

Multi-node Installation and Setup Guide

Qlik Catalog[®]
February 2023

TABLE OF CONTENTS

1.0 Qlik Catalog Overview and System Requirements	3
1.1 Hardware Configuration Requirements	3
1.2 Software Configuration Requirements & Support Matrix	3
2.0 User Setup and Security Prerequisites	5
2.1 Cluster User Setup and Impersonation	5
2.2 Active Directory	6
2.3 HDFS and Hive Permissions	6
2.4 Ranger	7
2.5 Kerberos	7
3.0 New Deployments	8
4.0 Installation Prerequisites	10
4.1 Java JDK Installation	11
4.2 Create Catalog Linux Account and Qlik Catalog Directory	12
4.3 Tomcat Installation	13
4.4 PostgreSQL Installation	15
4.5 Container Platform & Node.js	17
4.6 Create Qlik Catalog Base Storage Directory in HDFS	20
4.7 Create Hive 'user_views' Database	21
4.8 EMR Deployments Only: Setup Edge Node	22
5.0 Qlik Catalog Software Installation	22
5.1 First-time Installation Using the Installer	22
5.2 PostgreSQL Manual Database Creation	27
5.3 Upgrade of Qlik Catalog	29
5.4 Non-Interactive (“Silent”) Installation	30
5.5 Distribution Specific Configuration Parameters	31
6.0 Qlik Catalog Software Installation Reference	32
6.1 Masking and Obfuscation	32

6.2 Kerberos Configuration	32
6.3 Impersonation Configuration	36
6.4 Enabling SAML	39
6.5 Tomcat SSL Configuration	44
6.6 Port Redirection	45
6.7 Testing Licenses Connectivity to QlikCloud	46
6.8 Amazon Web Services EMR (Elastic MapReduce)	47
6.9 Ranger	54
6.10 Adding Qlik Core Docker Container to Existing Cluster Installation	62
6.11 Integration Setup Between Qlik Sense and Qlik Catalog	63
6.12 Enabling NextGen XML	63
6.13 Migrating to or Upgrading Tomcat 9	65
6.14 Disabling Tomcat Redirect / SSL	67
6.15 Upgrading PostgreSQL	68
6.16 Attaching a New Data Volume	70
6.17 SAML and PODIUM Authentication Using Single Tomcat Instance	71
6.18 RHEL 9, SELinux and podman	72

1.0 Qlik Catalog Overview and System Requirements

This document describes how to install the "multi-node" deployment option for Qlik Catalog. Once the installation is complete, upon first login to the Qlik Catalog application, you will be prompted for a license. A license for "**Qlik Catalog**" must be entered.

1.1 Hardware Configuration Requirements

Cluster Edge Node Recommendations

- Recommended Minimum Production Configuration
 - 12 Cores
 - 128GB RAM
 - System Drive 1TB
 - Data Drive 1TB
 - Ubuntu or RHEL8 & 9.1 deployments: 2gb **/home** directory OR set custom container storage directory according to Podman documentation for rootless containers.
 - Ethernet 10GB
 - Virtual Machine or bare metal

- Minimum POC/Dev Configuration
 - 4 Cores
 - 32GB RAM
 - System partition 100GB
 - Data partition 100GB
 - Ubuntu or RHEL 8/9.1 deployments: 2gb **/home** directory
 - Ethernet 10GB
 - Virtual Machine or bare metal

- Minimum Supported Screen Resolution: 1366x768px

1.2 Software Configuration Requirements & Support Matrix

Qlik Catalog and Qlik Enterprise Manager supported versions

- Qlik Enterprise Manager May 2021 and above
- Qlik Catalog November 2020 Service Release 1 and above (4.8.1+)

Qlik Catalog and Qlik Sense supported versions

- QSEoW/QSD May 2021 and above
- Qlik Catalog February 2021 Service Release 2 and above (4.9.2+)
- QSEoW/QSD February 2021 (latest patch) and November 2020 (latest patch)
- Qlik Catalog February 2021 Service Release 1 (4.9.1)
- QSEoW/QSD November 2020 patch 3
- Qlik Catalog February 2021 (4.9)

NOTE: Environment should be configured as a true edge node with all relevant Hadoop client tools.

System Requirements	Version
PostgreSQL Metadata Database	Custom Qlik Catalog PostgreSQL 14. Version 14.6 required for first-time installations.
Oracle Metadata Database	Not supported
Apache Tomcat	Tomcat 9. Version 9.0.70+ encouraged for first-time installations.
Java	OpenJDK 8 or JDK 11, minimum version 1.8.0_222
Podman (RHEL 8.x & 9.1 or Ubuntu 20.04/22.04 deployments)	3.4.2+
Certified Hadoop Distributions	
Cloudera CDP Private Cloud	7.1.6
AWS EMR	5.33 or 6.5.0 – please request script emr-create-edge-node.sh
Browsers	
Google Chrome	80.0 or higher
MS Internet Explorer	Not supported
Other browsers not actively tested	Issues must be reproducible using Chrome to be eligible for a fix.
Operating Systems	
CentOS 7.x	CentOS Linux release 7, certified on en_US locale
RHEL 7.x	All Red Hat Enterprise Linux installations require a valid Red Hat entitlement subscription.
RHEL 8.x	RHEL 7 installations require access to the following repositories:
RHEL 9.1+	<ul style="list-style-type: none"> • rhel-7-server-rpm • rhel-7-server-extras-rpms • rhel-7-server-optional-rpms
Ubuntu 20.04 LTS	
Ubuntu 22.04 LTS	

Additional Requirements

- 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 8443 (http or https) is open from user desktops to the Qlik Catalog node(s)

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 several 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 several 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. A group with the same name should also exist -- all permissions granted to the user should also be granted to the group.
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 (and group) should have ALL permissions (rwx) on the podium base directory in HDFS.
4. The Qlik Catalog service user should have a home directory in HDFS (example: `/user/qdc`) 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. A simple way to do this if Ranger is not being used is to add the user to the service group.
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 the [Ranger](#) section 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. If creating a custom AD group for the POC with associated users, and Ranger is in use, you must create appropriate Ranger policies as well.

Example of information for 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
    Users)(objectClass=group))"
```

2.3 HDFS and Hive 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, but 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/
```

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.a
cme.net
jdbc:hive2://hdmduv0005.test.group:10010/podium_test_01;principal=hive/hdmduv0005.machine.group@APPGL0
BAL.CHIPCORP.COM
jdbc:hive2://hdmduv0005.machine.test.group:10010/podium_test_01;principal=hive/_HOST@APPGLOBAL.CHIPCOR
P.COM
```

2.4 Ranger

Qlik Catalog supports Apache Ranger. First, Qlik Catalog will naturally honor Ranger security policies and report any access rights errors up through the Qlik Catalog UI because of its use of Hadoop standard APIs. Second, 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 [Ranger](#) section for more details.

2.5 Kerberos

For an application to use Kerberos, its source code must be modified to make the appropriate calls into the Kerberos libraries. Applications modified in this way are 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 6.2 Kerberos Configuration appendix section for more details.

3.0 New Deployments

The Qlik Catalog installation package includes a script named ***CatalogFirstTime.sh*** which combines prerequisite, application, and post-installation scripts for a start-to-finish installation. **It is intended only for new deployments of Qlik Catalog.**

Use of *CatalogFirstTime.sh* is recommended for new deployments to simplify the process, but it is not required. You may follow the next two sections of this guide and run the installation scripts individually.

The *CatalogFirstTime.sh* script works in conjunction with the ***CatalogInstaller.properties*** file. Additional documentation is present in that file. Most of the properties have been pre-populated and the defaults should work for most installations; however, they may be changed for customized installations. `PODIUM_RELEASE_FILE` is set to the current podium zip file name. (The properties `SUPERUSER_NAME` and `SUPERUSER_PASSWORD` are for the Qlik Catalog PostgreSQL instance).

There are three **empty** properties which **MUST** be populated prior to running *CatalogFirstTime.sh*:

- `INSTALL_TYPE` (input CDP or EMR)
- `QDC_SERVICE_ACCOUNT_PW` (password for Catalog Linux user that will be created)
- `SUPERUSER_PASSWORD` (password for PostgreSQL superuser that will be created)

In the event any of these properties are left empty, the *CatalogFirstTime.sh* script will prompt for input.

CatalogFirstTime.sh may be run in interactive mode or non-interactive “silent” mode:

Interactive installation allows users to confirm installer actions in a step-by-step manner.

Example: `$ sudo ./CatalogFirstTime.sh`

Non-Interactive or “silent” installation allows users to deploy Catalog in a scripted manner which does not require any user interaction. The following arguments must be added to the *CatalogFirstTime.sh* command to run in non-interactive mode:

- `-s` (silent)
- `-a` (Accept Qlik Customer Agreement)

Example: `$ sudo ./CatalogFirstTime.sh -sa`

NOTE: Invoking the “-a” option indicating acceptance of the Qlik Customer Agreement (QCA) is required to run the *CatalogFirstTime.sh* 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 CUSTOMER AGREEMENT (QCA) 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.

To use *CatalogFirstTime.sh*:

1. Install unzip (if not present)

All Red Hat Enterprise Linux & CentOS 7 Deployments

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

Ubuntu Deployments

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

2. Copy the CatalogInstaller.zip file into an installer “working” directory (example: /tmp)

Example:

```
(sudo) # cp CatalogInstaller.zip /tmp
```

3. Unzip CatalogInstaller.zip

```
(sudo) # cd /tmp
```

```
(sudo) # unzip CatalogInstaller.zip
```

4. Copy the podium zip file into the CatalogInstaller directory created in Step 3

Example:

```
(sudo) # cp podium-4.17-xxxxx.zip /tmp/CatalogInstaller/
```

5. Edit the CatalogInstaller.properties file located within the CatalogInstaller directory. Set the properties described above. (If properties are not set the script will prompt for user input).

Example:

```
(sudo) # vi /tmp/CatalogInstaller/CatalogInstaller.properties
```

6. Run CatalogFirstTime.sh

```
(sudo) # cd /tmp/CatalogInstaller
```

```
(sudo) # ./CatalogFirstTime.sh
```

Once CatalogFirstTime.sh completes successfully you must complete the following steps for multi-node deployments:



Create Qlik Catalog Base Storage Directory in HDFS

- [Instructions in section 4.6](#)



Create Hive 'user_views' Database

- [Instructions in section 4.7](#)

4.0 Installation Prerequisites

The prerequisite software need only be installed before a first-time Catalog installation, not an upgrade. There is also no requirement to upgrade prerequisites (e.g., Tomcat or PostgreSQL) when upgrading Catalog.

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 Linux 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 80/8080 (HTTP) and 443/8443 (HTTPS) must be opened to allow outbound communication to the Internet to allow software to be downloaded.

Prerequisite Installation Script (strongly encouraged)

There is an optional prerequisite installation script which may be used to install the prerequisite packages described in this section. It's use is strongly encouraged. It will automatically download and configure all required prerequisite software, including Java, Tomcat and PostgreSQL. Further, it will automatically configure Tomcat for HTTPS.

It is located within the CatalogInstaller.zip package and is named **CatalogPrereqs.sh**.

- ‘**sudo**’ permission is required to run *CatalogPrereqs.sh*
- *CatalogPrereqs.sh* works in conjunction with the **CatalogInstaller.properties** file. There are three **empty** properties which **MUST** be populated prior to running *CatalogPrereqs.sh*:
 - INSTALL_TYPE (input CDP or EMR)
 - QDC_SERVICE_ACCOUNT_PW (password for the Catalog Linux user that will be created)
 - SUPERUSER_PASSWORD (password for PostgreSQL superuser that will be created)
- All Red Hat Enterprise Linux installations require a valid Red Hat entitlement subscription. This prerequisite script will not run without a subscription.

To run *CatalogPrereqs.sh*:

7. Install unzip (if not present)

All Red Hat Enterprise Linux & CentOS 7 Deployments

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

Ubuntu 20.04 & Ubuntu 22.04 Deployments

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

8. Copy the CatalogInstaller.zip file into an installer “working” directory (example: /tmp)

```
Example: (sudo) # cp CatalogInstaller.zip /tmp
```

9. Unzip CatalogInstaller.zip

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

10. Copy the podium zip file into the CatalogInstaller directory created in Step 3

```
Example: (sudo) # cp podium-4.17-xxxxx.zip /tmp/CatalogInstaller/
```

11. Run CatalogPrereqs.sh

```
(sudo) # cd /tmp/CatalogInstaller
(sudo) # ./CatalogPrereqs.sh
```

12. After CatalogPrereqs.sh has been run successfully you may skip to [Section 4.1 First Time Installation Using the Installer](#)

Manual Prerequisite Installation

Important: During the prerequisite setup process, several items are needed from the Qlik Catalog software distribution, podium-4.17-xxxxx.zip. The instructions below assume the Qlik Catalog software distribution has been unzipped to /tmp:

1. Install unzip (if not present)

All Red Hat Enterprise Linux & CentOS 7 Deployments

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

Ubuntu Deployments

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

2. Expand the Qlik Catalog software distribution to /tmp

```
(sudo) # unzip <replace-path>/podium-4.17-xxxxx.zip -d /tmp/
```

4.1 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 a correct version 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)

2. JDK Installation:

➔ OpenJDK: Use package manager for installation:

OpenJDK 8 installation:

All Red Hat Enterprise Linux & CentOS 7 Deployments

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

Ubuntu Deployments

```
(sudo) # sudo apt install -y openjdk-8-jdk
```

OpenJDK 11 installation:

All Red Hat Enterprise Linux & CentOS 7 Deployments

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

Ubuntu Deployments

```
(sudo) # sudo apt install -y openjdk-11-jdk
```

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

4.2 Create Catalog Linux Account and Qlik Catalog Directory

Create a new Linux user "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 Linux user to run Qlik Catalog (the user which launches Tomcat)

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

Additional step for Ubuntu deployments:

```
(sudo) mkdir /home/qdc && chown qdc:qdc /home/qdc
```

2. Optionally, set a password for the Catalog Linux user - 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 Catalog Linux user and group being used

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

4.3 Tomcat Installation

Qlik Catalog is only supported on Tomcat version 9. Version 9.0.70+ is encouraged for first-time installs. There is no requirement to upgrade Tomcat when upgrading Catalog. If needed, Tomcat upgrade instructions are included elsewhere in this guide.

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

All Red Hat Enterprise Linux & CentOS 7 Deployments

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

Ubuntu Deployments

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

2. Become the Catalog Linux user

```
(sudo) # sudo su - qdc
```

3. Download Apache Tomcat 9.0.70+

```
(qdc) $ cd /usr/local/qdc
```

```
(qdc) $ wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.70/bin/apache-tomcat-9.0.70.tar.gz
```

4. Extract the Tomcat file

```
(qdc) $ tar -xvf apache-tomcat-9.0.70.tar.gz
```

5. The resulting directory, for example “/usr/local/qdc/apache-tomcat-9.0.70”, is known as the Tomcat home directory. When configuring the CatalogInstaller.sh file in the next section, please set TOMCAT_HOME to this value.

6. 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-9.0.70/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" maxHttpHeaderSize="65536"
  compression="on"
  compressionMinSize="150"
  noCompressionUserAgents="gozilla, traviata"
  compressibleMimeType="text/html,text/xml,text/plain,text/css,text/javas
  cript,application/javascript,application/json,application/xml"
  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 [%l]" />
```

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.

All Red Hat Enterprise Linux & CentOS 7 Deployments

```
(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
```

Ubuntu Deployments (if UFW firewall is enabled)

```
(sudo) # sudo ufw allow 8080,8443/tcp
```

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-9.0.70` 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
```

- Optional: Configure Tomcat to support HTTPS (see later section)

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

- Start Tomcat manually

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

- Browse to the following URL to verify that Tomcat is running

`http://<Qlik-Catalog-Node-IP-Address-OR-Hostname>:8080`

- Tomcat can be manually stopped at any time

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

- The Tomcat log can be monitored

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

4.4 PostgreSQL Installation

Qlik Catalog is certified on Qlik Catalog PostgreSQL 14, a customized version of PostgreSQL that supports long schema and table names. First-time Catalog installations must install Qlik Catalog PostgreSQL 14.6. There is no requirement to upgrade PostgreSQL when upgrading Catalog. If needed, PostgreSQL upgrade instructions are included elsewhere in this guide.

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.

- Create a “postgres” user and group

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

Additional step for Ubuntu deployments:

```
(sudo)     # sudo mkdir /home/postgres && chown postgres:postgres
           /home/postgres
```

- Add the “postgres” user to the “qdc” group

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

- Create directories for executables and data, and change their ownership


```
(sudo) # sudo mkdir -p /usr/pgsql/qdc14-6
(sudo) # sudo chown -R postgres:postgres /usr/pgsql
(sudo) # sudo mkdir -p /var/lib/pgsql/14-6/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 RHEL 7/CentOS 7 deployments:

```
(postgres) $ /tmp/podium/thirdParty/qdc_pg14-6_RHEL7-and-
CentOS7.bsx
```

Installer for RHEL 8 deployments:

```
(postgres) $ /tmp/podium/thirdParty/qdc_pg14-6_RHEL8-and-
CentOS8.bsx
```

Installer for RHEL 9.1+ deployments:

```
(postgres) $ /tmp/podium/thirdParty/qdc_pg14-6_RHEL9.bsx
```

Installer for Ubuntu 20.04 deployments:

```
(postgres) $ /tmp/podium/thirdParty/qdc_pg14-6_Ubuntu20.bsx
```

Installer for Ubuntu 22.04 deployments:

```
(postgres) $ /tmp/podium/thirdParty/qdc_pg14-6_Ubuntu22.bsx
```

5. Create symlinks to executables

```
(sudo) # sudo ln -s /usr/pgsql/qdc14-6/bin/psql /usr/bin/psql
(sudo) # sudo ln -s /usr/pgsql/qdc14-6/bin/pg_dump /usr/bin/pg_dump
(sudo) # sudo ln -s /usr/pgsql/qdc14-6/bin/pg_restore
/usr/bin/pg_restore
```

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.

All Red Hat Enterprise Linux & CentOS 7 Deployments

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

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

Ubuntu Deployments

```
(sudo) # sudo ufw allow 5432/tcp
```

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.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.service
/etc/systemd/system/
(sudo) # sudo systemctl daemon-reload
(sudo) # sudo systemctl enable qdc_pg.service
(sudo) # sudo systemctl start qdc_pg.service
```

4.5 Container Platform & Node.js

Qlik Catalog requires a container platform to run containers. Platform is determined by the operating system upon which Catalog is deployed:

CentOS 7: *Docker Community Edition*

RHEL 7: *Docker Enterprise*

RHEL 8, RHEL 9.1, Ubuntu 20.04, and Ubuntu 22.04: *Podman*

Node.js is required for integration with Qlik Sense.

1. Install Node.js (which includes npm)

All Red Hat Enterprise Linux Deployments

```
(sudo) # curl -sL https://rpm.nodesource.com/setup_18.x | sudo bash -
(sudo) # sudo yum install -y nodejs
```

CentOS 7 Deployments

```
(sudo) # curl -sL https://rpm.nodesource.com/setup_16.x | sudo bash -
(sudo) # sudo yum install -y nodejs
```

Ubuntu Deployments

```
(sudo) # curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
(sudo) # sudo apt install -y nodejs
```

2. Install Container Platform

CentOS 7 Deployments: *Docker Community Edition (CE)*

Docker CE installation instructions for CentOS 7

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
```

RHEL 7 Deployments: *Docker Enterprise*

- *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.
- Official 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

Docker Enterprise installation instructions for RHEL 7

1. Install Docker, set it to start automatically, and start it:

```
(sudo) # subscription-manager repos --enable=rhel-7-server-rpms
(sudo) # subscription-manager repos --enable=rhel-7-server-extras-rpms
(sudo) # subscription-manager repos --enable=rhel-7-server-optional-rpms

(sudo) # yum install docker device-mapper-libs device-mapper-event-libs
(sudo) # systemctl enable docker.service
(sudo) # systemctl start docker.service

(sudo) # groupadd docker
(sudo) # chown -R root:docker /run/docker /etc/sysconfig/docker /etc/docker
/var/lib/docker /usr/bin/docker /usr/share/bash-completion/completions/docker
/usr/libexec/docker
(sudo) # chmod -R 770 /var/lib/docker

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

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

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

3. Install Docker Compose

```
(sudo) sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.0/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
```

4. Test Docker and Docker Compose -- this should be done as the Catalog Linux 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				

RHEL 8 Deployments: *Podman*

- *Podman* is the officially supported container platform for RHEL 8.
- *Podman* is a "daemonless" container platform

RHEL 8 *Podman* installation instructions

```
(sudo) # yum module install -y container-tools
(sudo) # loginctl enable-linger qdc    (where "qdc" is the Catalog Linux user)
```

Test *Podman* command as the Catalog Linux user

```
(sudo) # sudo su - qdc
(qdc)  $ podman info
```

Verify Linger is enabled for the Catalog Linux user account

```
(qdc)  $ loginctl user-status qdc | grep inger
```

Result:

```
State: lingering
Linger: yes
```

RHEL 9 Deployments: *Podman*

- ***Podman*** is the officially supported container platform for RHEL 9.
- *Podman* is a "daemonless" container platform

RHEL 9 *Podman* installation instructions

```
(sudo) # yum install -y podman
(sudo) # loginctl enable-linger qdc    (where "qdc" is the Catalog Linux user)
```

Test *Podman* command as the Catalog Linux user

```
(sudo) # sudo su - qdc
(qdc)  $ podman info
```

Verify Linger is enabled for the Catalog Linux user account

```
(qdc)  $ loginctl user-status qdc | grep inger
```

Result:

```
State: lingering
Linger: yes
```

Ubuntu 20.04 & Ubuntu 22.04 Deployments: *Podman*

- *Podman* is the supported container platform for Ubuntu 20.04 & Ubuntu 22.04 deployments.

- *Podman* is a "daemonless" container platform

Ubuntu 20.04 *Podman* installation instructions

```
(sudo) # apt install curl gnupg2 -y
(sudo) # echo 'deb
http://download.opensuse.org/repositories/devel:kubic:libcontainers:stable/xUbuntu_20.04/
/' > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
(sudo) # wget -nv
https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable/xUbuntu_20.04/R
elease.key -O- | apt-key add -
(sudo) # apt-get update -qq -y
(sudo) # apt-get -qq --yes install podman
(sudo) # loginctl enable-linger qdc    (where "qdc" is the Catalog Linux user)
```

Test *Podman* command as the Catalog Linux user

```
(sudo) # sudo su - qdc
(qdc)  $ podman info
```

Verify Linger is enabled for the Catalog Linux user account

```
(qdc)  $ loginctl user-status qdc | grep inger
```

Result:

```
State: lingering
Linger: yes
```

Ubuntu 22.04 *Podman* installation instructions

```
(sudo) # apt-get -y install podman
(sudo) # loginctl enable-linger qdc    (where "qdc" is the Catalog Linux user)
```

Test *Podman* command as the Catalog Linux user

```
(sudo) # sudo su - qdc
(qdc)  $ podman info
```

Verify Linger is enabled for the Catalog Linux user account

```
(qdc)  $ loginctl user-status qdc | grep inger
```

Result:

```
State: lingering
Linger: yes
```

4.6 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 /podiumbase
```

2. Change ownership of <qlik catalog base directory> to the Catalog Linux running Qlik Catalog. Both the Catalog Linux user and group should have "rwx" access to the Catalog base directory:

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

```
[qdc]$ hadoop fs -ls /
Found 6 items
drwxr-xr-x   - hive hive                0 2020-07-17 10:43 /hive
drwxrwxr-x   - qdc  qdc                0 2021-08-05 11:26 /podiumbase
drwxrwxrwt   - hdfs supergroup         0 2021-08-05 14:15 /tmp
drwxr-xr-x   - hdfs supergroup         0 2021-07-23 17:58 /user
drwxr-xr-x   - hdfs supergroup         0 2020-07-17 10:43 /warehouse
drwxr-xr-x   - yarn hadoop             0 2020-07-17 10:43 /yarn
```

3. On all nodes in the cluster:
 - 1) create a user & group with the same name as the Catalog Linux user
 - 2) add user 'hive' to the Catalog Linux group

Example:

```
# adduser qdc
# usermod -aG hive qdc
```

4. Verify hive membership to the Catalog Linux group from Catalog node:

```
[qdc]$ hdfs groups hive
hive : hive qdc
```

5. When new files and directories are created under /podiumbase, both user and group qdc must have "rwx" access. This can be done using HDFS ACLs. First change any existing files and directories. Then, set the default for all future files and directories. Do this as owner or 'superuser' of hadoop cluster (e.g., 'hdfs'):

```
[qdc]$ sudo su - hdfs
Last login: Fri Jul 23 17:57:30 EDT 2021 on pts/0

[hdfs]$ hadoop fs -setfacl -m -R group:qdc:rwx /podiumbase

[hdfs]$ hadoop fs -setfacl -m -R default:group:qdc:rwx /podiumbase
```

<https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html#setfacl>

4.7 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.

- 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;
```

4.8 EMR Deployments Only: Setup Edge Node

The Qlik Catalog AWS EC2 instance must be configured as an "edge node" to interact with the EMR cluster. The process is detailed in [Section 5.6.1](#), and a sample shell script has been included to facilitate edge node setup. *Edge node setup should be completed prior to running the Qlik Catalog installer in Section 4.*

5.0 Qlik Catalog Software Installation

The Qlik Catalog installer is a shell script, ***CatalogInstaller.sh***, that is guided by a properties file containing configuration, ***CatalogInstaller.properties***. The same properties file can be used both for the initial install as well as later upgrades.

The shell script and properties files are included in a zip file, ***CatalogInstaller.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-4.17-xxxxx.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 Linux 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

5.1 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 Linux user:

```
(sudo) # sudo su - qdc
```

2. Unzip CatalogInstaller.zip into a working directory
3. Copy the Qlik Catalog software zip file (e.g., podium-4.17-xxxxx.zip) into the working directory

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

```
podium-4.17-xxxxx.zip CatalogInstaller.sh CatalogInstaller.sh CatalogInstaller.txt QCA.txt  
upgrade-scripts (directory)
```

5. Edit the installation parameters in **CatalogInstaller.properties** -- additional documentation is present in that file. There are three empty properties which MUST be populated prior to running CatalogInstaller.sh:
 - INSTALL_TYPE (input CDP or EMR)
 - QDC_SERVICE_ACCOUNT_PW (password for the Catalog Linux user created previously)
 - SUPERUSER_PASSWORD (password for PostgreSQL superuser set previously)

Additionally, the following properties must be defined for multi-node deployments:

- `PODIUM_BASE` -- change this to a path in HDFS (or S3) to which the service user has write permission
- `DISTRIBUTION_URI` -- set to the Hive JDBC URI (if known)
- If the install type is EMR, set `BASE_URI`, `BASE_URI_USERNAME` and `BASE_URI_PASSWORD` as suggested in the comments
- If PostgreSQL will be hosted 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 or 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: `./CatalogInstaller.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 `CatalogInstaller.sh` -- 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) # ./CatalogInstaller.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. Open 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 `CatalogInstaller.sh`, please replace “qdc” with the alternative webapp name.

```
http://<QDC node hostname or IP address>:8080/qdc
```

10. 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.

5.1.1 Post-Installation Tasks

The installer deployed four containers which are used by Qlik Catalog:

- Licenses
- Engine
- DCAAS

- Data Rest Connector

These containers should be configured to **auto-start during system boot**. (In fact, the Licenses container **MUST** be running in order to login to Qlik Catalog).

A **post-installation script** which simplifies the process of configuring the containers to run as services is included, and its use is strongly encouraged. It is located within the CatalogInstaller.zip package and is named **CatalogPostInstall.sh**:

NOTE: 'sudo' permission is required to run *CatalogPostInstall.sh*

To run **CatalogPostInstall.sh**:

1. Switch to a user account with 'sudo' privileges
2. Navigate to the CatalogInstaller working directory (used above to run the Catalog installer)

Example: \$ cd /tmp/CatalogInstaller

3. Run CatalogPostInstall.sh using an account with 'sudo'

(sudo) \$ sudo ./CatalogPostInstall.sh

Manual Container Service Configuration

If you did not run CatalogPostInstall.sh, the Qlik Licenses container must be setup as a service to automatically start when Linux restarts.

IMPORTANT:

- a) 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
- b) If deploying to RHEL 8, RHEL 9, Ubuntu 20.04, or Ubuntu 22.04 which use Podman to manage containers you must edit **/etc/systemd/system/qlikContainers.service** after the copy (cp) step below:
 - a. REMOVE line 13 completely:
Requires=docker.service
 - b. EDIT line 14 and remove the following text: docker.service

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! 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!".

Important Notes

- Any **JDBC drivers** for your licensed RDBMS should be placed in the directory called out by the following core_env property. This directory is preferred over placing drivers in \$TOMCAT_HOME/webapps/qdc/WEB-INF/lib, where they will be overwritten on upgrade and where they may interfere with other libraries.

```
# An alternate directory to WEB-INF/lib for JDBC driver jars.  
# May also be set directly, for a given driver, on table  
# podium_core.pd_jdbc_source_info, column alt_classpath.  
# Restart required. Default: not set  
jdbc.alternate.classpath.dir=/usr/local/qdc/jdbcDrivers
```

- If a JDBC driver is particularly complicated and consists of multiple jars (e.g., the Simba Google Big Query driver has dozens of jars), it can be further isolated into its own sub-directory. If you do this, you must run a SQL statement as follows (default password is "nvs2014!"; **update** path and name):

```
psql podium_md -U podium_md -c "update podium_core.pd_jdbc_source_info  
set alt_classpath = '/usr/local/qdc/jdbcDrivers/simbaBigQuery' where  
sname = 'BIGQUERY';"
```

- When running Qlik Catalog multi-node, 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.

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

Do not add any libraries from the Hadoop ecosystem, including Hive classes and the Hive JDBC jar, to the classpath, 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.service
 - test: (sudo) # psql
- RHEL 7/CentOS 7 Deployments using 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

Check for Running Containers:

- *Docker*:
(sudo) # sudo docker inspect -f '{{.State.Running}}' qlikcore_qix-engine_1
(sudo) # sudo docker inspect -f '{{.State.Running}}' licenses
- *Podman*:
(qdc) \$ podman inspect -f '{{.State.Running}}' catalog-pod-engine
(qdc) \$ podman inspect -f '{{.State.Running}}' catalog-pod-licenses
- 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 using *Docker*:
(qdc) \$ docker inspect -f '{{.State.Running}}' qlikcore_qix-engine_1
(qdc) \$ docker inspect -f '{{.State.Running}}' licenses
 - Test using *Podman*:
(qdc) \$ podman inspect -f '{{.State.Running}}' catalog-pod-engine
(qdc) \$ podman inspect -f '{{.State.Running}}' catalog-pod-licenses
- Tomcat: (qdc) \$ /usr/local/qdc/apache-tomcat-9.0.70/bin/startup.sh

5.2 PostgreSQL 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** 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 PostgreSQL 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-4.17-xxxxx.zip podium/config/core_ddl.txt -d /tmp/
```

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


5.3 Upgrade of Qlik Catalog

NOTE: Beginning with Qlik Catalog May 2021, **Apache Tomcat 9 is mandatory**. Prior installations using Tomcat 7 must be migrated. The installer will disallow use of any version of Tomcat other than version 9. Please see the section "Migrating to or Upgrading Tomcat 9". Notably, at the end, the installer should be run **WITHOUT** the upgrade option "-u". Please review the release notes and the below guidance before proceeding.

You should re-use the CatalogInstaller.sh file from the initial install, updating the value for PODIUM_RELEASE_FILE.

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 and later:

1. The installer must be run as the Qlik Catalog Linux user:

```
(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.
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 CatalogInstaller.zip into a working directory
6. Copy the Qlik Catalog software zip file (podium-4.17-xxxxx.zip) into the working directory
7. Change directory to the working directory. It should contain the following:

```
podium-4.17-xxxxx.zip CatalogInstaller.sh CatalogInstaller.sh CatalogInstaller.txt
QCA.txt upgrade-scripts (directory)
```

8. Edit the installation parameters in CatalogInstaller.sh -- 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, CATALOG_IPADDRESS, and 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 CatalogInstaller.sh -- 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 (or log4j2.xml) is automatically restored during upgrade.

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

10. **Restore** any manually copied Java library jars to <tomcat home>/webapps/qdc/WEB-INF/lib. 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
```

5.4 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 **CatalogInstaller.properties** -- additional documentation is present in that file. There are three empty properties which **MUST** be populated prior to running CatalogInstaller.sh:
 - INSTALL_TYPE (input CDP or EMR)
 - QDC_SERVICE_ACCOUNT_PW (password for the Catalog Linux user created previously)
 - SUPERUSER_PASSWORD (password for PostgreSQL superuser set previously)

Additionally, set the following properties accordingly for the environment:

- PODIUM_BASE -- change this to a path in HDFS (or S3) to which the service user has write permission
- DISTRIBUTION_URI -- set 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 PostgreSQL will be hosted on a different server, set CREATE_PODIUM_MD_DATABASE=false

2. the following options must be specified following the CatalogInstaller.sh command:
 - -s (silent)

- -a (Accept Qlik Customer Agreement)

Example: `./CatalogInstaller.sh -s -a`

NOTE: Invoking the “-a” option indicating acceptance of the Qlik Customer Agreement (QCA) 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 CUSTOMER AGREEMENT (QCA) 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.

5.5 Distribution Specific Configuration Parameters

5.5.1 Cloudera Private Cloud (CDP) Configuration

5.5.1.1 Tez Configuration

Prepare using Tez has been tested (in addition to local-mode and MapReduce).

1. Copy Tez to an HDFS folder (here the directory being used is the product base directory (aka podium.base in `core_env.properties`):

```
# hadoop fs -copyFromLocal /opt/cloudera/parcels/CDH/lib/tez/tez.tar.gz
hdfs://<NAME_SERVICE>:8020/<PODIUM_BASE_DIR>/
```

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

```
tez.lib.uris=hdfs://<NAME_SERVICE>:8020/<PODIUM_BASE_DIR>/tez.tar.gz
```

5.5.1.2 Hive User Defined Functions (UDFs) Configuration

Hive, used for queries and Publish, requires access to Catalog user defined functions (UDFs).

1. Copy `qdc-hudf-<VERSION>.jar` to an HDFS folder (here the directory being used is the product base directory (aka podium.base in `core_env.properties`):

```
# hadoop fs -copyFromLocal <TOMCAT_HOME>/webapps/qdc/WEB-INF/lib/qdc-
hudf-<VERSION>.jar hdfs://<NAME_SERVICE>:8020/<PODIUM_BASE_DIR>/
```

2. Specify the full path of hudf archive in `core_env.properties`:

```
podium.hive.udf.jar.location=hdfs://<NAME_SERVICE>:8020/<PODIUM_BASE_DIR>/qdc-hudf-4.15.jar
```

Additionally, you will need to add a property to the Hive service `hive-site.xml` to allow several properties to be SET when issuing Hive SQL queries. Append the following to the `hive-site.xml` file:


```

<property>
  <name>hive.security.authorization.sqlstd.confwhitelist</name>
  <value>podium\.obfuscation\.rules\.hdfs\.path\|hive\.server2\.thrift\.res
ultset\.default\.fetch\.size|hive\.fetch\.task\.conversion|hive\.merge\.m
apredfiles|hive\.merge\.size\.per\.task|hive\.merge\.smallfiles\.avgsizel
hive\.execution\.engine</value>
</property>

```

Note: You will need to restart the cluster Hive service for the changes to take effect.

6.0 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 contains all the primary and secondary configuration parameters necessary for desired operation. The file is found at `<tomcat home>/conf/core_env.properties`. 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.

6.1 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-<VERSION>.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-<VERSION>.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-<VERSION>.jar
podium.obfuscation.dictionary.hdfs.path=hdfs://apps/podium/
```

6.2 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).

6.2.1 Running Qlik Catalog in a Kerberized Environment

1. Create a Linux user for the Qlik Catalog application on all nodes in the cluster as well as the Qlik Catalog nodes. Example: `qdc`. 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 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 created 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 Catalog Linux user

```
kinit podium-kt podium.keytab
```

13. Start Qlik Catalog as the Catalog Linux user

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:

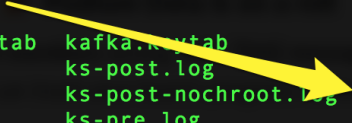
```
xst -kt <keytab-file-name> -norandkey user@KERBEROS.REALM
```

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 ~]#
```



6.2.2 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>

6.2.2.1 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_etypes = aes256-cts-hmac-sha1-96(user-defined)  
default_tkt_etypes = aes256-cts-hmac-sha1-96(user-defined)  
permitted_etypes = aes256-cts-hmac-sha1-96(user-defined)  
udp_preference_limit = 1  
useFirstPass=true  
doNotPrompt=true
```

6.2.2.2 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}$
```

6.2.2.3 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>
```

6.3 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 Ranger 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.

6.3.1 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 Catalog Linux user 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>
```

6.3.3 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.

6.3.4 Qlik Catalog Settings to Enable Impersonation

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

```
enable.impersonation=true
```

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

```
jdbc:hive2://domain.corp.mycompany.com:1000/default;principal=hive/domaincorp.mycompany.com@MYCOMPANY_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==}
```

6.3.5 User Rep (ServiceUser) Running Qlik Catalog Processes

The User running the Qlik Catalog process (referred to as ServiceUser at places) 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.

6.3.6 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.

6.3.7 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
```

6.4 Enabling SAML

This section covers configuring Catalog to use one of three SAML providers: Microsoft Azure Active Directory, Okta, and PingFederate. If using a different SAML provider, please review this section and apply the same approach to your provider.

Note for Other SAML Providers

Some organizations prohibit the HTTP-based exchange of metadata between their Identity Provider (typically on-premise) and Service Provider (Catalog). If so, Catalog can be configured to consume a local, exported copy of the Identity Provider (IdP) metadata by setting a `core_env` property to reference a local file:

```
saml.metadata.provider=file:///usr/local/qdc/idp.xml
```

Further, Catalog's Service Provider metadata can be exported to an XML file for hand delivery to the Identity Provider:

```
curl -k https://<REPLACE_WITH_FULL_CATALOG_HOSTNAME>:8443/qdc/saml2/service-provider-metadata/catalog > spExport.xml
```

The hostname in the above URL should be that used by end-users to access Catalog.

6.4.1 Enabling SAML using Microsoft Azure Active Directory (MS AAD)

(1) Qlik Catalog Tomcat must first be configured for HTTPS (see section below) -- new installs since May 2022 are automatically configured with HTTPS.

(2) Two properties must be set in the MS AAD "Single Sign-On with SAML" "Basic SAML Configuration" dialog – replace `<HOSTNAME>` with the Qlik Catalog hostname:

- **Identifier (Entity ID)** – `https://<HOSTNAME>:8443/qdc/saml2/service-provider-metadata/catalog`
- **Reply URL (Assertion Consumer Service URL)** – `https://<HOSTNAME>:8443/qdc/login/saml2/sso/catalog`

Set up Single Sign-On with SAML

An SSO implementation based on federation protocols improves security, reliability, and end user experiences and is easier to implement. Choose SAML single sign-on whenever possible for existing applications that do not use OpenID Connect or OAuth. [Learn more](#).

Read the [configuration guide](#) for help integrating QDC-mark.

1

Basic SAML Configuration

 Edit

Identifier (Entity ID)	https://dfn.local:8443/qdc/saml2/service-provider-metadata/catalog
Reply URL (Assertion Consumer Service URL)	https://dfn.local:8443/qdc/login/saml2/sso/catalog
Sign on URL	<i>Optional</i>
Relay State (Optional)	<i>Optional</i>
Logout Url (Optional)	<i>Optional</i>

(3) The following three `core_env` properties must be set (see notes immediately following):

```
authentication.mode=SAML
```

```
# this will log out of Qlik Catalog and MS AAD
# replace <tenant-id> and <HOSTNAME> (and possibly 8443 and qdc)
logout.url=https://login.windows.net/<tenant-id>/oauth2/logout?post_logout_redirect_uri=https%3A%2F%2F<HOSTNAME>%3A8443%2Fqdc%2Flogged-out
```

```
# replace <tenant-id> and <app-id>
saml.metadata.provider=https://login.microsoftonline.com/<tenant-id>/federationmetadata/2007-06/federationmetadata.xml?appid=<app-id>
```

Notes on `core_env` properties:

- property "saml.metadata.provider" is taken from MS AAD setup, "**App Federation Metadata Url**"
- property "logout.url" was formerly "saml.logout.url" – it applies to both manually initiated logout via the menu, as well as session timeout
- property "logout.url", when configured for login.windows.net, now supports "post_logout_redirect_uri" to allow redirection from MS back to the Catalog logged-out page
- property "saml.entity.baseurl" is no longer set (May 2021 change)
- property "saml.keystore.path" is no longer set, and there is no longer a need to download the Base64 Certificate and add it to samlKeystore.jks – in fact, samlKeystore.jks is no longer present (May 2021 change)
- do **NOT** copy "**Logout URL**" from MS AAD into the `core_env` property "logout.url"
- there are only two valid choices for "logout.url": /logged-out (log out of only Qlik Catalog) or https://login.windows.net/<tenant-id>/oauth2/logout (log out of MS AAD and Qlik Catalog)
- IMPORTANT:** when logging in to the UI, use URL "https://<HOSTNAME>:8443/qdc" and not "https://<HOSTNAME>:8443/qdc/login"

6.4.2 Enabling SAML using Okta

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 Catalog is the Service Provider (SP).

1. Log in to your Okta organization as a user with administrative privileges. You can 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 URL below into the "Single Sign On URL" field – replace <HOSTNAME> with the Qlik Catalog hostname:

```
https://<HOSTNAME>:8443/qdc
```

Then, paste the URL below into the "Audience URI (SP Entity ID)" [old] or "Audience Restriction" [new] field:

```
https://<HOSTNAME>:8443/qdc/saml2/service-provider-metadata/catalog
```

Then, check the box "Allow this app to request other SSO URLs" and paste the URL below into the "Requestable SSO URLs" field:

```
https://<HOSTNAME>:8443/qdc/login/saml2/sso/catalog
```

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-248822.okta.com/app/exk2z8xy1fc0t2tRg4x7/sso/saml/metadata
```
11. You can also add additional users in the **People** section. All these users will need to be added to podium 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 (Tomcat).

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

1. Log in to <https://<HOSTNAME>: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 qdc.

Important! A user with the same username must previously exist in Qlik Catalog.

2. Log in 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. If in the admin console, use the "My end user dashboard" from the top-right corner.

6.4.3 Enabling SAML using PingIdentity's *PingOne* & *PingFederate*

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

PingIdentity's *PingOne* SAML connection has been demonstrated to work with Catalog using the following steps:

1. From within PingOne admin console, add a SAML app by clicking on the **Connections** icon
2. Click the "+" icon & choose WEB APP
3. Select connection type "SAML"
4. Name new application. (Optionally, provide description and icon)
5. **Configure SAML** Connection:

Choose **Manually Enter** and accept all defaults except for the fields below

- **ACS URLS:** <https://<CATALOG-HOSTNAME>:8443/qdc/login/saml2/sso/catalog>
- **ENTITY ID:** <https://<CATALOG-HOSTNAME>:8443/qdc/saml2/service-provider-metadata/catalog>
- **ASSERTION VALIDITY DURATION:** 60

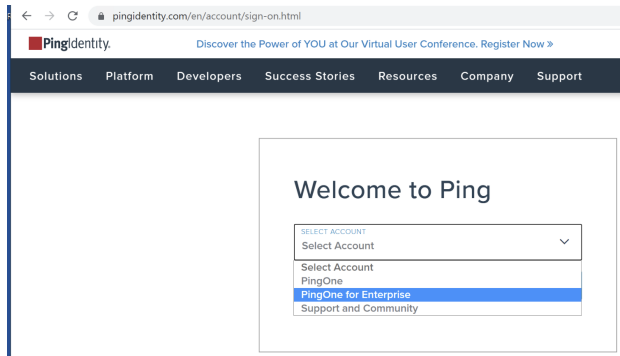
6. **Attribute Mapping:**

- Change default "User ID" attribute to "Username" and set SAML_SUBJECT to "Username". One customer (where Ping was backed by Active Directory) used "sAMAccountName" rather than "Username".

7. **Limit Access** (Optional):

As indicated within the console, all users have access to the application when no groups are defined. The Access tab serves to limit access to the Qlik Catalog application if desired. To limit access: create a group, add members, and add the group within the Access tab of the Qlik Catalog application.

Some customers may use Ping Federate in conjunction with the PingOne for Enterprise cloud-based product:



You must import the Ping Federate certificate into the Java Key Store (<https://stackoverflow.com/questions/2138940/import-pem-into-java-key-store>) or you will receive the following error during Tomcat startup after configuring SSO: "PKIX path building failed" and "unable to find valid certification path to requested target"

Instructions for exporting the certificate from Ping Federate may be found here: <https://docs.pingidentity.com/bundle/pingfederate-93/page/yij1564002984865.html>

Detailed instructions for setting up an **SP Connection** for use by Catalog are available here: <https://docs.pingidentity.com/bundle/solution-guides/page/ozz1597769517562.html>

Summary

SP Connection	
Connection Type	
Connection Role	SP
Browser SSO Profiles	true
Protocol	SAML 2.0
Connection Template	No Template
WS-Trust STS	false
Outbound Provisioning	false
Connection Options	
Browser SSO	true
IdP Discovery	false
Attribute Query	false
General Info	
Partner's Entity ID (Connection ID)	https://installer2.corp.podiumdata.com:8443/qdc/saml2/service-provider-metadata/catalog
Connection Name	CatalogFederate
Base URL	https://installer2.corp.podiumdata.com:8443/qdc
Company	qlik
Browser SSO	
SAML Profiles	
IdP-Initiated SSO	true
IdP-Initiated SLO	false
SP-Initiated SSO	true
SP-Initiated SLO	false
Assertion Lifetime	
Valid Minutes Before	5
Valid Minutes After	5
Assertion Creation	
Identity Mapping	
Enable Standard Identifier	true
Attribute Contract	
Attribute	SAML_SUBJECT
Subject Name Format	urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
Authentication Source Mapping	
Adapter instance name	PingOne HTML_Form Adapter
Adapter Instance	
Selected adapter	PingOne HTML_Form Adapter
Mapping Method	
Adapter	HTML_Form IdP Adapter

Integration	
SP Connections	Mapping Method: Use only the Adapter Contract values in the mapping
SP Adapters	Attribute Contract Fulfillment: SAML_SUBJECT: username (Adapter)
Target URL Mapping	Issuance Criteria: Criterion: (None)
SP Default URLs	Protocol Settings
Policy Contract Adapter Mappings	Assertion Consumer Service URL: Endpoint: URL: /login/saml2/soo/catalog (POST)
Adapter-to-Adapter Mappings	Allowable SAML Bindings: Artifact: false, POST: true, Redirect: false, SCAP: false
	Signature Policy: Require digitally signed AuthN requests: false, Always Sign Assertion: true, Sign Response As Required: false
	Encryption Policy: Status: Inactive
	Credentials
	Digital Signature Settings: Selected Certificate: 01-7B7D:ED:F9:76 (CN=david.jenkins@qlik.com), Include Certificate in KeyInfo: true, Include Raw Key in KeyInfo: false, Selected Signing Algorithm: RSA_SHA256

6.5 Tomcat SSL Configuration

Configure Tomcat to Support SSL (HTTPS)

NOTE: As of the May 2022 release, if the optional prerequisites script `CatalogPrereqs.sh` is used, Tomcat is **automatically configured for HTTPS** (including the generation of a self-signed certificate), and requests on port 8080 are automatically redirected to 8443.

The following steps are for customers that did not use the May 2022 (or more recent) `CatalogPrereqs.sh` script.

1. **Generate Keystore** - Use 'keytool' command to create a self-signed certificate. Optionally change the password from "changeit".

Example:

```
(qdc) # keytool -genkey -noprompt -validity 3650 -alias qdc -keyalg RSA -dname 'CN=Qlik Catalog,O=Qlik Technologies Inc.' -ext SAN=DNS:$(hostname) -ext EKU:c=serverAuth -keystore /usr/local/qdc/qdc.jks -storepass changeit -keypass changeit
```

The above created a certificate (with alias 'qdc') located at '/usr/local/qdc/qdc.jks' (the filename that was provided in the keytool command).

Certification Details

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

Example:

```
(qdc) # keytool -list -v -keystore /usr/local/qdc/qdc.jks -storepass changeit
```

2. **Add Connector in server.xml** - Locate your Tomcat's server configuration file at `$TOMCAT_HOME/conf/server.xml`; modify it by adding a connector element to support https connections as follows:

Under the existing Connector, which begins "`<Connector port="8080"`", add the following:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxHttpHeaderSize="65536"
server="Unknown Application Server"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="{qdc.home}/qdc.jks"
keystoreType="JKS"
```

```
keystorePass="changeit"
keyPass="changeit" />
```

Note: The passwords (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, create file \$TOMCAT_HOME/webapps/qdc/WEB-INF/web.xml by either:

```
(qdc) $ cp /tmp/podium/config/tomcat9-web.xml /usr/local/qdc/apache-tomcat-9.0.70/webapps/qdc/WEB-INF/web.xml
```

or manually adding the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4" id="WebApp_ID">
  <session-config>
    <session-timeout>60</session-timeout>
  </session-config>
  <listener>
    <listener-
class>org.springframework.web.context.request.RequestContextListener<
/listener-class>
  </listener>
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Entire Application</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
  </security-constraint>
</web-app>
```

4. Restart Tomcat

6.6 Port Redirection

Port redirection may be configured at the OS-level to redirect standard HTTPS traffic on port 443 to port 8443. Doing this will remove the need to specify the Tomcat port in the Catalog URL. For instance, instead of entering <https://catalog-server.mydomain.com:8443/qdc> a user may simply enter <https://catalog-server.mydomain.com/qdc>.

An OS-level redirect is strongly preferred to directly configuring Tomcat to use port 443, as ports below 1024 are privileged and would require granting root access to the "qdc" Linux user.

Firewalld Configuration (CentOS 7 & RHEL 7/8/9.1)

Firewalld is the default local firewall package used by CentOS & RHEL 7, 8, and 9.1. Enter the following firewalld command to redirect requests from 443 to 8443:

```
$ sudo firewall-cmd --permanent --direct --add-rule ipv4 nat PREROUTING 1 -p tcp --dport 443 -j REDIRECT --to-ports 8443
```

For CentOS 7 deployments using Docker additional configuration is required to allow the LICENSES container to bypass the new redirect so that it can properly route to <https://license.qlikcloud.com>:

1. Run the following command to create a rule for the LICENSES container to bypass the 443 redirect:

```
$ sudo firewall-cmd --permanent --direct --add-rule ipv4 nat PREROUTING 0 -s 172.20.0.0/16 -j ACCEPT
```

2. Run the following command to implement the new rules:

```
$ sudo firewall-cmd --reload
```

UFW Configuration (Ubuntu 20 & Ubuntu 22)

UFW ("uncomplicated firewall") may be installed via APT package manager: `$ sudo apt install ufw`

There are no **ufw** commands for setting up the port forwards. It must be done via configuration file.

1. Add the lines below to `/etc/ufw/before.rules` just prior to the "filter" section (near the top of the file):

```
*nat
:PREROUTING ACCEPT [0:0]
-A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 8443
COMMIT
```

2. Restart and enable ufw to start on boot:

```
$ sudo ufw enable
```

6.7 Testing Licenses Connectivity to QlikCloud

The *licenses* container must be able to contact <https://license.qlikcloud.com> periodically to verify the validity of the license being used for Catalog. You can login to the licenses container and manually verify connectivity to QlikCloud. This is particularly important after configuring port redirection.

1. Login to the licenses container according to the container manager in use (do this as the Catalog Linux user):

```
$ docker exec -it -u=root licenses /bin/sh
$ podman exec -it -u=root catalog-pod-licenses /bin/sh
```

2. Test connectivity by issuing the following command from within the container:

```
/ # wget https://license.qlikcloud.com/v1/definitions
```

3. A successful connection will return a “401 unauthorized” error:

```
Connecting to license.qlikcloud.com (107.20.116.99:443)
wget: server returned error: HTTP/1.1 401 Unauthorized
```

4. If the connection attempt hangs it is an indication that the licenses container is not able to make the connection to QlikCloud. Running a **traceroute** to QlikCloud from within the container may reveal a problem with the network path being used.

In the example below, the licenses container (10.0.2.100) exits the pod network at 10.0.2.2, hops to the LAN gateway at 10.118.0.1 and eventually gets out to the Internet.

```
/ # traceroute license.qlikcloud.com
traceroute to license.qlikcloud.com (52.73.12.217), 30 hops max, 46 byte packets
 1 10.0.2.2 (10.0.2.2)  0.113 ms  0.102 ms  0.098 ms
 2 10.118.0.1 (10.118.0.1)  1.088 ms  0.954 ms  0.803 ms
 3 10.118.17.6 (10.118.17.6)  0.229 ms  0.210 ms  0.207 ms
 4 144.121.57.97.lighttower.net (144.121.57.97)  6.823 ms  1.438 ms  1.289 ms
```

6.8 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
- **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>
```

6.8.1 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:

1. Copy the EC2 ssh key used during EMR cluster creation to the Qlik Catalog node.
2. Edit the **variables** at the beginning of the script according to your AWS environment

Note: The edge node script should be run following completion of Catalog prerequisite setup but prior to running the Catalog installation script (CatalogInstaller.sh).

```

--- BEGIN SCRIPT ---

#!/bin/bash

# setup logging
output_file=CreateEdgeNode_$(date +%s).log
exec > >(tee $output_file)
exec 2>&1

# Variables
# Set First Variables Section Specific To Your Environment
# Requires ssh key for master access to be set up prior to run

emrMasterDns="172.31.0.90"
CATALOG_SERVICE_ACCOUNT="qdc"
emrSshKey="/tmp/Dev2018.pem"
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"

#####

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
$instanceStorePath/var/log/pig /var/log/hive/user/$CATALOG_SERVICE_ACCOUNT /var/aws/emr/"

bashRc="/home/"$CATALOG_SERVICE_ACCOUNT"/.bashrc"

# 1. 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 $CATALOG_SERVICE_ACCOUNT:$CATALOG_SERVICE_ACCOUNT $libDir
    scp -r -i $emrSshKey $hadoop@$emrMasterDns:$libDir/* $libDir
done

# 2. Create supporting dirs and set ownership
for addDir in $additionalDirs; do
    echo "Setting up $addDir"
    sudo mkdir -p $addDir
    sudo chown -R $CATALOG_SERVICE_ACCOUNT:$CATALOG_SERVICE_ACCOUNT $addDir
done

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

# 4. 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

# 5. Create /mnt1 & grant the Catalog Linux group write access to /mnt & /mnt1
mkdir /mnt1
chown -R root:$CATALOG_SERVICE_ACCOUNT /mnt && chown -R root:$CATALOG_SERVICE_ACCOUNT /mnt1
chmod -R 775 /mnt && chmod -R 775 /mnt1

# 6. Set environment variables
newPath=$PATH
for libDir in $libDirs; do
    newPath="$newPath:$libDir/bin"
done

sed -i '#export JAVA_HOME=/ c\export JAVA_HOME=$JAVA_HOME' /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
sed -e '/ddb/ s/^\#*/#/' -i /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
sed -e '/goodies/ s/^\#*/#/' -i /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
sed -e '/kinesis/ s/^\#*/#/' -i /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
sed -e '/cloudwatch/ s/^\#*/#/' -i /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
sed -e '/security/ s/^\#*/#/' -i /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
echo "" >> /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
echo '# Added by Edge Node Script' >> /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
echo '# client needed for Sqoop' >> /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
echo 'export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/lib/hadoop/client/*"' >> /usr/lib/hadoop/etc/hadoop/hadoop-env.sh
echo 'export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/lib/hadoop-mapreduce/*"' >> /usr/lib/hadoop/etc/hadoop/hadoop-env.sh

echo '# Appended By Edge Node Script' >> $bashRc
echo 'export PATH="$newPath"' >> $bashRc
echo 'export JAVA_HOME="$JAVA_HOME"' >> $bashRc
echo 'export HADOOP_HOME="$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
echo 'export SPARK_HOME="/usr/lib/spark"' >> $bashRc

--- END SCRIPT ---

```

2) Hadoop Classpath

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

The edge node creation script above will modify `hadoop-env.sh` so that it 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-amd64

# 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/security/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_OPTS="--Dsun.net.inetaddr.ttl=3600
export HADOOP_DATANODE_OPTS="--Dsun.net.inetaddr.ttl=3600
export HADOOP_NAMENODE_HEAPSIZE=1740
export HADOOP_DATANODE_HEAPSIZE=757
export HADOOP_JOB_HISTORYSERVER_HEAPSIZE=2396
# Appended by Edge Node Script
# client needed for Sqoop
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/lib/hadoop/client/*"
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/lib/hadoop-mapreduce/*"
```

3) .bashrc for Qlik Catalog the Catalog Linux user

The edge node creation script above will append the following to the Catalog Linux user .bashrc file in /home/<Catalog-linux-user-name>:

```
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:/usr/lib/spark/bin:/etc/spark/conf/bin:/usr/lib/hadoop-
mapreduce/bin:/usr/share/aws/aws-java-sdk/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) EMR Buffer Directories

Qlik Catalog needs the ability to create & write to /mnt/ & /mnt1 as part of the EMR buffer process. The edge node script above will create /mnt1, and it will grant the Catalog Linux user group write access to /mnt & /mnt1.

5) 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. The tez.tar.gz file should be stored in a non-ephemeral location which is accessible by the Qlik Catalog node. An AWS S3 bucket is the suggested location for tez.tar.gz in EMR deployments:

```
$ aws s3 cp /tmp/tez.tar.gz s3://qlik-catalog-data/
$ aws s3 ls s3://qlik-catalog-data
```

3. On the edge node, set core_env property

```
tez.lib.uris=s3://qlik-catalog-data/tez.tar.gz
```

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

RHEL or CentOS deployments: # sudo yum install expect -y

Ubuntu 20.04 or Ubuntu 22.04 deployments: # sudo apt install expect -y

6) 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>/qdc-hudf-<VERSION>.jar
```

Example: podium.hive.udf.jar.location=s3://qdc-data/qdc-hudf-<VERSION>.jar

Note: Make sure the qdc-hudf-<VERSION>.jar file exists in the bucket location. If it does not, you can copy it from \$QDC_HOME/webapps/qdc/WEB-INF/lib/

7) If Using Transient EMR Clusters: Modify Hadoop "Site-XML" Files

Edit the following files (sudo permission required). Replace all master node hostname references with the secondary IP address assigned to the master node:

- /usr/lib/hadoop/etc/hadoop/core-site.xml
- /usr/lib/hadoop/etc/hadoop/hdfs-site.xml
- /usr/lib/hadoop/etc/hadoop/mapred-site.xml
- /usr/lib/hadoop/etc/hadoop/yarn-site.xml
- /usr/lib/hive/conf/hive-site.xml

Example:

Replace EC2 Instance *Hostname* (highlighted)

```
<property>
  <!-- URI of NN. Fully qualified. No IP.-->
  <name>fs.defaultFS</name>
  <value>ip-172-31-0-113.ec2.internal:8020 </value>
</property>
```

With *Secondary IP Address* Assigned To Master Node

```
<property>
  <!-- URI of NN. Fully qualified. No IP.-->
  <name>fs.defaultFS</name>
  <value>172.31.0.90:8020 </value>
</property>
```

6.9 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 user if HDFS or Hive access is restricted. However, security configuration scenarios vary from company to company.

There is one 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/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, 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 = podium_ducks_hive
```

#Specifies the Ranger service name for hdfs

example:

```
ranger.service.hdfs = podium_ducks_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 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:

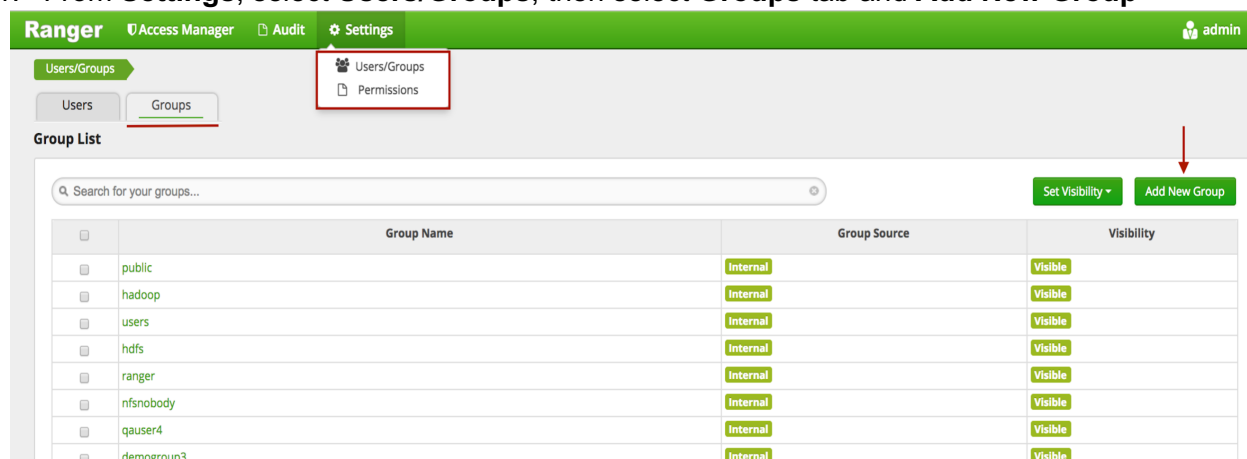
```
ranger.usersync ldap.username.caseconversion
```

```
ranger.usersync ldap.groupname.caseconversion
```

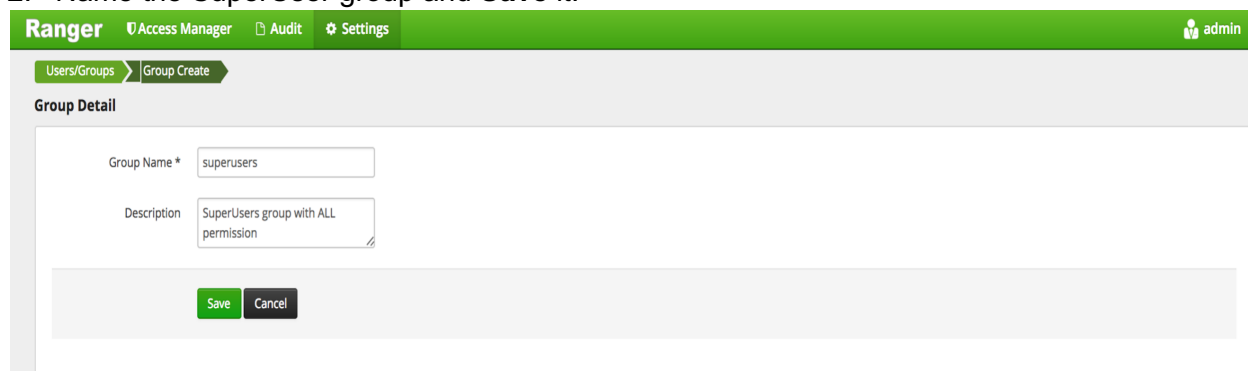
Setting Up SuperUser Group in Ranger for Qlik Catalog SuperUsers

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

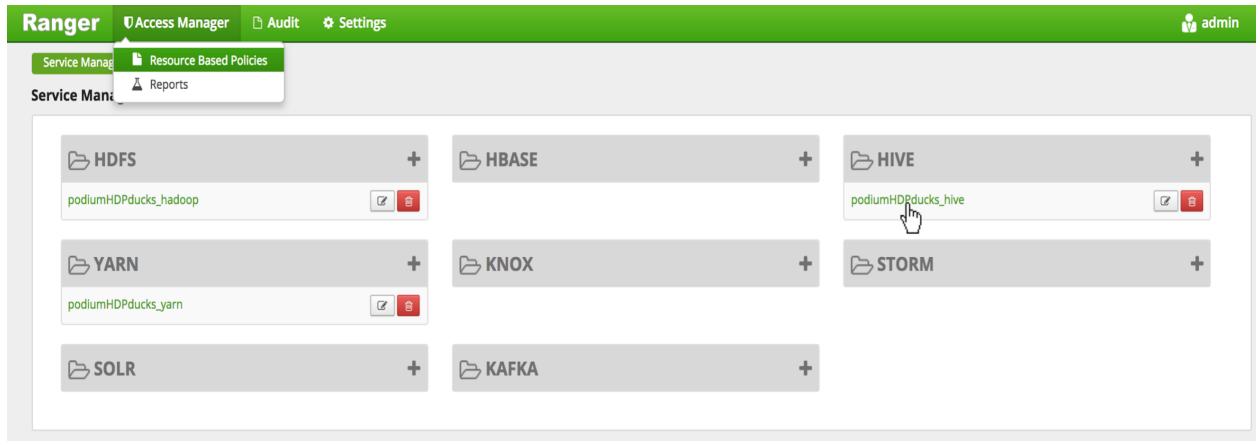
1. From **Settings**, select **Users/Groups**, then select **Groups** tab and **Add New Group**



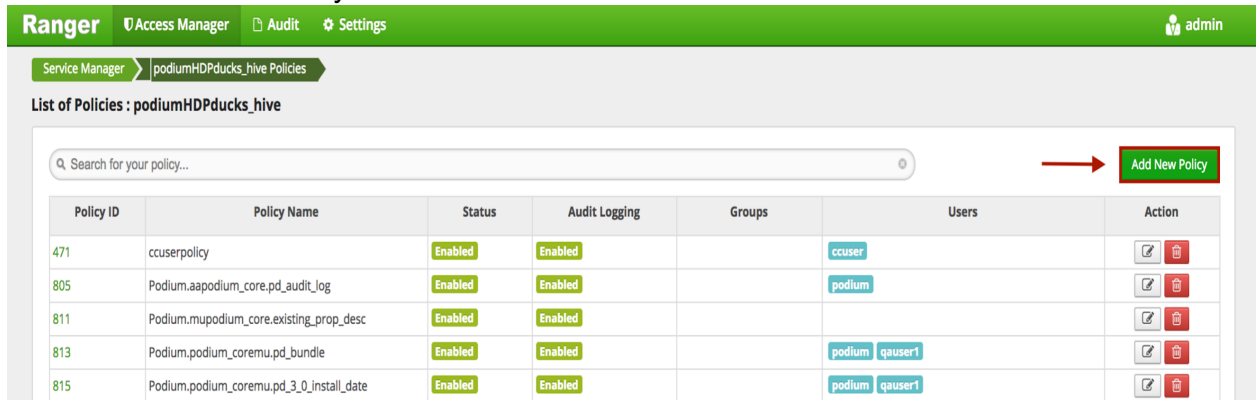
2. Name the SuperUser group and **Save** it.



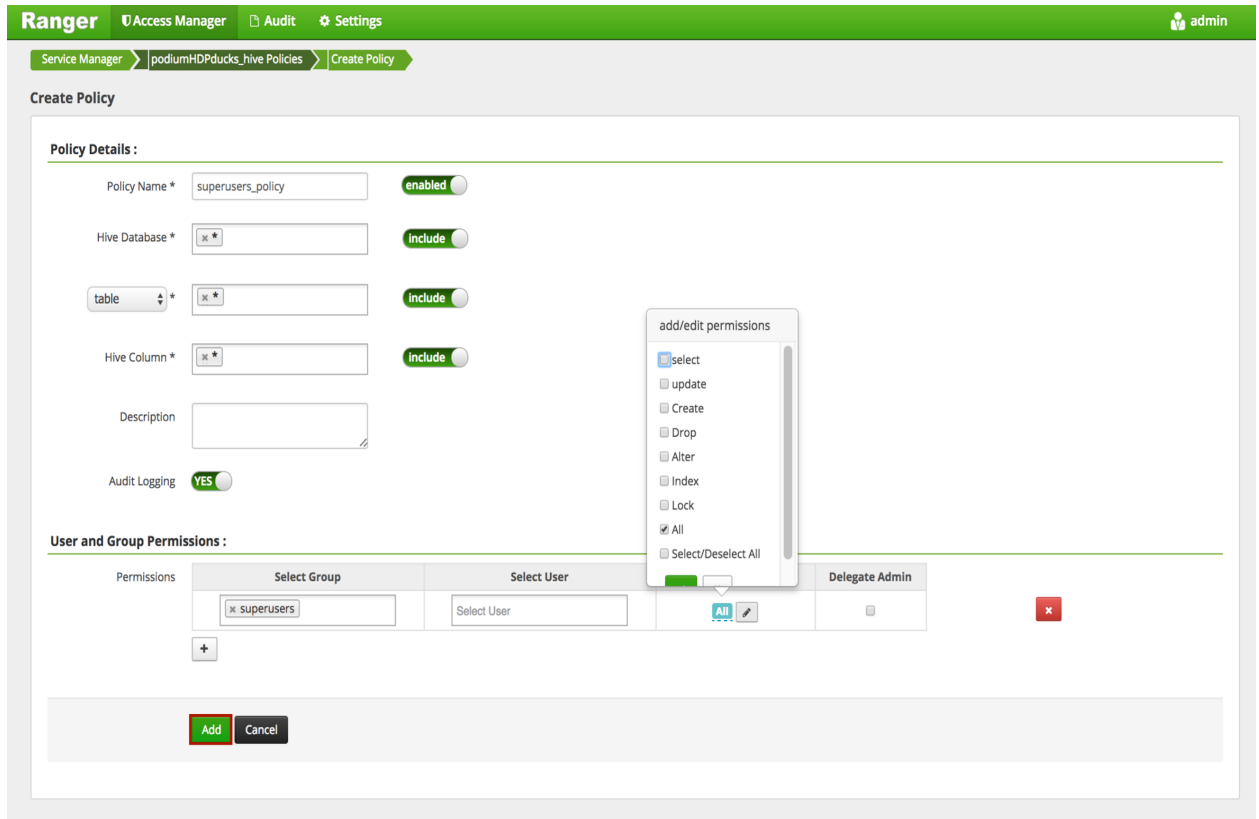
3. From **Access Manager**, select **Resource Based Policies** and open HIVE policies via the named hyperlink



4. Select Add New Policy



- 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.



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

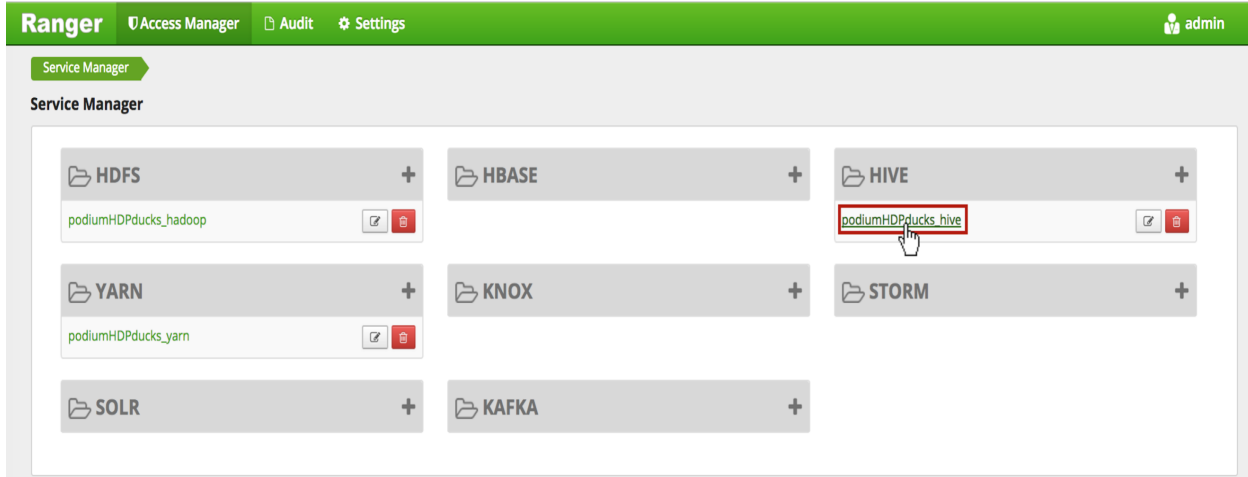
The **Service Manager** screen in Ranger is where users view the services they have enabled to sync with security policies on Qlik Catalog objects.

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

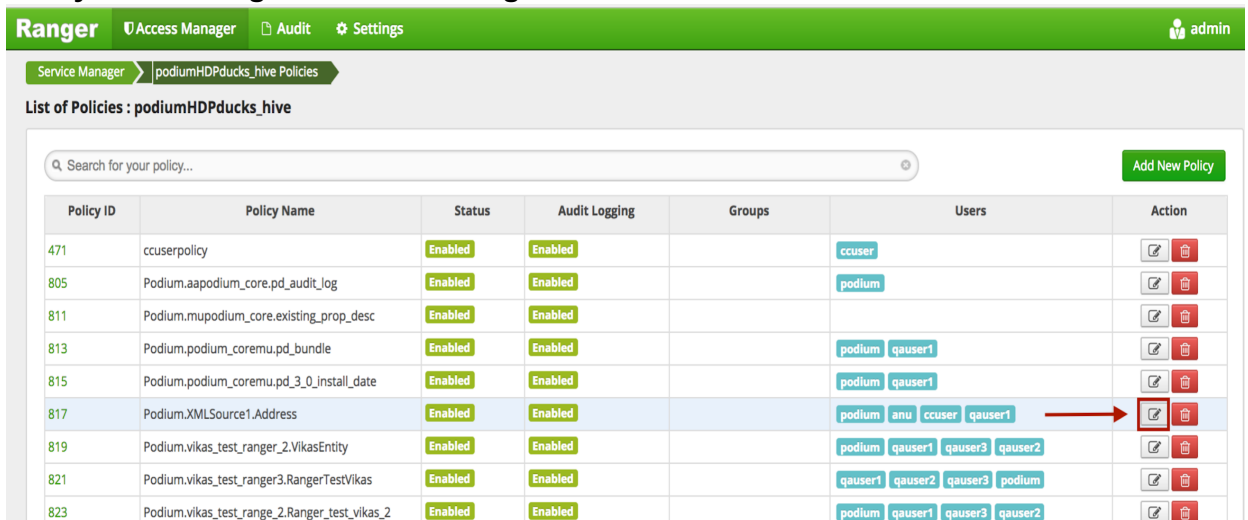
Select either HDFS or HIVE Service Name quicklink to access individual policies.



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



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_s
 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

The screenshot displays the Ranger Admin Console interface for configuring a Hive policy. The policy name is 'Podium.XMLSource1.Address' and is currently enabled. The Hive Database is set to 'XMLSource1'. A list of tables is provided, with 'Address' and 'Address_History' selected. The Hive Column is set to '*'. The 'Audit Logging' is set to 'YES'. Below the configuration, the 'User and Group Permissions' section is visible, showing a table with columns for 'Permissions', 'Select Group', 'Select User', 'Permissions', and 'Delegate Admin'. A dropdown menu is open for selecting a group, with 'public' selected.

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

The screenshot shows the Ranger Admin Console Policy Manager. The 'Service Manager' section is active, displaying a grid of service policies. The 'HDFS' service is expanded, showing a policy named 'podiumHDPducks_hadoop' which is highlighted with a red box. Other services like HBASE, HIVE, YARN, KNOX, SOLR, and KAFKA are also visible.

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>/<entity name>/* /tmp/user/<servicehost>/shipping/<sourcename>/<entity name>/* /tmp/user/<servicehost>/receiving/<sourcename>/<entity name>/* /tmp/user/<servicehost>/archive/<sourcename>/<entity name>/* 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

The screenshot shows the Ranger web interface for editing a policy. The breadcrumb path is: Service Manager > podumHDPoducks_hadoop Policies > Edit Policy. The user 'admin' is logged in.

Policy Details:

- Policy ID: 818
- Policy Name: Podum.XMLSource1.Address (enabled)
- Resource Path:
 - /tmp/halma/podumonjungle/loadingdock/XMLSource1/Address*
 - /tmp/halma/podumonjungle/shipping/XMLSource1/Address*
 - /tmp/halma/podumonjungle/receiving/XMLSource1/Address*
 - /tmp/halma/podumonjungle/archive/XMLSource1/Address*
- recursive:
- Description: Policy created from Podum
- Audit Logging:

User and Group Permissions:

Permissions	Select Group	Select User	Permissions	Delegate Admin
	Select Group	<input type="text" value="x atnu"/>	Execute Read Write	<input checked="" type="checkbox"/>
		<input type="text" value="x ccuser"/>	Execute Read Write	<input checked="" type="checkbox"/>
		<input type="text" value="x podum"/>	Execute Read Write	<input checked="" type="checkbox"/>
		<input type="text" value="x quser1"/>	Execute Read Write	<input checked="" type="checkbox"/>

A dropdown menu for 'Select Group' is open, showing a list of groups including 'quser4', 'demogroup3', 'plumar', 'djordan', 'jgrage', 'mpathyll', and 'Kannan'. A 'permissions' dialog is also open, showing 'Read', 'Write', 'Execute', and 'Select/Deselect All' options, with 'Execute' checked.

Audit Screen

Changes to entities and corresponding policies can be 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.

Ranger Access Manager Audit Settings admin

Access Admin Login Sessions Plugins

Search for your access logs...

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 10:03:32 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

6.10 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, Container Platform & Node.js Installation, has been completed.

The commands below assume the Qlik Catalog home directory is `/usr/local/qdc` and that the Catalog Linux 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/qlikcore` 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 Engine docker image:

RHEL 7/CentOS 7 Deployments using *Docker*

```
(qdc) $ docker load < engine.tar
```

RHEL 8/RHEL 9/Ubuntu 20.04/Ubuntu 22.04 Deployments using *Podman*

```
(qdc) $ podman load < engine.tar
```

4. To test, list images:

RHEL 7/CentOS 7 Deployments using *Docker*

```
(qdc) $ docker images
```

RHEL 8/RHEL 9.1/Ubuntu 20.04/Ubuntu 22.04 Deployments using *Podman*

```
(qdc) $ podman images
```

5. Register node modules:

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

6. To test, list modules:

```
(qdc) $ npm ls
```

7. Start the Qlik Engine container

```
(qdc) $ ./launch_qlikContainers.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.

6.11 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

6.12 Enabling NextGen XML

As of the May 2022 release, NextGen XML (along with Tomcat HTTPS) is automatically enabled for first-time installs that use CatalogPrereqs.sh.

NextGen XML Containers

- To use the 'next-generation' XML support in Qlik Catalog, the Catalog installer (CatalogInstaller.sh) installs two Qlik Sense containers: DCaaS (a connector lookup service) and a REST Connector (that parses XML files and converts them to flattened data).
- 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.

IMPORTANT:

- a) If the Qlik Catalog NextGen XML 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
- b) If deploying to RHEL 8, RHEL 9.1, Ubuntu 20.04, or Ubuntu 22.04 which use Podman to manage containers you must edit **/etc/systemd/system/nextgen-xml.service** after the copy (cp) step below:

- a. **REMOVE** line 13 completely:

Requires=docker.service

- b. EDIT line 14 and remove the following text: **docker.service**

```
(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.

▪ `core_env.properties`

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

Locate the following property and replace `$CATALOG_IPADDRESS` with the IP address of the Qlik Catalog server:

- `base.xml.callback.url=http://$CATALOG_IPADDRESS:8082/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`

Modify Tomcat Web Server Configuration

NOTE: As of the May 2022 release, if the optional prerequisites script `CatalogPrereqs.sh` is used, the NextGen XML feature and Tomcat are ***automatically configured to work together*** and this section can be ignored.

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-9.0.70/conf/server.xml
```

2) Add a dedicated Service and Connector

A dedicated service and connector are setup to listen on port 8082. The connector is configured to only allow localhost / local subnet connections (e.g., 127.0.0.1, 192.168.*.*, 172.16.*.*), thereby prohibiting outside access.

This new Service should come after the existing Service and is likely near the end of the document, near the closing `</Server>` tag. Here is the XML:

`<!-- The qdc-xmlstore directory is a temporary location for serving XML files to the DCaaS REST connector. By default, it is configured`

`to allow localhost and private IP access only. To allow access from an external host, add the host's IP address with a pipe '|' separator`

`to the end of the 'allow' value. For instance,`

`allow="127.\.d+\.d+\.d+|::1|0:0:0:0:0:0:1|...|11.111.3.241" -->`

```
<Service name="qdc-xmlstore">
```

```
  <Connector port="8082" protocol="org.apache.coyote.http11.Http11NioProtocol"
    connectionTimeout="20000" />
```

```
  <Engine name="Catalina" defaultHost="localhost">
```

```
    <Host name="localhost" appBase="qdc-xmlstore" unpackWARs="true" autoDeploy="true">
```

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

```

        <Valve className="org.apache.catalina.valves.RemoteAddrValve"
            allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:0:1|10\.\d+\.\d+\.\d+|172\.\d+\.\d+|172\.\d+\.\d+|172\.\d+\.\d+|172\.\d+\.\d+|172\.\d+\.\d+|172\.\d+\.\d+|172\.\d+\.\d+|192\.\d+\.\d+|192\.\d+\.\d+|168\.\d+\.\d+" />
    </Context>
</Host>
</Engine>
</Service>

```

Port 8082 can be changed -- if changed, make sure to update core_env property base.xml.callback.url. The above Context is standard for NextGen XML and prohibits outside access.

4) Save the changes to server.xml and **restart Tomcat**

5) Restart NextGen XML docker containers:

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

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

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

Note: If SELinux is enabled (or other security framework), you may need to open port 50060 to allow routing between the two containers:

```
(sudo) # sudo firewall-cmd --zone=public --permanent --add-port=50060/tcp
```

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

6.13 Migrating to or Upgrading Tomcat 9

Beginning with the May 2021 release, only Apache Tomcat 9 is supported. The installer will prohibit other versions. If using Tomcat 7, please first initiate a migration to Tomcat 9 before installing this release. Then, when installing, the upgrade option (-u) is **NOT** used.

These instructions may also be used to upgrade from an older version of Tomcat 9 to a newer version.

Step	Sample Commands
Shutdown and rename old Tomcat 7 or 9	<pre>cd /usr/local/qdc (or cd /usr/local/podium) ./apache-tomcat-<OLD_VERSION>/bin/shutdown.sh mv apache-tomcat-<OLD_VERSION> old-apache-tomcat</pre>

Step	Sample Commands
Download and expand Tomcat 9 - NOTE: adjust version 9.0.70 to use latest 9.0.x series	<pre>wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.70/bin/apache-tomcat-9.0.70.tar.gz tar -xf apache-tomcat-9.0.70.tar.gz rm apache-tomcat-9.0.70.tar.gz</pre>
Copy core_env.properties from old Tomcat 7 to new Tomcat 9	<pre>cp old-apache-tomcat/conf/core_env.properties apache-tomcat-9.0.70/conf/</pre>
If migrating from Tomcat 7: Extract server.xml from podium.zip and copy to new Tomcat	<pre>unzip -j podium-4.17-xxxxx.zip podium/config/tomcat9-server.xml -d . mv ./tomcat9-server.xml apache-tomcat-9.0.70/conf/server.xml</pre>
If upgrading Tomcat 9: Copy server.xml from old Tomcat 9 to new Tomcat 9	<pre>cp old-apache-tomcat/conf/server.xml apache-tomcat-9.0.70/conf/</pre> <p>If the old Tomcat 9 was configured for HTTPS, and the keystore (jks file) was stored in the old Tomcat directory, migrate it to the new Tomcat directory, and update conf/server.xml to reference it. Consider placing the keystore file in a non-Tomcat directory such as /usr/local/qdc/keystore.</p>
Configure CatalogInstaller.sh for Tomcat 9	<p>Whether using an existing CatalogInstaller.sh file from a previous install, or configuring one for the first time, ensure that it is updated to point to Tomcat 9:</p> <pre>TOMCAT_HOME=/usr/local/podium/apache-tomcat-9.0.70</pre>

Step	Sample Commands
<p>Finally, when the installer is run, <u>do NOT specify upgrade mode (-u)</u>, as some files should be created as if it were a first-time install.</p>	<pre>./CatalogInstaller.sh</pre>
<p>After installation, restore webapp web.xml, then restart Tomcat</p>	<pre>cp old-apache-tomcat/webapps/qdc/WEB-INF/web.xml apache-tomcat-9.0.70/webapps/qdc/WEB-INF/</pre>

At this point, Tomcat 9, if newly installed, will support only HTTP on port 8080.

Verify successful Qlik Catalog startup and basic functionality.

Additional configuration will be required to configure HTTPS on port 8443, apply security headers, etc. If Tomcat 7 used HTTPS, the keystore (jks file) containing the public-private keypair should be copied to Tomcat 9 and conf/server.xml updated. For more information, see **Error! Reference source not found.**

In addition, Tomcat 7 may have been configured as a service. It should be disabled. Tomcat 9 may be configured as a service to automatically start. For more information, see setting up Tomcat as a service.

6.14 Disabling Tomcat Redirect / SSL

As of the May 2022 release, when using the optional prerequisites script CatalogPrereqs.sh, Tomcat is automatically configured for HTTPS (including the generation of a self-signed certificate), and requests on port 8080 are automatically redirected to 8443.

In certain situations, these may need to be disabled (e.g., when the SSL connection is terminated at a load balancer). Perform either or both of the following and then restart Tomcat.

Disable Redirect from 8080 to 8443

Edit <tomcat home>/webapps/qdc/WEB-INF/web.xml

Remove the entire "<security-constraint>" element.

Disable HTTPS Port 8443

First, disable the redirect (see above).

Edit <tomcat home>/conf/server.xml

Remove the entire "<Connector>" element which begins "<Connector port="8443"

6.15 Upgrading PostgreSQL

These instructions may be used to upgrade to the latest version of Qlik Catalog custom PostgreSQL. Reminder: there is no requirement to upgrade PostgreSQL when upgrading Catalog. If a PostgreSQL upgrade is planned, please do so before upgrading Catalog.

Step	Sample Commands
Shutdown Tomcat	<pre>./bin/shutdown.sh</pre>
Backup databases. Default password for podium_md is "nvs2014!" For licenses is "licenses". For postgres is "postgres".	<pre>pg_dump -U podium_md --format=c --file=podium_md.sqlc podium_md pg_dump -U licenses --format=c --file=licenses.sqlc licenses</pre>
Find the Linux PostgreSQL service file name. Will be something like "qdc_pg.service" or "qdc_pg-11.2.service" -- USE THIS NAME FOR NEXT TWO STEPS	<pre>ls -la /etc/systemd/system/</pre>
Stop the service and verify PostgreSQL is not running	<pre>sudo systemctl stop qdc_pg-11.2.service ps -ef grep "postgres:"</pre>
Remove the service	<pre>sudo systemctl disable qdc_pg-11.2.service sudo rm /etc/systemd/system/qdc_pg-11.2.service</pre>

Step	Sample Commands
Prepare for new PostgreSQL -- adjust version as needed	<pre> sudo mkdir -p /usr/pgsql/qdc11-16 sudo chown -R postgres:postgres /usr/pgsql sudo mkdir -p /var/lib/pgsql/11-16/qdc_data sudo chown -R postgres:postgres /var/lib/pgsql </pre>
Install new PostgreSQL -- adjust versions as needed	<pre> cp podium-4.17-xxxxx_Linux_X64.zip /tmp/ cd /tmp unzip podium-4.17-xxxxx_Linux_X64.zip sudo su - postgres /tmp/podium/thirdParty/qdc_pg11-16_RHEL7-and-CentOS7.bsx OR /tmp/podium/thirdParty/qdc_pg11-16_RHEL8-and-CentOS8.bsx OR /tmp/podium/thirdParty/qdc_pg11-16_Ubuntu.bsx exit </pre>
Recreate symlinks	<pre> sudo rm /usr/bin/psql sudo ln -s /usr/pgsql/qdc11-16/bin/psql /usr/bin/psql psql --version sudo rm /usr/bin/pg_dump sudo ln -s /usr/pgsql/qdc11-16/bin/pg_dump /usr/bin/pg_dump pg_dump --version sudo rm /usr/bin/pg_restore sudo ln -s /usr/pgsql/qdc11-16/bin/pg_restore /usr/bin/pg_restore pg_restore --version </pre>

Step	Sample Commands
Recreate Linux service	<pre>sudo cp /tmp/podium/config/qdc_pg.service /etc/systemd/system/ sudo systemctl daemon-reload sudo systemctl enable qdc_pg.service</pre>
Ensure Tomcat service depends on new PostgreSQL service -- all references should be to "qdc_pg.service"	<pre>sudo vi /etc/systemd/system/tomcat.service</pre>
Start the service and verify PostgreSQL is running	<pre>sudo systemctl start qdc_pg.service ps -ef grep "postgres:"</pre>
Restore podium_md. Default password for podium_md is "nvs2014!" For licenses is "licenses". For postgres is "postgres".	<pre>INTERNAL_DB_PASSWORD=nvs2014! psql template1 -U postgres -c "create role podium_md with encrypted password '\$INTERNAL_DB_PASSWORD' createdb login;" psql template1 -U podium_md -c "create database podium_md;" pg_restore -U podium_md --dbname=podium_md podium_md.sqlc</pre>
Restore licenses	<pre>psql template1 -U postgres -c "create user licenses with password 'licenses'" psql template1 -U postgres -c "create database licenses owner licenses" pg_restore -U licenses --dbname=licenses licenses.sqlc</pre>
Proceed to upgrade Catalog	

6.16 Attaching a New Data Volume

The following documents how to use a symbolic link (aka symlink) to transparently migrate data to a new, larger data volume.

1. Stop Tomcat

2. Create a new data directory on the mounted volume (here assumed to be /mountedvol). Do this as root (or use sudo):

```
# cd /mountedvol
# mkdir data
# chown qdc:qdc data
# chmod 775 data
```

3. Copy and backup the original data (backup can later be deleted). Do this as the Catalog Linux user (e.g., qdc):

```
$ cd /usr/local/qdc/data
$ cp -R * /mountedvol/data/
$ cd /usr/local/qdc
$ mv data data-BACKUP
```

4. Create the symlink. Do this as root (or use sudo):

```
# ln -s /mountedvol/data /usr/local/qdc/data
```

The core_env property should be left as "podium.base=/usr/local/qdc/data" and NOT changed to "podium.base=/mountedvol/data".

5. Restart Tomcat

6.17 SAML and PODIUM Authentication Using Single Tomcat Instance

This solution should apply to any version of Catalog from May 2021 onwards.

These instructions assume the original "qdc" webapp has been configured for SAML authentication. A second "qdcadmin" webapp will use PODIUM (username + password) authentication – it may be used for administration, API access, QEM API access, etc.

(1) Shutdown Tomcat.

(2) Make complete copies of the qdc.war file and qdc directory – here is the result:

```
$ pwd
/usr/local/qdc/apache-tomcat-9.0.70

$ ls -la webapps/
total 880464
drwxr-x---  7 qdc  qdc          224 Sep 28 16:33 .
drwxr-xr-x 17 qdc  qdc          544 Apr 20 15:16 ..
drwxr-xr-x  5 qdc  qdc          160 Sep 28 16:33 qdc
-rw-r--r--  1 qdc  qdc 225386798 Sep 28 16:33 qdc.war
drwxr-xr-x  5 qdc  qdc          160 Sep 28 16:33 qdcadmin
-rw-r--r--  1 qdc  qdc 225386798 Sep 28 16:33 qdcadmin.war
```

(3) Create file webapps/qdcadmin/WEB-INF/classes/core_env.local.properties – it should contain:


```
authentication.mode=PODIUM
logout.url=/logged-out
external.job.runner.enable=false
external.job.runner.spawn=false
```

NOTE: if you plan on running load, prepare or publish jobs in the qdcadmin webapp, do not include the two "external.job.runner" properties.

NOTE: until the first release after August 2022 SR1 (likely November 2022), Job Runner will run in the qdcadmin webapp despite its disablement above (this is ok) – a code fix is required to address this.

(4) Restart Tomcat.

(5) The two webapps are accessed using:

- SAML – `https://<HOSTNAME>:8443/qdc`
- PODIUM – `https://<HOSTNAME>:8443/qdcadmin`

Tomcat's log file, `logs/catalina.out`, is shared by the two webapps.

Upgrades

The following must be repeated each time Catalog is upgraded.

Before upgrade:

- Shutdown Tomcat
- Make a backup of `webapps/qdcadmin/WEB-INF/classes/core_env.local.properties`
- Execute: `rm -fr webapps/qdcadmin*`

After upgrade, repeat the initial process.

6.18 RHEL 9, SELinux and podman

If Catalog is installed on Red Hat Enterprise Linux (RHEL) 9.1+ (and some versions of 8) and SELinux is enabled (the default on RHEL 9), configuration changes must be made to successfully use the podman containers included by Catalog to import (load) and export (publish) QVDs, and to load XML data. NOTE: the alternative to the below procedure is to disable SELinux.

For QVD import and export, please do the following, which will create a dedicated "temp" directory and configure SELinux and Catalog to use it. This temporary directory is where the engine container writes its output files. It must be in the root directory as many of the other standard locations (e.g., `/tmp` and `/usr/local`) have been blacklisted.

As root:

```
# mkdir /tmp-catalog
# chown qdc:qdc /tmp-catalog
# semanage fcontext -a -s system_u -t container_file_t /tmp-catalog
# restorecon -F -v /tmp-catalog
```

As the Catalog service account user (e.g., `qdc`), edit `core_env.properties` and find and set the following property:

```
qvd.conversion.temp.directory=/tmp-catalog
```

Also, as the Catalog service account, edit `/usr/local/qdc/qlikcore/pod_qlikContainers.yml` and uncomment the following two blocks:

```
# Uncomment the following volumeMount.
- mountPath: /tmp-catalog
  name: tmp-catalog
```

and:

```
# Uncomment the following volume.
- hostPath:
  path: /tmp-catalog
  type: Directory
  name: tmp-catalog
```

To ensure the podman containers are started on host reboot, the container start must be made dependent on the user service for the service account user (e.g., qdc).

As root, edit `/etc/systemd/system/qlikContainers.service` and follow the comment:

```
#After=syslog.target network.target qdc_pg.service
# A dependency on the service account user service is required for RHEL 9 with SELinux.
# Please see the November 2022 SR1 release notes or the February 2023 installation guide.
# Uncomment the below Requires and After; comment off the above Requires and After.
Requires=user@1000.service qdc_pg.service
After=user@1000.service qdc_pg.service
```

You can ensure user "qdc" has the correct ID by running "id -u qdc". If the correct numeric value is not used in the service file, update the values above.

Then, uncomment similar lines in file `/etc/systemd/system/nextgen-xml.service`.

When all changes have been made, run "systemctl reboot" to restart the system.



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.

qlik.com

© 2022 QlikTech International AB. All rights reserved. Qlik®, Qlik Sense®, QlikView®, QlikTech®, Qlik Cloud®, Qlik DataMarket®, Qlik Analytics Platform®, Qlik NPrinting®, Qlik Connectors®, Qlik GeoAnalytics®, Qlik Core®, Associative Difference®, Lead with Data™, Qlik Catalog®, Qlik Associative Big Data Index™, Qlik Insight Bot™, Qlik World™ and the QlikTech logos® are trademarks of QlikTech International AB that, where indicated by an "®", have been registered in one or more countries. Attunity® and the Attunity logo™ are trademarks of Qlik Analytics (ISR) Ltd. Other marks and logos mentioned herein are trademarks or registered trademarks of their respective owners.