Multi-node Installation Guide

Qlik Catalog®

February 2021

QLIK.COM

LEAD WITH DATA™ Qlik

TABLE OF CONTENTS

1.0 Qlik Catalog Overview and System Requirements

This document describes how to install the "Multi-node" deployment option for Qlik Catalog.

This guide is meant to be read top to bottom as an end-to-end run book to configure your cluster and Qlik Catalog EXCEPT for section 5+, which is meant to address one-off configuration items and optional appendix sections that are not required for all installations.

1.1 Supported Hadoop Versions

- ➢ CDH Cloudera Enterprise 5.7+
- ➢ Hortonworks (HDP) 2.6.4+, 3.0+, 3.1+
- ➢ AWS EMR 5.19, 5.29

NOTE: Qlik Catalog will partner with all prospects and customers to ensure a needed version is certified. Please contact your pre-sales engineer or sales rep if a required version is not shown.

1.2 Qlik Catalog Node Configuration

Qlik Catalog ***must*** be installed on a dedicated Edge Node, which can be a bare metal or virtual machine.

1.3 Hardware Configuration Requirements

Cluster Edge Node Recommendations

- ➢ Recommended Minimum Production Configuration
  - o 12 Cores
  - o 128GB RAM
  - o System Drive 1TB
  - o Data Drive 1TB
  - o Ethernet 10GB
  - o Virtual Machine or bare metal

- ➢ Minimum POC/Dev Configuration
  - o 4 Cores
  - o 32GB RAM
  - o System partition 100GB
  - o Data partition 100GB
  - o Ethernet 10GB
  - o Virtual Machine or bare metal

- ➢ Minimum Supported Screen Resolution: 1366x768px

1.4 Software Configuration Requirements

- ➢ Configure environment as a true Hadoop edge node with all relevant Hadoop client tools
- ➢ OS: QDC is compatible with either RHEL 7 or CentOS 7 (en_US locales only) linux distributions
- ➢ Java: Oracle JDK 1.8, OpenJDK 1.8, or OpenJDK 11 (see Section 3).
  - o 8u222 is recommended version
  - o 8u162 began to include the Java Cryptography Extension (JCE); if for some reason an older version is used, the JCE must be separately downloaded and enabled
- ➢ Qlik Catalog Postgres 11.10 or Oracle 12.1 (see Section 3)
  - o Custom build of PostgreSQL 11.10 included with Qlik Catalog
  - o NOTE: the ***QVD Import*** feature is NOT currently supported for Oracle deployments
- ➢ Apache Tomcat (see Section 3)
  - o 9.0.38 is recommended for all installs and is required for those that will be using Hortonworks Data Platform (HDP) 3.x or Cloudera Distribution Hadoop (CDH) 6.x
- ➢ Sqoop version supported by your Hadoop distribution (should naturally be included as part of the edge node)
- ➢ Beeline (should naturally be included as part of the edge node)
- ➢ Hadoop client (should naturally be included as part of the edge node)
- ➢ Kerberos tools (krb5-workstation.x86_64) if Kerberos will be used
- ➢ Apache Phoenix (if Hbase access is needed)
- ➢ All JDBC drivers needed for database connectivity
- ➢ Ensure port 8080 or 443 (http or https) is open from user desktops to the Qlik Catalog node(s)

1.5 Qlik Catalog Browser Support

Officially Supported:

- ➢ Google Chrome - 80.0+
- ➢ IE11

Other browsers not actively tested. Issues must be reproducible on Chrome or IE to be eligible for a fix.

2.0 User Setup and Security Prerequisites

Please review this section carefully to determine the user access plan for deploying Qlik Catalog on your Hadoop cluster. There are a number of nuances that are important to consider up front based on expected usage of the application. For example, in a POC you might not enable all the security components to reduce install complexity, but for production you would enable a number of them.

2.1 Cluster User Setup and Impersonation

Qlik Catalog supports Impersonation. Impersonation allows a managed user to login as themselves and execute work on the cluster represented by their user ID using privileges of a service account.

If impersonation is DISABLED:

1. A service user is needed to run Qlik Catalog (the user who is running Tomcat on the edge node) and it should exist on all cluster nodes as an OS user.
2. The Qlik Catalog service user should have ALL access to node manager local directories specified in yarn.nodemanager.local-dirs property in yarn-site.xml
3. The Qlik Catalog service user should have ALL permissions on the podium base directory in HDFS.
4. The Qlik Catalog service user should have a home directory in HDFS (example: /user/podium) and should have all permissions on it.
5. The Qlik Catalog service user should have all permissions in Hive including create/drop database and create/drop function.
    a. If this is not possible the Hive databases can be created in advance and a property set to allow this to happen vs default behavior which is dynamic databases creation when sources are on-boarded.

If impersonation is ENABLED:

1. All Qlik Catalog users should exist on all cluster nodes as OS users.
2. The Qlik Catalog service user should have ALL access to node manager local directories specified in yarn.nodemanager.local-dirs property in yarn-site.xml
3. All Qlik Catalog users should have all permissions on podium base directory in HDFS
4. All Qlik Catalog users should have a home directory in HDFS (example: /user/username1) and should have all permissions on it.
5. In case the hive.admin.user is specified, it should have all permissions in Hive including create/drop databases and create/drop function. All other Qlik Catalog users should have read permissions on their source tables
    a. NOTE: hive.admin.user (specified in core_env.properties) allows you to override impersonation settings specifically for Hive access
6. In case the hive.admin.user is NOT specified, all Qlik Catalog users should have all permissions in Hive including create/drop databases and create/drop function.

Please see Sentry and Ranger sections for more details.

2.2 Active Directory

Qlik Catalog can sync with existing users and their groups in AD by specifying the necessary parameters in the Qlik Catalog UI within the admin section. Qlik Catalog will provide a validation shell script prior to installation that will ask for the active directory query string to find the users and groups required so please have that ready. If creating a custom AD group for the POC with associated users you must create appropriate Ranger or Sentry policies as well.

Example of information to gather that corresponds to the validation script and UI parameters for registering the connection.

```
#active_directory_ldap_host=sid.ad.podiumdata.net
#active_directory_ldap_port=636
#active_directory_ldap_user_dn="CN=Podium Data,DC=ad,DC=podiumdata,DC=net"
#active_directory_ldap_user_pw="Qwerty123!"
#active_directory_ldap_is_ssl=true
#active_directory_ldap_search_base_dn="DC=ad,DC=podiumdata,DC=net"
#active_directory_ldap_search_filter="(&(cn=Domain Admins)(objectClass=group))"
```

2.3 HDFS Permissions

Qlik Catalog stores all ingested data in HDFS within a defined taxonomy. The prefix structure can be completely user defined, as well as the named folders Qlik Catalog creates, put Qlik Catalog will create its own structures within those folders. All users who want to run loads, transforms, or publish jobs in Qlik Catalog must have RWX access to at least one of the predefined directory structures.

Example:

- /user/defined/directory/for/podium

Within this structure Qlik Catalog will store sources, tables, and field information associated with Hive tables.

- /user/defined/directory/for/podium/receiving/source_name/entity_name/partition_timestamp/2.
4 Hive and Permissions

As part of the data on-boarding process, Qlik Catalog will automatically create Hive external tables for the data it copies to HDFS (see above section). If no Hive database exists, Qlik Catalog will dynamically create one as the service user (if impersonation is OFF or if it's explicitly set) or as the username that runs the load job (if impersonation is ON). This can be bypassed if it violates security policy (give the create permissions) by pre-creating Hive databases and setting a parameter in core_env.properties called validate.hive.database=false.

Example Hive JDBC URIs

```
jdbc:hive2://master.hostname.acme.net:10000/default;principal=hive/master.hostname.acme.net@hostname.acme.net
jdbc:hive2://hdmduv0005.test.group:10010/podium_test_01;principal=hive/hdmduv0005.machine.group@APPGLOBAL.CHIPCORP.COM
jdbc:hive2://hdmduv0005.machine.test.group:10010/podium_test_01;principal=hive/_HOST@APPGLOBAL.CHIPCORP.COM
```

2.4 Ranger and Sentry

Qlik Catalog supports Apache Ranger and Sentry. First, Qlik Catalog will naturally honor Ranger and Sentry security policies and report any access rights errors up through the Qlik Catalog UI as a result of its use of Hadoop standard APIs. Second, for Apache Ranger only, in the unique scenario where impersonation is enabled, but a service account is used for Hive, Qlik Catalog can dynamically create Ranger policies based on the work done by the service account based on the user executing the work in Qlik Catalog. This is an OPTIONAL capability for this unique security scenario. Please see the Apache Ranger appendix section for more details on this.

Please see Sentry and Ranger sections for more details.

2.5 Kerberos

For an application to use Kerberos, its source must be modified to make the appropriate calls into the Kerberos libraries. Applications modified in this way are considered to be Kerberos-aware or Kerberized. The following will enable Qlik Catalog to run (under Tomcat) as a Kerberized application. Qlik Catalog functionality will be authenticated by Kerberos for the user which has been kinit (obtained/cached Kerberos ticket-granting tickets) before Tomcat is started.

When Kerberos is enabled, Qlik Catalog specifies a set of rules in the property hadoop.security.auth_to_local in core-site.xml that maps Kerberos principals to local OS users. Usually, the rules are configured to just strip the domain names from the principals to get the local user name. Those local users must be OS users on all nodes.
If Kerberos is configured with Active Directory, this process is simplified as the AD users are already available as OS users on all nodes (e.g., `id adccuser` returns the AD user/group).

- ➢ Ensure Java Cryptography Extension (JCE) unlimited security jars are up to date. They are provided with OpenJDK, but not with Oracle JDK. JCE is automatically included and enabled with Java 8u162 or above.
- ➢ Optimal Kerberos properties for single realm include ticket lifecycle and encryption. Add the following properties to: krb5.conf dns_lookup_kdc = false dns_lookup_realm = false
- ➢ Multi realm settings are supported.

Please see 5.2 Kerberos Configuration appendix section for more details.

3.0 Installation Prerequisites

NOTE: In all commands below, the user that the command should be “executed as” is in parentheses at the beginning of the line:

- ∉ “(sudo)” means the command should be run as a user with sudo permission
- ∉ “(qdc)” means the command should be run as the Qlik Catalog *service account* user -- “sudo su - qdc” may be used to become this user
- ∉ “(postgres)” means the command should be run as the PostgreSQL superuser -- “sudo su - postgres” may be used to become this user

Note: Outside ports 80 (HTTP) and 443 (HTTPS) must be open to allow outbound communication to the Internet in order to allow software to be downloaded.

Prerequisite Installation Script for CentOS 7 deployments (Optional)

There is an optional prerequisite installation script which may be used for **CentOS 7 deployments**. This script will install all of the prerequisites detailed in this section. It is located within the QDCinstaller.zip package and is named: ***QDCprereqs.sh***.

- ‘**sudo’** permission is required to run *QDCprereqs.sh*
- There are two environment variables at the beginning of the script which may be defined by end-users:
    - **QDC_HOME**: the directory where Qlik Catalog will be installed. (default value: /usr/local/qdc)
    - **QDC_SERVICE_ACCOUNT**: the local user account which will be used to run Qlick Catalog. (default value: qdc)

To run *QDCpreqs.sh*:

1. Install unzip (if not present)

```
(sudo)    # sudo yum install -y unzip
```

2. Copy the podium.zip & QDCinstaller.zip files into /tmp
3. Unzip QDCinstaller.zip within /tmp

```
(sudo)   # cd /tmp
(sudo)   # unzip QDCinstaller.zip
```

4. Run QDCprereqs.sh

```
(sudo)   # sudo ./QDCinstaller/QDCprereqs.sh
```

Manual Prerequisite Installation

Important: During the prerequisite setup process, several items are needed from the Qlik Catalog software distribution: a file named podium-4.6-14761.zip. The instructions below assume the Qlk Catalog software distribution has been unzipped to /tmp:

1. Install unzip (if not present)

(sudo) # sudo yum install -y unzip

2. Expand the Qlik Catalog software distribution to /tmp

(sudo) # unzip <replace-path>/podium-4.6-14761.zip -d /tmp/

3.1 Java JDK Installation

Qlik Catalog is supported on the following JDK platforms:

- OpenJDK 8
- OpenJDK 11
- Oracle JDK 8 (license required)

1. Check if JDK exists. If it exists skip this step.

(sudo) # java -version

JDK 8 results:

```
Openjdk version "1.8.0_222"
OpenJDK Runtime Environment (build 1.8.0_222-b10)
OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

JDK 11 results:

```
openjdk version "11.0.6" 2020-01-14 LTS
OpenJDK Runtime Environment 18.9 (build 11.0.6+10-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.6+10-LTS, mixed mode, sharing)
```

1. JDK Installation:
   ➔ OpenJDK: Use YUM for installation:

   OpenJDK 8 installation:

   (sudo) # sudo yum install -y java-1.8.0-openjdk-devel

   OpenJDK 11 installation:

   (sudo) # sudo yum install -y java-11-openjdk-devel

   ➔ **Oracle JDK 8 (license required): Download the package directly from Oracle and install.**

3.1.1 Mandatory JCE Upgrade for JDK Before 8u162

Customers are **strongly** discouraged from running a JDK prior to 8u162. If you do so, it is mandatory to download and enable the Java Cryptography Extension (JCE).

3.2 Create Service Account and Qlik Catalog Directory

Create a ***service account*** to run Qlik Catalog. Tomcat will be started as this user. Typically, this user is named “qdc” or “qdcsvc”. Throughout the remainder of this document “qdc” will be used -- please replace “qdc” with a different user if so desired. In a similar fashion, “qdc” is also used as a group name.

1. Create a service account to run Qlik Catalog (the user which launches Tomcat)

   ```
   (sudo)    # sudo groupadd qdc
   (sudo)    # sudo useradd -s /bin/bash -g qdc qdc
   ```

2. Optionally, set a password for the service account – this is not needed if “sudo” is used to become this user (e.g., “sudo su - qdc”)

   ```
   (sudo)    # sudo passwd qdc
   ```

3. Create a directory for all Qlik Catalog artifacts, including Tomcat

   ```
   (sudo)    # sudo mkdir /usr/local/qdc
   ```

4. Change ownership of /usr/local/qdc to the service account and group being used

   ```
   (sudo)    # sudo chown -Rf qdc:qdc /usr/local/qdc
   ```

3.3 Tomcat Installation

Qlik strongly encourages the use of SSL with Tomcat for securing Qlik Catalog sessions. Instructions for configuring Tomcat to support SSL connections are provided later in this document.

1. Install wget (if not present) while a sudo capable user

   ```
   (sudo)     # sudo yum install -y wget
   ```

2. Become the service account user

   ```
   (sudo)     # sudo su – qdc
   (qdc)    $ cd /usr/local/qdc
   ```

3. Download Apache Tomcat:

- **Tomcat 9.0.38** is recommended for all installations and is ***required*** for the following:
    - Hortonworks Data Platform (HDP) 3.x
    - Cloudera Distribution Hadoop (CDH) 6

```
(qdc)   $ wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.38/bin/apache-tomcat-9.0.38.tar.gz
```

4. Extract the Tomcat file

```
(qdc)   $ tar -xvf apache-tomcat-9.0.38.tar.gz
```

5. The resulting directory, for example “/usr/local/qdc/apache-tomcat-9.0.38”, is known as the Tomcat home directory. When configuring the QDCinstaller.properties file in the next section, please set TOMCAT_HOME to this value.

6. If using **Tomcat 7.0.94**, overwrite <tomcat home>/conf/server.xml with the version expanded from the Qlik Catalog zip file or edit the existing server.xml manually:

   Overwrite Instructions (recommended)

```
(qdc)   $ cp /tmp/podium/config/tomcat7-server.xml /usr/local/qdc/apache-tomcat-7.0.94/conf/server.xml
```

   OR

   Manual Edit Instructions:

   In the HTTP Connector element, add the bold attributes to turn compression on

```
<Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        server="Unknown Application Server"
        useSendfile="false"
        compression="on"
        compressionMinSize="150"
        noCompressionUserAgents="gozilla, traviata"
        compressableMimeType="text/html,text/xml,text/plain,text/css,text/java
        script,application/x-javascript,application/javascript,application/json"
        redirectPort="8443" />
```

   In the AccessLogValve element, change the bold attributes prefix, suffix and pattern

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
prefix="localhost_access" suffix=".log"
pattern="%h %l %u %t &quot;%r&quot; %s %b %{podiumUser}s
%{podiumSession}s [%I]" />
```

- If using **Tomcat 9.0.38** overwrite <tomcat home>/conf/server.xml with the version expanded from the Qlik Catalog zip file or edit the existing server.xml manually:

  Overwrite Instructions (recommended)

```
(qdc)   $ cp /tmp/podium/config/tomcat9-server.xml /usr/local/qdc/apache-tomcat-9.0.38/conf/server.xml
```

OR

Manual Edit Instructions:

In the HTTP Connector element, add the bold attributes to turn compression on

```
<Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        useSendfile="false"
        compression="on"
        compressionMinSize="150"
        noCompressionUserAgents="gozilla, traviata"
        compressableMimeType="text/html,text/xml,text/plain,text/css,text/java
        script,application/x-javascript,application/javascript,application/json"
        redirectPort="8443" />
```

In the AccessLogValve element, change the bold attributes prefix, suffix and pattern

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
        prefix="localhost_access" suffix=".log"
        pattern="%h %l %u %t &quot;%r&quot; %s %b %{podiumUser}s
        %{podiumSession}s [%I]" />
```

7. Port 8080 needs to be open on the Qlik Catalog node firewall for HTTP connections. Port 8443 needs to be opened on the Qlik Catalog node firewall for secure HTTPS connections.

   ➔ It is recommended that Tomcat be configured to ***redirect*** insecure HTTP connections directly to a secure HTTPS session. Instructions for configuring **HTTPS redirect** are provided later in this document.

```
(sudo)    # sudo firewall-cmd --zone=public --permanent --add-port=8080/tcp
(sudo)    # sudo firewall-cmd --zone=public --permanent --add-port=8443/tcp
(sudo)    # sudo systemctl restart firewalld.service
```

8. Setup Apache Tomcat as a service to automatically start when Linux restarts

   Possible edit: if the Tomcat home directory is not /usr/local/qdc/apache-tomcat-9.0.38 or the service user/group is not qdc, the file /etc/systemd/system/tomcat.service must be edited after the copy (cp) step below.

   The copy (cp) command below assumes the Qlik Catalog software (a.k.a., podium zip) has been expanded to /tmp -- see Section 3.0. You will not be able to start the service until Qlik Catalog PostgreSQL is later installed, as a dependency exists.

```
(sudo)    # sudo cp /tmp/podium/config/tomcat.service /etc/systemd/system/
(sudo)    # sudo systemctl daemon-reload
(sudo)    # sudo systemctl enable tomcat.service
```

9. Optional: Configure Tomcat to support HTTPS

Configuring Tomcat to support HTTPS may be done now or at a later time.

10. Start Tomcat manually

```
(qdc)   $ cd <tomcat home>
(qdc)   $ ./bin/startup.sh
```

11. Browse to the following URL to verify that Tomcat is running

https://<Qlik-Catalog-Node-IP-Address-OR-Hostname>:8443

12. Tomcat can be manually stopped at any time

```
(qdc)   $ cd <tomcat home>
(qdc)   $ ./bin/shutdown.sh
```

13. The Tomcat log can be monitored

```
(qdc)   $ tail -F <tomcat home>/logs/catalina.out
```

3.4 PostgreSQL Installation

Qlik Catalog is only certified on Qlik Catalog PostgreSQL 11.10. To ensure this version is used, the Qlik Catalog PostgreSQL installer has been included in the Qlik Catalog zip file. The directions below describe how to extract and install this custom version of PostgreSQL, and then configure it.

Do **NOT** install PostgreSQL using rpm, yum or otherwise download it from the Internet.

NOTE: If you already have a different version of PostgreSQL installed, please first uninstall it.

NOTE: The below instructions assume that the Qlik Catalog zip file has already been extracted to /tmp -- see Section 3.0.

1. Create a “postgres” user and group

```
(sudo)     # sudo groupadd postgres
(sudo)     # sudo useradd -s /bin/bash -g postgres postgres
```

2. Add the “postgres” user to the “qdc” group

```
(sudo)     # sudo usermod -aG qdc postgres
```

3. Create directories for executables and data, and change their ownership

```
(sudo)     # sudo mkdir -p /usr/pgsql/qdc11
(sudo)     # sudo chown -R postgres:postgres /usr/pgsql
(sudo)     # sudo mkdir -p /var/lib/pgsql/11/qdc_data
(sudo)     # sudo chown -R postgres:postgres /var/lib/pgsql
```

4. Run the custom Qlik Catalog PostgreSQL installer appropriate for your operating system as the “postgres” user:

```
(sudo)          # sudo su - postgres
```

Installer for RHEL7/CentOS7 deployments**:**

```
(postgres)     $ /tmp/podium/thirdParty/qdc_pg11-10_RHEL7-and-CentOS7.bsx
```

Installer for RHEL8/CentOS8 deployments:

```
(postgres)     $ /tmp/podium/thirdParty/qdc_pg11-10_RHEL8-and-CentOS8.bsx
```

5. Create a symlink to the psql and pg_dump executables

```
(sudo)     # sudo ln -s /usr/pgsql/qdc11/bin/psql /usr/bin/psql
(sudo)     # sudo ln -s /usr/pgsql/qdc11/bin/pg_dump /usr/bin/pg_dump
```

6. IMPORTANT! Port 5432 needs to be opened on the Qlik Catalog node firewall to allow connections to PostgreSQL. The Qlik Licenses container is dependent upon communication with the PostgreSQL database.

```
(sudo)     # sudo firewall-cmd --zone=public --permanent --add-port=5432/tcp
```

```
(sudo)     # sudo systemctl restart firewalld.service
```

7. Set PostgreSQL to start automatically, then start it.

   Possible edit: if the directories in step 3 were altered, or the user/group is not "postgres", the file /etc/systemd/system/qdc_pg-11.10.service must be edited after the copy (cp) step below.

   The copy (cp) command below assumes the Qlik Catalog software (a.k.a., podium zip) has been expanded to /tmp -- see Section 2.0.

```
(sudo)     # sudo cp /tmp/podium/config/qdc_pg-11.10.service /etc/systemd/system/
(sudo)     # sudo systemctl daemon-reload
(sudo)     # sudo systemctl enable qdc_pg-11.10.service
(sudo)     # sudo systemctl start qdc_pg-11.10.service
```

3.5 Docker & Node.js

Qlik Catalog is dependent upon the Docker Engine (https://www.docker.com/products/container-runtime). Additionally, Node.js is also required for integration with Qlik Sense.

The Docker Engine installation package differs between RHEL 7 and CentOS 7:

- RHEL 7:
  - ***Docker Enterprise*** is the officially supported Docker platform for RHEL 7.
  - A support subscription with Red Hat Software is required to access the RHEL repository containing the Docker Enterprise engine.
  - RHEL 7 Docker Enterprise installation instructions may be found here: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html-single/getting_started_with_containers/index#using_the_docker_command_and_service

- CentOS 7:
  - ***Docker Community Edition*** **(CE)** may be used with CentOS 7.
  - Installation of Docker CE on RHEL 7 is not recommended and is not supported.

The following instructions detail the installation of **Docker Community Edition** on CentOS 7.

If Qlik Catalog will catalog Qlik Sense QVDs, please do the following.

1. Install Node.js (which includes npm)

```
(sudo)     # curl -sL https://rpm.nodesource.com/setup_10.x | sudo bash -
(sudo)     # sudo yum install -y nodejs
```

2. Install Docker, set it to start automatically, and start it

```
(sudo)    # sudo yum install -y yum-utils device-mapper-persistent-data lvm2
(sudo)    # sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
(sudo)    # sudo yum install -y docker-ce

(sudo)    # sudo systemctl enable docker.service
(sudo)    # sudo systemctl start docker.service
```

3. Add the service user to the “docker” group

```
(sudo)    # sudo usermod -aG docker qdc
```

4. Install Docker Compose

```
(sudo) sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   617  100   617    0     0   2114      0 --:--:-- --:--:-- --:--:--  2127
100 11.2M  100 11.2M    0     0  13.9M      0 --:--:-- --:--:-- --:--:-- 32.9M
```

```
(sudo)    # sudo chmod +x /usr/local/bin/docker-compose
```

5. Test Docker and Docker Compose -- this should be done as the service account user

```
(sudo)  # sudo su - qdc

(qdc)    $ docker ps
```

```
CONTAINER ID   IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
```

```
(qdc)    $ docker-compose version
```

```
docker-compose version 1.23.2, build 1110ad01
docker-py version: 3.6.0
CPython version: 3.6.7
OpenSSL version: OpenSSL 1.1.0f  25 May 2017
```

3.6 Oracle Configuration

To configure Oracle as the storage engine for Qlik Catalog metadata, please refer to your DBA team for Oracle instance access. (As noted previously, the QVD Import Feature is NOT currently supported for Oracle deployments). Follow the steps below to configure the Oracle instance for Qlik Catalog metadata.

On Oracle Server
1. upload Qlik Catalog oracle metadata package (/tmp/podium/thirdParty/podium_oracle_md_2.tar.gz) to oracle server

```
scp ./podium_oracle_md_2.tar.gz oracle@oracle12c-1c.corp.podiumdata.com:/tmp/
```

2. untar the package -- this must occur in the /tmp directory as the script has hardcoded /tmp

```
[oracle@oracle2 tmp]$ cd /tmp; tar -xzvf ./podium_oracle_md_2.tar.gz
./podium/
./podium/.DS_Store
./podium/bin/
./podium/bin/.DS_Store
./podium/bin/oracle/
./podium/bin/oracle/podium_constraints_oracle.sql
./podium/bin/oracle/.DS_Store
./podium/bin/oracle/podium_oracle_md_build.sh
./podium/bin/oracle/podium_drop_oracle.sql
./podium/bin/oracle/podium_create_users.
... ...
```

3. change ownership of shell script file

```
chmod 766 /tmp/podium/bin/oracle/podium_oracle_md_build.sh
```

4. execute the shell script, it will create all the oracle objects (users & tables) and insert configuration data into tables. Monitor podium_oracle_md_build.log & import.log for errors.

```
cd /tmp/podium/bin/oracle/; ./podium_oracle_md_build.sh >> ./podium_oracle_md_build.log 2>&1 &
```

3.7 Create Qlik Catalog Base Storage Directory in HDFS

Note: If you have not already read section 2.1 “Cluster User Setup” please do so now.

The following operations should be performed by the owner of the cluster directory (usually ‘hdfs’) or user who is a member of the cluster ‘supergroup.’

1. Qlik Catalog requires a base directory to work from in HDFS. You can specify any base directory and Qlik Catalog will create sub-directories from there.

   ```
   # hadoop fs –mkdir /qlik-catalog_base
   ```

2. Change ownership of <qlik catalog base directory> to the service account running Qlik Catalog or to group permissions that reflect access for all impersonated users.

   ```
   # hadoop fs -chown -R <service user>:<service group> /qlik-catalog_base
   ```

3.8 Create Hive ‘user_views’ Database

This database is used as ‘transient scratch space’ for Publishing data from Qlik Catalog to an external target. Views are dynamically created when data needs to be masked/obfuscated with the appropriate function required in order to land the data in the required state. This database can be any name and is set in the core_env.properties file.

1. Log into Hive and create a database called 'user_views'. Ensure that the Hive/Podium user has full permissions for the 'user_views' database.

   ```
   # create database user_views;
   ```

3.9 Run the pre-installation checklist tool

1. Your pre-sales engineer should provide a zip file with a shell script that will validate security and access rights for the Qlik Catalog application against a given cluster. This helps considerably in addressing underlying cluster access rights and configuration issues before introducing the Qlik Catalog application.

2. Unzip the package and edit the preInstallationChecklistTool.properties and edit all applicable properties. This drives preInstallationChecklistTool.sh that tries to access AD, HDFS, Hive, Postgres, and tests Kerberos and Ranger policies naturally.

3. If you would like to disable one of the checks because you may not be using Active Directory for example just comment out the associated function at the bottom of preInstallationChecklistTool.sh

4. Once the properties file is configured, run the shell script and address any errors that are produced.

   ```
   # ./preInstallationChecklistTool.sh
   ```

4.0 Qlik Catalog Software Installation

The Qlik Catalog installer is a shell script, ***QDCinstaller.sh***, that is guided by a properties file, ***QDCinstaller.properties***.

The shell script and properties files are included in a zip file, **QDCinstaller.zip**. Required installation parameters are first configured in the properties file. Then, the shell script is run and begins by confirming the installation parameters.

Password Encryption

Passwords may be encrypted at any time using a utility. It will use stdin to prompt for the password and output the encrypted password to stdout.

```
# unzip -j podium-4.6-14761.zip podium/lib/podium-encrypt-util.jar -d .
# java -cp podium-encrypt-util.jar com.nvs.core.utils.PodiumEncrUtil
```

NOTE: In all commands below, the user that the command should be "executed as" is in parentheses at the beginning of the line:

- "(sudo)" means the command should be run as a user with sudo permission
- "(qdc)" means the command should be run as the Qlik Catalog service account user -- "sudo su - qdc" may be used to become this user
- "(postgres)" means the command should be run as the PostgreSQL superuser -- "sudo su - postgres" may be used to become this user

4.1 First-time Installation Using the Installer

Execute the following steps to perform a first-time install:

1. The installer must be run as the Qlik Catalog service account:

   (sudo) `# sudo su - qdc`

2. Unzip QDCinstaller.zip into a working directory

3. Copy the Qlik Catalog software zip file (e.g., podium-4.6-14761.zip) into the working directory

4. Change directory to the working directory. It should contain the following:

   ```
   podium-4.6-14761.zip  QDCinstaller.properties  QDCinstaller.sh  QDCinstaller.txt  QULA.txt  upgrade-scripts (directory)
   ```

5. Edit the installation parameters in **QDCinstaller.properties** -- additional documentation is present in that file. Ensure the following are set -- the defaults should work for most. PODIUM_RELEASE_FILE is set to the current podium zip file. SUPERUSER_NAME and SUPERUSER_PASSWORD are for QDC PostgreSQL and by default are both "postgres":

   - INSTALL_TYPE -- **CDH, HDP, or EMR**
   - QDC_HOME
   - TOMCAT_HOME
   - PODIUM_RELEASE_FILE

- ⊄ WEBAPP_NAME
- ⊄ POSTGRES_HOSTNAME
- ⊄ POSTGRES_IPADDRESS
- ⊄ SUPERUSER_NAME
- ⊄ SUPERUSER_PASSWORD
- ⊄ JAVA_HOME
- ⊄ PODIUM_BASE -- change this to a path in HDFS (or S3) to which the service user has write permission
- ⊄ If the install is CDH, HDP or EMR, set DISTRIBUTION_URI to the Hive JDBC URI if known
- ⊄ If the install is EMR, set BASE_URI, BASE_URI_USERNAME and BASE_URI_PASSWORD as suggested in the comments
- ⊄ If Qlik Catalog metadata is being stored in Oracle rather than PostgreSQL (or the PostgreSQL database will be manually created on a different server), set CREATE_PODIUM_MD_DATABASE=false

6. Interactive & Non-Interactive installation modes:

   The Qlik Catalog installer shell script may be run in interactive mode on non-interactive "silent" mode.

   Interactive installation allows users to confirm installer actions in a step-by-step manner. To run the installer in interactive mode simply run: ./QDCinstaller.sh

   Non-Interactive or "silent" installation allows users to deploy QDC in a scripted manner which does not require any user interaction. Details for running the installer in non-interactive mode are noted later in this document.

7. Run the shell script. You may abort the script at any time by entering control-C. The installer will begin by asking you to confirm the data you entered in QDCinstaller.properties -- it will also validate the parameters. It will expand the Qlik Catalog software zip and create all necessary configuration files. It will then setup the database (if PostgreSQL is used).

   (qdc) # ./QDCinstaller.sh

8. When complete, Tomcat is automatically started. It may be started and shutdown using the following:

   (qdc) # <tomcat home>/bin/startup.sh
   (qdc) # <tomcat home>/bin/shutdown.sh

   The log can be viewed as follows:

   (qdc) # tail –F <tomcat home>/logs/catalina.out

9. If Qlik Catalog metadata is being stored in Oracle rather than PostgreSQL, the Oracle database and schema must be created. Stop tomcat, perform the following, and then restart Tomcat.

   Copy the appropriate Oracle JDBC driver (e.g., ojdbc8.jar) to <tomcat home>/webapps/qdc/WEB-INF/lib

   Edit <tomcat home>/conf/core_env.properties and replace the following JDBC and Hibernate properties:

```
# JDBC driver
jdbc.driver=oracle.jdbc.driver.OracleDriver

# JDBC data source
datasource.classname=oracle.jdbc.pool.OracleDataSource

# JDBC URL
jdbc.url=jdbc:oracle:thin:@<oracle_server_hostname>:1521:orcl

# Database user
jdbc.user=podium_core
jdbc.password jdbc.password=<encrypted password string>

# Hibernate Dialect
hibernate.dialect=org.hibernate.dialect.Oracle12cDialect
```

10. Open up a browser and go to the following URL to validate that you can display the homepage. If a WEBAPP_NAME other than “qdc” was specified in QDCinstaller.properties, please replace “qdc” with the alternative webapp name.

    https://<QDC node hostname or IP address>:8443/qdc

11. Attempt to login for the first time (user: podium, password: nvs2014!) and a prompt will appear to enter a license key. Enter the provided key and click register.

12. The Qlik Licenses container must be setup as a service to automatically start when Linux restarts.

    Possible edit: If the Qlik Catalog/Qlik Sense integration directory is not /usr/local/qdc/qlikcore or the service user/group is not qdc, the file /etc/systemd/system/qlikContainers.service must be edited after the copy (cp) step below.

    The copy (cp) command below assumes the Qlik Catalog software zip has been expanded to /tmp -- see the prerequisites section.

    ```
    (sudo)   # sudo cp /tmp/podium/config/qlikContainers.service /etc/systemd/system/
    (sudo)   # sudo systemctl daemon-reload
    (sudo)   # sudo systemctl enable qlikContainers.service
    ```

Congratulations! The base Qlik Catalog software installation is now complete.

If your cluster uses any combination of Kerberos, SAML, Impersonation or Active Directory, please refer to each section that relates to these topics for configuration. Note: You’ll need to sync Active Directory users first before enabling Kerberos to ensure that users registered in Qlik Catalog are authorized to login.

The installer created the Qlik Catalog metadata database (podium_md). The default user (role) for this database is podium_md, with a default password of “nvs2014!”.

<u>Important Notes</u>

- Any **JDBC drivers** for your licensed RDBCS should be place in the following directory:

  $TOMCAT_HOME/webapps/qdc/WEB-INF/lib

- When running Qlik Catalog in unbundled mode, the expectation is that all cluster configuration is found on the classpath generated by executing ‘hadoop classpath’. Typically, a directory like /etc/hadoop/conf is at the front of the classpath and it contains files like core-site.xml.

<u>No Hadoop configuration files should be placed in the Qlik Catalog WEB-INF/classes directory</u>, nor should symlinks be created from WEB-INF/classes to *-site.xml files elsewhere.

<u>Do not add any libraries from the Hadoop ecosystem, including Hive classes and the Hive JDBC jar, to the classpath</u>, unless directed to do so by Qlik personnel.

REMINDER: Reboot Procedure

When the Qlik Catalog server is restarted, several required processes must be started.

The following are **autostarted** services. To manually restart these services:

- ∉ PostgreSQL: (sudo) # sudo systemctl restart qdc_pg-11.10.service
  - test: (sudo) # psql
- ∉ Docker: (sudo) # sudo systemctl restart docker.service
  - test: (sudo) # sudo docker ps

The following should be **autostarted** services, if configured correctly above. To manually restart these services:

- ∉ Qlik Licenses & Engine Containers: (sudo) # sudo systemctl restart qlikContainers.service
  - test: (sudo) # sudo docker inspect -f '{{.State.Running}}' qlikcore_qix-engine_1
- ∉ Tomcat: (sudo) # sudo systemctl restart tomcat.service

If the following were not configured to be autostarted services, they must be manually restarted after reboot. First, become the service user: (sudo) # sudo su - qdc

- ∉ Qlik Licenses & Engine Containers: (qdc) $ cd /usr/local/qdc/qlikcore && ./launch_qlikContainers.sh
  - test: (qdc) $ docker inspect -f '{{.State.Running}}' qlikcore_qix-engine_1
- ∉ Tomcat: (qdc) $ /usr/local/qdc/apache-tomcat-9.0.38/bin/startup.sh

4.2 Upgrade of Qlik Catalog 4.0 and Later

4.2.1 PostgreSQL Upgrade

If you are upgrading from Qlik Catalog June 2019 and you want **long entity** and **source name** support, you must upgrade to the custom build of PostgreSQL 11.; otherwise, you can continue using the version of PostgreSQL 11.10 that was installed with the Qlik Catalog June 2019 release.

If you are upgrading from Qlik Catalog 4.0.x, you must upgrade to the custom build of **PostgreSQL 11.10** included in the Qlik Catalog software download.

4.2.2 PostgresSQL Manual Database Creation

When “psql” is present on the edge node, the installer will perform a variant of the below actions. **Only perform these actions manually if the installer could not create the Podium metadata database in PostgreSQL** (see earlier section for manual Oracle database creation), or if the database is located on a different server.

1. Connect to PostgreSQL as **superuser** and create the Qlik Catalog database role:

```
$ psql template1 -U postgres -c "create role podium_md” with encrypted password 'input-new-role-password-here' createdb login;"
```

2. Connect to PostgresSQL as **podium_md** user and create the Qlik Catalog metadata database and schema

```
$ psql template1 -U podium_md -c "create database podium_md;"

$ psql podium_md -U podium_md -c "create schema podium_core;"
```

3. Initialize Qlik Catalog metadata tables in the above database schema

```
$ unzip -j podium-4.8-XXX.zip podium/config/core_ddl.txt -d /tmp/

$ psql podium_md -U podium_md -f /tmp/core_ddl.txt
```

4.2.3 Qlik Catalog Installer Upgrade Mode

The installer script has an upgrade mode, which also performs a backup of the WEB-INF/classes directory. Execute the following steps to perform an upgrade of Qlik Catalog June 2019 (4.2) and later:

1. The installer must be run as the Qlik Catalog service account:

   (sudo)   `# sudo su - qdc`

2. Stop Tomcat. Ensure it is no longer running.

```
(qdc)     # cd <tomcat home>
(qdc)     # ./bin/shutdown.sh
(qdc)     # ps -ef | grep Boot
```

3. **Backup** any manually copied Java library jars from <tomcat home>/webapps/qdc/WEB-INF/lib (e.g., JDBC drivers).

4. **Backup** the PostgreSQL database, in case the upgrade must be reverted.

   ```
   (sudo)     # pg_dump -U postgres --format=c --file=<backupFileName> podium_md
   ```

5. Unzip QDCinstaller.zip into a working directory

6. Copy the Qlik Catalog software zip file (podium-4.6-14761.zip) into the working directory

7. Change directory to the working directory. It should contain the following:

   ```
   podium-4.6-14761.zip  QDCinstaller.properties  QDCinstaller.sh  QDCinstaller.txt   QULA.txt  upgrade-scripts (directory)
   ```

8. Edit the installation parameters in QDCinstaller.properties -- additional documentation is present in that file

   - Only the following are used for upgrade: INSTALL_TYPE, QDC_HOME, TOMCAT_HOME, PODIUM_RELEASE_FILE, WEBAPP_NAME, POSTGRES_HOSTNAME, POSTGRES_IPADDRESS, database SUPERUSER_NAME and SUPERUSER_PASSWORD.

9. Run the shell script **with the "-u" argument**. You may abort the script at any time by entering control-C. The installer will begin by asking you to confirm the data you entered in QDCinstaller.properties -- it will also validate the parameters. It will expand the Qlik Catalog software zip and update the webapp. A **backup** of WEB-INF/classes is automatically made in <tomcat home>/backups. The file WEB-INF/classes/log4j.xml is automatically restored during upgrade.

   ```
   (qdc) # ./QDCinstaller.sh -u
   ```

10. **Restore** any manually copied Java library jars to <tomcat home>/webapps/qdc/WEB-INF/lib (e.g., JDBC drivers). If files were restored, restart Tomcat.

11. The Qlik Licenses container must be setup as a service to automatically start when Linux restarts.

    Possible edit: If the Qlik Catalog Qlik Sense integration directory is not /usr/local/qdc/qlikcore or the service user/group is not qdc, the file /etc/systemd/system/qlikContainers.service must be edited after the copy (cp) step below.

    The copy (cp) command below assumes the Qlik Catalog software (a.k.a., podium zip) has been expanded to /tmp -- the prerequisites section.

    ```
    (sudo)     # sudo cp /tmp/podium/config/qlikContainers.service /etc/systemd/system/
    (sudo)     # sudo systemctl daemon-reload
    (sudo)     # sudo systemctl enable qlikContainers.service
    ```

12. If previously using the QVD Import feature then **remove the existing *qlikcore.service* file**. The Qlik Engine container required for the QVD Import feature has been included in the unified *qlikContainers.service* file configured in step 11.

    (sudo) # sudo rm /etc/systemd/system/qlikcore.service

4.2.4 Non-Interactive ("Silent") Installation

Note: If upgrading from a Catalog version prior to 4.7 non-interactive mode is not allowed by the installer due to mandatory upgrade scripts which must be run manually.

To run the installer in non-interactive mode:

1. Edit the installation parameters in **QDCinstaller.properties** -- additional documentation is present in that file. Ensure the following are set -- the defaults should work for most. PODIUM_RELEASE_FILE is set to the current podium zip file. The properties SUPERUSER_NAME and SUPERUSER_PASSWORD are for Qlik Catalog PostgreSQL and by default are both "postgres".
   - INSTALL_TYPE -- **SINGLE**
   - QDC_HOME
   - TOMCAT_HOME
   - PODIUM_RELEASE_FILE
   - WEBAPP_NAME
   - POSTGRES_HOSTNAME
   - POSTGRES_IPADDRESS
   - SUPERUSER_NAME
   - SUPERUSER_PASSWORD
   - JAVA_HOME
   - PODIUM_BASE

2. the following options must be specified following the QDCinstaller.sh command:

   - -s (silent)
   - -a (Accept QULA)

   Example: ./QDCinstaller.sh -s -a

NOTE: Invoking the "-a" option indicating acceptance of the Qlik User License Agreement (QULA) is required to run the installer in silent mode. By selecting this installation option the user agrees to the following:

> BY DOWNLOADING, INSTALLING, OR OTHERWISE USING QLIK PRODUCTS, THE CUSTOMER ACKNOWLEDGES AND AGREES THAT THE USE OF ALL QLIK PRODUCTS IS SUBJECT TO THE TERMS AND CONDITIONS OF THE QLIK USER LICENSE AGREEMENT (QULA) FOUND ON https://qlik.com. ANY SUCH USE WILL CONSTITUTE CUSTOMER'S ACCEPTANCE AND RESULT IN A BINDING AND LEGALLY ENFORCEABLE AGREEMENT BETWEEN THE CUSTOMER AND THE QLIK ENTITY IDENTIFIED IN TABLE 1 OF THE AGREEMENT ("QLIK"). IF YOU ACCEPT THESE TERMS ON BEHALF OF ANY CORPORATION, PARTNERSHIP, OR OTHER ENTITY, YOU REPRESENT AND WARRANT THAT YOU ARE

AUTHORIZED TO LEGALLY BIND SUCH ENTITY TO THIS AGREEMENT AND SUCH ENTITY IS DEEMED THE CUSTOMER HEREUNDER.

4.3 Upgrade of Qlik Catalog 3.4 or Earlier

Postgres Upgrade Required: You must upgrade to the custom build of **PostgreSQL 11.10** included in the Qlik Catalog software download.

With the release of Qlik Catalog 4.0, Hadoop libraries are no longer bundled (included) with Qlik Catalog. As Qlik Catalog Multi-Node is installed on an edge node, the Hadoop libraries and configuration files can be found on the Hadoop classpath of the edge node (e.g., run command 'hadoop classpath'). The following process can be used to upgrade a "bundled" version of Qlik Catalog 3.4 or earlier. Execute it as the Qlik Catalog service account.

1. Become the service account

   ```
   # sudo su - qdc
   ```

2. Stop Tomcat. Ensure it is no longer running.

   ```
   # cd <tomcat home>
   # ./bin/shutdown.sh
   # ps -ef | grep Boot
   ```

3. **Backup** the <tomcat home>/webapps/podium/WEB-INF/classes folder to a backup directory of your choice.

4. **Backup** any manually copied Java library jars from WEB-INF/lib (e.g., JDBC drivers).

5. **Backup** the PostgreSQL database, in case the upgrade must be reverted.

   (sudo)
   ```
   # pg_dump -U postgres --format=c --file=<backupFileName> podium_md
   ```

6. Move <tomcat home>/webapps/podium/WEB-INF/classes/core_env.properties to <tomcat home>/conf

7. Modify this property in <tomcat home>/conf/core_env.properties:

   ```
   entitymanager.packages.to.scan=com.nvs.core,com.nvs.reporting,com.podiumdata.sentinel,com.
   podiumdata.propertysystem.model,com.podiumdata.dashboard.model
   ```

8. Delete the old Qlik Catalog webapps:

   ```
   # cd <tomcat home>/webapps
   # rm -rf podium
   # rm -f podium.war
   # rm -rf podium-webhelp
   # rm -f podium-webhelp.war
   ```

9. Extract the new Qlik Catalog unbundled webapp (aka localhadoop) into the webapps directory:

   ```
   # cd <tomcat home>/webapps
   # unzip -j <full path to>/podium-4.6-14761.zip podium/lib/podium_localhadoop.war -d .
   ```

```
# mv podium_localhadoop.war podium.war
# mkdir podium
# cd podium
# unzip ../podium.war
```

10. Extract the Qlik Catalog webhelp:

```
# cd <tomcat home>/webapps
# unzip -j <full path to>/podium-4.6-14761.zip podium/lib/podium-webhelp.war -d .
# mkdir podium-webhelp
# cd podium-webhelp
# unzip ../podium-webhelp.war
```

11. **Restore** WEB-INF/classes/log4j.xml from the backup.

12. **Restore** any manually copied Java library jars to WEB-INF/lib (e.g., JDBC drivers).

13. Edit <tomcat home>/conf/catalina.properties and set:

```
shared.loader=${PODIUM_EXTERNAL_CLASSPATH}
```

14. Use and **adapt** the appropriate Tomcat setenv.sh from the reference section below to update <tomcat home>/bin/setenv.sh

15. Restart Tomcat.

4.4 Reference Tomcat setenv.sh

The following Tomcat setenv.sh files are generated by the installer and written to /bin. They may be educational and are required when manually upgrading from a 3.4 or earlier "bundled" install.

Cloudera

Validate that the "/opt/cloudera/parcels/CDH/lib/hive/lib" directory exists. If not, validate and use "/usr/lib/hive/lib" below.
Ensure /usr/java/latest points to the correct JVM; if not, update JAVA_HOME to be specific.

```
export JAVA_HOME=/usr/java/latest

export PODIUM_EXTERNAL_CLASSPATH=`hadoop classpath | sed -e 's/:/,/g' -e 's/*/*.jar/g' -e 's/*.jar.jar/*.jar/g'`
export PODIUM_EXTERNAL_CLASSPATH="$PODIUM_EXTERNAL_CLASSPATH,/opt/cloudera/parcels/CDH/lib/hive/lib/*.jar"
export PODIUM_EXTERNAL_CLASSPATH="$CATALINA_BASE/podiumLibPreHadoop/*.jar,$PODIUM_EXTERNAL_CLASSPATH, $CATALINA_BASE/podiumLibPostHadoop/*.jar"

export JAVA_OPTS="$JAVA_OPTS -Xms1g -Xmx8g -Djava.library.path=/lib64/:/opt/cloudera/parcels/CDH/lib/hadoop/lib/native/"
export JAVA_OPTS="$JAVA_OPTS -
DPODIUM_EXTERNAL_CLASSPATH=$PODIUM_EXTERNAL_CLASSPATH -
Dpodium.conf.dir=$CATALINA_BASE/conf"
export JAVA_OPTS="$JAVA_OPTS -Djavax.security.auth.useSubjectCredsOnly=false -
Djava.security.egd=file:/dev/./urandom"

export SPARK_MASTER=yarn-client
```

Hortonworks

Update HDP_VERSION is set to the correct version (see out of "hadoop classpath", verify directory).
Ensure /usr/java/latest points to the correct JVM; if not, update JAVA_HOME to be specific.

```
export JAVA_HOME=/usr/java/latest

export HDP_VERSION=2.6.5.0-292

export PODIUM_EXTERNAL_CLASSPATH=`hadoop classpath | sed -e 's/:/,/g' -e 's/*/*.jar/g' -e
's/*.jar.jar/*.jar/g'`
export
PODIUM_EXTERNAL_CLASSPATH="$PODIUM_EXTERNAL_CLASSPATH,/usr/hdp/$HDP_VERSI
ON/hive/lib/*.jar"
export
PODIUM_EXTERNAL_CLASSPATH="$CATALINA_BASE/podiumLibPreHadoop/*.jar,$PODIUM_EX
TERNAL_CLASSPATH, $CATALINA_BASE/podiumLibPostHadoop/*.jar"

export JAVA_OPTS="$JAVA_OPTS -Xms1g -Xmx8g -Dhdp.version=$HDP_VERSION -
Djava.library.path=/lib64/:/usr/hdp/$HDP_VERSION/hadoop/lib/native" export
JAVA_OPTS="$JAVA_OPTS -
DPODIUM_EXTERNAL_CLASSPATH=$PODIUM_EXTERNAL_CLASSPATH -
Dpodium.conf.dir=$CATALINA_BASE/conf"
export JAVA_OPTS="$JAVA_OPTS -Djavax.security.auth.useSubjectCredsOnly=false -
Djava.security.egd=file:/dev/./urandom"

export SPARK_MASTER=yarn-client
```

4.5 Distribution Specific Configuration Parameters

4.5.1 Hortonworks Configuration

Add the following property to the cluster's core-site.xml. This setting ensures that new sessions are created for each Pig Tez job. If TRUE, jobs will re-use the same session, which can cause execution issues.

```
<property>
  <name>pig.tez.session.reuse</name>
  <value>false</value>
</property>
```

4.5.1.1 Hortonworks Spark Configuration

Only Spark 1.x is supported -- Spark 2 found on HDP 3.0 is not supported. The standard Spark configuration property spark.yarn.jar must be set in one of the cluster *-site.xml files, or it must be set in core_env.properties (documented below). The preference is to use the spark-hdp-assembly.jar that is distributed with Qlik Catalog. However, if there are classpath dependency issues, the Spark assembly found on the edge node may be substituted (e.g., /usr/hdp/2.6.4.0-91/spark/lib/spark-assembly-1.6.3.2.6.4.0-91-hadoop2.7.3.2.6.4.0-91.jar).

1. Extract spark-hdp-assembly.jar to the “podiumLibPreHadoop” directory in the Tomcat home directory:

```
 # unzip -j podium-4.6-14761.zip podium/thirdParty/spark-hdp-assembly.jar -d <tomcat
home>/podiumLibPreHadoop/
```

2. Copy spark-hdp-assembly.jar from the edge node to an HDFS folder:

```
 # hadoop fs -mkdir -p /apps/spark
 # hadoop fs -copyFromLocal <tomcat home>/podiumLibPreHadoop/spark-hdp-assembly.jar
/apps/spark/
```

3. Specify the full path to spark-hdp-assembly.jar in core_env.properties:

```
spark.yarn.jar=hdfs://<host:port>/apps/spark/spark-hdp-assembly.jar
```

4.5.1.2 Hortonworks Tez Configuration

If Tez is already installed you don’t have to perform any steps in this section. Make sure your Tez version works with your Pig version by checking the corresponding Apache project website.

1. Copy the edge node's tez.tar.gz to an HDFS folder:

```
# hadoop fs -mkdir -p /apps/tez
# hadoop fs -copyFromLocal /usr/hdp/2.6.3.0-235/tez/lib/tez.tar.gz /apps/tez/
```

2. Specify the full path of Tez assembly archive in core_env.properties

```
tez.lib.uris=hdfs://nameservice1:8020/apps/tez/tez.tar.gz
```

4.5.1.3 Hortonworks Ranger Configuration

Qlik Catalog naturally honors Ranger policies due to only use Hadoop standard API’s. If dynamic Ranger policy creation is needed due to using a specific Hive service account (rare) please see Ranger Appendix and then return directly to this section.

As stated before, a specific directory structure in HDFS needs to be provided for Qlik Catalog to store data. All Qlik Catalog service accounts and/or impersonated users must have Ranger policies with RWX to these directories in order to use most Qlik Catalog capabilities.

4.5.2 Cloudera Configuration

4.5.2.1 Spark on Cloudera 5

Only Spark 1.x is supported. Please follow the above instructions for “Hortonworks Spark Configuration”, replacing “spark-hdp-assembly.jar” with “spark-cdh-assembly.jar”.

NOTE: Prepare using Spark is not supported on Cloudera 6 -- please see below section for Tez support on Cloudera 6.

4.5.2.2 Sentry Add JAR

This is a one-time configuration required if Sentry is enabled. Qlik Catalog requires that this permission be set via Cloudera Manager. If this is not set, the 'ADD JAR' command will throw 'Insufficient privileges to execute add', even for an Admin role. Obfuscation will not work unless this setting is enabled. Read here about this limitation.

Add the JAR globally using hive.aux.jars.path and restart Hive. If Sentry is enabled, podium.hive.udf.jar.location must be empty or comment out this property entirely to disable the ADD JAR statements. Set the following core_env.property to enable obfuscation: hive.service.user = hive

To ADD JAR globally when Sentry is enabled:

1. Copy <tomcat home>/webapps/podium/WEB-INF/lib/qdc-hudf-<version>.jar to the host on which HiveServer2 is running. Save the JAR to any directory you choose, give the hive user read, write, and execute access to this directory, and make a note of the path (for example, /opt/local/hive/lib/qdc-hudf-4.8.jar).

2. In the Cloudera Manager Admin Console, go to the Hive service.

3. Click the Configuration tab.

4. Expand the Service-Wide > Advanced categories.

5. Configure the Hive Auxiliary JARs Directory property with the HiveServer2 host path from Step 1
   example: /opt/local/hive/lib/

6. Click **Save Changes**. The JARs are added to HIVE_AUX_JARS_PATH environment variable.

7. Redeploy the Hive client configuration; from the Actions menu at the top right of the Hive service page select **Deploy Client Configuration.**

8. Restart the Hive service. **Hive Auxiliary JARs Directory property** If the is configured but the directory does not exist, HiveServer2 will not start.

9. Grant privileges on the JAR files to the roles that require access. Login to Beeline and use the Hive SQL GRANT statement to do so. For example:

   ```
   GRANT ALL ON URI 'file:///opt/local/hive/lib/qdc-hudf-4.8.jar' TO ROLE EXAMPLE_ROLE;
   ```

See the following for more info: https://www.cloudera.com/documentation/enterprise/5-9-x/topics/cm_mc_hive_udf.html

4.5.2.3 Sentry

Qlik Catalog naturally honors Sentry policies to only use Hadoop standard API's. If dynamic Sentry policy creation is needed due to using a specific Hive service account (rare), please see Sentry section in the appendix and then return directly to this section.

As stated before, a specific directory structure in HDFS needs to be provided for Qlik Catalog to store data. All Qlik Catalog service accounts and/or impersonated users must have Ranger policies with RWX to these directories in order to use most Qlik Catalog capabilities.

4.5.2.4 Cloudera 6 Tez Configuration

Prepare using Tez is supported for Cloudera 6. For Cloudera 5, please see the above section and use Prepare on Spark.

1. Copy Tez 0.9.2 to an HDFS folder:

```
# unzip podium-<VERSION>-<BUILD>.zip podium/thirdParty/tez-0.9.2-cdh-6.tar.gz -d /tmp/
# hadoop fs -mkdir -p /apps/tez
# hadoop fs -copyFromLocal /tmp/podium/thirdParty/tez-0.9.2-cdh-6.tar.gz /apps/tez/
```

2. Specify the full path of Tez assembly archive in core_env.properties:

```
tez.lib.uris=hdfs://<NAME_SERVICE>:8020/apps/tez/tez-0.9.2-cdh-6.tar.gz
```

5.0 Qlik Catalog Software Installation Reference

All following sections are not part of the base installation runbook and as such are treated in isolation.

The core_env.properties file is used to indicate to the Qlik Catalog application all the primary and secondary configuration parameters necessary for desired operation. The file is found at <tomcat home>/conf/core_env.properties (on legacy installs, it may still be found at <tomcat home>/webapps/podium/WEB-INF/classes). It is self describing in that the explanation for all parameters is included in the file itself. Please see the file and modify the parameters as needed. Once Qlik Catalog is launched, you can edit the file and then use the button in the admin section to refresh core_env.properties if any changes are made, which prevents having to restart Tomcat for these changes.

5.1 Masking and Obfuscation

The Qlik Catalog code performing masking and obfuscation must be placed in a directory accessible to the cluster, as well as any needed dictionary files.

1. Define or use an existing HDFS location for the .jar file and dictionary files that enable the data masking/obfuscation capability. This must be a location all users, or the service user that runs on the cluster, have access to. Example:

   ```
   # hadoop fs –mkdir /apps/podium
   ```

2. Copy qdc-hudf-4.8.jar and any dictionary files to the directory you created above.

   ```
   # hadoop fs –put US_census_first_names.txt /apps/podium/
   # hadoop fs –put <tomcat home>/webapps/podium/WEB-INF/lib/qdc-hudf-<version>.jar
   /apps/podium/qdc-hudf-4.8.jar
   ```

3. Open core_env.properties and modify the following properties that reference the file and directory above

   ```
   podium.hive.udf.jar.location=hdfs://apps/podium/qdc-hudf-4.8.jar
   podium.obfuscation.dictionary.hdfs.path=hdfs://apps/podium/
   ```

5.2 Kerberos Configuration

When kerberos is enabled, Qlik Catalog specifies a set of rules in the property hadoop.security.auth_to_local in core-site.xml that maps kerberos principals to local OS users. Theoretically a customer can map all kerberos principals to a few fixed OS users that already exist but normally its not done. Usually the rules are configured to just strip the domain names from the principals to get the local user name. Those local users must be OS users on all nodes. If kerberos is configured with Active Directory, this process is simplified as the AD users are already available as OS users on all nodes (e.g., `id adccuser` returns the AD user/group).

5.2.1 Running Qlik Catalog in a Kerberized Environment

1. Create a service account for the Qlik Catalog application on all nodes in the cluster as well as the Qlik Catalog nodes. Example: Qlik Catalog. This user, as well as all other users who will execute work through Qlik Catalog if impersonation is enabled, must exist on all nodes and be part of appropriate access right groups and aligned with Ranger or Sentry policies.

2. Create a Qlik Catalog principal and principals for all users who need access if impersonation will be turned on.

3. Make sure user home directories exist on HDFS for all accounts

4. Make sure HDFS directory ownership and access rights align via policies for the users who will be accessing them.

5. Install Kerberos workstation on the Qlik Catalog server

   ```
   # sudo yum install -y krb5-workstation.x86_64
   ```

6. Ensure these files are listed in the core_env.properties file in the hadoop.conf.files property

   ```
   hadoop.conf.files=core-site.xml,hdfs-site.xml,mapred-site.xml,yarn-site.xml,ssl-client.xml
   ```

7. If SSL is enabled also copy the hadoop.truststore file from the cluster to the Qlik Catalog classes folder on the Qlik Catalog node. If SSL is not enabled, skip this step.

   Podium classpath folder
   <tomcat home>/webapps/podium/WEB-INF/classes

8. Generate a Qlik Catalog keytab file on the cluster from kadmin.local command line. Use norand to prevent a new random password from being generated.

   ```
   xst -kt -norand podium
   ```

9. Copy the keytab file that was just created to the Qlik Catalog server. It will have been create in the local directory where kadmin.local was started.

10. On the Qlik Catalog server, make podium.keytab read-only for the Qlik Catalog user

11. Modify the Qlik Catalog core_env.properties file to include the JDBC URI for Hive as Kerberos Authentication

    ```
    jdbc:hive2://master.hostname.acme.net:10000/default;principal=hive/master.hostname.acme.net@hostname.acme.net

    jdbc:hive2://hdmduv0005.machine.test.group:10010/podium_test_01;principal=hive/hdmduv0005.machine.test.group@APPGLOBAL.CHIPCORP.COM

    jdbc:hive2://hdmduv0005.machine.test.group:10010/podium_test_01;principal=hive/_HOST@APPGLOBAL.CHIPCORP.COM
    ```

12. On the Qlik Catalog server kinit as the service account

kinit podium-kt podium.keytab

13. Start Qlik Catalog as the service account

14. Add kinit to crontab for lease refresh

crontab -e

- 0 0,5,10,15,20 * * * kinit podium -kt /home/podium/podium.keytab

15. Add kinit to the .bash_profile for the service user

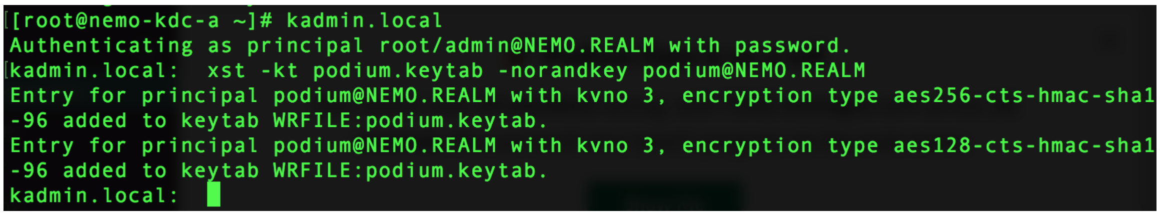
kinit podium -kt /home/podium/podium.keytab

TIP: The klist command will provide details of who the user is kinited as. To get the exact Hive realm user for the jdbc connection url in core_env; run kadmin.local on the kerberos server and execute listprincs example: kadmin.local:listprincs

kadmin.local is only going to work if the user is logged in to the actual kerberos server
once logged into the kerberos server via kadmin.local OR remotely using kadmin, user can export keytabs with the following syntax:

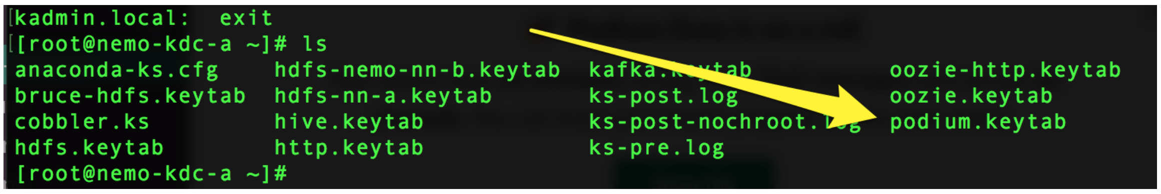
xst -kt <keytab-file-name> -norandkey user@KERBEROS.REALM

example: xst -kt podium.keytab -norandkey podium@NAME.REALM
if using kadmin utility remotely, a user would need to be logged in w/ admin permission to export keytabs



```
[root@nemo-kdc-a ~]# kadmin.local
Authenticating as principal root/admin@NEMO.REALM with password.
kadmin.local:  xst -kt podium.keytab -norandkey podium@NEMO.REALM
Entry for principal podium@NEMO.REALM with kvno 3, encryption type aes256-cts-hmac-sha1
-96 added to keytab WRFILE:podium.keytab.
Entry for principal podium@NEMO.REALM with kvno 3, encryption type aes128-cts-hmac-sha1
-96 added to keytab WRFILE:podium.keytab.
kadmin.local:
```

extracted keytab will be located in whatever directory you ran the kadmin command from:



```
kadmin.local:  exit
[root@nemo-kdc-a ~]# ls
anaconda-ks.cfg    hdfs-nemo-nn-b.keytab  kafka.keytab           oozie-http.keytab
bruce-hdfs.keytab  hdfs-nn-a.keytab       ks-post.log            oozie.keytab
cobbler.ks         hive.keytab            ks-post-nochroot.log   podium.keytab
hdfs.keytab        http.keytab            ks-pre.log
[root@nemo-kdc-a ~]#
```

5.2.2 Using Kerberos to Authenticate as a Qlik Catalog User

Kerberos provides transitive authentication—the client authenticates once, and when he requests subsequent services the servers are aware of, prior authentication is extended to those services. A Kerberos user is automatically authenticated in Qlik Catalog as long as the username is the same in both applications and the local Qlik Catalog user has a valid Qlik Catalog group association.

Ensure Java Cryptography Extension (JCE) unlimited security jars are up to date on your system:

```
US_export_policy.jar
local_policy.jar
```

They can be downloaded from:
http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

5.2.2.1 Enable Kerberos in core_env.properties

The following properties are used to enable Kerberos for UI (or API) logon.

```
# Authentication modes (case-insensitive): PODIUM, KERBEROS, SAML, NONE (no authentication
# required). This is for authenticating access to the Qlik Catalog server (UI or API). Default: PODIUM
authentication.mode=KERBEROS

# Kerberos realm / Active Directory domain name. This is appended to the username entered into
# the login page. For legacy reasons, it is the same name as the Java system property above.
java.security.krb5.realm=QLIK_SECURE.COM
```

Debug should be enabled when a user faces an issue, this property is enabled for the application to run/generate logs, otherwise the log files will unnecessarily log all details of operations being executed.

```
debug=false
```

Optimal Kerberos properties for single realm including ticket lifecycle and encryption.
Add the following properties to: krb5.conf

```
dns_lookup_kdc = false
dns_lookup_realm = false
ticket_lifetime = 86400 (user-defined)
renew_lifetime = 604800 (user-defined)
forwardable = true
default_tgs_enctypes = aes256-cts-hmac-sha1-96(user-defined)
default_tkt_enctypes = aes256-cts-hmac-sha1-96(user-defined)
permitted_enctypes = aes256-cts-hmac-sha1-96(user-defined)
udp_preference_limit = 1
useFirstPass=true
doNotPrompt=true
```

5.2.2.2 Kerberos: Setting Up Cross Realm Authentication

To allow users with the same username to sign on from a different realm in the same company, comment out the following property in Qlik Catalog core.env:

#another.realm.username.pattern=^([A-Za-z0-9]+)([._-]([0-9A-Za-z_-]+))*@([A-Za-z0-9]+)([.]([0-9A-Za-z]+))*([.]([A-Za-z0-9]+){2,4})$

5.2.2.3 Configuring Publish for Target HDFS (Cross Realm)

All name nodes and node managers for Source cluster and Target cluster need the following lines added to /etc/krb5.conf
(default Kerberos config file):

```
target-nn-a.corp.podiumdata.com = <PRINCIPAL>
target-cloudera.corp.podiumdata.com = <PRINCIPAL>

<sourceclustername>-nn-a.corp.<mycompany>.com=<REALMNAME>
<sourceclustername>-nn-b.corp.<mycompany>.com=<REALMNAME>
<sourceclustername>-cloudera.corp.<mycompany>.com=<REALMNAME>
```

5.3 Impersonation Configuration

Impersonation support enables Qlik Catalog running as a ServiceUser with executive access to all commands and directories to execute tasks/run jobs on the cluster on behalf of a logged in user in a secured way. Hence Any permissions or privileges as applicable for that user (via Sentry or similar tools) are honored. When impersonation is enabled:

The user logged in to Qlik Catalog will be the user impersonated by ServiceUser (for all interactions with cluster including MapReduce jobs, HDFS operations etc.) User authentication (when logging into Qlik Catalog) will be performed using Kerberos

5.3.1 Cluster Configuration Requirements

Configuring Hadoop to Allow Impersonation By Creating Proxy Users
To allow ServiceUser to impersonate, the following settings must be configured in core-site.xml. For the following example, assume that ServiceUser is 'podium'. The value * can be used for 'any':

hadoop.proxyuser.podium.hosts==(machine hosts from which proxy/impersonation is allowed)
hadoop.proxyuser.podium.groups==(user groups that can be proxied/impersonated)

example:

```
hadoop.proxyuser.podium.groups==group1,group2
hadoop.proxyuser.podium.hosts==domain1.mycompany.net, domain2.mycompany.net
hadoop.proxyuser.podium.hosts==67.186.188.83,67.186.188.91-98
```

(hadoop.proxyuser.<serviceuser>.hosts accepts IP address lists and IP address ranges in CIDR format and/or host names)

If the cluster uses a KMS, modify the following properties for your environment, adding these properties in the KMS configuration
(kms-site.xml) to enable impersonation.

Note: you must remove **ServiceUser** from the below properties and substitute in the service account name used as a proxy user for impersonation.

```
<property>
<name>hadoop.kms.proxyuser.ServiceUser.users</name>
<value>*</value>
</property>
<property>
<name>hadoop.kms.proxyuser.ServiceUser.groups</name>
<value>*</value>
</property>
<property>
<name>hadoop.kms.proxyuser.ServiceUser.hosts</name>
<value>*</value>
</property>
```

5.3.2 Cloudera Specific--Configuring Apache Sentry to allow Impersonation

1. If Apache Sentry is configured for Hive, the following additional requirements must be met before loading data:
2. Sentry roles must be created for groups of all the users in the application— these groups are the OS or Active Directory groups that a Kerberos user belongs to.
3. The roles should at least have Select permissions to access Qlik Catalog Receiving folder.
4. Individual roles should be given permissions on the Hive database for the sources users will load data to.
5. If users are allowed to create new sources in Qlik Catalog, either:
    - Users should be given CREATE DATABASE permission in Sentry
    - The Hive database should first be created by the ServiceUser and then permission to insert data should be granted to other users
    - Add the following property in core_env.properties to run DDL statements as admin user (in example, as 'podiumuser')
    - hive.admin.user=podiumuser

5.3.3 Hortonworks Specific--Configuring Ranger to Allow Impersonation

If Apache Ranger is configured for Hive, the following additional requirements must be met before loading data:

1. Ranger policies should be created for all users/groups in the application— these groups of users are the OS or Active Directory groups that a Kerberos user belongs to.
2. Individual roles should be given permissions on the Hive database for the sources users will load data to.
3. Set the flag hive.server2.enable.doAs=false in hive-site.xml in Ambari [If old property hive.server2.enable.impersonation is in Ambari, remove it] -- Impersonation is enabled through core_env properties (enable.impersonation=true)
4. If users are allowed to create sources in Qlik Catalog, either the users should be given CREATE permission in Ranger for all databases (asterisk * selects ALL) or

the Hive database should first be created by the ServiceUser and then the permission to insert data should be granted to other users.

5. Add hdfs and hive users to wildcard policies defined in hdfs and hive ranger services.

5.3.4 Qlik Catalog Settings to Enable Impersonation

Set the following property in core_env.properties to enable impersonation:

```
enable.impersonation=true
enable.delegation=false
```

When impersonation is enabled, Hive Principal property must be set (modify to environment):

```
#hive.principal=hive/domain.corp.mycompany.com@MYCOMPANY_SECURE.COM
```

When impersonation is enabled, a delegation token has to be obtained for the logged in Qlik Catalog user to access Hive. Authorization method is set as

```
auth=delegationToken
jdbc:hive2://domain.corp.mycompany.com:10000/default;auth=delegationToken
```

When impersonation is disabled; The Hive principal is automatically appended to the URL so it doesn't require a delegation token:

```
jdbc:hive2://domain.corp.mycompany.com:1000/default;principal=hive/domaincorp.mycompany.com@MYCO
MPANY_SECURE.COM
```

If the environment is not Kerberized (unusual for most clients) , the older url form and username/pw properties are set where the user executing the operations will be the one specified in username property:

```
distribution.uri=jdbc:hive2://domain.mycompany.com:10000/default
distribution.username=#0{VbhLwQqs3SgU86cJSIhA/w==}
distribution.password=#0{BKWujXAJ1nE6txneNeHdXw==}
```

5.3.5 User Rep (ServiceUser) Running Qlik Catalog Processes

The User running the Qlik Catalog process (referred to as ServiceUser or superuser at places, not to be confused with Qlik Catalog SuperUser role) and hence performing impersonation on the cluster must have rights to proxy other users
See above (Configuring Hadoop to Allow Impersonation By Creating Proxy Users**)**
The ServiceUser must have valid Kerberos credentials to be able to impersonate another user.

5.3.6 Users Logging into Qlik Catalog (requirements)

Users logged into Qlik Catalog must be valid Kerberos users and hence known to the cluster. Qlik Catalog allows a mix of local and AD/Kerberos-based users. However if a local user tries to perform a cluster operation and impersonation is enabled for that Qlik Catalog install, those operations may fail if the user is not known to the cluster.

5.3.7 Enabling Hive Service id for DDL Operations

Set the following property to run Hive DDL statements as admin user (in the following example, admin user is 'cruser') as opposed to running them as logged in user:

example:
hive.admin.user=cruser

5.4 Enabling SAML using Okta (Okta is optional)

Instructions below are a reference with examples. Modifications will be required for client-specific SAML authentication and client environment. In this example setup, Okta is used as the Identity Provider (IDP) while Qlik Catalog is the Service Provider (SP).

1. Log in to your Okta organization as a user with administrative privileges. There is a sample organization created with the following information
   URL: https://dev-519244.oktapreview.com/
   Username: catalog.user@gmail.com
   Password: Password
   You can also create a free Okta Developer Edition organization with your own email here: https://www.okta.com/developer/signup/

2. Click on the blue Admin button on the top right corner.

3. Click on the Add Applications shortcut in the right panel.

4. Click on the green Create New **App** button.

5. In the dialog that opens, select the **SAML 2.0** option, then click the green **Create** button

6. In Step 1 General Settings, enter the application name (e.g. HostName SAML Application) in App name field, then click the green Next button.

7. In Step 2 Configure SAML Paste the URLs like those given below into the Single Sign On URL field: (the sample urls are for internal Qlik Catalog install)

   https://hostname-qdc.qlik.com:8443/qdc/saml/SSO

   Then paste the URL below into the Audience URI (SP Entity ID) field:

   https://hostname-qdc.qlik.com:8443/qdc/saml/metadata

8. In Step 3 Feedback click the checkbox next to the text This is an internal application that we created then click the green **Finish** button.

9. You will now see the Sign On section of your newly created Spring Security SAML application

10. Copy the Identity Provider metadata link and paste it in the core_env.properties saml.metadata.provider.

example:
saml.metadata.provider=https://dev-519244.oktapreview.com/app/exk7y30wlbho83ej70h7/sso/saml/metadata

11. You can also add additional users in the **People** section. All these users will need to be added to Qlik Catalog application as well with the same username.

12. Open the core_env.properties and add this line to it.
    authentication.mode=SAML

13. Restart the Qlik Catalog application.

There are now multiple ways to log into Qlik Catalog using Okta SAML Provider

1. Log in to https://hostname-qdc.corp.qlik.com:8443/qdc/ as usual. It will redirect you to Okta IDP from where you will have to authenticate using username/password. After successful authentication it will redirect to Qlik Catalog
   Important! The user with the same username must exist in Qlik Catalog as well.

2. Login to your Okta account and on the home page, click on the application icon you just created. This will login to the Qlik Catalog application using the account you signed in with.

3. If you have already authenticated using Okta using some other application, you can directly send a POST request to https://hostname-qdc.qlik.com:8443/qdc/saml/SSO with the SAMLResponse parameter containing the value of the response from IDP.

5.5 Tomcat SSL Configuration

5.5.1 Local

Configure Tomcat to Support SSL or https as Local Web Server

1. **Generate Keystore** - Use 'keytool' command to create a self-signed certificate. During the keystore creation process, assign a password and fill in the certificate detail.

   Example:
   ```
   [root@hostname bin]# keytool -genkey -alias podium -keyalg RSA –keystore
   /home/hostname/catalogkeystore

   Enter keystore password:
   Re-enter new password:
   What is your first and last name?
   [Unknown]: <Peter>
   What is the name of your organizational unit?
   [Unknown]: <CC >
   What is the name of your organization?
   [Unknown]: <BestCompany>
   What is the name of your City or Locality?
   [Unknown]: <Boston>
   What is the name of your State or Province?
   [Unknown]: <MA>
   What is the two-letter country code for this unit?
   [Unknown]: 12
   Is CN=Peter, OU=CC, O=BestCompany, L=Boston, ST=MA, C=12 correct?
   [no]: yes
   Enter key password for <catalog>
   (RETURN if same as keystore password):
   Re-enter new password:
   ```

This process created a certificate ('catalogkeystore') located at 'home/hostname/catalogkeystore'(the address that was provided in the keytool command).

Certification Details
Use same 'keytool' command to list the existing certificate's details:

Example:

```
[root@dopey bin]# keytool -list -keystore /home/hostname/catalogkeystore
Enter keystore password:
Keystore type: JKS
Keystore provider: SUN
Your keystore contains 1 entry
podium, Dec 14, 2015, PrivateKeyEntry,
Certificate fingerprint (SHA1): E7:6F:31:29:6F:87:29:6B:D7:6F:1C:E6:96:6E:3D:DB:D8:CF:C1:35
```

2. **Add Connector in server.xml -** Locate your Tomcat's server configuration file at $Tomcat\conf\server.xml; modify it by adding a connector element to support for SSL or https connection as follows:

   Search for comment: <!—"Define a SSL HTTP/1.1 Connector on port 8443"->

   Add following code (after the comments searched above)

   ```
   <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
   server="Unknown Application Server"
   maxThreads="150" scheme="https" secure="true"
   clientAuth="false" sslProtocol="TLS"
   keystoreFile="path/to/keystore"
   keystoreType="JKS"
   keystorePass="pass4keystore"
   keyPass="pass4key"/>
   ```

   Note: The password (for both 'keystorePass' and 'keyPass') must be the same passwords given when the certificate was generated.

   Make sure that redirect port is available for the connector with the default (http) port you are using.
   For example, default server.xml may show:

   ```
   <Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>
   ```

HTTPS Redirect Configuration: 8080 is the http port and 8443 is the https port:

1. Start the server

2. Browse http://qdc-node-hostname:8080/qdc
3. If the application doesn't redirect to https, add the following code to $TOMCAT_HOME/conf/web.xml at the end of the file just before the web-app tag ends:

```
<security-constraint>
  <web-resource-collection>
  <web-resource-name>securedapp</web-resource-name>
  <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
```

```
    </user-data-constraint>
  </security-constraint>
```

4. Restart Tomcat

Troubleshooting
ISSUE: 'Server has a weak ephemeral Diffie-Hellman public key'
ERR_SSL_WEAK_SERVER_EPHEMERAL_DH_KEY

RESOLUTION:
Replace the connector-port-redirect code (in #2) with below code then restart the server and browse the URL. The following workaround (adding allowed ciphers to the SSL code in server.xml) will enable your site to run as a secure HTTPS site in Chrome and Firefox.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true
maxThreads="150" scheme="https" secure="true
clientAuth="false" sslProtocol="TLS"
keystoreFile="path/to/keystore"
keystoreType="JKS"
keystorePass="pass4keystore"
keyPass="pass4key"
ciphers="TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_AES_1
28_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_
SHA,TLS_ECDHE_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_
WITH_AES_256_CBC_SHA256,
TLS_RSA_WITH_AES_256_CBC_SHA,SSL_RSA_WITH_RC4_128_SHA"/>
```

What causes this error?
This error displays when the website is trying to set up a secure connection but is actually *in*secure. The Diffie-Hellman key exchange uses/requires 1024 bits of parameters and the SSL/TLS on the site is using a smaller, conflicting Diffie-Hellman group size. This conflict prevents the shared encryption 'handshake'. Try browsing the same URL in Internet Explorer if you don’t see this error, the site will likely work fine for IE. Chrome and Firefox enforce a stronger public key.

5.5.2 Distributed

Configure Tomcat to Support SSL or https for Distributed Applications (Running on Tomcat)

1. Generate Keystore
   Use 'keytool' command to create a self-signed certificate. During the keystore creation process, assign a password and fill in the certificate's detail.

   Example:

   ```
   [root@hostname bin]# keytool -genkey -alias podium -keyalg RSA –keystore /home/hostname/
   catalogkeystore

   Enter keystore password:
   Re-enter new password:
   What is your first and last name?
   [Unknown]: <Peter>
   What is the name of your organizational unit?
   [Unknown]: <CC>
   What is the name of your organization?
   [Unknown]: <BestCompany
   What is the name of your City or Locality?
   [Unknown]: <Boston>
   What is the name of your State or Province?
   [Unknown]: <MA>
   What is the two-letter country code for this unit?
   [Unknown]: 12
   Is CN=Peter, OU=CC, O=BestCompany, L=Boston, ST=khi, C=12 correct?
   [no]: yes
   Enter key password for <catalog>
   (RETURN if same as keystore password):
   Re-enter new password:
   ```

   This process created a certificate ('catalogkeystore') located at 'home/hostname/catalogkeystore', the address that was provided in the keytool command.

   Certification Details
   Use same 'keytool' command to list the existing certificate's details:

   ```
   Example:
   [root@dopey bin]# keytool -list -keystore /home/hostname/catalogkeystore
   Enter keystore password:
   Keystore type: JKS
   Keystore provider: SUN
   Your keystore contains 1 entry
   catalog, Dec 14, 2020, PrivateKeyEntry,
   Certificate fingerprint (SHA1): E7:6F:31:29:6F:87:29:6B:D7:6F:1C:E6:96:6E:3D:DB:D8:CF:C1:35
   ```

2. Add **Connector in server.xml**
   Locate your Tomcat's server configuration file at $Tomcat\conf\server.xml; modify it by adding a connector element to support for SSL or https connection as follows:

   Update the code for defined services in server.xml as described below. Note that if there is more than one service defined in server.xml the redirect ports should be different for each of the defined services (i.e., multiple applications running on various ports 8081, 8082, etc.) require redirection within each of the services with different redirect ports (8444, 8445, etc.)

```
<Service name="catalog">
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" compression="off"
compressableMimeType="text/html,text/xml,text/plain,text/css,text/javascript,application/javascript,applicatio
n/json" />
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="/usr/home/hostname/catalogkeystore"
keystoreType="JKS"
keystorePass="password"
keyPass="password"/>
<Engine name="Catalina80" defaultHost="localhost">
<Host name="localhost" appBase="qdc" unpackWARs="true" autoDeploy="true" />
[Where 8080 is the http port and 8443 is the https port]
```

Notes: The location of 'keystorefile' must be the same that was used in the 'keytool'command. The password (for both 'keystorePass' and 'keyPass') must be the same passwords given when the certificate was generated.

3. Constraints in web.xml
   To make your web applications work on SSL, locate your tomcat's web.xml file at $Tomcat\conf\web.xml and modify it by adding the following code (before web-app tag ends).

```
<security-constraint>
<web-resource-collection>
<web-resource-name>appfoldername</web-resource-name>
<url-pattern>/*</url-pattern>
</web-resource-collection>
<user-data-constraint>
<transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
</security-constraint>
```

   Important:
   - The url-pattern is set to /* so that any page resource from your application is secure (it can be only accessed with https).
   - Set <transport-guarantee>CONFIDENTIAL</transport-guarantee> to make sure your app will work on SSL.
   - If you want to turn off the SSL, you don't need to delete the code above from web.xml, simply change CONFIDENTIAL to NONE.
   - Note that if there are multiple apps running on different ports, security constraint must be defined for each one by reusing the same code and setting the web-resource-name tag with the correspondent app folder name.

4. **Finalize**
   Once all changes have been made, shutdown the server, restart it and browse the application with the existing URL that will automatically redirect to https.

5.6 Amazon Web Services EMR (Elastic MapReduce)

Qlik Catalog supports AWS EMR Managed Cluster Platform as a Hadoop distribution option for on-demand computing resources and services in the cloud with pay-as-you-go pricing. This functionality enables Qlik Catalog clients to spin-up a cluster on Amazon EMR using Qlik Catalog for on-demand data management, query and processing functions that are available when data sources are accessible to the cluster.

Users are able to select various types of AWS instances launched from an AMI (Amazon Machine Image) configured for environment and project needs with variable combinations of CPU, memory, storage and networking capacity. Performance options include Fixed or Burstable Performance Instances that allow users to choose and scale according to data delivery requirements.

What you can do with Qlik Catalog on AWS when the cluster is down

When the cluster is up, users are able manage their data as if running Qlik Catalog on physical on-premise servers. The following details some of the limitations that users will encounter when the AWS cluster is offline:

- **DataSource**
  - All metadata is viewable, loads will fail at the MapReduce phase, or the distribution phase if in local mode
- **Discovery**
  Everything except
  - Interactive query doesn't work
  - Profile/Sample data doesn't work if stored in Hive
- **Prepare**
  - Workflows are editable and new workflows can be added
  - Execution fails at the script execution phase for non-local, fails at the distribution phase for local
- **Publish**
  - New publish jobs and targets can be created
  - Publish fails at the data preparation (Hive) phase
- **Reports**
  - No issues
- **Security/Admin**
  - Everything except policy sync

Known Limitations

1. String Hive tables on true-type Parquet /ORC are not supported because EMR has its own version of Hive that does not support true-type Parquet/ORC
2. "Decorated" types (VARCHAR, DECIMAL) are not supported on Hive tables for Parquet/ORC because EMR has its own version of Hive that does not support these types
3. Kerberos & Impersonation are not an officially supported part of EMR yet, therefore it is also not supported for Qlik Catalog EMR
4. Qlik Catalog Prepare on Spark is not supported.

EMR Node Sizing

Running a MapReduce job on EMR takes a lot of memory even for small datasets, so EMR "worker" nodes should have at least 16 GB of memory.

Network

The Qlik Catalog server can be run anywhere so long as there is network connectivity between the EMR "master" node and the Qlik Catalog server. This includes anywhere within an AWS VPC or an internal network with a VPN connection to the VPC. The Qlik Catalog node must have network access to both the EMR "master" & "worker" nodes. Modify the EMR security group OR add the Master & Slave security groups to the Qlik Catalog node during creation of the Qlik Catalog node.

EMR Cluster Setup

- EMR clusters are expensive to run 24/7, so it is best to shutdown the cluster when it is not being used.
- The best way to bring a reusable cluster up & down in a consistent & repeatable manner is to use an AWS ***Cloudformation*** script.
- A working cloudformation template is available upon request.

Cluster Setup Considerations

1) Assign a ***secondary IP address*** to the EMR "master" node:

   1. Because the EMR "master" node will likely obtain a different internal IP address each time a cluster is launched, the cloudformation script should assign a static **secondary IP address** to the master node.
   2. The Qlik Catalog node should be configured to point at this secondary static address so that it can always find the cluster following subsequent launches & terminations. This is accomplished by editing the various ***-site.xml*** files on the Qlik Catalog node and replacing the internal hostname with the static secondary address.

2) ***Externalize*** Hive data which needs to persist beyond cluster terminations:

   - Use a valid S3 location for the hive warehouse data
   - Use a small RDS instance for the Hive metastore db

   Example Cloudformation Script:

```
"Configurations": [{
"Classification": "hive-site",
"ConfigurationProperties": {
"hive.mapred.mode": "nonstrict",
"hive.metastore.warehouse.dir": "s3://emr-
functional/warehouse/",
"javax.jdo.option.ConnectionURL": "jdbc:mysql:\/\/emr-
functional.cuyutghxiclj.us-east-
1.rds.amazonaws.com:3306\/hive?createDatabaseIfNotExist=true",
"javax.jdo.option.ConnectionDriverName":
"org.mariadb.jdbc.Driver",
"javax.jdo.option.ConnectionUserName": "username",
"javax.jdo.option.ConnectionPassword": "password"
```

   Manual Edits for Externalizing Hive Data (if not creating via cloudformation script):

   1). Add the following property and value into hive-site.xml:

```
<property>
   <name>hive.mapred.mode</name>
   <value>nonstrict</value>
</property>
```

2). ADD the following property to hive-site.xml:

hive.metastore.warehouse.dir

Property value should be a valid S3 location: s3://<S3-bucket-name>/

example:

```
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>s3://emr-hive-warehouse-test-bucket1/warehouse/</value>
    <description>location of default database for the warehouse</description>
        </property>
```

5.6.1 Qlik Catalog Configuration as EMR "Edge Node"

An EC2 "edge node" which will contain the Qlik Catalog application must be created using library files from the EMR cluster "master" node. You may may create an edge node using the following process:

1) Run Edge Node Script

Use the following bash script to copy the Hadoop libraries from the EMR master node to the Qlik Catalog edge node.

Note: edit the ***variables*** at the beginning of the script according to your AWS environment

```
--- BEGIN SCRIPT ---
#!/bin/bash

# Requires ssh key for master access to be set up prior to run
# Variables
emrMasterDns="172.31.0.93"
emrSshKey="/home/ec2-user/keypair.pem"
hadoopHome="/usr/lib/hadoop"

libDirs="/usr/lib/pig $hadoopHome /usr/share/aws/emr/emrfs /usr/lib/hadoop-lzo /usr/lib/hive
/usr/lib/tez /usr/lib/sqoop /usr/lib/hive-hcatalog /etc/tez/conf /usr/lib/spark /etc/spark/conf
/usr/lib/hadoop-mapreduce /usr/share/aws/aws-java-sdk"

instanceStorePath="/mnt"

additionalDirs="$instanceStorePath/var/lib/hadoop/tmp $instanceStorePath/tmp $instanceStorePath/s3

$instanceStorePath/var/log/pig /var/log/hive/user/ec2-user /var/aws/emr/"

bashRc="/home/ec2-user/.bashrc"

# 1. Install java and vim
sudo yum install -y vim java-1.8.0-*
```

```
# 2. Create package dirs and pull files from master node
for libDir in $libDirs; do
  echo "-------------------------"

  echo "Setting up $(echo $libDir | rev | cut -d"/" -f1 | rev )"
  sudo mkdir -p $libDir
  sudo chown -R ec2-user:ec2-user $libDir
  scp -r -i $emrSshKey hadoop@$emrMasterDns:$libDir/* $libDir
 done

# 3. Create supporting dirs and set ownership

for addDir in $additionalDirs; do
  echo "Setting up $addDir"
  sudo mkdir -p $addDir
  sudo chown -R ec2-user:ec2-user $addDir
 done

# 4. Add empty json object to userdata file
echo '{}' > /var/aws/emr/userData.json

# 5. Adjust the s3 buffer dir config value

sed -i 's#<value>/mnt/s3,/mnt1/s3</value>#<value>/mnt/s3</value>#g' /usr/lib/hadoop/etc/hadoop/core-
site.xml

# 6. Set environment variables

newPath=$PATH
 for libDir in $libDirs; do
  newPath="$newPath:$libDir/bin"
 done

echo 'export PATH="'$newPath'"' >> $bashRc
echo 'export JAVA_HOME="/etc/alternatives/jre"' >> $bashRc
echo 'export HADOOP_HOME="'$hadoopHome'"' >> $bashRc
echo 'export HADOOP_CONF_DIR="'$hadoopHome'/etc/hadoop"' >> $bashRc
echo 'export HADOOP_MAPRED_HOME="'$hadoopHome'"' >> $bashRc
echo 'export HADOOP_COMMON_HOME="'$hadoopHome'"' >> $bashRc
echo 'export HADOOP_HDFS_HOME="'$hadoopHome'"' >> $bashRc
echo 'export YARN_HOME="'$hadoopHome'"' >> $bashRc
echo 'export HADOOP_COMMON_LIB_NATIVE_DIR="'$hadoopHome'/lib/native"' >> $bashRc
echo 'export HIVE_HOME="/usr/lib/hive"' >> $bashRc

--- END SCRIPT ---
```

2) Edit Hadoop Classpath

The output of 'hadoop classpath' can be controlled by editing /usr/lib/hadoop/etc/hadoop/hadoop-env.sh.

Edit hadoop-env.sh so that is contains the following:

```
# set JAVA_HOME for the java implementation to use – Example below
# NOTE: This is an example – make sure the path defined matches location of java installation
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-11.0.9.11-2.el7_9.x86_64

# tez environment, needed to enable tez
export TEZ_CONF_DIR=/etc/tez/conf
export TEZ_JARS=/usr/lib/tez

# Add tez into HADOOP_CLASSPATH
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:${TEZ_CONF_DIR}:${TEZ_JARS}/*:${TEZ_JARS}/lib/*
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/lib/hadoop-lzo/lib/*"
export JAVA_LIBRARY_PATH="$JAVA_LIBRARY_PATH:/usr/lib/hadoop-lzo/lib/native"
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:/usr/share/aws/aws-java-sdk/*
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/emr/emrfs/lib
/*:/usr/share/aws/emr/emrfs/auxlib/*"

export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/ddb/lib/emr-ddb-hadoop.jar"
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/goodies/lib/emr-hadoop-goodies.jar"
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/kinesis/lib/emr-kinesis-hadoop.jar"

# Add CloudWatch sink jar to classpath
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/cloudwatch-sink/lib/*"



# Add security artifacts to classpath
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/security/conf:/usr/share/aws/emr/securi
ty/lib/*"

# client needed for Sqoop
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/lib/hadoop/client/*"

export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/lib/hadoop-mapreduce/*"
export HADOOP_OPTS="$HADOOP_OPTS -server -XX:OnOutOfMemoryError='kill -9 %p'"
export HADOOP_NAMENODE_HEAPSIZE=1740
export HADOOP_DATANODE_HEAPSIZE=757
export HADOOP_JOB_HISTORYSERVER_HEAPSIZE=2396
```

3) Edit .bashrc for Qlik Catalog service account

Within the service account home directory, append the following to the .bashrc file:

```
export
PATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/lib/pig/bin:/usr/lib/hadoop/bin:/usr/share/aws/emr/emrfs/bin
:/usr/lib/hadoop-lzo/bin:/usr/lib/hive/bin:/usr/lib/tez/bin:/usr/lib/sqoop/bin:/usr/lib/hive-
hcatalog/bin:/etc/tez/conf/bin"
```

```
export JAVA_HOME="/etc/alternatives/jre"
export HADOOP_HOME="/usr/lib/hadoop"
export HADOOP_CONF_DIR="/usr/lib/hadoop/etc/hadoop"
export HADOOP_MAPRED_HOME="/usr/lib/hadoop"
export HADOOP_COMMON_HOME="/usr/lib/hadoop"
export HADOOP_HDFS_HOME="/usr/lib/hadoop"
export YARN_HOME="/usr/lib/hadoop"
export HADOOP_COMMON_LIB_NATIVE_DIR="/usr/lib/hadoop/lib/native"
export HIVE_HOME="/usr/lib/hive"
export SPARK_HOME="/usr/lib/spark"
```

4) Tomcat setenv.sh

Edit/create $TOMCAT_HOME/bin/setenv.sh. It should contain the text below:

NOTE: **JAVA_HOME** will have to be adjusted for the version that was installed. For instance, the example below uses version **1.8.222**.b10-1.el7_7.x86_64.

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.222.b10-1.el7_7.x86_64
export PODIUM_EXTERNAL_CLASSPATH=`hadoop classpath | sed -e 's/:/,/g' -e 's/*/*.jar/g' -
e 's/*.jar.jar/*.jar/g'`

# YARN libraries are not part of hadoop classpath, unusual
export PODIUM_EXTERNAL_CLASSPATH="$PODIUM_EXTERNAL_CLASSPATH,/usr/lib/hadoop/client/*.jar"

# only use subset of Hive jars for JDBC access, else get classpath conflicts

export PODIUM_EXTERNAL_CLASSPATH="$PODIUM_EXTERNAL_CLASSPATH,/usr/lib/hive/lib/hive-
exec.jar,/usr/lib/hive/lib/hive-jdbc.jar,/usr/lib/hive/lib/hive-metastore.jar,/usr/lib/hive/lib/hive-
service.jar"

# pig
export
PODIUM_EXTERNAL_CLASSPATH="$PODIUM_EXTERNAL_CLASSPATH,/usr/lib/pig/*.jar,/usr/lib/pig/lib/*.jar"

# place EMRFS libraries in front of all other hadoop libraries
export PODIUM_EXTERNAL_CLASSPATH="/usr/share/aws/emr/emrfs/lib/*.jar,$PODIUM_EXTERNAL_CLASSPATH"

export
PODIUM_EXTERNAL_CLASSPATH="$CATALINA_BASE/podiumLibPreHadoop/*.jar,$PODIUM_EXTERNAL_CLASSPATH,$CATALI
NA_BASE/podiumLibPostHadoop/*.jar"

# add hadoop and lzo native libraries
export JAVA_OPTS="$JAVA_OPTS -Xms1g -Xmx8g -
Djava.library.path=/lib64/:/usr/lib/hadoop/lib/native/:/usr/lib/hadoop-lzo/lib/native/"

export JAVA_OPTS="$JAVA_OPTS -DPODIUM_EXTERNAL_CLASSPATH=$PODIUM_EXTERNAL_CLASSPATH -
Dpodium.conf.dir=$CATALINA_BASE/conf"
```

```
export JAVA_OPTS="$JAVA_OPTS -Djavax.security.auth.useSubjectCredsOnly=false -
Djava.security.egd=file:/dev/./urandom"
```

5) S3 Buffer

Located within core-site.xml is the fs.s3.buffer.dir property. The default value is /mnt/s3. **Qlik Catalog needs to ability to create & write to /mnt/s3** as part of the buffer process, so you will need to modify permissions accordingly on that directory.

6) TEZ Configuration (optional)

The following is required to get prepare-on-tez working:

1. On the edge node, build tez.tar.gz:

```
$ cd /usr/lib/tez/
$ tar -czvf /tmp/tez.tar.gz *.jar lib/*.jar
```

2. Use scp to copy tez.tar.gz to the master node (where support for scheme hdfs exists), and from there copy it to hdfs (the second /tmp argument is in hdfs):

```
$ hadoop fs -copyFromLocal /tmp/tez.tar.gz /tmp/
$ hadoop fs -ls /tmp/
```

3. On the edge node, set core_env property

```
tez.lib.uris=/tmp/tez.tar.gz
```

3. Prepare-on-tez requires the ***EXPECT*** package:

```
# sudo yum install expect -y
```

7) Define core_env.properties values

Set the following properties within the core_env.properties file :

```
shippingdock.use.fulluri=true

hadoop.conf.files=core-site.xml,hdfs-site.xml,mapred-site.xml,yarn-site.xml

sample.data.method=HDFS

profile.data.method=HDFS

podium.hive.udf.jar.location=s3://<bucket-name>/podiumhudf-4.6.jar
```

Example: podium.hive.udf.jar.location=s3://qdc-data/podiumhudf-4.6.jar

Note: Make sure the podiumhudf-4.6.jar file exists in the bucket location. If it does not you can copy it from $QDC_HOME/webapps/qdc/WEB-INF/lib/

5.7 Ranger

Qlik Catalog only uses Hadoop standard API's for accessing Hadoop. As a result, Qlik Catalog will naturally take advantage of in-place ranger policies by reporting a "permission denied" result to the end use if HDFS or Hive access is restricted. However, security configuration scenarios vary from company to company.

There is one particular scenario where it's mandatory that a special admin user perform all work in Hive while impersonation is enabled in Qlik Catalog. If so, Qlik Catalog can directly interface with Ranger and dynamically create Hive/HDFS policies that apply to the USER who executed the work in Qlik Catalog and not just the Hive admin user. This allows the user to access this data outside of Qlik Catalog.

Ranger centralized security policy management has been tested in configuration with Hortonworks distribution on Qlik Catalog.

Ranger/Qlik Catalog Integration: Prerequisites and core_env.properties

- Impersonation must be enabled. See here for details on enabling impersonation in Ranger.
- Ranger and Qlik Catalog have the same Users and Groups and associations through integration with Active Directory (Refer to Apache documentation for instructions on pulling in users from UNIX, LDAP, or AD to populate Ranger's local user tables via UserSync.)
- User specified in this core_env property must have permission to make updates in Ranger (example: hive.admin.user=newuser)
- A SuperUser Group (of Qlik Catalog SuperUsers) with ALL permissions must be created in Ranger; refer to instructions below to manually create this group
- Ensure that Group names in Ranger are in lowercase.
- The following **core_env properties** must be configured:

Qlik Catalog (core_env) Properties for Apache Ranger
Authorization Properties
This property specifies the policy management tool for the cluster. # Possible values (case-insensitive): RANGER, SENTRY, NONE **example:** authorization.mode = RANGER
#This property specifies what components on the cluster are authorized with the policy management tool enabled with authorization.mode property. #Possible values (case-insensitive): HIVE, HDFS, ALL #Default value: ALL **example:** authorization.component = ALL #This property decides whether distribution (tables) are to be created or not while syncing policies (before data is loaded). If value is true (default), distribution tables will be created when the policy is created/synced. If the value is false, distribution tables are created when data is loaded. create.distribution.on.sync = true
Apache Ranger Properties (core_env)
#Specifies Ranger Admin portal URL **example:**

ranger.admin.portal.url = http:// <rangerhostportal.url.com>:6080
#Specifies Ranger Admin portal username **example:** ranger.admin.portal.username = admin
#Specifies Admin portal password **example:** ranger.admin.portal.password = #0{5QaBqDr7Ks+16WCqam5Z6Q==}
#Specifies the Ranger service name for hive **example:** ranger.service.hive = podiumHDPducks_hive
#Specifies the Ranger service name for hdfs **example:** ranger.service.hdfs = podiumHDPducks_hadoop

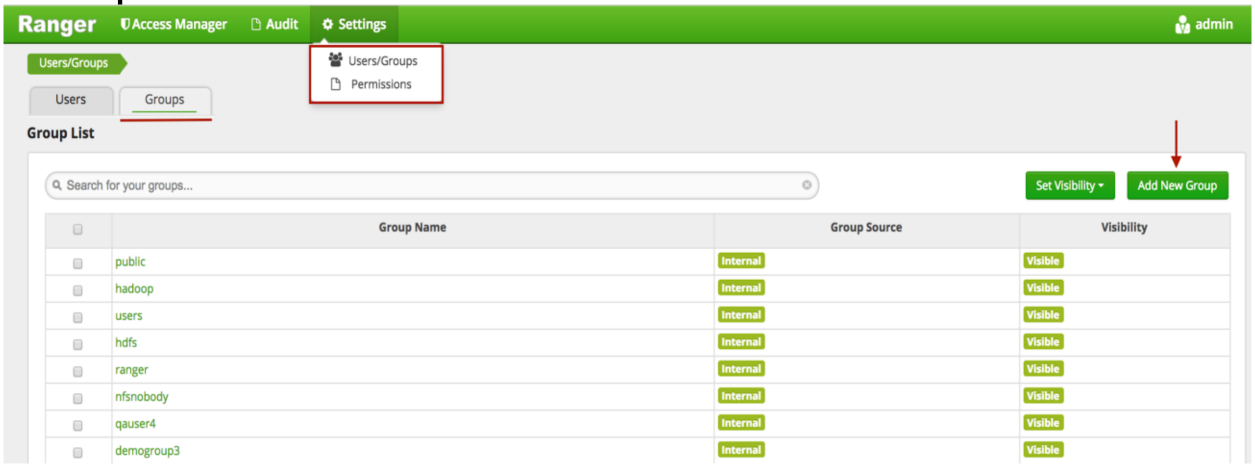
#Use full URI for shipping paths
shippingdock.use.fulluri=true

Apache Ranger Setting in Ambari to address case sensitivity
Ranger authorization is case sensitive therefore if the username/group id doesn't match the id returned from the Directory (AD/LDAP) authorization will be denied. Particularly where AD/LDAP environments are using HDP with Ranger, case must match between AD/LDAP, Qlik Catalog, and Ranger. Set the following properties in Ranger with Ambari to ensure matching case across the applications to “upper” or “lower” [default is “none”]. Note that Ranger service on Ambari must be restarted for the changes to take effect: ranger.usersync.ldap.username.caseconversion ranger.usersync.ldap.groupname.caseconversion

Setting Up SuperUser Group in Ranger for Qlik Catalog SuperUsers

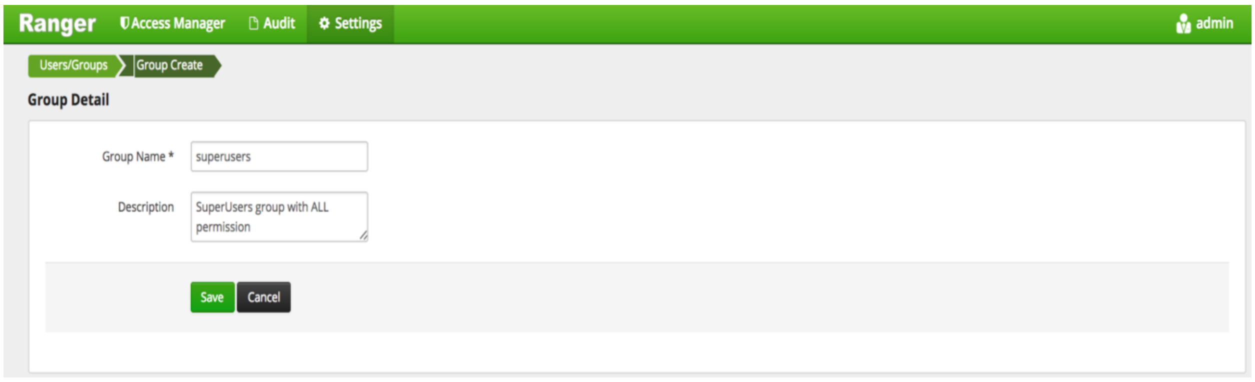
The following instructions set up a Ranger group of Qlik Catalog SuperUsers into a SuperUser group with ALL Privileges.

1. From **Settings**, select **Users/Groups**, then select **Groups** tab and **Add New Group**



	Group Name	Group Source	Visibility
☐	public	Internal	Visible
☐	hadoop	Internal	Visible
☐	users	Internal	Visible
☐	hdfs	Internal	Visible
☐	ranger	Internal	Visible
☐	nfsnobody	Internal	Visible
☐	qauser4	Internal	Visible
☐	demogroup3	Internal	Visible

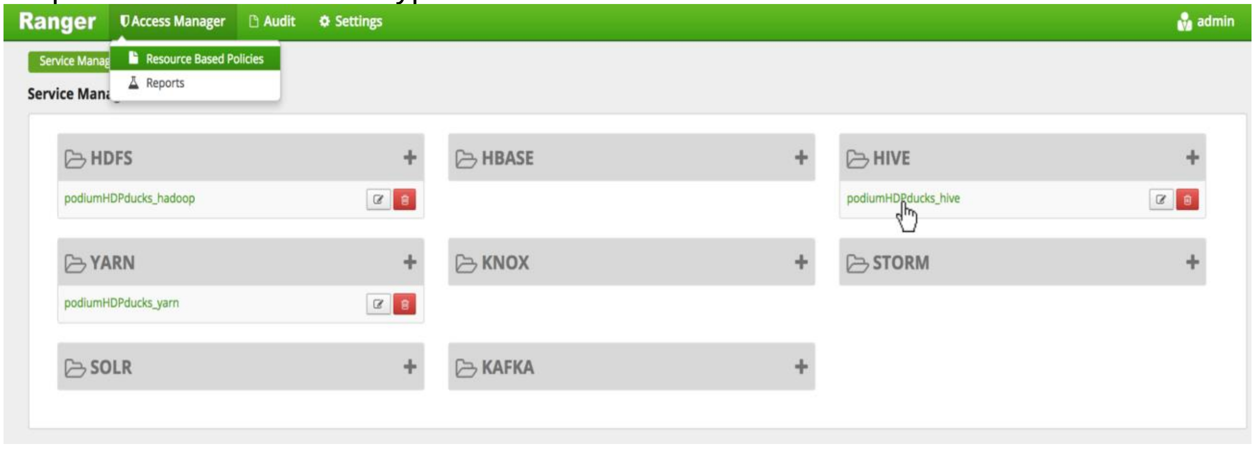
2. Name the SuperUser group and **Save** it.



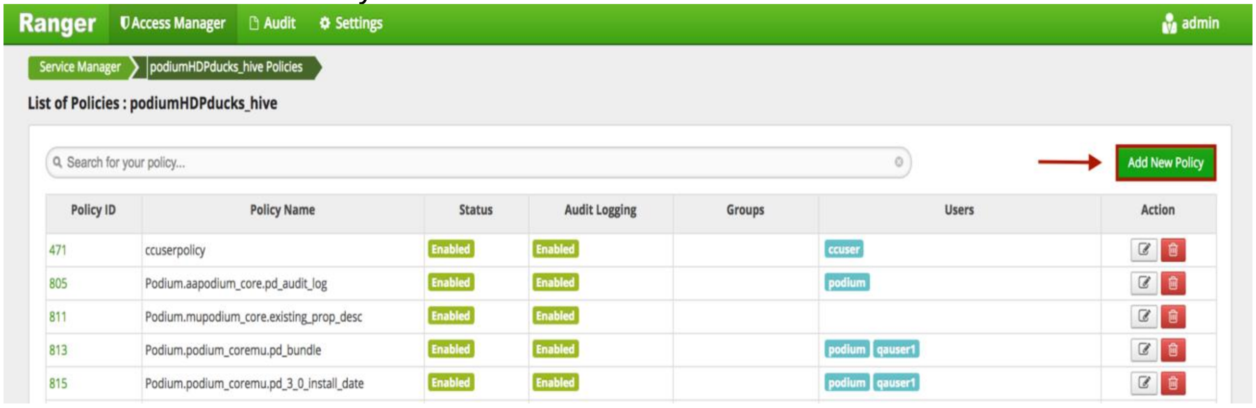
Group Detail

Group Name *	superusers
Description	SuperUsers group with ALL permission

3. From **Access Manager**, select **Resource Based Policies** and open HIVE policies via the named hyperlink



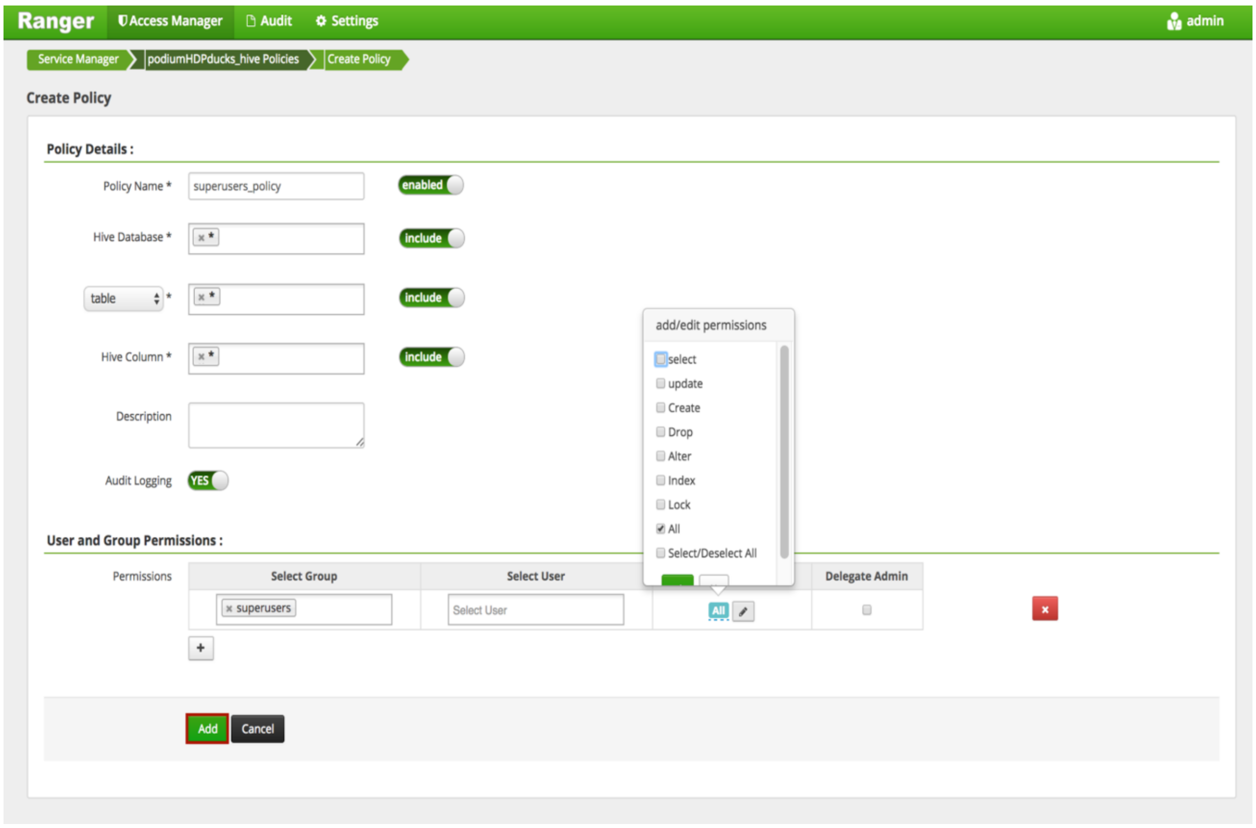
4. Select Add New Policy



List of Policies : podiumHDPducks_hive

Policy ID	Policy Name	Status	Audit Logging	Groups	Users	Action
471	ccuserpolicy	Enabled	Enabled		ccuser	
805	Podium.aapodium_core.pd_audit_log	Enabled	Enabled		podium	
811	Podium.mupodium_core.existing_prop_desc	Enabled	Enabled			
813	Podium.podium_coremu.pd_bundle	Enabled	Enabled		podium qauser1	
815	Podium.podium_coremu.pd_3_0_install_date	Enabled	Enabled		podium qauser1	

5. In **Create Policy** screen, name the Policy name with a name that will identify the SuperUser group.
Specify asterisk (*) for **Hive Database**, **table**, **Hive Column**.
Select the newly created SuperUser group from the **Select Group** dropdown in User and Group Permissions. Skip over **User Select.**
[Note that Qlik Catalog only applies permission at group level and does not employ column-level permissions at this time.]
Select the plus '**+**' to the right of **Add Permissions** in the Permissions column, specifying **All**, save the selected permissions, select **Add** in the lower left of the screen.
The new group (superusers) has an associated policy with **All** permissions for all Hive Tables.



Create Policy

Policy Details :

Policy Name *	superusers_policy	enabled
Hive Database *	*	include
table *	*	include
Hive Column *	*	include
Description		
Audit Logging	YES	

add/edit permissions

- [ ] select
- [ ] update
- [ ] Create
- [ ] Drop
- [ ] Alter
- [ ] Index
- [ ] Lock
- [x] All
- [ ] Select/Deselect All

User and Group Permissions :

Permissions

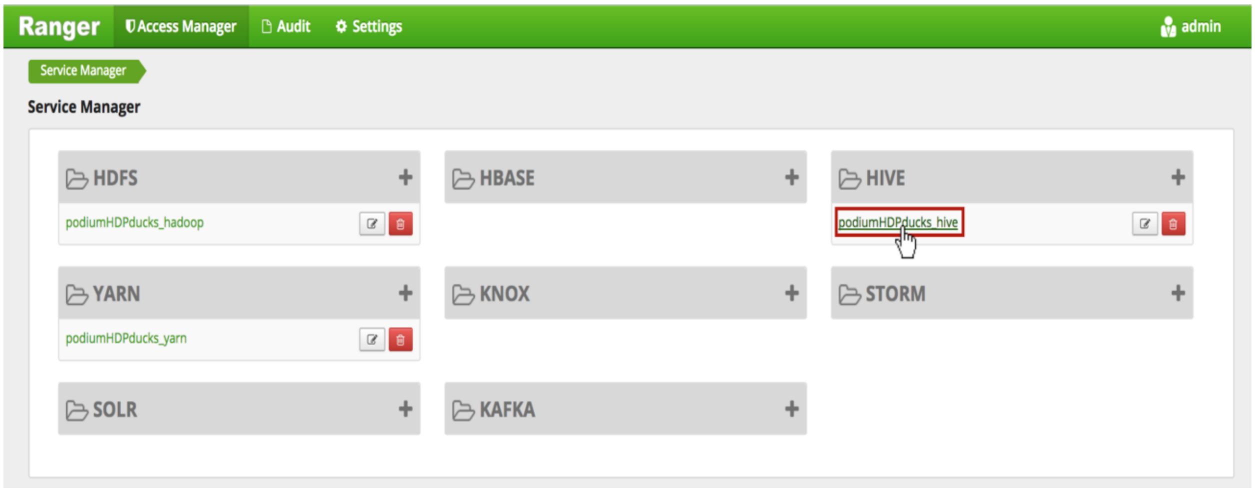
Select Group	Select User	Permissions	Delegate Admin
superusers	Select User	All	☐

Setting Up SuperUser Group in Ranger for Qlik Catalog SuperUsers
The following instructions set up a Ranger group of Qlik Catalog SuperUsers into a SuperUser group with ALL Privileges.

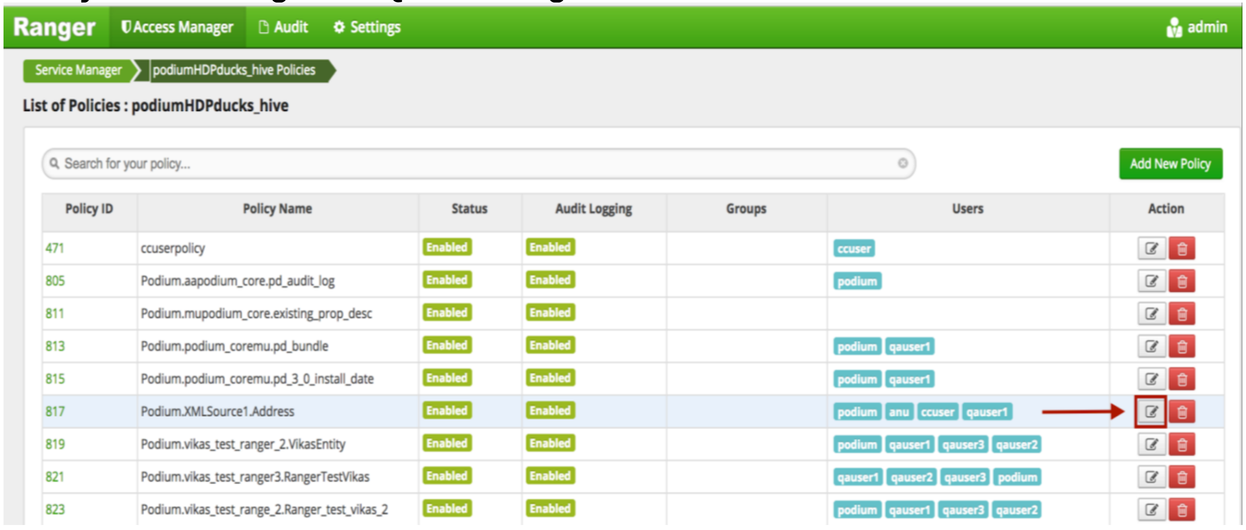
From **Settings**, select **Users/Groups**, then select **Groups** tab and **Add New Group**

Ranger Admin Console
The **Service Manager** screen in Ranger is where users view the services they have enabled to sync with security policies on Qlik Catalog objects.
Hadoop Administrators can add services by selecting the plus icon next to the Hadoop component, enter Service Name/Description, enable/disable policies, and configure service properties and authentication specifications particular to those services.
For Qlik Catalog, Hadoop administrators should enable Hive and HDFS services for their cluster. For every entity, Qlik Catalog creates two policies, one for HDFS and one for Hive.

Ranger Admin Console: Service Manager
Select either HDFS or HIVE Service Name quicklink to access individual policies.



Ranger Hive Policies
Select the edit icon to the right of individual entities to access and update individual Hive entity policies.
Policy List in Ranger for Qlik Catalog Entities



List of Policies : podiumHDPducks_hive

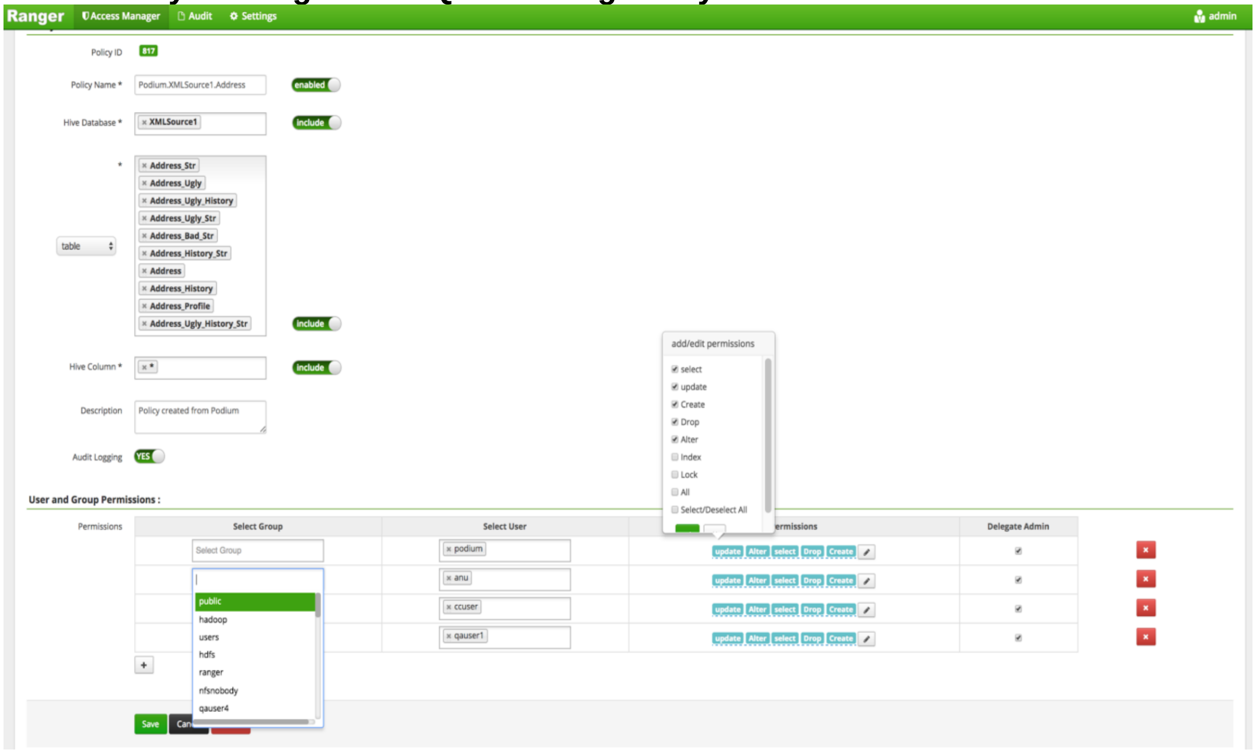
Policy ID	Policy Name	Status	Audit Logging	Groups	Users	Action
471	ccuserpolicy	Enabled	Enabled		ccuser	
805	Podium.aapodium_core.pd_audit_log	Enabled	Enabled		podium	
811	Podium.mupodium_core.existing_prop_desc	Enabled	Enabled			
813	Podium.podium_coremu.pd_bundle	Enabled	Enabled		podium qauser1	
815	Podium.podium_coremu.pd_3_0_install_date	Enabled	Enabled		podium qauser1	
817	Podium.XMLSource1.Address	Enabled	Enabled		podium anu ccuser qauser1	
819	Podium.vikas_test_ranger_2.VikasEntity	Enabled	Enabled		podium qauser1 qauser3 qauser2	
821	Podium.vikas_test_ranger3.RangerTestVikas	Enabled	Enabled		qauser1 qauser2 qauser3 podium	
823	Podium.vikas_test_range_2.Ranger_test_vikas_2	Enabled	Enabled		podium qauser1 qauser3 qauser2	

Editing policies: Synchronization between Qlik Catalog Group level access and Ranger policies will be automatic. Note that while all users in Qlik Catalog have different roles and varying permission levels in the Qlik Catalog application, permissions ('ALL') are granted at the Group level in Ranger. Hive policies in Ranger for Qlik Catalog entities remain in sync with Qlik Catalog Hive tables via Qlik Catalog Sentinel Sync Admin Wizard.
****Add hdfs and hive users to all wildcard policies (ex. “/home*”)**

Hive Policies in Ranger: Settings	
Policy ID: Each policy receives an ID (HDFS policy ID is the Hive policy number + 1)	
Policy Name: Policy naming convention is [Podium.<sourcename>.<entityname>]	
Hive Database: Source Name	
Hive Tables:	entityname entityname_str entityname_history entityname_history_str entityname_history_bad_str entityname_bad_str

	entityname_ugly entityname_ugly_str entityname_history_ugly entityname_history_ugly_str entityname_profile
Hive Column: While Ranger supports column level access, Qlik Catalog does not currently support column level security integration.	
Description: User-defined information about the policy.	
Audit Logging: Enables Audit logging in Ranger.	
User and Group Permissions: All users will have 'ALL' permissions (update, Alter, select, Drop, Create)	

Hive Policy in Ranger for a Qlik Catalog Entity



Policy ID: 817

Policy Name *: Podium.XMLSource1.Address — enabled

Hive Database *: XMLSource1 — include

table *: Address_Str, Address_Ugly, Address_Ugly_History, Address_Ugly_Str, Address_Bad_Str, Address_History_Str, Address, Address_History, Address_Profile, Address_Ugly_History_Str — include

Hive Column *: * — include

Description: Policy created from Podium

Audit Logging: YES

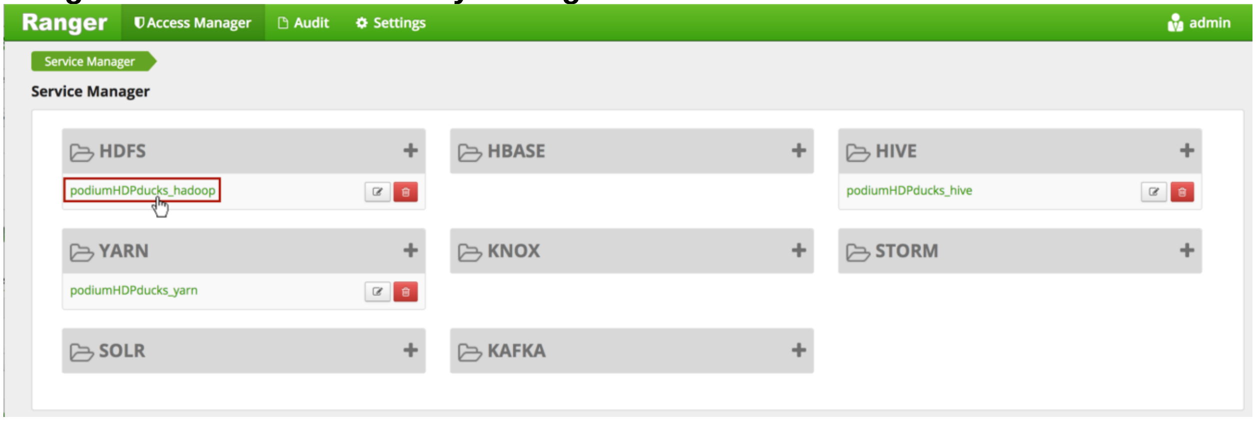
User and Group Permissions :

add/edit permissions: [✓ select] [✓ update] [✓ Create] [✓ Drop] [✓ Alter] Index Lock All Select/Deselect All

Select Group	Select User	Permissions	Delegate Admin
Select Group	podium	update Alter select Drop Create	✓
	anu	update Alter select Drop Create	✓
	ccuser	update Alter select Drop Create	✓
	qauser1	update Alter select Drop Create	✓

Ranger HDFS Policies
Select the edit icon to the right of the Service Name to access individual HDFS entity policies in Ranger.

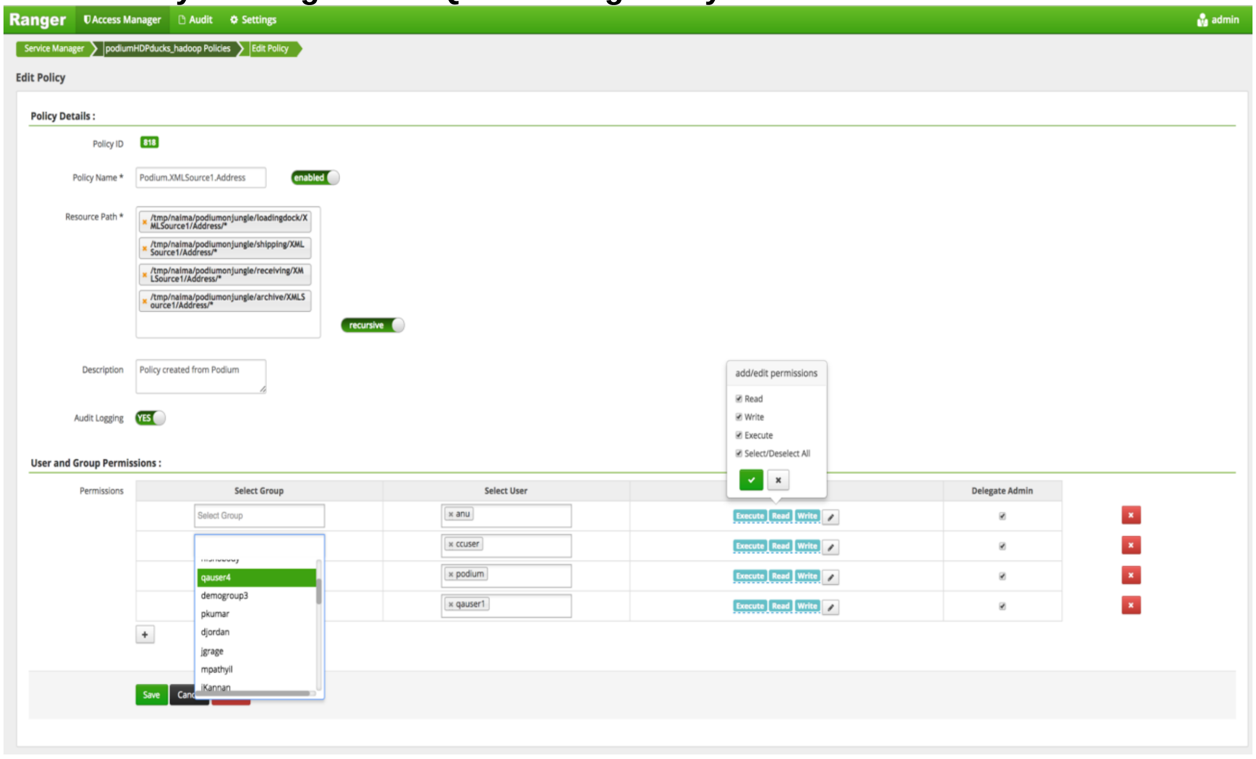
Ranger Admin Console: Policy Manager



Editing policies: Synchronization between Qlik Catalog Group level access and Ranger policies will be automatic though Hadoop administrators can go into individual policies to review and disable/enable the policies. Note that Ranger serves as the backend metastore to manage and sync policies at group level.

HDFS Policies in Ranger: Settings	
Policy ID: Each policy receives an ID (HDFS policy ID is the Hive policy number + 1)	
Policy Name: Policy naming convention is [Catalog.<sourcename>.<entityname>] Note that when a user makes a change to the source/entity this change is reflected/persisted in the corresponding Ranger policy.	
Resource Path: URI for HDFS directories: loading dock|shipping|receiving|archive.<sourcename>.<entityname>/*	
Resource Path	URIs for Qlik Catalog HDFS Directories (loading dock, receiving, archive, shipping) **examples:** /tmp/user/<servicehost>/loadingdock/<sourcename>/<entityname>/* /tmp/user/<servicehost>/shipping/<sourcename>/<entityname>/* /tmp/user/<servicehost>/receiving/<sourcename>/<entityname>/* /tmp/user/<servicehost>/archive/<sourcename>/<entityname>/* Note: Select 'recursive' to apply policies across all directories and its contents including the contents of any subdirectories.
Description: User-defined information about the policy.	
Audit Logging: Enable Audit logging in Ranger	
User and Group Permissions: Synchronization of Directory permissions and ACLS (These will always be ALL: Read, Write, Execute)	

HDFS Policy in Ranger for a Qlik Catalog Entity



Service Manager > podiumHDPducks_hadoop Policies > Edit Policy

Edit Policy

Policy Details :

Policy ID: 818

Policy Name *: Podium.XMLSource1.Address (enabled)

Resource Path *:
- /tmp/naima/podiumonjungle/loadingdock/XMLSource1/Address/*
- /tmp/naima/podiumonjungle/shipping/XMLSource1/Address/*
- /tmp/naima/podiumonjungle/receiving/XMLSource1/Address/*
- /tmp/naima/podiumonjungle/archive/XMLSource1/Address/*

(recursive)

Description: Policy created from Podium

Audit Logging: YES

User and Group Permissions :

add/edit permissions: ✓ Read, ✓ Write, ✓ Execute, ✓ Select/Deselect All

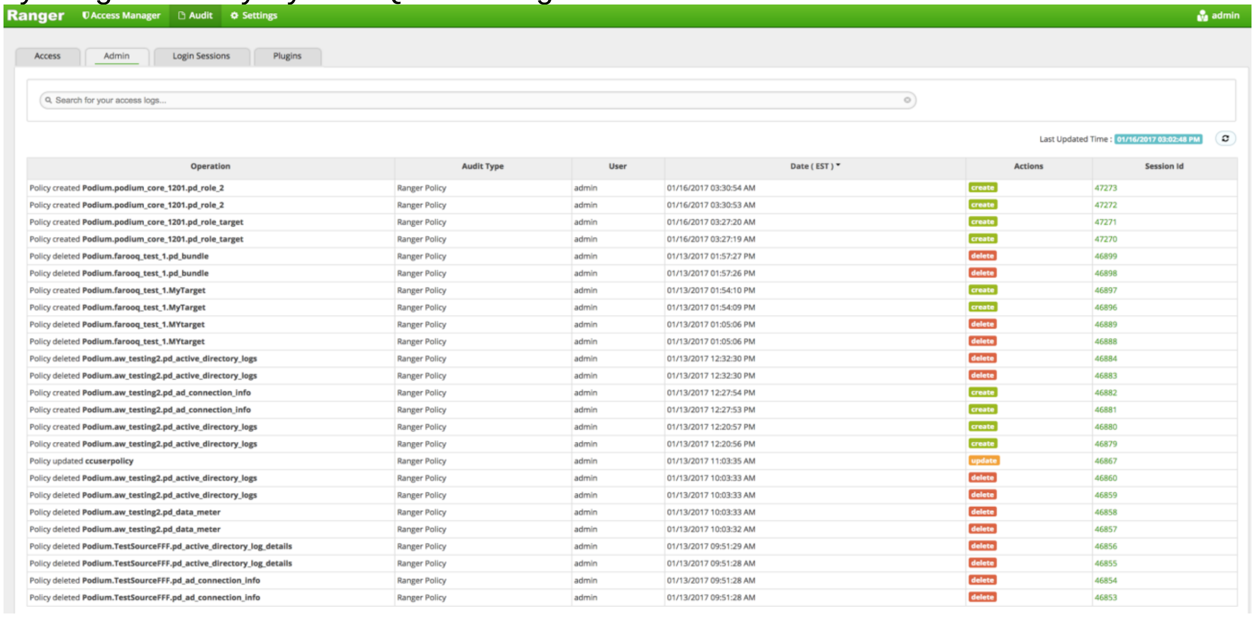
Permissions

Select Group	Select User	Permissions	Delegate Admin
Select Group	anu	Execute Read Write	✓
	ccuser	Execute Read Write	✓
	podium	Execute Read Write	✓
	qauser1	Execute Read Write	✓

Select Group list: qauser4, demogroup3, pkumar, djordan, jgrage, mpathyil, iKannan

Audit Screen

Changes to entities and corresponding policies can tracked through an Audit screen in the Ranger Admin console. It is recommended to manage and monitor Qlik Catalog object syncing via Policy Sync in Qlik Catalog UI.



Last Updated Time : 01/16/2017 03:02:48 PM

Operation	Audit Type	User	Date (EST)	Actions	Session Id
Policy created **Podium.podium_core_1201.pd_role_2**	Ranger Policy	admin	01/16/2017 03:30:54 AM	create	47273
Policy created **Podium.podium_core_1201.pd_role_2**	Ranger Policy	admin	01/16/2017 03:30:53 AM	create	47272
Policy created **Podium.podium_core_1201.pd_role_target**	Ranger Policy	admin	01/16/2017 03:27:20 AM	create	47271
Policy created **Podium.podium_core_1201.pd_role_target**	Ranger Policy	admin	01/16/2017 03:27:19 AM	create	47270
Policy deleted **Podium.farooq_test_1.pd_bundle**	Ranger Policy	admin	01/13/2017 01:57:27 PM	delete	46899
Policy deleted **Podium.farooq_test_1.pd_bundle**	Ranger Policy	admin	01/13/2017 01:57:26 PM	delete	46898
Policy created **Podium.farooq_test_1.MyTarget**	Ranger Policy	admin	01/13/2017 01:54:10 PM	create	46897
Policy created **Podium.farooq_test_1.MyTarget**	Ranger Policy	admin	01/13/2017 01:54:09 PM	create	46896
Policy deleted **Podium.farooq_test_1.MYtarget**	Ranger Policy	admin	01/13/2017 01:05:06 PM	delete	46889
Policy deleted **Podium.farooq_test_1.MYtarget**	Ranger Policy	admin	01/13/2017 01:05:06 PM	delete	46888
Policy deleted **Podium.aw_testing2.pd_active_directory_logs**	Ranger Policy	admin	01/13/2017 12:32:30 PM	delete	46884
Policy deleted **Podium.aw_testing2.pd_active_directory_logs**	Ranger Policy	admin	01/13/2017 12:32:30 PM	delete	46883
Policy created **Podium.aw_testing2.pd_ad_connection_info**	Ranger Policy	admin	01/13/2017 12:27:54 PM	create	46882
Policy created **Podium.aw_testing2.pd_ad_connection_info**	Ranger Policy	admin	01/13/2017 12:27:53 PM	create	46881
Policy created **Podium.aw_testing2.pd_active_directory_logs**	Ranger Policy	admin	01/13/2017 12:20:57 PM	create	46880
Policy created **Podium.aw_testing2.pd_active_directory_logs**	Ranger Policy	admin	01/13/2017 12:20:56 PM	create	46879
Policy updated **ccuserpolicy**	Ranger Policy	admin	01/13/2017 11:03:35 AM	update	46867
Policy deleted **Podium.aw_testing2.pd_active_directory_logs**	Ranger Policy	admin	01/13/2017 10:03:33 AM	delete	46860
Policy deleted **Podium.aw_testing2.pd_active_directory_logs**	Ranger Policy	admin	01/13/2017 10:03:33 AM	delete	46859
Policy deleted **Podium.aw_testing2.pd_data_meter**	Ranger Policy	admin	01/13/2017 10:03:33 AM	delete	46858
Policy deleted **Podium.aw_testing2.pd_data_meter**	Ranger Policy	admin	01/13/2017 10:03:32 AM	delete	46857
Policy deleted **Podium.TestSourceFFF.pd_active_directory_log_details**	Ranger Policy	admin	01/13/2017 09:51:29 AM	delete	46856
Policy deleted **Podium.TestSourceFFF.pd_active_directory_log_details**	Ranger Policy	admin	01/13/2017 09:51:28 AM	delete	46855
Policy deleted **Podium.TestSourceFFF.pd_ad_connection_info**	Ranger Policy	admin	01/13/2017 09:51:28 AM	delete	46854
Policy deleted **Podium.TestSourceFFF.pd_ad_connection_info**	Ranger Policy	admin	01/13/2017 09:51:28 AM	delete	46853

5.8 Sentry

Qlik Catalog integrates with Sentry policy administration tool for centralized Hadoop security management. Note that Sentry centralized security policy management has been tested in configuration with Cloudera distribution on Qlik Catalog.

Sentry/Qlik Catalog Integration: Prerequisites and core_env.properties

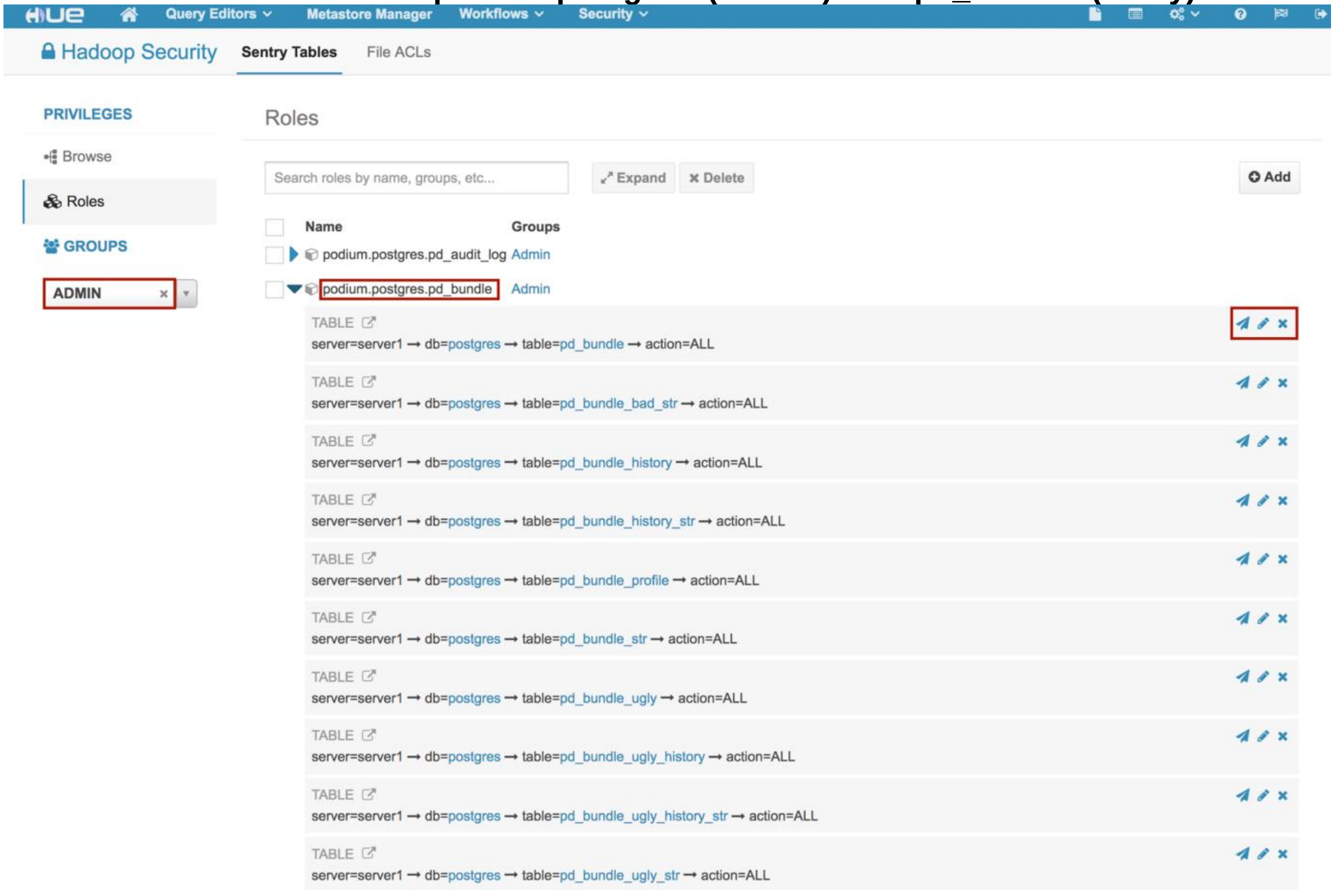
- Impersonation must be enabled in Sentry. See here for details on enabling impersonation in Sentry.
- Sentry and Qlik Catalog have the same Users and Groups and associations through integration with Kerberos/Active Directory
- Qlik Catalog recommends downloading Hue open-source security app for administering and integrating Sentry permissions
- Qlik Catalog uses core_env values for Qlik Catalog (Hive) Distribution URI (Found in About screen, DISTRIBUTION: **distribution.uri** and **distribution.username**) to integrate with and manage Hive policies
- User specified in this core_env property must have permission to make updates in Sentry: hive.admin.user=ccuser
- When Sentry is enabled, this setting allows Hive to access the rules table in HDFS required for publish/obfuscation to work:
- hive.service.user=hive
- A SuperUser Group (of Qlik Catalog SuperUsers) with ALL permissions must be created in Sentry; refer to instructions below to manually create this group
- Ensure that Group names in Sentry must be in lowercase, corresponding groups in Qlik Catalog can be any case (lowercase, uppercase, camelcase)
- The following core_env properties must be configured:

Qlik Catalog (core_env) Properties for Sentry
Authorization Properties
This property specifies the policy management tool for the cluster. # Possible values (case-insensitive): RANGER, SENTRY, NONE **example:** authorization.mode = SENTRY
#This property specifies what components on the cluster are authorized with policy management tool. #Possible values (case-insensitive): HIVE, HDFS, ALL (HDFS is not an option for Sentry, therefore HIVE and ALL are effectively the same setting for Sentry) #Default value: ALL **example:** authorization.component = ALL

#This property decides whether distribution (tables) are to be created or not while syncing policies (before data is loaded). If value is true (default), distribution tables will be created when the policy is created/synced. If the value is false, distribution tables are created when data is loaded.
create.distribution.on.sync = true

Hue-Sentry Interface
The user who logs into Hue must have access to the Security Tab/Sentry Tables.
ROLES: Policies and associated Groups.
Select GROUPS to view associated policies.
Name: Naming convention for roles (policies) is 'source.entity'. Policy names must be in lower-case.

HUE-SENTRY ROLE/POLICY EXAMPLE: ADMIN GROUP has ALL permissions for the Hive tables associated with 'podium.postgres' (source) and 'pd_bundle' (entity)



Name	Groups
podium.postgres.pd_audit_log	Admin
podium.postgres.pd_bundle	Admin

TABLE
server=server1 → db=postgres → table=pd_bundle → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_bad_str → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_history → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_history_str → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_profile → action=ALL

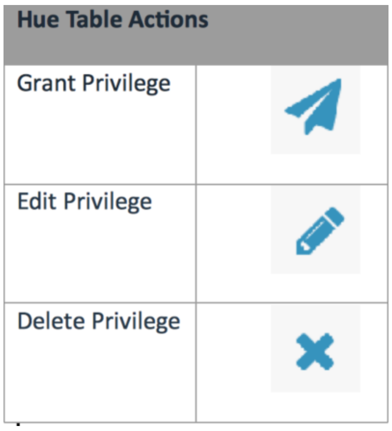
TABLE
server=server1 → db=postgres → table=pd_bundle_str → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_ugly → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_ugly_history → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_ugly_history_str → action=ALL

TABLE
server=server1 → db=postgres → table=pd_bundle_ugly_str → action=ALL

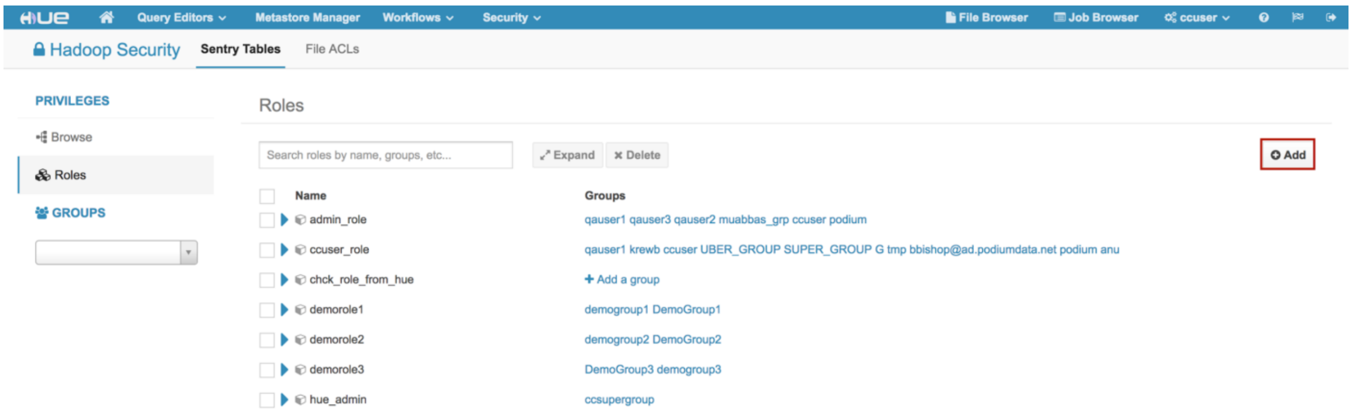


Hue Table Actions	
Grant Privilege	➤
Edit Privilege	✎
Delete Privilege	✖

Editing policies: Synchronization between Qlik Catalog Group level access and Sentry policies will be automatic. Note that while all users in Qlik Catalog have different roles and varying permission levels in the Qlik Catalog application, permissions ('ALL') are granted at the Group level in Ranger. Hive policies in Ranger for Qlik Catalog entities remain in sync with Qlik Catalog Hive tables via Qlik Catalog Sentinel Sync Admin Wizard.

Setting Up SuperUser Group in Sentry for Qlik Catalog SuperUsers
The following instructions set up a Sentry group of Qlik Catalog SuperUsers into a SuperUser group with ALL Privileges.

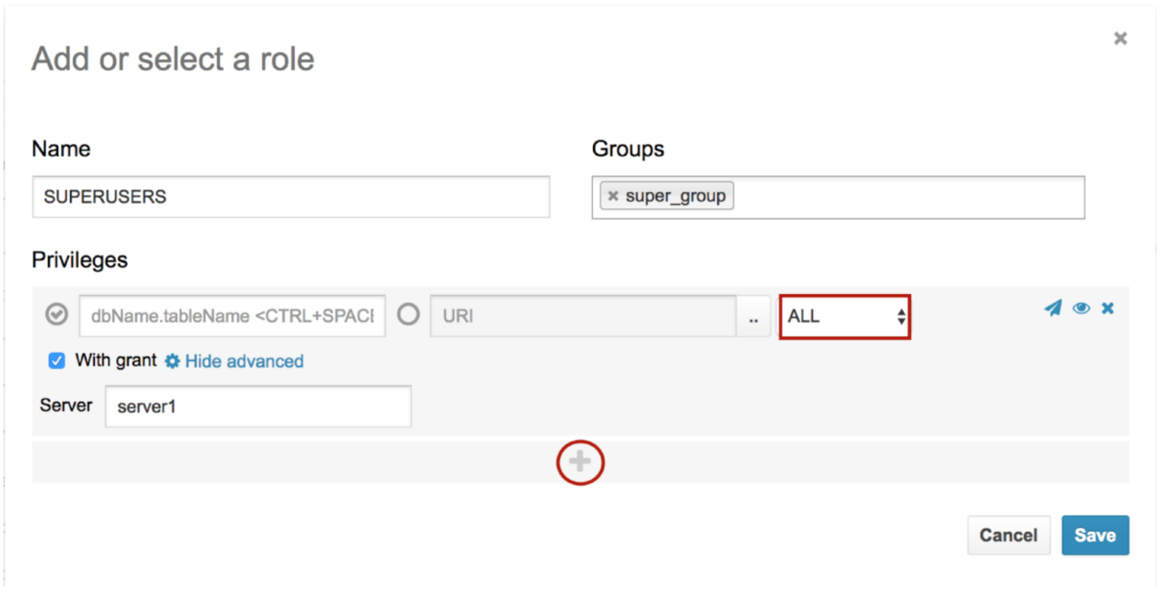
1. From the Security dropdown on the Navigation bar tab in Hue, select Sentry Tables tab. Select Add+ from the top-right screen.



Roles

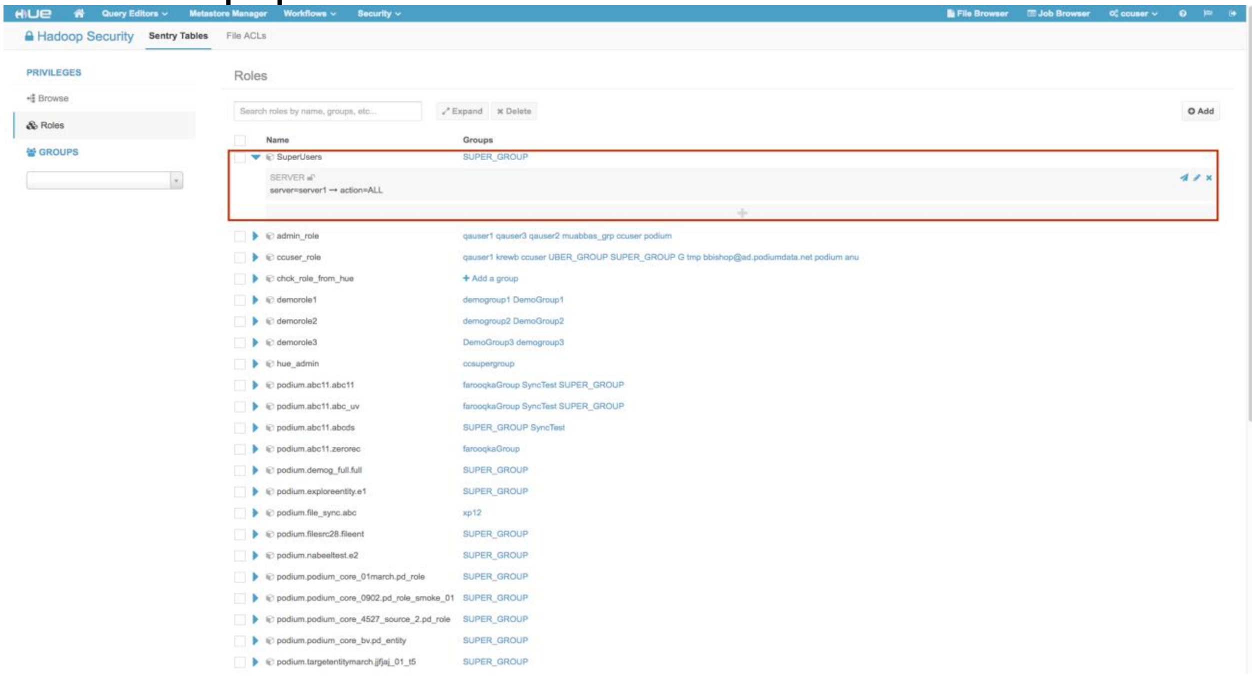
Name	Groups
admin_role	qauser1 qauser3 qauser2 muabbas_grp ccuser podium
ccuser_role	qauser1 krewb ccuser UBER_GROUP SUPER_GROUP G tmp bbishop@ad.podiumdata.net podium anu
chck_role_from_hue	+ Add a group
demorole1	demogroup1 DemoGroup1
demorole2	demogroup2 DemoGroup2
demorole3	DemoGroup3 demogroup3
hue_admin	ccsupergroup

2. Name the SuperUser Group and add the Qlik Catalog Group of SuperUsers (pre-created in Qlik Catalog), select the '+' to add a privilege, Select ALL and select the checkbox for With grant, specifying the server name as specified in core_env property hive.server.name. Save.



Name: SUPERUSERS
Groups: super_group
ALL
With grant ✓
Server: server1

User Group has been created with ALL permissions on the Hive server as specified in Authorization properties.



Roles

Name	Groups
SuperUsers	SUPER_GROUP
SERVER server=server1 → action=ALL	
admin_role	qauser1 qauser3 qauser2 muabbas_grp ccuser podium
ccuser_role	qauser1 krewb ccuser UBER_GROUP SUPER_GROUP G tmp bbishop@ad.podiumdata.net podium anu
chck_role_from_hue	+ Add a group
demorole1	demogroup1 DemoGroup1
demorole2	demogroup2 DemoGroup2
demorole3	DemoGroup3 demogroup3
hue_admin	ccsupergroup
podium.abc11.abc11	farooqkaGroup SyncTest SUPER_GROUP
podium.abc11.abc_uv	farooqkaGroup SyncTest SUPER_GROUP
podium.abc11.abcds	SUPER_GROUP SyncTest
podium.abc11.zerorec	farooqkaGroup
podium.demog_full.full	SUPER_GROUP
podium.exploreentity.e1	SUPER_GROUP
podium.file_sync.abc	xp12
podium.filesrc28.fileent	SUPER_GROUP
podium.nabeeltest.e2	SUPER_GROUP
podium.podium_core_01march.pd_role	SUPER_GROUP
podium.podium_core_0902.pd_role_smoke_01	SUPER_GROUP
podium.podium_core_4527_source_2.pd_role	SUPER_GROUP
podium.podium_core_bv.pd_entity	SUPER_GROUP
podium.targetentitymarch.jjfjaj_01_t5	SUPER_GROUP

5.9 RESTful API Python libraries

The following Python libraries must be installed to use Python (Sample Client) Code to talk to Qlik Catalog RESTful APIs.
A Python Package Manager can assist with installs.

Standard Python libraries (included with Python)

- sys
- os
- glob
- datetime
- json

Libraries requiring install:

- psycopg2
- requests

5.10 Ingestion from Kafka

The installer performs the following for a first-time install on an HDP 3.0 edge node. It is documented for reference purposes and may help troubleshoot issues with loading from Kafka on alternative edge nodes.

Qlik Catalog uses kafka-clients-1.1.0.jar -- an older version, kafka-clients-0.8.2.1.jar, is found on an HDP 3.0 edge node. The two jars are so incompatible that Qlik Catalog cannot be made to work with the older library.

The fix for this issue is to use the newer, Qlik Catalog version of the Kafka client jar, ensuring it takes precedence over the one on the Hadoop classpath:

- Extract kafka-clients-1.1.0.jar from the thirdparty directory of the Qlik Catalog zip to <tomcat home>/podiumLibPreHadoop/
- Change the order of classpath in setenv.sh so that the above folder comes before the Hadoop classpath:

  ```
  export
  PODIUM_EXTERNAL_CLASSPATH=$CATALINA_BASE/podiumLibPreHadoop/*.jar,$PODIUM_
  EXTERNAL_CLASSPATH
  ```

- Add a property, mapreduce.job.user.classpath.first, to core_env.properties, which is by default set to false but here is set to true. This is a standard Hadoop configuration property that allows the data load job to use the updated kafka-clients library instead of the one on the cluster. Without this change, the data load job will fail at the receiving step.

5.11 Adding Qlik Core Docker Container to Existing Cluster Installation

The installer performs the following for a first-time install of June 2019 (4.2) or later. This section describes how to manually add Qlik Core support to a Qlik Catalog cluster-capable installation. It is documented for reference purposes.

First, ensure section 3.5, Qlik Sense Integration, has been completed.

The commands below assume the Qlik Catalog home directory is /usr/local/qdc and that the service account user is qdc. Do the following:

1. Make a Qlik Core integration directory (perform as service user: sudo su - qdc):

   ```
   (qdc) # cd /usr/local/qdc
   (qdc) # mkdir qlikcore
   (qdc) # cd qlikcore
   ```

2. Copy the Qlik Sense integration files, included in the Qlik Catalog (podium.zip), to '/usr/local/qdc/qlikIntegration' directory. Replace the path '/tmp/podium/bin/qlikSenseIntegration/' with the path to your unzipped podium.zip.

   ```
   (qdc) # cp /tmp/podium/bin/qlikSenseIntegration/* .
   ```

3. Load the Qlik Core docker image:

   ```
   (qdc) # docker load < docker-qdc
   ```

4. To test, list images:

   ```
   (qdc) # docker images
   ```

5. Register node modules:

   ```
   (qdc) # npm install --only=prod
   ```

6. To test, list modules:

   ```
   (qdc) # npm ls
   ```

7. Start the Qlik Core Docker container

   ```
   (qdc) # ./launch_qlikcore.sh
   ```

8. Set the following property in Qlik Catalog’s <tomcat home>/conf/core_env.properties

   ```
   qvd.openconnector.script.path=/usr/local/qdc/qlikcore/qdc_qvd_2_csv.sh
   %prop.qvd.file.linux.folder %prop.qvd.file.entity.original.name
   %loadingDockLocation %loadingDockUri
   ```

Please refer to other documentation for guidance on configuring Qlik Catalog and Qlik Sense integration, including using the Qlik Core Docker container to load a QVD.

5.12 Integration Setup Between Qlik Sense and Qlik Catalog

The configuration process for Integrating *Qlik Catalog* with *Qlik Sense* is detailed in a document called QlikDataCatalystQlikSense_<Month.year>_IntegrationGuide.pdf

5.13 Enabling NextGen XML

NextGen XML Docker Containers

- In order to use the 'next-generation' XML support in Qlik Catalog, you will need to install two Qlik Sense Docker containers: Dcaas (a connector lookup service) and a REST Connector (that parses XML files and converts them to flattened data).
- NextGen XML does NOT support HTTPS redirection.
- The QDC Installer will setup the required containers.
- The NextGen XML Docker containers must be setup as a service to automatically start when Linux restarts.

  The copy (cp) command below assumes the Qlik Catalog software (a.k.a., podium zip) has been expanded to /tmp -- the prerequisites section.

  Note: If the Qlik Catalog/Qlik Sense integration directory is not /usr/local/qdc/dcaasIntegration or the service user/group is not qdc, the file /etc/systemd/system/nextgen-xml.service must be edited after the copy (cp) step below

```
(sudo)    # sudo cp /tmp/podium/config/nextgen-xml.service /etc/systemd/system/
(sudo)    # sudo systemctl daemon-reload
(sudo)    # sudo systemctl enable nextgen-xml.service
```

Qlik Catalog Configuration

Qlik Catalog must be configured to support NextGen XML. The Qlik Catalog server **hostname** value must be set manually within two files. (This is the hostname or IP address of the host where the Tomcat server runs):

- **core_env.properties**:

  The core_env.properties file is located here: $TOMCAT_HOME/conf/core_env.properties.

  Locate the following property and replace <QDC_HOST_NAME> with the hostname of the Qlik Catalog server:

  - base.xml.callback.url=http://<**QDC_HOST_NAME**>:8080/qdc-xmlstore

  Note: the following two NextGen XML properties are also present in the core_env.properties file and are set automatically by the Qlik Catalog Installer:

  - enable.new.xml.ingestion=true

- dcaas.connector.staging.dir=<QDC_HOME>/dcaasIntegration/dcaas-connector-staging

- **docker-compose.dcaas-connectors.yml**

  The docker-compose.dcaas-connectors.yml file is located here: $QDC_HOME/dcaasIntegration

  Locate the following property and replace ${DCAAS_HOST_NAME} with the hostname of the Qlik Catalog server:

  - CONNECTOR_SERVICE=${DCAAS_HOST_NAME}:data-connector-rest:50060

Modify Tomcat Web Server Configuration

Manual configuration changes to the Tomcat web server are also required. Follow the instructions below to modify the Tomcat server.xml file to support NextGen XML:

1) Using a text editor, open **$TOMCAT_HOME/conf/server.xml:**

```
<qdc>  $ vi /usr/local/qdc/apache-tomcat-9.0.38/conf/server.xml
```

2) Locate the following text:

```
<!-- UNCOMMENT THIS for the next-gen xml feature. The 'qdc.home' variable
should be defined in setenv.sh -->

<!--

<Context docBase="${qdc.home}/dcaasIntegration/qdc-xmlstore"  path="/qdc-
xmlstore" >

<Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|10\.\d+\.\d+\.\d+|172\.1[6-
9]\.\d+\.\d+|172\.2[0-9]\.\d+\.\d+|172\.3[0-
1]\.\d+\.\d+|192\.168\.\d+\.\d+"/>

</Context>

-->
```

3) “Uncomment” this section of text by removing the first & last lines of the section so that it appears as follows:

```
<Context docBase="${qdc.home}/dcaasIntegration/qdc-xmlstore"  path="/qdc-
xmlstore" >

<Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|10\.\d+\.\d+\.\d+|172\.1[6-
9]\.\d+\.\d+|172\.2[0-9]\.\d+\.\d+|172\.3[0-
1]\.\d+\.\d+|192\.168\.\d+\.\d+"/>

</Context>
```

4) Save the changes to server.xml

5) Restart NextGen XML docker containers:

```
(sudo)  systemctl restart nextgen-xml.service
```

6) **To verify that Docker and its containers are running:**

- Enter this on a command line: docker ps
- Expected result: you should see the dcaas and data-connector-rest containers in the list of active containers.

About Qlik

Qlik is on a mission to create a data-literate world, where everyone can use data to solve their most challenging problems. Only Qlik's end-to-end data management and analytics platform brings together all of an organization's data from any source, enabling people at any skill level to use their curiosity to uncover new insights. Companies use Qlik products to see more deeply into customer behavior, reinvent business processes, discover new revenue streams, and balance risk and reward. Qlik does business in more than 100 countries and serves over 48,000 customers around the world.