CPS Installation Guide for OpenStack, Release 9.1.0

First Published: April 29, 2016

Last Modified: May 23, 2016

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

About this Guide

The *CPS Installation Guide for OpenStack* assists System Administrators and Network Engineers with installing Cisco Policy Suite (CPS) in an OpenStack environment.

For installations that need to be performed within VMware virtualized environments, refer to the *CPS Installation Guide for VMware*.

Note The host addresses, names of VLANs, and so on used in the examples in this document may be different for your deployment.

Audience

This guide is best used by these readers:

- Network administrators
- Network engineers
- Network operators
- System administrators

This document assumes a general understanding of network architecture, configuration, and operations.

Additional Support

For further documentation and support:

- Contact your Cisco Systems, Inc. technical representative.
- Call the Cisco Systems, Inc. technical support number.
- Write to Cisco Systems, Inc. at support@cisco.com.
- Refer to support matrix at http://www.support.cisco.com and to other documents related to Cisco Policy Suite.

Conventions (all documentation)

This document uses the following conventions.

Conventions	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
courier font	Terminal sessions and information the system displays appear in courier font.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means reader take note. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.

IMPORTANT SAFETY INSTRUCTIONS.

Means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Warning Provided for additional information and to comply with regulatory and customer requirements.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the What's New in Cisco Product Documentation RSS feed. RSS feeds are a free service.

CHAPTER 1

Preinstallation Tasks

Overview and Requirements

The Cisco Policy Suite offers a carrier-grade, high capacity, high performance, virtualized software solution, capable of running on VMWare, OpenStack/KVM hypervisors or cloud infrastructures. To meet the stringent performance, capacity, and availability demands, the Cisco software requires that all allocated hardware system resources be 100% available when needed, and not oversubscribed or shared across separate VM's.

For customers operating a cloud infrastructure, the infrastructure must be configured to guarantee CPU, memory, network, and I/O availability for each CPS VM. Oversubscription of system resources will reduce the performance and capacity of the platform, and may compromise availability and response times. CPU core requirements are listed as pCPUs (physical cores) not vCPU's (hyperthreaded virtual cores).

In addition, the CPS carrier-grade platform requires:

- RAM reservation is enabled for all memory allocated to the CPS VM.

- CPU Hyperthreading must be disabled (no over-subscription of CPU cores). If Hyperthreading cannot be disabled (ENABLED), then CPU pinning should be ENABLED, and all vCPU allocations should be doubled.
- CPU must be a high performance Intel x86 64-bit chipset.
- RAM latency should be lower than 15 ns.
- RAM should be error-correcting ECC memory.
- Disk storage performance should be less than 2 millisecond average latency.
- Disk storage performance needs to support greater than 5000 input/output operations per second (IOPS) per CPS VM.
- Disk storage must provide redundancy and speed, such as RAID 0+1.
- Hardware and hardware design must be configured for better than 99.999% availability.
- For HA deployments, Cisco requires the customer designs comply with the Cisco CPS HA design guidelines.
- At least two of each CPS VM type (qns, lb, pcrfclient, sessionmgr) for each platform.
- Each CPS VM type must not share common HW zone with the same CPS VM type.
- The number of CPU cores, memory, NICs, and storage allocated per CPS VM must meet or exceed the requirements.

Install OpenStack

CPS is supported on OpenStack Juno or Kilo.

For more information about installing these versions of OpenStack, see the following URLs:

- http://docs.openstack.org/juno/
- http://docs.openstack.org/kilo/

Note If you are installing Juno and intend to use the Orchestration API, ensure that Heat, the orchestration component of OpenStack, is enabled. For more information about enabling Heat, refer to the OpenStack documentation at: http://docs.openstack.org/juno/install-guide/install/yum/content/heat-install-controller-node.html

After you install OpenStack, you must perform some prerequisite tasks. The following sections describe these prerequisite tasks.

CPU Pinning

CPU pinning is supported and recommended in OpenStack deployments where hyperthreading is enabled. This enables CPS VMs to be pinned to dedicated physical CPU cores.

Prerequisites

- OpenStack Kilo (OSP 7)
- Numactl must be installed on control and compute nodes.

Refer to the following link for general instructions to enable CPU pinning for guest VMs: http://redhatstackblog.redhat.com/2015/05/05/cpu-pinning-and-numa-topology-awareness-in-openstack-compute/

Install numactl

The numactl package provides a command to examine the NUMA layout of the blades. Install this package on compute nodes to help determine which CPUs to set aside for pinning.

Run the following command to install numactl:

```
yum install numactl
```

Identify the Physical CPUs to Use for Pinning

Step 1 Run the following command on the compute nodes where you want to set aside physical CPUs for pinning.

```
numactl --hardware
```

```
[root@os6-compute-1 ~]# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 8 9 10 11
node 0 size: 98245 MB
node 0 free: 87252 MB
node 1 cpus: 4 5 6 7 12 13 14 15
node 1 size: 98304 MB
node 1 free: 95850 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

Step 2 Determine the pool of CPUs you want to set aside for pinning.
At least 2 CPUs should be set aside for the Hypervisor in each node if it is a compute only blade. If the blade is operating as both a control and compute node, set aside more CPUs for OpenStack services.

Select the remaining CPUs for pinning.

In the above example, the following CPUs could be selected for pinning: **2, 3, 6, 7, 8-11, 12-15**.

Prevent Hypervisor from Using CPUs Set Aside for Pinning

To configure the hypervisor so that it will not use the CPUs identified for CPU Pinning:

Step 1 Open the KVM console for the Compute node.

Step 2 Reset the blade and wait for the Grub bootloader options to be displayed.

Step 3 Type **e** to edit the configuration for the Kernel which is highlighted by default (this is the kernel which is used to boot Linux on the blade).

Step 4 Add the following line to the kernel boot parameters. Based on the above example, you would enter:
set isolcpus=2,3,6,7,8,9,10,11,12,13,14,15

```
Setparams 'Red Hat Enterprise Linux Server (3.10.0-327.el7.x86_64) 7.2 (Maipo)\'
   load_video
   set gfxpayload=keep
   insmod gzio
   insmod part_msdos
   insmod xfs
   set root='hd0,msdos1'
   set iso1cpus=2,3,6,7,8,9,10,11,12,13,14,15
   ...
```

Step 5 Enter Ctrl+X to continue booting Linux.

Step 6 After Linux has finished the boot process, enter the following command to verify the above Kernel boot options:

```
cat /proc/cmdline
```

The iso1cpus options you defined will be displayed, for example:

```
BOOT_IMAGE=/vmlinuz-3.10.0-327.el7.x86_64 root=/dev/mapper/cinder--volumes-slash ro rd.lvm.lv=rhel/swap
 crashkernel=auto rd.lvm.lv=cinder-volumes/slash rhgb quiet LANG=en_US.UTF-8
isolcpus=2,3,6,7,8,9,10,11,12,13,14,15
```

Configure Host Aggregates and Update Flavors

Step 1 Follow the rest of the instructions in the above blog post (refer to Prerequisites, on page 3) to create host-aggregate, add compute hosts to the aggregate and set the CPU pinning metadata.

Step 2 Update Flavors which are NOT used for CPU pinning (non-CPS VM flavors) with the following command:

```
nova flavor-key <id> set "aggregate_instance_extra_specs:pinned"="false"
```

Step 3 Update Flavors which are CPS VM flavors (or a sub-set, if you are planning to bring up only certain VMs in CPS with CPU pinning) with the following command:

```
nova flavor-key <id> set hw:cpu_policy=dedicated
nova flavor-key <id> set aggregate_instance_extra_specs:pinned=true
```

Step 4 Launch a CPS VM with the performance enhanced Flavor. Note the host on which the instance is created and the instance name.
`nova show <id>` will show the host on which the VM is created in the field: `OS-EXT-SRV-ATTR:host`

`nova show <id>` will show the virsh Instance name in the field: `OS-EXT-SRV-ATTR:instance_name`

Step 5 To verify that vCPUs were pinned to the reserved physical CPUs, log in to the Compute node on which the VM is created and run the following command:

```
virsh dumpxml <instance_name>
```

The following section will show the physical CPUs in the field `cpuset` from the list of CPUs that were set aside earlier. For example:

```
<vcpu placement='static'>4</vcpu>
  <cputune>
    <shares>4096</shares>
    <vcpupin vcpu='0' cpuset='11'/>
    <vcpupin vcpu='1' cpuset='3'/>
    <vcpupin vcpu='2' cpuset='2'/>
    <vcpupin vcpu='3' cpuset='10'/>
    <emulatorpin cpuset='2-3,10-11'/>
  </cputune>
```

Configure OpenStack Users and Networks

For more information about keystone commands, refer to the keystone command reference at: http://docs.openstack.org/cli-reference/content/keystoneclient_commands.html

Step 1 Create an OpenStack tenant with the name core (under which you can install the VMs) as shown in the following command:

```
source /root/keystonerc_admin

keystone tenant-create --name "core" --description "PCRF Admin"
```

Step 2 For the above tenant, create an OpenStack user with the name core as shown in the following command:

```
source /root/keystonerc_admin

keystone user-create --name "core" --tenant "core" --pass "Core123" --email "core@cisco.com"
```

Step 3 The tenant must have access to three provider VLANs. In this guide, the names of the VLANs are:

- Internal - Used for inter-VM communication of CPS VMs.
- Gx - Used by CPS to access the PCRF
- Management - Used to access the management functions of the CPS

You can specify any names.

Set up the VLANs as shown in the following table:

Name	Subnet	VLAN-ID	Allocation Pool	Purpose
Internal	Specific to environment, for example:172.xx.xx.0/24	Specific to environment, for example: 20xx	Specific to environment, for example:172.xx.xx.16 to 172.xx.xx.220	The neutron network used by Cisco Policy Suite VMs for internal / private communication.
Management	Specific to environment, for example: 10.xx.xx.0/24	Specific to environment, for example: 20xx	Specific to environment, for example: 10.xx.xx.100 to 10.xx.xx.120	The neutron network used by Cisco Policy Suite VMs to connect on Management network.
Gx	Specific to environment, for example:192.xx.xx.0/24	Specific to environment, for example: 20xx	Specific to environment, for example: 192.xx.xx.16 to 192.xx.xx.220	The neutron network used by Cisco Policy Suite VMs to connect on Gx network.

The following example illustrates how to create networks and subnets:

```
source /root/keystonerc_admin
# $1 network name
# $2 vlan_id
# $3 gw ip
# $4 pool start
# $5 pool end
# $6 subnet x.x.x.x./y
# $OSTACKTENANT  core
#### Networks
echo "neutron net-create "$1" --shared --tenant_id $OSTACKTENANT --provider:network_type "vlan"
--provider:physical_network
"physnet1" --provider:segmentation_id "$2""
neutron net-create "$1" --shared --tenant_id $OSTACKTENANT --provider:network_type
"vlan" --provider:physical_network "physnet1" --provider:segmentation_id "$2"
echo "neutron subnet-create --disable-dhcp --tenant-id $OSTACKTENANT --gateway "$3" --allocation-pool
 "start=$4,end=$5" "$1" "$6""
neutron subnet-create --disable-dhcp --tenant-id $OSTACKTENANT --gateway "$3" --allocation-pool
"start=$4,end=$5" "$1" "$6"
```

All of these networks must be either shared or accessible by Cisco Policy Suite OpenStack network.

Define Availability Zones

Step 1 A core user must have administrator role to launch VMs on specific hosts in OpenStack. Add the administrator role to the core user in the core tenant as shown in the following command:

```
keystone user-role-add --user "core" --role admin --tenant "core"
```

Note The administrator role for the core user is not required if you do not intend to launch the VM on a specific host and if you prefer to allow nova to select the host for the VM.

Step 2 You must define at least one availability zone in OpenStack. Nova hypervisors list will show list of available hypervisors. For example:

```
[root@os24-control]# nova hypervisor-list
+----+--------------------------+
| ID | Hypervisor hostname      |
+----+--------------------------+
| 1  | os24-compute-2.cisco.com |
| 2  | os24-compute-1.cisco.com |
+----+--------------------------+
```

Step 3 Create availability zones specific to your deployment. The following commands provide an example of how to create the availability zones:

```
nova aggregate-create osXX-compute-1 az-1

nova aggregate-add-host osXX-compute-1 osXX-compute-1.cisco.com

nova aggregate-create osXX-compute-2 az-2

nova aggregate-add-host osXX-compute-2 osXX-compute-2.cisco.com
```

Note The above command creates two availability zones az-1 and az-2. You need to specify the zones az-1 or az-2 using Nova boot commands (see Create CPS VMs using Nova Boot Commands, on page 11), or in the Heat environment files (see Create CPS VMs using Heat, on page 16). You can also put more than one compute node in an availability zone. You could create az-1 with both blades, or in a 6-blade system, put three blades in each and then use `az-1:osXX-compute-2.cisco.com` to lock that VM onto that blade.

Availability zone for svn01 volume should be the same as that of pcrfclient01, svn02 volume as that of pcrfclient02, similarly for mongo01 and sessionmgr01, mongo02 and sessionmgr02. The same concept is applicable to cluman – the ISO volume and Cluster Manager (cluman) should be in same zone.

Step 4 Configure the compute nodes to create volumes on availability zones: Edit the `/etc/cinder/cinder.conf` file to add the `storage_availability_zone` parameter below the `[DEFAULT]` line. For example:

```
ssh root@os24-compute-1.cisco.com

 [DEFAULT]
storage_availability_zone=az-1:os24-compute-1.cisco.com
```

For Juno, along with above change, update `storage_availability_zone=nova` to

```
storage_availability_zone=az-1:os24-compute-1.cisco.com
```

After adding the storage availability zone lines in `cinder.conf` file, restart the cinder volume with following command:

```
systemctl restart openstack-cinder-volume
```

Repeat Step 4 for other compute nodes.

Download the ISO Image

Download the CPS ISO image file (CPS_x.x.x.release.iso) for the release from software.cisco.com and load it on the OpenStack control node.

Download the Base Image

Download the base CPS base image file that is available along with the ISO and extract the file as shown in the following command:

```
tar -zxvf CPS_x.x.x_Base.release.tar.gz
```

Locate the base image that is the root disk used by Cisco Policy Suite VM.

Import Images to Glance

Import the Cisco Policy Suite base VM image into the OpenStack glance repository as shown in the following command:

```
source /root/keystonerc_admin

glance image-create --name "base_vm" --is-public "True" --disk-format "vmdk" --container
"bare" --file <name of vmdk file>
```

Import the ISO image by running following command:

```
source /root/keystonerc_admin

glance image-create --name "CPS_x.x.x.release.iso" --is-public "True" --disk-format "iso"
--container "bare" --file <name of iso file>
```

Create Cinder Volumes

Create a cinder volume to map the glance image to the volume. This ensures that the cinder volume (and also the ISO that you imported to glance) can be automatically attached to the Cluster Manager VM when it is launched.

In the core tenant, create and format the following cinder volumes to be attached to various VMs:

- svn01
- svn02
- mongo01
- mongo02
- CPS_*x.x.x*.release.iso

It is recommended you work with Cisco AS to determine the size of each volume.

Note For mongo01 and mongo02, the minimum recommended size is 60 GB.

The following commands illustrate how to create the cinder volumes:

```
source /root/keystonerc_user
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name svn01
--availability-zone az-1:os24-compute-1.cisco.com 2
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name svn02
--availability-zone az-2:os24-compute-2.cisco.com 2
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name mongo01
--availability-zone az-1:os24-compute-1.cisco.com 60
cinder create --metadata fstype=ext4 fslabel=newfs dio=yes --display-name mongo02
--availability-zone az-2:os24-compute-2.cisco.com 60
cps_iso_id=$(glance image-show $cps_iso_name | awk -F '|'  '/id/ {print $3}' | sed 's/ //')
cinder create --display-name $cps_iso_name --image-id $cps_iso_id --availability-zone
az-1:os24-compute-1.cisco.com 3
```

Note

- Replace `$cps_iso_name` with the ISO filename. For example: `CPS_9.0.0.release.iso`
- If any host in the availability zone may be used, then only the zone needs to be specified. Currently, the recommendation only specifies `zone:host`

Verify or Update Default Quotas

OpenStack must have enough Default Quotas (that is, size of RAM, number of vCPUs, number of instances) to spin up all the VMs.

Update the Default Quotas in the following page of the OpenStack dashboard: **Admin** > **Defaults** > **Update Defaults**.

For example:

- instances: 20
- ram: 1024000
- cores: 100

Create Flavors

OpenStack flavors define the virtual hardware templates defining sizes for RAM, disk, vCPUs, and so on.

To create the flavors for your CPS deployment, run the following commands, replacing the appropriate values for your CPS deployment.

```
 source /root/keystonerc_admin
    nova flavor-create --ephemeral 0 pcrfclient01 auto 16384 0 2
    nova flavor-create --ephemeral 0 pcrfclient02 auto 16384 0 2
    nova flavor-create --ephemeral 0 cluman auto 8192 0 4
    nova flavor-create --ephemeral 0 qps auto 10240 0 4
    nova flavor-create --ephemeral 0 sm auto 16384 0 4
    nova flavor-create --ephemeral 0 lb01 auto 8192 0 6
    nova flavor-create --ephemeral 0 lb02 auto 8192 0 6
```

Set up Access and Security

Allow access of the following TCP and UDP ports from the OpenStack dashboard **Project** > **Access & Security** > **default / Manage Rules** or from the CLI as shown in the following example:

```
source /root/keystonerc_user
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
nova secgroup-add-rule default tcp 53 53 0.0.0.0/0
nova secgroup-add-rule default udp 53 53 0.0.0.0/0
nova secgroup-add-rule default tcp 80 80 0.0.0.0/0
nova secgroup-add-rule default tcp 443 443 0.0.0.0/0
nova secgroup-add-rule default tcp 7443 7443 0.0.0.0/0
nova secgroup-add-rule default tcp 8443 8443 0.0.0.0/0
nova secgroup-add-rule default tcp 9443 9443 0.0.0.0/0
nova secgroup-add-rule default tcp 5540 5540 0.0.0.0/0
nova secgroup-add-rule default tcp 1553 1553 0.0.0.0/0
nova secgroup-add-rule default tcp 3868 3868 0.0.0.0/0
nova secgroup-add-rule default tcp 9160 9160 0.0.0.0/0
nova secgroup-add-rule default tcp 27717 27720 0.0.0.0/0
nova secgroup-add-rule default tcp 5432 5432 0.0.0.0/0
nova secgroup-add-rule default tcp 61616 61616 0.0.0.0/0
nova secgroup-add-rule default tcp 9443 9450 0.0.0.0/0
nova secgroup-add-rule default tcp 8280 8290 0.0.0.0/0
nova secgroup-add-rule default tcp  7070 7070 0.0.0.0/0
nova secgroup-add-rule default tcp 8080 8080 0.0.0.0/0
nova secgroup-add-rule default tcp 8090 8090 0.0.0.0/0
nova secgroup-add-rule default tcp 7611 7611 0.0.0.0/0
nova secgroup-add-rule default tcp 7711 7711 0.0.0.0/0
nova secgroup-add-rule default udp  694 694 0.0.0.0/0
nova secgroup-add-rule default tcp 10080 10080 0.0.0.0/0
nova secgroup-add-rule default tcp 11211 11211 0.0.0.0/0
nova secgroup-add-rule default tcp 111 111 0.0.0.0/0
nova secgroup-add-rule default udp 111 111 0.0.0.0/0
nova secgroup-add-rule default tcp 2049 2049 0.0.0.0/0
nova secgroup-add-rule default udp 2049 2049 0.0.0.0/0
nova secgroup-add-rule default tcp 32767 32767 0.0.0.0/0
nova secgroup-add-rule default udp 32767 32767 0.0.0.0/0
nova secgroup-add-rule default tcp 9763 9763 0.0.0.0/0
nova secgroup-add-rule default tcp 8140 8140 0.0.0.0/0
nova secgroup-add-rule default tcp 8161 8161 0.0.0.0/0
nova secgroup-add-rule default tcp 12712 12712 0.0.0.0/0
nova secgroup-add-rule default tcp 9200 9200 0.0.0.0/0
nova secgroup-add-rule default tcp 5060 5060 0.0.0.0/0
nova secgroup-add-rule default udp 5060 5060 0.0.0.0/0
nova secgroup-add-rule default tcp 8458 8458 0.0.0.0/0
nova secgroup-add-rule default udp 8458 8458 0.0.0.0/0
```

Where: default is the name of the security group.

CHAPTER 2

Installation

Installation Overview

Cisco Policy Suite VMs can be deployed using either Nova boot commands or Heat templates.

- Create CPS VMs using Nova Boot Commands, on page 11
- Create CPS VMs using Heat, on page 16

Create CPS VMs using Nova Boot Commands

Step 1 Create the following cloud configuration files for each VM to be deployed (xxx-cloud.cfg). Refer to Sample Cloud Config Files, on page 14 to create these files.

Step 2 Run the following command on the control node:

```
source ~/keystonerc_core
```

Step 3 Deploy each CPS VM with the following nova boot command:

```
nova boot --config-drive true --user-data=<node>-cloud.cfg --image "base_vm" --flavor
"<cluman|pcrfclient0x|sm|lb0x|qns0x>" --nic net-id="<Internal n/w id>,v4-fixed-ip=<Internal network
 private IP>"
--nic net-id="<Management network id>,v4-fixed-ip=<Management n/w public ip>" --block-device-mapping
 "/dev/vdb=<Volume id of iso>:::0" --availability-zone "<availability zone:Host info>" "cluman"
```

Note The networks, internal IPs, management IPs and availability zones in the commands need to be configured according to your environment.

The following example shows the nova boot commands to deploy a Cluster Manager (cluman), 2 pcrfclients, 2 sessionmgrs, 2 loadbalancers, and 4 qns VMs.

In the following example:

- 172.16.2.200 is the Internal vip address.
- 172.18.11.156 is the management vip address.
- 192.168.2.200 is the Gx vip address

```
nova boot --config-drive true --user-data=pcrfclient01-cloud.cfg --image "base_vm" --flavor
"pcrfclient01" --nic net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.20" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.152" --block-device-mapping
"/dev/vdb=139f2b90-eb74-4d5e-9e20-2af3876a7572:::0" --availability-zone "az-1:os8-compute-1.cisco.com"
 "pcrfclient01"

nova boot --config-drive true --user-data=pcrfclient02-cloud.cfg --image "base_vm" --flavor
"pcrfclient02"  --nic net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.21" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.153" --block-device-mapping
"/dev/vdb=27815c35-c5e8-463b-8ce4-fb1ec67d9446:::0" --availability-zone "az-2:os8-compute-2.cisco.com"
 "pcrfclient02"

nova boot --config-drive true --user-data=sessionmgr01-cloud.cfg --image "base_vm" --flavor "sm"
--nic net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.22" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.157"  --block-device-mapping
"/dev/vdb=8c3577d2-74f2-4370-9a37-7370381670e4:::0" --availability-zone "az-1:os8-compute-1.cisco.com"
 "sessionmgr01"

nova boot --config-drive true --user-data=sessionmgr02-cloud.cfg  --image "base_vm" --flavor "sm"
--nic net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.23" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.158" --block-device-mapping
"/dev/vdb=67aa5cbd-02dd-497e-a8ee-797ac04b85f0:::0" --availability-zone "az-2:os8-compute-2.cisco.com"
 "sessionmgr02"

nova boot --config-drive true --user-data=lb01-cloud.cfg --image "base_vm" --flavor "lb01" --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.201" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.154" --nic
net-id="d0a69b7f-5d51-424a-afbe-5f6486c6e90d,v4-fixed-ip=192.168.2.201" --availability-zone
"az-1:os8-compute-1.cisco.com" "lb01"

nova boot --config-drive true --user-data=lb02-cloud.cfg --image "base_vm" --flavor "lb02" --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.202" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.155" --nic
net-id="d0a69b7f-5d51-424a-afbe-5f6486c6e90d,v4-fixed-ip=192.168.2.202" --availability-zone
"az-2:os8-compute-2.cisco.com" "lb02"

nova boot --config-drive true --user-data=qns01-cloud.cfg --image "base_vm" --flavor "qps" --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.24" --availability-zone
"az-1:os8-compute-1.cisco.com" "qns01"

nova boot --config-drive true --user-data=qns02-cloud.cfg --image "base_vm" --flavor "qps" --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.25" --availability-zone
```

```
"az-1:os8-compute-1.cisco.com" "qns02"

nova boot --config-drive true --user-data=qns03-cloud.cfg --image "base_vm" --flavor "qps" --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.26" --availability-zone
"az-2:os8-compute-2.cisco.com" "qns03"

nova boot --config-drive true --user-data=qns04-cloud.cfg --image "base_vm" --flavor "qps" --nic
net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.27" --availability-zone
"az-2:os8-compute-2.cisco.com" "qns04"
```

Step 4 Allow VIPs on the neutron ports:

a) Get the neutron port ID for the lb01 internal IP address:

```
neutron port-list | grep "<lb01_internal_IP>"
```

b) Get the neutron port ID for the lb02 internal IP address:

```
neutron port-list | grep "<lb02_internal_IP>"
```

c) Update the above two neutron ports to allow Internal VIP address by running the following command for each of the above ports:

```
neutron port-update <id> --allowed-address-pairs type=dict list=true ip_address=<Internal VIP
address>
```

For example:

```
[root@os8-control cloud(keystone_core)]# neutron port-list | grep "172.16.2.201"
| db8944f3-407d-41ef-b063-eabbab43c039 |      | fa:16:3e:30:17:02 | {"subnet_id":
"afa2a847-0241-4363-8b0e-88ec3dcaf13b", "ip_address": "172.16.2.201"}  |
[root@os8-control cloud(keystone_core)]# neutron port-update db8944f3-407d-41ef-b063-eabbab43c039
 --allowed-address-pairs type=dict list=true ip_address=172.16.2.200
Updated port: db8944f3-407d-41ef-b063-eabbab43c039
[root@os8-control cloud(keystone_core)]# neutron port-list | grep "172.16.2.202"
| 741e535c-1e27-4af6-8f89-1641fa3420c9 |      | fa:16:3e:ab:bd:3e | {"subnet_id":
"afa2a847-0241-4363-8b0e-88ec3dcaf13b", "ip_address": "172.16.2.202"}  |
[root@os8-control cloud(keystone_core)]# neutron port-update 741e535c-1e27-4af6-8f89-1641fa3420c9
 --allowed-address-pairs type=dict list=true ip_address=172.16.2.200
Updated port: 741e535c-1e27-4af6-8f89-1641fa3420c9
[root@os8-control cloud(keystone_core)]# neutron port-list | grep "172.18.11.154"
| 45bf8eeb-664d-48a8-9f7d-a61f951c0cf4 |      | fa:16:3e:8f:ed:8d | {"subnet_id":
"e153de30-aab3-4c8d-83f8-662d6d5eaca8", "ip_address": "172.18.11.154"} |
[root@os8-control cloud(keystone_core)]# neutron port-update 45bf8eeb-664d-48a8-9f7d-a61f951c0cf4
 --allowed-address-pairs type=dict list=true ip_address=172.18.11.156
Updated port: 45bf8eeb-664d-48a8-9f7d-a61f951c0cf4
[root@os8-control cloud(keystone_core)]# neutron port-list | grep "172.18.11.155"
| 67e37edb-581f-49a2-9eca-2bedaf81a466 |      | fa:16:3e:d2:61:58 | {"subnet_id":
"e153de30-aab3-4c8d-83f8-662d6d5eaca8", "ip_address": "172.18.11.155"} |
[root@os8-control cloud(keystone_core)]# neutron port-list | grep "192.168.2.201"
| 270b4526-bef7-48f0-a24d-4be8f3bad8dc |      | fa:16:3e:05:4c:f1 | {"subnet_id":
"4c4fc222-f690-4ca1-a889-9b3842f8b060", "ip_address": "192.168.2.201"} |
[root@os8-control cloud(keystone_core)]# neutron port-update 270b4526-bef7-48f0-a24d-4be8f3bad8dc
 --allowed-address-pairs type=dict list=true ip_address=192.168.2.200
Updated port: 270b4526-bef7-48f0-a24d-4be8f3bad8dc
[root@os8-control cloud(keystone_core)]# neutron port-list | grep "192.168.2.202"
| 671ba09b-6e4b-4399-9ad7-ed2d81829c87 |      | fa:16:3e:ed:27:98 | {"subnet_id":
"4c4fc222-f690-4ca1-a889-9b3842f8b060", "ip_address": "192.168.2.202"} |
[root@os8-control cloud(keystone_core)]# neutron port-update 671ba09b-6e4b-4399-9ad7-ed2d81829c87
```

```
 --allowed-address-pairs type=dict list=true ip_address=192.168.2.200
Updated port: 671ba09b-6e4b-4399-9ad7-ed2d81829c87
```

d) Repeat Step c for External VIP address using neutron ports for the lb01/lb02 Management IP address and also Gx VIP address using neutron ports for lb01/lb02 Gx IP addresses.

Step 5 Wait approximately 10 minutes for the Cluster Manager VM to be deployed, then check the readiness status of the Cluster Manager VM using the following API:
GET http://<*Cluster Manager IP*>:8458/api/system/status/cluman

Refer to /api/system/status/cluman, on page 37 for more information.

When this API responds that the Cluster Manager VM is in a ready state (`"status": "ready"`), continue with Deploy CPS, on page 33.

Refer also to the `/var/log/cloud-init-output.log` on the Cluster Manager VM for deployment details.

Sample Cloud Config Files

For nova boot installation of CPS, you must create a cloud configuration file for each CPS VM to be deployed.

The following sections show an example Cluster Manager cloud configuration (`cluman-cloud.cfg`), and a pcrflient01 cloud configuration (`pcrfclient01-cloud.cfg`).

These files must be placed in the directory in which you execute the nova launch commands, typically `/root/cps-install/`.

Cluster Manager Configuration File

```
#cloud-config
write_files:
 - path: /etc/sysconfig/network-scripts/ifcfg-eth0
   encoding: ascii
   content: |
     DEVICE=eth0
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.16.2.19    <---- Internal IP to access via private IP
     NETMASK=255.255.255.0
     GATEWAY=172.16.2.1    <----- Internal network Gateway
     NETWORK=172.16.2.0    <------ Internal network
   owner: root:root
   permissions: '0644'
 - path: /etc/sysconfig/network-scripts/ifcfg-eth1
   encoding: ascii
   content: |
     DEVICE=eth1
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.18.11.101   <---- Management IP to access via public IP
     NETMASK=255.255.255.0
     GATEWAY=172.18.11.1
     NETWORK=172.18.11.0
   owner: root:root
   permissions: '0644'
 - path: /var/lib/cloud/instance/payload/launch-params
   encoding: ascii
   owner: root:root
   permissions: '0644'
 - path: /root/.autoinstall.sh
   encoding: ascii
   content: |
     #!/bin/bash
     if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
       /mnt/iso/install.sh << EOF
     mobile
     y
     1
     EOF
     fi
   permissions: '0755'
mounts:
 - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
runcmd:
 - ifdown eth0
 - ifdown eth1
 - echo 172.16.2.19 installer >> /etc/hosts   <---- Internal/private IP of cluman
 - ifup eth0
 - ifup eth1
 - /root/.autoinstall.sh
```

Non-Cluster Manager Configuration File

- The following example configuration file is for pcrfclient01. You must create separate configuration files for each CPS VM to be deployed.

  For each file, modify the `NODE_TYPE`, and network settings (`IPADDR`, `GATEWAY`, `NETWORK`) accordingly.

  A typical CPS deployment would require the following files:

  ◦
    - pcrfclient01-cloud.cfg
    - pcrfclient02-cloud.cfg
    - lb01-cloud.cfg
    - lb02-cloud.cfg
    - sessionmgr01-cloud.cfg
    - sessionmgr02-cloud.cfg
    - qns01-cloud.cfg
    - qns02-cloud.cfg
    - qns03-cloud.cfg
    - qns04-cloud.cfg

- Modify `IPADDR` to the IP address used in nova boot command for that interface.
- Set `NETMASK`, `GATEWAY`, and `NETWORK` according to your environment.

```
#cloud-config
hostname: pcrfclient01
write_files:
 - path: /etc/sysconfig/network-scripts/ifcfg-eth0
   encoding: ascii
   content: |
     DEVICE=eth0
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.16.2.20
     NETMASK=255.255.255.0
     GATEWAY=172.16.2.1
     NETWORK=172.16.2.0
   owner: root:root
   permissions: '0644'
 - path: /etc/sysconfig/network-scripts/ifcfg-eth1
```

```
    encoding: ascii
    content: |
      DEVICE=eth1
      BOOTPROTO=none
      NM_CONTROLLED=none
      IPADDR=172.18.11.152
      NETMASK=255.255.255.0
      GATEWAY=172.18.11.1
      NETWORK=172.18.11.0
    owner: root:root
    permissions: '0644'
  - path: /var/lib/cloud/instance/payload/launch-params
    encoding: ascii
    owner: root:root
    permissions: '0644'
  - path: /etc/broadhop.profile
    encoding: ascii
    content: "NODE_TYPE=pcrfclient01\n"
    owner: root:root
    permissions: '0644'
runcmd:
  - ifdown eth0
  - ifdown eth1
  - echo 172.16.2.19 installer >> /etc/hosts
  - ifup eth0
  - ifup eth1
  - sed -i '/^HOSTNAME=/d' /etc/sysconfig/network && echo HOSTNAME=pcrfclient01 >>
/etc/sysconfig/network
```

Create CPS VMs using Heat

To create the CPS VMs using OpenStack Heat, you must first create an environment file and a Heat template containing information for your deployment.

These files include information about the ISO, base image, availability zones, management IPs, and volumes. Modify the sample files provided below with information for your deployment.

- Sample Heat Template File, on page 17
- Sample Heat Environment File, on page 16

After populating these files, continue with Create Heat Stack, on page 31

Sample Heat Environment File

Note Update the network/vlan names, internal and management IPs, VIPs, and volumes for your environment.

`az-1`, `az-2` shown in the following sample are for example purposes only. Update these for your environment accordingly.

Also update the heat template (hot-cps.yaml) with your availability zone variables (for example: `cps_az_1`, `cps_az_2`) after updating this heat environment file.

```
# cat hot-cps.env
# This is an example environment file

parameters:
  cps_iso_image_name: CPS_9.0.0.release.iso
  base_vm_image_name: CPS_9.0.0_Base.release
  cps_az_1: az-1
  cps_az_2: az-2

  internal_net_name: internal
  internal_net_cidr: 172.16.2.0/24

  management_net_name: management
  management_net_cidr: 172.18.11.0/24
  management_net_gateway: 172.18.11.1

  gx_net_name: gx
  gx_net_cidr: 192.168.2.0/24

  cluman_flavor_name: cluman
  cluman_internal_ip: 172.16.2.19
  cluman_management_ip: 172.18.11.151

  lb_internal_vip: 172.16.2.200
  lb_management_vip: 172.18.11.156
  lb_gx_vip: 192.168.2.200
  lb01_flavor_name: lb01
  lb01_internal_ip: 172.16.2.201
  lb01_management_ip: 172.18.11.154
  lb01_gx_ip: 192.168.2.201
  lb02_flavor_name: lb02
  lb02_internal_ip: 172.16.2.202
  lb02_management_ip: 172.18.11.155
  lb02_gx_ip: 192.168.2.202

  pcrfclient01_flavor_name: pcrfclient01
  pcrfclient01_internal_ip: 172.16.2.20
  pcrfclient01_management_ip: 172.18.11.152
  pcrfclient02_flavor_name: pcrfclient02
  pcrfclient02_internal_ip: 172.16.2.21
  pcrfclient02_management_ip: 172.18.11.153

  qns01_internal_ip: 172.16.2.24
  qns02_internal_ip: 172.16.2.25
  qns03_internal_ip: 172.16.2.26
  qns04_internal_ip: 172.16.2.27

  sessionmgr01_internal_ip: 172.16.2.22
  sessionmgr01_management_ip: 172.18.11.157
  sessionmgr02_internal_ip: 172.16.2.23
  sessionmgr02_management_ip: 172.18.11.158

  mongo01_volume_id: "54789405-f683-401b-8194-c354d8937ecb"
  mongo02_volume_id: "9694ab92-8ddd-407e-8520-8b0280f5db03"
  svn01_volume_id: "5b6d7263-40d1-4748-b45c-d1af698d71f7"
  svn02_volume_id: "b501f834-eff9-4044-90c3-a24378f3734d"
  cps_iso_volume_id: "ef52f944-411b-42b1-b86a-500950f5b398"
```

Sample Heat Template File

Note Update the following sample heat template according to your environment, such as to add more VMs, networks to the VMs, and so on.

```
#cat hot-cps.yaml
heat_template_version: 2014-10-16

description: A minimal CPS deployment for big bang deployment

parameters:
#=========================
# Global Parameters
#=========================
  base_vm_image_name:
    type: string
    label: base vm image name
    description: name of the base vm as imported into glance
  cps_iso_image_name:
    type: string
    label: cps iso image name
    description: name of the cps iso as imported into glance
  cps_install_type:
    type: string
    label: cps installation type (mobile|wifi|mog|arbiter)
    description: cps installation type (mobile|wifi|mog|arbiter)
    default: mobile
  cps_az_1:
    type: string
    label: first availability zone
    description: az for "first half" of cluster
    default: nova
  cps_az_2:
    type: string
    label: second availability zone
    description: az for "second half" of cluster
    default: nova

#=========================
# Network Parameters
#=========================
  internal_net_name:
    type: string
    label: internal network name
    description: name of the internal network
  internal_net_cidr:
    type: string
    label: cps internal cidr
    description: cidr of internal subnet

  management_net_name:
    type: string
    label: management network name
    description: name of the management network
  management_net_cidr:
    type: string
    label: cps management cidr
    description: cidr of management subnet
  management_net_gateway:
    type: string
    label: management network gateway
    description: gateway on management network
    default: ""

  gx_net_name:
    type: string
    label: gx network name
    description: name of the gx network
  gx_net_cidr:
    type: string
    label: cps gx cidr
    description: cidr of gx subnet
  gx_net_gateway:
    type: string
    label: gx network gateway
    description: gateway on gx network
    default: ""

  cps_secgroup_name:
    type: string
    label: cps secgroup name
    description: name of cps security group
    default: cps_secgroup

#=========================
# Volume Parameters
#=========================
  mongo01_volume_id:
    type: string
    label: mongo01 volume id
    description: uuid of the mongo01 volume

  mongo02_volume_id:
    type: string
    label: mongo02 volume id
    description: uuid of the mongo02 volume

  svn01_volume_id:
    type: string
    label: svn01 volume id
    description: uuid of the svn01 volume

  svn02_volume_id:
    type: string
    label: svn02 volume id
    description: uuid of the svn02 volume

  cps_iso_volume_id:
    type: string
    label: cps iso volume id
    description: uuid of the cps iso volume

#==========================
# Instance Parameters
#==========================
  cluman_flavor_name:
    type: string
    label: cluman flavor name
    description: flavor cluman vm will use
    default: cluman
  cluman_internal_ip:
    type: string
    label: internal ip of cluster manager
    description: internal ip of cluster manager
  cluman_management_ip:
    type: string
    label: management ip of cluster manager
    description: management ip of cluster manager

  lb_internal_vip:
    type: string
    label: internal vip of load balancer
    description: internal vip of load balancer
  lb_management_vip:
    type: string
    label: management vip of load balancer
    description: management vip of load balancer
  lb_gx_vip:
    type: string
    label: gx ip of load balancer
    description: gx vip of load balancer
  lb01_flavor_name:
    type: string
    label: lb01 flavor name
    description: flavor lb01 vms will use
    default: lb01
  lb01_internal_ip:
    type: string
    label: internal ip of load balancer
    description: internal ip of load balancer
  lb01_management_ip:
    type: string
    label: management ip of load balancer
    description: management ip of load balancer
  lb01_gx_ip:
    type: string
    label: gx ip of load balancer
    description: gx ip of load balancer
  lb02_flavor_name:
    type: string
    label: lb02 flavor name
    description: flavor lb02 vms will use
    default: lb02
  lb02_internal_ip:
    type: string
    label: internal ip of load balancer
    description: internal ip of load balancer
  lb02_management_ip:
    type: string
    label: management ip of load balancer
    description: management ip of load balancer
  lb02_gx_ip:
    type: string
    label: gx ip of load balancer
    description: gx ip of load balancer

  pcrfclient01_flavor_name:
    type: string
    label: pcrfclient01 flavor name
    description: flavor pcrfclient01 vm will use
    default: pcrfclient01
  pcrfclient01_internal_ip:
    type: string
    label: internal ip of pcrfclient01
    description: internal ip of pcrfclient01
  pcrfclient01_management_ip:
    type: string
    label: management ip of pcrfclient01
    description: management ip of pcrfclient01
  pcrfclient02_flavor_name:
    type: string
    label: pcrfclient02 flavor name
    description: flavor pcrfclient02 vm will use
    default: pcrfclient02
  pcrfclient02_internal_ip:
    type: string
    label: internal ip of pcrfclient02
    description: internal ip of pcrfclient02
  pcrfclient02_management_ip:
    type: string
    label: management ip of pcrfclient02
    description: management ip of pcrfclient02

  qns_flavor_name:
    type: string
    label: qns flavor name
    description: flavor qns vms will use
    default: qps
  qns01_internal_ip:
    type: string
    label: internal ip of qns01
    description: internal ip of qns01
  qns02_internal_ip:
    type: string
    label: internal ip of qns02
    description: internal ip of qns02
  qns03_internal_ip:
    type: string
    label: internal ip of qns03
    description: internal ip of qns03
  qns04_internal_ip:
    type: string
    label: internal ip of qns04
    description: internal ip of qns04


  sessionmgr_flavor_name:
    type: string
    label: sessionmgr flavor name
    description: flavor sessionmgr vms will use
    default: sm
  sessionmgr01_internal_ip:
    type: string
    label: internal ip of sessionmgr01
```

```
    description: internal ip of sessionmgr01
  sessionmgr01_management_ip:
    type: string
    label: management ip of sessionmgr01
    description: management ip of sessionmgr01
  sessionmgr02_internal_ip:
    type: string
    label: internal ip of sessionmgr02
    description: internal ip of sessionmgr02
  sessionmgr02_management_ip:
    type: string
    label: management ip of sessionmgr02
    description: management ip of sessionmgr02

resources:
#==========================
# Instances
#==========================

  cluman:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: cluman_flavor_name }
      networks:
        - port: { get_resource: cluman_internal_port }
        - port: { get_resource: cluman_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: cps_iso_volume_id }
      user_data_format: RAW
      user_data: { get_resource: cluman_config }
  cluman_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_internal_ip }}]
  cluman_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_management_ip }}]
  cluman_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
            permissions: "0644"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            permissions: "0644"
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: cluman_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            permissions: "0644"
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
```

```
                $ip: { get_param: cluman_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /root/.autoinstall.sh
          permissions: "0755"
          content:
            str_replace:
              template: |
                #!/bin/bash
                if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
                /mnt/iso/install.sh << EOF
                $install_type
                y
                1
                EOF
                fi
              params:
                $install_type: { get_param: cps_install_type }
      mounts:
        - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=cluman >> /etc/sysconfig/network
        - hostname cluman
        - /root/.autoinstall.sh

lb01:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: lb01_flavor_name }
    networks:
      - port: { get_resource: lb01_internal_port }
      - port: { get_resource: lb01_management_port }
      - port: { get_resource: lb01_gx_port }
    user_data_format: RAW
    user_data: { get_resource: lb01_config }
lb01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_internal_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_internal_vip }
lb01_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_management_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_management_vip }
lb01_gx_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: gx_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_gx_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_gx_vip }
lb01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=lb01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: lb01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb01_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth2
          content:
            str_replace:
              template: |
                DEVICE=eth2
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb01_gx_ip }
                $gateway: { get_param: gx_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
            params:
              $cidr: { get_param: gx_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - ifdown eth2 && ifup eth2
        - echo HOSTNAME=lb01 >> /etc/sysconfig/network
        - hostname lb01

lb02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: lb02_flavor_name }
    networks:
      - port: { get_resource: lb02_internal_port }
```

```
        - port: { get_resource: lb02_management_port }
        - port: { get_resource: lb02_gx_port }
      user_data_format: RAW
      user_data: { get_resource: lb02_config }
  lb02_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: lb02_internal_ip }}]
      allowed_address_pairs:
        - ip_address: { get_param: lb_internal_vip }
  lb02_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: lb02_management_ip }}]
      allowed_address_pairs:
        - ip_address: { get_param: lb_management_vip }
  lb02_gx_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: gx_net_name }
      fixed_ips: [{ ip_address: { get_param: lb02_gx_ip }}]
      allowed_address_pairs:
        - ip_address: { get_param: lb_gx_vip }
  lb02_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=lb02\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: lb02_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: lb02_management_ip }
                  $gateway: { get_param: management_net_gateway }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth2
            content:
              str_replace:
                template: |
                  DEVICE=eth2
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: lb02_gx_ip }
                  $gateway: { get_param: gx_net_gateway }
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
              params:
                $cidr: { get_param: gx_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - ifdown eth2 && ifup eth2
          - echo HOSTNAME=lb02 >> /etc/sysconfig/network
          - hostname lb02

  pcrfclient01:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: pcrfclient01_flavor_name }
      networks:
        - port: { get_resource: pcrfclient01_internal_port }
        - port: { get_resource: pcrfclient01_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: svn01_volume_id }
      user_data_format: RAW
      user_data: { get_resource: pcrfclient01_config }
  pcrfclient01_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: pcrfclient01_internal_ip }}]
  pcrfclient01_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: pcrfclient01_management_ip }}]
  pcrfclient01_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=pcrfclient01\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: pcrfclient01_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: pcrfclient01_management_ip }
                  $gateway: { get_param: management_net_gateway }
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=pcrfclient01 >> /etc/sysconfig/network
          - hostname pcrfclient01

  pcrfclient02:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_2 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: pcrfclient02_flavor_name }
      networks:
        - port: { get_resource: pcrfclient02_internal_port }
        - port: { get_resource: pcrfclient02_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: svn02_volume_id }
      user_data_format: RAW
      user_data: { get_resource: pcrfclient02_config }
  pcrfclient02_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: pcrfclient02_internal_ip }}]
  pcrfclient02_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: pcrfclient02_management_ip }}]
  pcrfclient02_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=pcrfclient02\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: pcrfclient02_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: pcrfclient02_management_ip }
                  $gateway: { get_param: management_net_gateway }
        runcmd:
```

```
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=pcrfclient02 >> /etc/sysconfig/network
        - hostname pcrfclient02

qns01:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns01_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns01_config }
qns01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns01_internal_ip }}]
qns01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns01_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns01 >> /etc/sysconfig/network
        - hostname qns01

qns02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns02_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns02_config }
qns02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns02_internal_ip }}]
qns02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns02_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns02 >> /etc/sysconfig/network
        - hostname qns02

qns03:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns03_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns03_config }
qns03_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns03_internal_ip }}]
qns03_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns03\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns03_internal_ip }
      runcmd:
```

```
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns03 >> /etc/sysconfig/network
        - hostname qns03

qns04:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns04_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns04_config }
qns04_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns04_internal_ip }}]
qns04_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns04\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns04_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns04 >> /etc/sysconfig/network
        - hostname qns04

sessionmgr01:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: sessionmgr_flavor_name }
    networks:
      - port: { get_resource: sessionmgr01_internal_port }
      - port: { get_resource: sessionmgr01_management_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: mongo01_volume_id }
    user_data_format: RAW
```

```
      user_data: { get_resource: sessionmgr01_config }
  sessionmgr01_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_internal_ip }}]
  sessionmgr01_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_management_ip }}]
  sessionmgr01_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=sessionmgr01\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: sessionmgr01_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: sessionmgr01_management_ip }
                  $gateway: { get_param: management_net_gateway }
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=sessionmgr01 >> /etc/sysconfig/network
          - hostname sessionmgr01

  sessionmgr02:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_2 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: sessionmgr_flavor_name }
      networks:
        - port: { get_resource: sessionmgr02_internal_port }
        - port: { get_resource: sessionmgr02_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: mongo02_volume_id }
      user_data_format: RAW
      user_data: { get_resource: sessionmgr02_config }
  sessionmgr02_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr02_internal_ip }}]
  sessionmgr02_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr02_management_ip }}]
  sessionmgr02_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=sessionmgr02\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: sessionmgr02_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: sessionmgr02_management_ip }
                  $gateway: { get_param: management_net_gateway }
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=sessionmgr02 >> /etc/sysconfig/network
          - hostname sessionmgr02
```

Create Heat Stack

Before beginning, verify you have populated your information in the environment (.env) file and heat template (.yaml) file and loaded both files on the control node.

Step 1 Run the following command on control node at the location where your environment and heat template files are located:

```
source ~/keystonerc_core
```

Step 2 Add/assign the heat stack owner to core tenant user:

```
keystone user-role-add --user=core --tenant=core --role=heat_stack_owner
```

Step 3 Verify that no existing CPS stack is present:

```
[root@os8-control ~(keystone_core)]# heat stack-list

+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
+--------------------------------------+------------+-----------------+----------------------+
```

Step 4 Create the stack using the heat template (hot-cps.yaml) and environment file (hot-cps.env) you populated earlier.

```
[root@os8-control mbuild(keystone_core)]# heat stack-create --environment-file hot-cps.env
--template-file hot-cps.yaml cps
+--------------------------------------+------------+--------------------+----------------------+
| id                                   | stack_name | stack_status       | creation_time        |
+--------------------------------------+------------+--------------------+----------------------+
| 3f1ab6c2-673d-47b3-ae01-8946cac9e9e9 | cps        | CREATE_IN_PROGRESS | 2016-03-03T16:58:53Z |
+--------------------------------------+------------+--------------------+----------------------+
```

Step 5 Check the status using the `heat stack-list` command:

```
[root@os8-control mbuild(keystone_core)]# heat stack-list
+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
| 3f1ab6c2-673d-47b3-ae01-8946cac9e9e9 | cps        | CREATE_COMPLETE | 2016-01-19T16:58:53Z |
+--------------------------------------+------------+-----------------+----------------------+
```

`CREATE_COMPLETE` will be reported when the heat stack is finished.

Step 6 Wait approximately 10 minutes for the Cluster Manager VM to be deployed, then check the readiness status of the Cluster Manager VM using the following API:
GET http://<*Cluster Manager IP*>:8458/api/system/status/cluman

Refer to /api/system/status/cluman, on page 37 for more information.

When this API responds that the Cluster Manager VM is in a ready state (`"status": "ready"`), continue with Deploy CPS, on page 33.

Refer also to the `/var/log/cloud-init-output.log` on the Cluster Manager VM for deployment details.

Deploy CPS

The following steps outline how to create a consolidated CPS configuration file and use the CPS platform orchestration APIs to deploy the CPS VMs on OpenStack:

Step 1 Create a consolidated CPS configuration file. This file contains all the information necessary to deploy VMs in the CPS cluster, including a valid CPS license key. Contact your Cisco representative to receive the CPS license key for your deployment.

a) Refer to Sample YAML Configuration File, on page 46 for a sample CPS configuration to use as a template.

b) Refer to Configuration Parameters, on page 40 for a description of all parameters within this file.

Step 2 Load the consolidated configuration file you created in Step 1 using the following API:
POST http://<*Cluster Manager IP*>:8458/api/system/config/

Refer to /api/system/config/, on page 38 for more information.

Step 3 (Optional) To confirm the configuration was loaded properly onto the Cluster Manager VM, perform a GET with the same API:
GET http://<*Cluster Manager IP*>:8458/api/system/config/

Step 4 Apply the configuration using the following API:
POST http://<*Cluster Manager IP*>:8458/api/system/config/apply

Refer to Apply the Loaded Configuration, on page 39 for more information.

This API applies the CPS configuration file, triggers the Cluster Manager VM to deploy and bring up all CPS VMs, and performs all post-installation steps.

Validate CPS Deployment

Step 1 To monitor the status of the deployment, use the following API:
GET http://<*Cluster Manager IP*>:8458/api/system/config/status

Refer to /api/system/config/status , on page 51 for more information.

Step 2 After the deployment has completed, verify the readiness of the entire CPS cluster using the following API:
GET http://<*Cluster Manager IP*>:8458/api/system/status/cps

Refer to /api/system/status/cps, on page 53 for more information.

Step 3 Connect to the Cluster Manager and issue the following command to run a set of diagnostics and display the current state of the system.

```
/var/qps/bin/diag/diagnostics.sh
```

SR-IOV Support

CPS supports single root I/O virtualization (SR-IOV) on Intel NIC adapters.

CPS also supports bonding of SR-IOV sub-interfaces for seamless traffic switchover.

The Intel SR-IOV implementation includes anti-spoofing support that will not allow MAC addresses other than the one configured in the VF to communicate. As a result, the active failover mac policy is used. Further, it is recommended that VLAN interfaces are created directly on top of the VF interfaces (for example eth0.123 and eth1.123) and then those interfaces are bonded. If VLAN interfaces are created on top of a bond, their MAC address will not follow the bonds when a failover occurs and the old MAC will be used for the new active interface. This may result in failover not working as expected and so this configuration should be avoided.

The following sample configuration shows the bonding of two interfaces using a single IP address:

```
[root@qns0x ~]# cat /proc/net/bonding/bond0310
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: None
Currently Active Slave: eth1.310
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1.310
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:aa:a5:c8
Slave queue ID: 0

Slave Interface: eth2.310
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:26:e3:9e
Slave queue ID: 0
[root@qns02 ~]# cat /proc/net/bonding/bond0736
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: None
Currently Active Slave: eth1.736
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1.736
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:aa:a5:c8
Slave queue ID: 0

Slave Interface: eth2.736
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:26:e3:9e
```

```
Slave queue ID: 0

[root@qns0x ~]# more /etc/sysconfig/network-scripts/ifcfg-*
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond0310
::::::::::::::
DEVICE=bond0310
BONDING_OPTS="mode=active-backup miimon=100 fail_over_mac=1"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV6INIT=no
IPADDR=172.16.255.11
NETMASK=255.255.255.192
NETWORK=172.16.255.0
IPV4_FAILURE_FATAL=no
IPV6INIT=no
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
ONBOOT=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond0736
::::::::::::::
DEVICE=bond0736
BONDING_OPTS="mode=active-backup miimon=100 fail_over_mac=1"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV6INIT=yes
IPV6ADDR=fd00:4888:1000:30c2::23/64
IPV6_DEFAULTGW=fd00:4888:1000:30c2::1
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
ONBOOT=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth0
::::::::::::::
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.66.34
NETMASK=255.255.255.0
GATEWAY=192.168.66.1
NETWORK=192.168.66.0
IPV6INIT=yes
IPV6ADDR=fd00:4888:1000:f000::aab1/64
IPV6_DEFAULTGW=fd00:4888:1000:f000::1
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1
::::::::::::::
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=none
USRCTL=no
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1.310
::::::::::::::
DEVICE=eth1.310
```

```
ONBOOT=yes
MASTER=bond0310
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1.736
::::::::::::::
DEVICE=eth1.736
ONBOOT=yes
MASTER=bond0736
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2
::::::::::::::
DEVICE=eth2
ONBOOT=yes
BOOTPROTO=none
USRCTL=no
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2.310
::::::::::::::
DEVICE=eth2.310
ONBOOT=yes
MASTER=bond0310
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2.736
::::::::::::::
DEVICE=eth2.736
ONBOOT=yes
MASTER=bond0736
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
```

CHAPTER 3

Orchestration API

Installation APIs

/api/system/status/cluman

Purpose

This API returns the readiness status of the Cluster Manager VM.

Cluster Manager VM Readiness

If `/mnt/iso/install.sh` is executing, the status is returned as 'not ready'.

If `/mnt/iso/install.sh` has completed executing, status is returned as 'ready'.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/status/cluman
- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** JSON
- **Response:** 200 OK: success

  The following example shows the status reported for a new CPS deployment:

  ```
  {
      "status": "ready",
  }
  ```

API logs are at written to: `/var/log/orchestration-api-server.log`

/api/system/config/

Purpose

This API is used to load an initial configuration or return (GET) the current CPS cluster configuration.

This API is also used to apply the loaded configuration to all VMs within the CPS cluster.

API logs are at written to: `/var/log/orchestration-api-server.log`

Retrieve the Current Configuration

To retrieve (GET) the current CPS cluster configuration that is loaded on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/
- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200: OK.

  Example Response (No Configuration Present) XML:

  ```
  ---
  configVersion: null
  hosts: null
  vlans: null
  additionalHosts: null
  config: null
  licenses: null
  replicaSets: null
  ```

  For a response showing an example configuration file refer to Sample YAML Configuration File, on page 46.

Load a Configuration

Note This API can only be used once for initial deployment. Once a configuration has been applied (/system/config/apply) as described below, this API is no longer available.

Note Before loading the configuration file to your CPS cluster, verify that the YAML file uses the proper syntax. There are many publicly-available websites which you can use to validate your YAML configuration file.

Note When this API is issued, the following basic validations are performed on the consolidated configuration (YAML) file submitted in the payload:

- The replica set hosts are included in hosts or additionalHosts section
- Standard CPS aliases are present (lb01, lb02, and so on)
- Standard CPS vlan names are present (Internal, Management, and so on)
- Range checking (for example, IPv4/IPv6 IP address syntax validation)
- Cross-referencing of vlans with hosts

If a validation error is detected, an appropriate message is provided in the API response, and reported in `/var/log/orchestration-api-server.log`.

To load a new CPS cluster configuration on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/
- **Header:** Content-Type: application/yaml
- **Method:** POST
- **Payload:** Include the YAML configuration file in the POST. Refer to Sample YAML Configuration File, on page 46 for more information about this configuration file.
- **Response:** 200: success; 400: malformed or invalid; 403: Configuration may not be changed at this time (for example, after it has been applied).

To verify the configuration was properly loaded, perform another GET to http://<*Cluster Manager IP*>:8458/api/system/config/

Apply the Loaded Configuration

Note This API can only be used once for initial deployment. After a configuration has been applied, the API is no longer available.

Once a new configuration file has been uploaded to the Cluster Manager VM, you must apply the configuration. This triggers the Cluster Manager VM prepare and push out the new configurations to all VMs in the cluster, as well as perform any post-update steps.

During an initial deployment of a CPS cluster, the CPS VMs in the cluster will remain in an inactive/waiting state until this configuration file is applied.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/action/apply
- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** There is no payload.
- **Response:** 200: success; 400: malformed or invalid; 403: Configuration may not be applied at this time; 500: System error. See logs.

To check the status of the CPS cluster after applying a configuration, refer to /api/system/config/status , on page 51.

Configuration Parameters

The following parameters can be defined in the CPS configuration file. Refer also to: Sample YAML Configuration File, on page 46.

In this file, the Internal, Management and Gx networks must have an exact case match of "Internal", "Management" and " Gx" in the following sections:

- hosts: interfaces: value of "network"
- vlans: value of "name"

All parameters and values are case sensitive.

Note Before loading the configuration file to your CPS cluster, verify that the YAML file uses the proper syntax. There are many publicly-available websites which you can use to validate your YAML configuration file.

Parameter	Description
`configVersion`	The version of the configuration file. Release 9.0.0 should be set to `configVersion: 1.0`.
`hosts:`	This section defines the host entries for each of the CPS VMs in the deployment.
`- name:`	Defines the hostname of the VM. This name must be resolvable in the enterprise DNS environment.
`alias:`	Defines the internal host name used by each CPS VMs for internal communication, such as lb0x, pcrfclient0x, sessionmgr0x, or qns0x.
`interfaces`	This section defines the network details for each VM.
`network:`	The network name which must match a VLAN name (see below).
`ipAddress:`	The IP interface address.
`vlans:`	This section defines the separate VLANs to be configured. The "Internal" and "Management" VLANs are always needed. For additional networks, add more as needed.
`- name:`	Defines the name for a particular VLAN. It can be anything but it is recommended to have a name representing the network for certain traffic. The VLAN names defined here must be used in the `network` field in the `hosts` section above.
`vipAlias:`	The hostname associated with virtual interfaces on the Load Balancers (LBs), typically "Internal", "Management", and "Gx".

Parameter	Description
`vip:`	The Virtual IP address used on this VLAN. The virtual addresses are used to distribute the traffic between two load balancers.
`guestNIC:`	The Name of the interface specified in the host cloud config or Heat definition.
`pcrfVipAlias:`	The pcrfclient's vip alias.
`additionalHosts`	This section defines any hosts not configured in the hosts section above. **Note** LB VIPs are defined in this section as '`lbvip01`' and '`lbvip02`', as well as the '`arbitervip`' which defines the prcfclient01 internal IP. Any other hosts which CPS must interact with, such as NTP or NMS servers, must be defined in this section. Any hosts defined here are added to each CPS VM /etc/hosts file.
`- name:`	The hostname of the host.
`alias:`	The internal host name used by CPS nodes for internal communication, such as qns01.
`ipAddress:`	The IP address to use in the `/etc/hosts` file.
`config:`	This section defines general global parameters used to deploy CPS.
`qpsUser:`	Do not change.
`selinuxState:`	Do not change. Security Enhanced Linux (SELinux) support: disabled | enforcing. Default: disabled
`selinuxType:`	Do not change.
`broadhopVar:`	Do not change. Default: broadhop
`tacacsEnabled:`	Enter TRUE to enable TACACS+ authentication. For more information, refer to the *CPS Installation Guide for VMware*. Default: FALSE
`tacacsServer:`	Defines the IP address of the TACACS+ server.
`tacacsSecret:`	Defines the password/secret of the TACACS+ server.
`nmsManagers:`	Defines the SNMP Network Management Stations (NMS) address or hostname. Multiple NMS can be defined with a space separated list, for example: 10.105.10.10 10.202.10.10. Any hostnames defined here must also be added under the additionalHosts section above. To change the NMS trap receiver port, include the port appended to the IP address or hostname. For example: 10.105.10.10:6100.

Parameter	Description
`freeMemPer:`	By default, a low memory alert is generated when the available memory of any CPS VM drops below 10% of the total memory. To change the default threshold, enter a new value (0.0-1.0) for the alert threshold. The system will generate an alert trap whenever the available memory falls below this percentage of total memory for any given VM. Default: 0.10 (10% free).
`syslogManagers:`	Entries are space separated tuples consisting of protocol:hostname:port. Only UDP is supported at this time. Default: 514. For example: udp:corporate_syslog_ip:514 udp:corporate_syslog_ip2:514
`syslogManagersPorts:`	A comma separated list of port values. This must match values in the syslog_managers_list.
`logbackSyslogDaemonPort:`	Port value for the rsyslog proxy server to listen for incoming connections, used in the rsyslog configuration on the Load Balancer (lb) and in the logback.xml on the pcrfclient. Default: 6515
`logbackSyslogDaemonAddr:`	IP address value used in the `/etc/broadhop/controlcenter/logback.xml` on the pcrfclient. Default: lbvip02
`cpuUsageAlertThreshold:`	The following cpu_usage settings are related to the High CPU Usage Alert and High CPU Usage Clear traps that can be generated for CPS VMs. Refer to the *CPS Mobile Orchestration Gateway SNMP and Alarms Guide* for more details about these SNMP traps. Set the higher threshold value for CPU usage. The system generates an Alert trap whenever the CPU usage is higher than this value.
`cpuUsageClearThreshold:`	The lower threshold value for CPU usage. The system generates a Clear trap whenever the CPU usage is than this value and Alert trap is already generated.
`cpuUsageTrapIntervalCycle:`	The interval period to execute the CPU usage trap script. The interval value is calculated by multiplying five with the given value. For example, if set to one, then the script will get executed every five seconds. The default value is 12, which means the script is executed every 60 seconds.
`snmpTrapCommunity:`	The SNMP trap community string. Default: broadhop
`snmpRoCommunity:`	This value is the SNMP read-only community string. Default: broadhop
`monQnsLb:`	Do not change.

Parameter	Description
`freeMemoryPerAlert:`	By default, a low memory alert is generated when the available memory of any CPS VM drops below 10% of the total memory. To change the default threshold, enter a new value (0.0-1.0) for the alert threshold. The system will generate an alert trap whenever the available memory falls below this percentage of total memory for any given VM. Default: 0.10 (10% free)
`freeMemoryPerClear:`	Enter a value (0.0-1.0) for the clear threshold. The system will generate a low memory clear trap whenever available memory for any given VM is more than 30% of total memory. Default: 0.3 (30% of the total memory)
`monitorReplicaTimeout:`	This value is used to configure the replica-set timeout value. The default value is 540 seconds considering four replica sets. The customer can set timeout value according to the number of replica sets in their network. To recover a single session replica-set, it takes approximately 120 sec and adding 20% buffer to it; we are using 540 sec for default (for four replica sets). Without any latency between sessionmgr VMs, one replica-set will recover in ~135 seconds. If latency (40 -100 ms) is present between sessionmgr VMs, add a 10% buffer to 135 seconds and set the timeout value for the required number of replica sets in the deployment.
`sctpEnabled:`	Enables (`TRUE`) or disables (`FALSE`) Stream Control Transmission Protocol (SCTP) support for Diameter interfaces. Default: `TRUE`
`firewallState:`	Enables or disables linux firewall (IPtables) on all VMs. Valid Options: enabled / disabled Default: enabled

Parameter	Description
`snmpv3:`	Enable SNMPv3 support within CPS by deleting `null` and uncommenting (removing #) the following snmpv3 object parameters: • `v3User:` Username to be used for SNMPv3 request/response and trap. This parameter is required. Default: cisco_snmpv3 • `engineId:` This value is used for SNMPv3 request/response and on which NMS manager can receive the trap. It must be a hex value. This parameter is required. Default: 0x0102030405060708 • `authProto:` SHA or MD5. This value specifies the authentication protocol to be used for SNMPv3. This parameter is required. Default: SHA • `authPass:` This value specifies the authentication password to be used for SNMPv3 requests. It should have minimum length as 8 characters. This parameter is required. Default: cisco_12345 • `privProto:` This value specifies Privacy/Encryption protocol to be used in SNMPv3 request/response and SNMP trap. User can use AES/DES protocol. This parameter is required. Default: AES • `privPass:` This value specifies Privacy/Encryption password to be used in SNMPv3. If it is blank then value specified in `authPass` is used as privPass. This parameter is optional. Default: *blank (no value)*
`sysUsers:`	This section defines CPS system users.
`- name:`	The username of this user.
`  password:`	The clear text or encrypted password for this user. Refer to the *CPS Installation Guide for VMware* for instructions to generate an encrypted password.
`  groups:`	This section defines the groups to which this user belongs.
`  - `*`<group>`*	List each group on a separate line.
`hvUsers`	This section defines the hypervisor users.
`- name:`	The username of a user with root access to the host/blade. If installing CPS to multiple blade servers, it is assumed that the same username and password can be used for all blades.

Parameter	Description
`password:`	The password for this user. To pass special characters, they need to be replaced with the “% Hex ASCII” equivalent. For example, “$” would be “%24” or “hello$world” would be “hello%24world”.
`additionalUsers:`	This section defines additional CPS system users, such as those given access to Control Center.
`- name:`	The username of this user.
`password:`	The clear text or encrypted password for this user. Refer to the *CPS Installation Guide for VMware* for instructions to generate an encrypted password.
`groups:`	This section defines the groups to which this user belongs.
`- <group>`	List each group on a separate line.
`licenses:`	This section is used to enter the CPS license information. Contact your Cisco representative to receive your CPS license key(s).
`- feature:`	The name of the feature license, for example: "MOBILE_CORE".
`license:`	The license key for this feature.
`replicaSets:`	This section defines the CPS MongoDB replica sets.
`- title:`	The database for which the replica set is being created.
`setName:`	The name of the replica set.
`oplogSize:`	MongoDB operations log (oplog) size, in MB. Default: 5120
`arbiter:`	The hostname and port of the arbiter.
`arbiterDataPath:`	The data directory on the arbiter VM.
`members:`	The list of members for the replica set. Each list element will be a session manager hostname:port, for example sessionmgr01:27718.
`- <member>`	List each member hostname:port on a separate line.
`dataPath:`	The data directory path on the Session Manager VM.

Sample YAML Configuration File

Use the following file as a template to create the YAML configuration file for your CPS deployment. Refer to Configuration Parameters, on page 40 for a description of the available parameters.

```
---
#
#  CPS system configuration
#
#  CPS configuration is a YAML file with all the configuration required
#  to bring up a new installation of CPS.
#
#  This example file lists all possible configuration fields.
#  Fields that are not marked as required can be left out of
#  the configuration. Fields that are not provided will use
#  the default value. If not default is indicated the default
#  is an empty string.

# The version of the configuration file. The installation documentation
# for the version of the CPS you are installing will indicate which
# configuration version you must use.
# REQUIRED
configVersion: 1.0

# Configuration section for CPS hosts
# REQUIRED
hosts:
  # The host section must specify all hosts that are members of the CPS
  # deployment. Host entries consist of the following REQUIRED fields
  #  name: the string to be used as a hostname for the VM
  #  alias: the string to be used in hostname lookup for the VM
  #  interfaces: Network details consisting of the following REQUIRED fields
  #    network: The network name which must match a VLAN name (see below)
  #    ipAddress: The interface address
  - name: "lb01"
    alias: "lb01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.201"
      - network: "Management"
        ipAddress: "172.18.11.154"
      - network: "Gx"
        ipAddress: "192.168.2.201"
  - name: "lb02"
    alias: "lb02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.202"
      - network: "Management"
        ipAddress: "172.18.11.155"
      - network: "Gx"
        ipAddress: "192.168.2.202"
  - name: "sessionmgr01"
    alias: "sessionmgr01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.22"
      - network: "Management"
        ipAddress: "172.18.11.157"
  - name: "sessionmgr02"
    alias: "sessionmgr02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.23"
      - network: "Management"
        ipAddress: "172.18.11.158"
  - name: "qns01"
    alias: "qns01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.24"
  - name: "qns02"
    alias: "qns02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.25"
  - name: "qns03"
    alias: "qns03"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.26"
  - name: "qns04"
    alias: "qns04"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.27"
  - name: "pcrfclient01"
    alias: "pcrfclient01"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.20"
      - network: "Management"
        ipAddress: "172.18.11.152"
  - name: "pcrfclient02"
    alias: "pcrfclient02"
    interfaces:
      - network: "Internal"
        ipAddress: "172.16.2.21"
      - network: "Management"
        ipAddress: "172.18.11.153"

# Configuration section for CPS VLANs
# REQUIRED
vlans:
  # VLAN entries consist of the following REQUIRED fields
  #  name: The VLAN name. This name must be used in the "network" field
  #        host interfaces (see above)
  #  vipAlias: Hostname associated with the vip
  #  vip: Virtual IP used no this network, if any.
  #  guestNic: The name of the interface specified in the host cloud config
  #            or the Heat definition.
  #
  - name: "Internal"
    vipAlias: "lbvip02"
    vip: "172.16.2.200"
  - name: "Management"
    vipAlias: "lbvip01"
    vip: "172.18.11.156"
  - name: "Gx"
    vipAlias: "gxvip"
    vip: "192.168.2.200"

# Configuration section for hosts not configured in the hosts section above.
# REQUIRED
additionalHosts:
  # additionalHosts entries consist of the following REQUIRED fields
  #  name: The hostname
  #  alias: The string to be used in the etc/host file.
  #  ipAddress: The IP address to use in the etc/host file.
  #
  # the "arbitervip" to the pcrfclient01 internal ip is mandatory.
  #
  - name: "lbvip01"
    ipAddress: "172.18.11.156"
    alias: "lbvip01"
  - name: "lbvip02"
    ipAddress: "172.16.2.200"
    alias: "lbvip02"
  - name: "diam-int1-vip"
    ipAddress: "192.168.2.200"
    alias: "gxvip"
  - name: "arbitervip"
```

```
    ipAddress: "172.16.2.20"
    alias: "arbitervip"

# Configuration section for general configuration items.
# REQUIRED
config:
  # Do not change. See install documentation for details.
  # default: sys_user_0
  qpsUser: "sys_user_0"

  # Do not change. See install documentation for details.
  # default: disabled
  selinuxState: "disabled"

  # Do not change. See install documentation for details.
  # default: targeted
  selinuxType: "targeted"

  # See install documentation for details.
  # default: broadhop
  broadhopVar: "broadhop"

  # Set true to enable TACACS+ authentication.
  # default: FALSE
  tacacsEnabled: "FALSE"

  # The IP Address of the TACACS+ server
  tacacsServer: "127.0.0.1"

  # The password/secret of the TACACS+ server.
  tacacsSecret: "CPE1704TKS"

  # A set of SNMP Network Management Stations.
  # NMS can be specified as IP addresses or IP
  # addresses. Entries are space separated.
  # Hostnames must also be specified in Additional
  # Host configuration.
  # See install documentation for details.
  nmsManagers:

  # Low Memory alert threshold %.
  # default: 0.1 (10% free)
  freeMemPer: "0.1"

  # A space separated set of protocol:hostname:port
  # entries. UDP is the only supported protocol.
  # Example:
  # upd:corporate_syslog_ip:514 udp:corporate_syslog_ip2:514
  syslogManagers:

  # A comma separated set of port values.
  # This must match values in the syslog_managers_list.
  # default: 514
  syslogManagersPorts: "514"

  # Port value for the rsyslog proxy server to listen
  # for incoming connections
  # default: 6515
  logbackSyslogDaemonPort: "6515"

  # IP address value used in the
  # /etc/broadhop/controlcenter/logback.xml
  # on the pcrfclient.
  # default: lbvip02
  logbackSyslogDaemonAddr: "lbvip02"


  # High CPU alert threshold.
  # The system will alert whenever the usage is
  # higher than this value.
  # default: 80
  cpuUsageAlertThreshold: "80"
```

```
 # Clear High CPU Trap threshold.
 # The system will generate a clear trap when a
 # High CPU trap has been generated and the CPU
 # usage is lower than this value.
 # default: 40
 cpuUsageClearThreshold: "40"

 # The number of 5 sec intervals to wait between
 # checking the CPU usage.
 # default: 12 (60 seconds)
 cpuUsageTrapIntervalCycle: "12"

 # The SNMP trap community string.
 snmpTrapCommunity: "broadhop"

 #The SNMP read community string.
 snmpRoCommunity: "broadhop"

 #
 monQnsLb:

# The memory alert threshold (0.1 is 10%)
 freeMemoryPerAlert: "0.1"

 # The memory clear threshold (0.3 is 30%)
 freeMemoryPerClear: "0.3"

 #
 monitorReplicaTimeout: "540"

 # Enable SCTP
 # TRUE - feature enabled
 # FALSE - feature disabled
 sctpEnabled: "TRUE"


 # Enables or disables linux firewall on all VMs (IPtables).
 # default: disabled
 firewallState: "disabled"

 # enable SNMP V3.
 # If null, SNMP V3 is disabled.
 # To enabled add the following:
 #   v3User: The SNMP V3 user: REQUIRED
 #   engineId: hex value (ie, 0x0102030405060708):  REQUIRED
 #   authProto: SHA or MD5: REQUIRED
 #   authPass: at least 8 characters:  REQUIRED
 #   privProto: AES or DES: REQUIRED
 #   privPass: OPTIONAL
 snmpv3:
   null
 #  v3User: "cisco_snmpv3"
 #  engineId: "0x0102030405060708"
 #  authProto: "SHA"
 #  authPass: "cisco_12345"
 #  privProto: "AES"
 #  privPass: ""


 # Users
 # There are different categories of users specified for the CPS.
 # All users have the following fields:
 #
 #  name: The user name. REQUIRED
 #  password: The password for the user. REQUIRED
 #            The password will need to be either in cleartext or
 #            encrypted. Please refer to Install documentation for details.
 #  groups: The groups for the user. Groups are specified as a list
 #          of group names.

 # System Users
 # Note that there must be a system use named sys_user_0
 sysUsers:
    - name: "qns"
      password: "$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTk"
      groups:
        - pwauth

    - name: "qns-svn"
      password: "$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTk"
    - name: "qns-ro"
      password: "$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTk"

  # Hypervisor Users
  hvUsers:
    - name: "root"
      password: "cisco123"

  # Other Users for the CPS
  # e.g. Control Center Users
  additionalUsers:
    - name: "admin"
      password: "qns123"
      groups:
        - qns

# Configuration section for feature licenses
# REQUIRED
licenses:
  # Licenses have the following required fields:
  # feature: The name of the feature license.
  # license: The license key for the feature.
  # - feature: "feature 1  Name"
  #   license: "license 1 key string"
   - feature: "MOBILE_CORE"
     license:
"25D220C6817CD63603D72ED51C811F9B7CB093A53B5CE6FB04FF6C5C6A21ED1962F0491D4EED4441D826F1BC110B05EE35B78CF43B8B8B7A8127B4545538E365"

   - feature: "RADIUS_AUTH"
     license:
"118D767CE11EC2CB1E3AAA846A916FA57CB093A53B5CE6FB04FF6C5C6A21ED1962F0491D4EED4441D826F1BC110B05EE35B78CF43B8B8B7A8127B4545538E365"

# Configuration section for mongo replica sets.
# REQUIRED
replicaSets:
  #
  # Mongo replica sets have the following REQUIRED fields
  # <Mongo Set Identifier> : The database for which the replica
  #                          set is being created.
  #   setName: The name of the replica set
  #   oplogSize: Mongo Oplog size
  #   arbiter: The Arbiter hosthame and port
  #   arbiterDataPath: The data directory on the arbiter VM
  #   members: List of members for the replica set. Each list element
  #            will be a session manager hostname:port
  #   dataPath: The data directory path on the session manager VMs
  - title: SESSION-SET1
    setName: set01
    oplogSize: 5120
    arbiter: pcrfclient01:27717
    arbiterDataPath: /var/data/sessions.1
    members:
      - sessionmgr01:27717
      - sessionmgr02:27717
    dataPath: /var/data/sessions.1
  - title: BALANCE-SET1
    setName: set02
    oplogSize: 5120
    arbiter: pcrfclient01:27718
    arbiterDataPath: /var/data/sessions.2
    members:
      - sessionmgr01:27718
      - sessionmgr02:27718
    dataPath: /var/data/sessions.2
  - title: REPORTING-SET1
    setName: set03
    oplogSize: 5120
    arbiter: pcrfclient01:27719
    arbiterDataPath: /var/data/sessions.3
    members:
      - sessionmgr01:27719
      - sessionmgr02:27719
    dataPath: /var/data/sessions.3
  - title: SPR-SET1
    setName: set04
    oplogSize: 3072
    arbiter: pcrfclient01:27720
    arbiterDataPath: /var/data/sessions.4
    members:
      - sessionmgr01:27720
      - sessionmgr02:27720
    dataPath: /var/data/sessions.4
  - title: AUDIT-SET1
    setName: set05
    oplogSize: 3072
    arbiter: pcrfclient01:27017
    arbiterDataPath: /var/data/sessions.5
    members:
      - sessionmgr01:27017
      - sessionmgr02:27017
    dataPath: /var/data/sessions.5
  - title: ADMIN-SET1
    setName: set06
    oplogSize: 3072
    arbiter: pcrfclient01:27721
    arbiterDataPath: /var/data/sessions.6
    members:
      - sessionmgr01:27721
      - sessionmgr02:27721
    dataPath: /var/data/sessions.6
```

/api/system/config/status

Purpose

This API retrieves the status of individual install and deploy tasks run when a new or updated configuration is applied on the Cluster Manager VM.

This API can be called while the installation and deployment tasks are actively running.

The status reports:

- timestamp: timestamp in milliseconds.
- taskname: name of the individual task.
- status:
  - START: start of task.
  - INFO: general information about the task.
  - WARNING: error information about the task.
  - SUCCESS: task was successfully completed.
  - FAILURE: task failed and deployment failed.
- details: information about this task.

Retrieve Deployment Status

To retrieve the deployment status:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/status
- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success.

  Example Response:

```
---
[
{"timestamp":"1454367943000","taskName":"CPS
Installation","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"Cluman Setup","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"Cluman Setup","status":"SUCCESS","details":"Wait
 for Puppet to complete"},
{"timestamp":"1454367943000","taskName":"Post Install","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"SyncSvn","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"SyncSvn","status":"WARNING","details":"Failed
 to sync SVN."},
{"timestamp":"1454367943000","taskName":"SyncSvn","status":"SUCCESS","details":""},
{"timestamp":"1454367943000","taskName":"build_set","status":"START","details":"Building
 replica sets"},
{"timestamp":"1454367943000","taskName":"build_set","status":"INFO","details":"Wrote
mongo config"},
{"timestamp":"1454367943000","taskName":"build_set","status":"INFO","details":"Syncing
 mongo config to other hosts"},
{"timestamp":"1454367943000","taskName":"build_set","status":"SUCCESS","details":"Replica
 sets have been created successfully"},
{"timestamp":"1454367943000","taskName":"SetPriority","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"SetPriority","status":"SUCCESS","details":""},
{"timestamp":"1454367943000","taskName":"AddAdditionalUsers","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"AddAdditionalUsers","status":"SUCCESS","details":""},
{"timestamp":"1454367943000","taskName":"Licenses","status":"START","details":""},
{"timestamp":"1454367943000","taskName":"Licenses","status":"SUCCESS","details":""},
{"timestamp":"1454367943000","taskName":"Post Install","status":"SUCCESS","details":""}
]
```

The deployment process is complete when the following response is received: `"Post Install","status":"SUCCESS"`

Note The amount of time needed to complete the entire deployment process depends on the number of VMs being deployed, as well as the hardware on which it is being deployed. A typical deployment can take 45 minutes or more.

Startup status logs are written to: `/var/log/startupStatus.log` on the Cluster Manager VM.

API logs are written to: `/var/log/orchestration-api-server.log`

Refer to the /api/system/status/cps, on page 53 to determine the readiness status of the CPS cluster.

/api/system/status/cps

Purpose

This API returns the readiness status of CPS cluster.

Cluster Readiness

This API returns the "readiness" status of the CPS cluster.

The cluster is deemed "ready" when Puppet has run to completion on all VMs and the Replica set creation is complete on the Session Manager VMs. The Orchestrator can use this API to check when the cluster is ready so that it can then invoke the Service Creation APIs.

This API reports an aggregate status of MongoDB replica sets, qns processes, and the cluster (Puppet) for all VMs.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/status/cps
- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** JSON
- **Response:** 200 OK: success; 404: Unknown entity; 500: Script config not found

  The following example shows the readiness status for a CPS cluster:

  ```
  {
      "clusterStatus": "ready",
      "mongoStatus": "ready",
      "qnsStatus": "ready"
  }
  ```

mongoStatus and clusterStatus can report "ready", "not ready", or "error". qnsStatus can report "ready" or "not ready". MongoDB is the only application that reports the "error" status. If mongoStatus reports an "error" status, the clusterStatus also will report an "error" status.

API logs are at written to: `/var/log/orchestration-api-server.log`

/api/system

Purpose

This API is to used to determine the current state of the CPS system, and if necessary, override it in the event the reported state does not match the actual system state.

Many CPS orchestration APIs are accepted only when the CPS system is in a particular state. This API provides a method of overriding the reported API system state. It does not rectify or correct the underlying issue. For example setting the state to pre_deploy does not un-deploy the CPS deployment.

API logs are at written to: `/var/log/orchestration-api-server.log`

Retrieve the Current API State

To determine the current system state:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/
- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200: OK.

  Example Response:

  ```
  {
    "state": "pre_config"
  }
  ```

  This API can be used at any time.

The following states can be reported:

- **pre_config:** no configuration has been loaded onto the system (/api/system/config).
- **pre_deploy:** a configuration has been loaded, but not applied (api/system/config/apply).
- **deploying:** the system is in the process of being deployed.
- **deployed:** the system has finished the installation/deployment.
- **upgrading:** unsupported

Override the Current API State

Caution This API should only be used as directed by a Cisco representative. Improper use can cause irreparable harm to the CPS deployment.

To override the current system state:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/
- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** JSON payload with the new state specified as one of the following options: `pre_config`, `pre_deploy`, `deploying`, `deployed`, or `upgrading`.

  For example:

  ```
  {
    "state": "pre_config"
  }
  ```

- **Response Codes:** 400: Invalid state, please use: [pre_config, pre_deploy, deploying, deployed, upgrading]; 500: System error. See logs.

  Example Response:

  ```
  {
    "state": "pre_config"
  }
  ```

Upgrade APIs

Upgrade API Prerequisites

The following sequence of commands should be executed in OpenStack before running the CPS upgrade APIs.

Note These commands are for illustration purpose only and do not override any setup specific constraints. The specific commands may differ on your environment.

Step 1 Create a glance image of the new CPS ISO.

```
glance image-create --name <name of CPS ISO> --disk-format iso --container-format bare --is-public
True --file <Absolute path to new CPS ISO>
```

Step 2 Create a cinder volume based on the glance image of the new CPS ISO.

```
cinder create --image-id <glance image id of new CPS ISO> --display-name <name of new CPS ISO volume>
--availability-zone <optional zone> <size of ISO in GBs>
```

Step 3 Detach the existing CPS ISO volume from the Cluster Manager VM.

```
nova volume-detach <nova instance ID of cluman> <cinder volume ID of old CPS ISO volume>
```

Step 4 Attach the new CPS ISO volume to the Cluster Manager VM. This will require either the name of device at which volume is attached to the Cluster Manager, or "auto" to attach the volume as any available device name. In either case, the following command will output name of device to which new CPS ISO volume is attached.

```
nova volume-attach <nova instance ID of cluman> <cinder volume ID of new CPS ISO volume> <Name of
device, e.g. /dev/vdb or auto for autoassign>
```

/api/system/upgrade

Purpose

The following APIs are used to mount and unmount an ISO image to the Cluster Manager VM, trigger an out-of-service upgrade of a CPS deployment, and view the status of the upgrade.

Note Before invoking any of these APIs, refer to Upgrade API Prerequisites, on page 55.

Logs are at written to: `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

Unmount ISO

To unmount an existing CPS ISO image from `/mnt/iso` directory on the Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/action/unmount
- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The mount parameters are invalid; 500: System Error. See logs.

Note After invoking this API, it is recommended to detach the ISO image from the Cluster Manager VM using relevant command in OpenStack.

Mount ISO

Note Before invoking this API:

- A new cinder volume must be created in OpenStack based on the CPS ISO, and then attached to the Cluster Manager VM using relevant command in OpenStack. Refer to Upgrade API Prerequisites, on page 55 for more details.
- Run the **lsblk** command on the Cluster Manager VM to check the device name before running mount API. This needs to be checked after the CPS ISO volume has been attached to the Cluster Manager VM.

To mount the CPS ISO image onto /mnt/iso directory on the Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/action/mount
- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:**
  ```
  {
  deviceName: <filename of the block device at which the cinder volume is attached Ex:
  /dev/vdb>
  }
  ```
  `/dev/vdb` is for illustration only. Replace with the device name to which the CPS ISO volume is attached on your Cluster Manager VM.
- **Response Codes:** 200 OK: success; 400: The mount parameters are invalid; 500: System Error. See logs.

Upgrade CPS

Caution This API must only be used during a planned maintenance window. This API does not perform an in-service software upgrade. CPS processes will be restarted during this process and traffic will be affected.

This API can only be used once the CPS has been deployed and is in a ready state. Prior to that time this API will not be available.

To upgrade CPS using the mounted ISO:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/action/apply
- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:**

  type: Only `"OUT_OF_SERVICE"` is supported.

  config: The SVN/policy repository configuration to back up prior to upgrade. This repository will be backed up and restored during the upgrade.

  installType: The type of CPS deployment. Only `mobile` is supported.

  Example:
  ```
  {
  "config": "run",
  "installType": "mobile",
  "type": "OUT_OF_SERVICE"
  }
  ```
- **Response Codes:** 200 OK: success; 400: The input parameters are malformed or invalid.

The upgrade logs are at written to: /var/log/Upgrade_<*timestamp*>.log on the Cluster Manager VM.

Upgrade Status

To view the status of an upgrade:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/upgrade/status
- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 500: Script config not found

  Example Response:
  ```
  {
          "status": "In-Process"
      }
  ```
  - Not-Started - No upgrade has been initiated
  - In-Process - Upgrade is currently underway
  - Completed - Upgrade has completed
  - Error - There is a problem with the upgrade

This API is only valid after the operator has issued an upgrade.

System Configuration APIs

/api/system/mongo/config

Purpose

This API is used to retrieve the contents of `/etc/broadhop/mongoConfig.cfg`. This API is also used to add members to existing Mongo replica sets.

Important This API does **not** support modifications to any other parameters within the Mongo configuration. It will only add members to existing Mongo replica sets.

API logs are at written to: `/var/log/orchestration-api-server.log`

Workflow

1 Retrieve Current Mongo Configuration, on page 58

2 Manually edit the YAML file retrieved in step 1 to add members to the existing replica sets.

3 Load Updated Configuration, on page 59

4 Apply Loaded Configuration, on page 59

Retrieve Current Mongo Configuration

To retrieve (GET) the current configuration:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/mongo/config
- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Example Response (YAML format):

```
---
- title: "SESSION-SET1"
  setName: "set01"
  oplogSize: "5120"
  arbiter: "pcrfclient01:27717"
  arbiterDataPath: "/var/data/sessions.1"
  members:
  - "sessionmgr01:27717"
  - "sessionmgr02:27717"
  dataPath: "/var/data/sessions.1"
- title: "BALANCE-SET1"
  setName: "set02"
  oplogSize: "5120"
  arbiter: "pcrfclient01:27718"
```

```
    arbiterDataPath: "/var/data/sessions.2"
    members:
    - "sessionmgr01:27718"
    - "sessionmgr02:27718"
    dataPath: "/var/data/sessions.2"
  - ...
```

Note The response will include the complete Mongo configuration in YAML format.

Load Updated Configuration

Note This API can only be used once CPS has been deployed and is in a ready state. Prior to that time this API will not be available.

Use this API to load an updated Mongo configuration on the CPS Cluster Manager:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/mongo/config/
- **Header:** Content-Type: application/yaml
- **Method:** PUT
- **Payload:** The updated `mongoConfig.cfg` file in YAML format must be submitted. The entire contents of the Mongo config must be included.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Example Response:

  The updated contents of `/etc/broadhop/mongoConfig.cfg` is returned in the response in YAML format.

  Note After using this API to load the updated mongo configuration, you must apply the configuration. Refer to Apply Loaded Configuration, on page 59.

Apply Loaded Configuration

Note This API can only be used once the CPS has been deployed and is in a ready state. Prior to that time this API will not be available.

Use this API to apply the updated Mongo configuration that you loaded using Load Updated Configuration, on page 59:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/mongo/action/addMembers
- **Header:** Content-Type: application/json
- **Method:** POST

- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  This API returns immediately and does not wait for the members to be added. Refer to the log file to check the status.

  Example Response:
  ```
  {
      "logfile": "/var/log/broadhop/scripts/orch_api_03122016_203220.log"
  }
  ```

/api/system/config/additional-hosts

Purpose

This API enables you to configure new peer nodes such as PCEF, NTP, NMS, and so on, by modifying the `/etc/hosts` files on all CPS VMs.

The API logs are written in the `/var/log/orchestration-api-server.log` and `/var/log/startupStatus.log` files.

Note This API does not add a CPS VM to the CPS cluster.

Retrieve AdditionalHosts Configuration

To retrieve (GET) the AdditionalHosts configuration from the CPS Cluster Manager VM:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/additional-hosts
- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload
- **Response Codes:** 200 OK: success

  Example Response (YAML format):

  ```
  ---
   - name: "Host1 name"
     alias: "Host1 internal name"
     ipAddress: "Host1 IP address"
  - name: "Host2 name"
     alias: "Host2 internal name"
     ipAddress: "Host2 IP address"
  - name: "Host3 name"
     alias: "Host3 internal name"
     ipAddress: "Host3 IP address"
  ```

Add or Update AdditionalHosts Entry

This API adds or updates a new AdditionalHosts entry in the configuration file.

When this API call completes, the Cluster Manager is configured with the new `/etc/hosts` file. All the other deployed VMs are then updated asynchronously and the status is reported in http://:<*Cluster Manager IP*>8458/api/system/config/status.

To add or update an AdditionalHosts configuration:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/additional-hosts
- **Header:** Content-Type: application/yaml
- **Method:** PUT
- **Payload:** YAML

  Example Request:

```
---
 - name: "Host name"
   alias: "Host internal name"
   ipAddress: "Host IP address"
 - name: "NewHost name"
   alias: "NewHost internal name"
   ipAddress: "NewHost IP address"
```

Important

- To add or update AdditionalHosts, update new payload with existing additional hosts information along with new or updated additional hosts. This request replaces all the additional hosts with new additional hosts information.
- To modify or delete AdditionalHosts, update new payload with modified or deleted additional hosts and perform PUT request. This request replaces additional hosts information in the `/etc/hosts` file of both Cluster Manager and CPS VMs.
- To verify that the AdditionalHosts configuration is properly loaded, perform another GET request to http://<*Cluster ManagerIP*>:8458/api/system/config/additional-hosts.

- **Response Codes:** 200 OK: success; 400: malformed or invalid; 500: system error

Configuration Parameters

The following parameters can be defined in the AdditionalHosts YAML configuration file:

Parameter	Description
`- name:`	Defines the hostname of the VM. This name must be resolvable in the enterprise DNS environment.
`alias:`	Defines the internal host name used by CPS nodes for internal communication, such as qns01.
`ipAddress:`	Defines the IP address to use in the `/etc/hosts` file.