less /var/log/marathon.log
sudo less /var/log/mesos-master.log
```

15.10   Troubleshooting Tools

Tools to assist with troubleshooting issues in your cloud. Additional troubleshooting information is available at *Section 15.1, "General Troubleshooting"*.

15.10.1   Retrieving the SOS Report

The SOS report provides debug level information about your environment to assist in troubleshooting issues. When troubleshooting and debugging issues in your HPE Helion OpenStack environment you can run an ansible playbook that will provide you with a full debug report, referred to as a SOS report. These reports can be sent to the support team when seeking assistance.

15.10.1.1   Retrieving the SOS Report

1. Log in to the Cloud Lifecycle Manager.

2. Run the SOS report ansible playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts sosreport-run.yml
```

3. Retrieve the SOS report tarballs, which will be in the following directories on your Cloud Lifecycle Manager:

```
/tmp
/tmp/sosreport-report-archives/
```

4. You can then use these reports to troubleshoot issues further or provide to the support team when you reach out to them.

⬡ Warning

The SOS Report may contain sensitive information because service configuration file data is included in the report. Please remove any sensitive information before sending the SOSReport tarball externally.