Single Node Installation Guide

Qlik Data Catalyst®

December 2019 SR1

TABLE OF CONTENTS

1.0 Qlik Data Catalyst Single Node Overview + System Requirements 2

1.1 Single Node Configuration 2

1.2 QDC Software Configuration Requirements 2

1.3 Data Catalyst Browser Support 3

2.0 Installation Prerequisites 4

2.1 Java JDK 8 Installation 4

2.2 Create Service Account and QDC Directory 4

2.3 Tomcat Installation 5

2.4 PostgreSQL Installation 7

2.5 Qlik Sense Integration 8

3.0 QDC Software Installation 9

3.1 First-time Installation Using the Installer 9

3.2 Upgrade of QDC June 2019 and Later 11

4.0 Qlik Data Catalyst Software Installation Reference 13

4.1 Enabling SAML using Okta 13

4.2 Tomcat SSL Configuration 14

4.3 Configuring Impersonation 18

4.4 Implementation Setup for Security Policy Sync with Impersonation 19

4.5 Configuration Recommendations for Qlik Data Catalyst Installations 20

1.0 Qlik Data Catalyst Single Node Overview + System Requirements

This document describes how to install the "Single Node" deployment option for Qlik Data Catalyst. Once the installation is complete, upon first login to the QDC application, you will be prompted for a license. A license for either "**Qlik Data Catalyst for QVDs**" or "**Qlik Data Catalyst**" must be entered. **Qlik Data Catalyst for QVDs** is a streamlined offering of Qlik Data Catalyst that is specially focused on making QVDs easier to discover, understand and consume.

1.1 Single Node Configuration

Single Node Recommendations

- Recommended Minimum Production Configuration
  - 12 Cores
  - 128GB RAM
  - System Drive 250GB
  - Data Drive 3x expected data
  - Ethernet 10GB
  - Virtual Machine or bare metal

- Minimum POC/Dev Configuration
  - 8 Cores
  - 32GB RAM
  - System partition 250GB
  - Data partition 3x expected data
  - Ethernet 10GB
  - Virtual Machine or bare metal

- Minimum Supported Screen Resolution: 1366x768px

1.2 QDC Software Configuration Requirements

- OS: QDC is compatible with either RHEL or Centos Linux 6 or 7 (en_US locales only); however, RHEL 7 or Centos 7 are the only distributions that will support integration with Qlik Sense (due to a limitation of the Docker Engine).
- Java: Oracle JDK 1.8 or OpenJDK 1.8 (see Section 2)
  - 8u222 is recommended version
  - 8u162 began to include the Java Cryptography Extension (JCE); if for some reason an older version is used, the JCE must be separately downloaded and enabled
- QDC PostgreSQL 11.2 (see Section 2)
  - Custom build of PostgreSQL 11.2 included with QDC
- Apache Tomcat 7.0.94 (see Section 2)
- All JDBC drivers needed for database connectivity
- Ensure port 8080 or 443 (http or https) is open from user desktops to the Qlik Data Catalyst node

1.3 Data Catalyst Browser Support

Officially Supported:

- Google Chrome - 45.0+
- IE11

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

2.0 Installation Prerequisites

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

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

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

Important: During the prerequisite setup process, several items are needed from the QDC software distribution: a file named podium-4.3-13932.zip. The instructions below assume the QDC software distribution has been unzipped to /tmp:

1. Install unzip (if not present)

   (sudo) # sudo yum install -y unzip

2. Expand the QDC software distribution to /tmp

   (sudo) # unzip <replace-path>/podium-4.3-13932.zip -d /tmp/

2.1 Java JDK 8 Installation

1. Check if JDK 1.8 exists. If it exists and is the correct version (8u191 or later) skip this step.

   (sudo) # java -version
   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)

2. Either directly download and install the JDK from Oracle (if you are a commercial licensee), or use yum to install OpenJDK

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

2.1.1 Mandatory JCE Upgrade for JDK Before 8u162

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

2.2 Create Service Account and QDC Directory

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

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

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

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

   (sudo) # sudo passwd qdc

3. Create a directory for all QDC artifacts, including Tomcat

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

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

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

2.3 Tomcat Installation

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

   (sudo) # sudo yum install -y wget

2. Become the service account user

   (sudo) # sudo su - qdc

3. Download Apache Tomcat 7.0.94

   (qdc) $ cd /usr/local/qdc
   (qdc) $ wget http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.94/bin/apache-tomcat-7.0.94.tar.gz

4. Extract the Tomcat file

   (qdc) **$** tar -xvf apache-tomcat-7.0.94.tar.gz

5. The resulting directory, for example “/usr/local/qdc/apache-tomcat-7.0.94”, is known as the Tomcat home directory. When configuring the QDCinstaller.properties file in the next section, please set TOMCAT_HOME to this value.

6. Overwrite <tomcat home>/conf/server.xml with the version expanded from the QDC zip

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

   Or, as an alternative to the above, manually edit the original:

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

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

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

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

7. Port 8080 needs to be open on the QDC node firewall

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

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

   Possible edit: if the Tomcat home directory is not /usr/local/qdc/apache-tomcat-7.0.94 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 QDC software (a.k.a., podium zip) has been expanded to /tmp -- see Section 2.0. You will not be able to start the service until QDC PostgreSQL is later installed, as a dependency exists.

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

9. Start Tomcat manually

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

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

    http://<QDC-Node-IP-Address-OR-Hostname>:8080

11. Tomcat can be manually stopped at any time

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

12. The Tomcat log can be monitored

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

2.4 PostgreSQL Installation

QDC is only certified on Qlik Data Catalyst PostgreSQL 11.2. To ensure this version is used, the QDC PostgreSQL installer has been included in the QDC 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 QDC zip file has already been extracted to /tmp -- see Section 2.0.

1. Create a “postgres” user and group

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

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

   (sudo) # sudo usermod -aG qdc postgres

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

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

4. Run the custom QDC PostgreSQL installer, after becoming the “postgres” user

   (sudo) # sudo su - postgres

   (postgres) $ /tmp/podium/thirdParty/qdc-pg-installer.bsx

5. Create a symlink to the psql and pg_dump executables

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

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

   Possible edit: if the directories in step 3 were altered, or the user/group is not “postgres”, the file /etc/systemd/system/qdc_pg-11.2.service must be edited after the copy (cp) step below.

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

   (sudo) # sudo cp /tmp/podium/config/qdc_pg-11.2.service /etc/systemd/system/
   (sudo) # sudo systemctl daemon-reload
   (sudo) # sudo systemctl enable qdc_pg-11.2.service
   (sudo) # sudo systemctl start qdc_pg-11.2.service

2.5 Qlik Sense Integration

QDC integration with Qlik Sense is dependent upon the Docker Engine (https://www.docker.com/products/container-runtime). If QDC will be used to catalog Qlik Sense QVDs, a Docker installation is required.
As indicated earlier in this guide, either RHEL 7 or CentOS 7 is required for Docker.

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

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

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

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

1. Install Node.js (which includes npm)

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

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

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

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

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

   (sudo) # sudo usermod -aG docker qdc

4. Install Docker Compose

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

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

   (sudo) # sudo su - qdc

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

3.0 QDC Software Installation

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

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

Password Encryption

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

```
$ unzip -j podium-4.3-13932.zip podium/lib/podium_localhadoop.jar -d .
$ java -cp podium_localhadoop.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 QDC service account user -- "sudo su - qdc" may be used to become this user
- "(postgres)" means the command should be run as the PostgreSQL superuser -- "sudo su - postgres" may be used to become this user

3.1 First-time Installation Using the Installer

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

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

   (sudo) # sudo su - qdc

2. Unzip QDCinstaller.zip into a working directory

3. Copy the QDC software zip file (e.g., podium-4.3-13932.zip) into the working directory

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

   ```
   podium-4.3-13932.zip  QDCinstaller.properties  QDCinstaller.sh  QDCinstaller.txt
   ```

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

- JAVA_HOME
- PODIUM_BASE

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

   (qdc) $ ./QDCinstaller.sh

7. 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

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

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

9. 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.

10. If QDC will be integrated with Qlik Sense in order to catalog QVDs, and Node and Docker were installed in the prerequisites section, the Qlik Core Docker container must be setup as a service to automatically start when Linux restarts.

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

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

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

Congratulations! The base QDC software installation is now complete.

The installer created two databases: the QDC metadata database (podium_md) and the distribution database (podium_dist). The default users (roles) for these databases are podium_md and podium_dist, both with a default password of “nvs2014!”.

REMINDER: Reboot Procedure

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

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

- PostgreSQL: (sudo) # sudo systemctl restart qdc_pg-11.2.service
  - test: (sudo) # psql
- Docker: (sudo) # sudo systemctl restart docker.service
  - test: (sudo) # sudo docker ps

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

- Qlik Core Docker Container: (sudo) # sudo systemctl restart qlikcore.service
  - test: (sudo) # sudo docker inspect -f '{{.State.Running}}' qlikintegration_qix-engine_1
- Tomcat: (sudo) # sudo systemctl restart tomcat.service

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

- Qlik Core Docker Container: (qdc) # cd /usr/local/qdc/qlikIntegration && ./launch_qlikcore.sh
  - test: (qdc) # docker inspect -f '{{.State.Running}}' qlikintegration_qix-engine_1
- Tomcat: (qdc) # /usr/local/qdc/apache-tomcat-7.0.94/bin/startup.sh

3.2 Upgrade of QDC June 2019 and Later

If you are upgrading from QDC June 2019 or later and you want **long entity** and **source name** support, you must upgrade to the custom build of PostgreSQL 11.2 (included in the QDC software download); otherwise, you can continue using the version of PostgreSQL that was installed with your current release.

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 QDC June 2019 and later:

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

   (sudo) # sudo su - qdc

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

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

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

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

   (sudo) # pg_dump -U postgres --format=c --file=<backupFileName1> podium_md
   (sudo) # pg_dump -U postgres --format=c --file=<backupFileName2> podium_dist

5. Unzip QDCinstaller.zip into a working directory

6. Copy the QDC software ZIP file (e.g., podium-4.3-13932.zip) into the working directory

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

   podium-4.3-13932.zip QDCinstaller.properties QDCinstaller.sh QDCinstaller.txt

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

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

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

   (qdc) # ./QDCinstaller.sh -u

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

4.0 Qlik Data Catalyst Software Installation Reference

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

The **core_env.properties** file is used to indicate to the Qlik Data Catalyst application 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 QDC 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.

4.1 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 podium is the Service Provider (SP).

1. Log in to your Okta organization as a user with administrative privileges. There is a sample organization created with the following information
   URL: https://dev-519244.oktapreview.com/
   Username: qdc.user@gmail.com
   Password: Password
   You can also create a free Okta Developer Edition organization with your own email here: https://www.okta.com/developer/signup/.
2. Click on the blue Admin button on the top right corner.
3. Click on the Add Applications shortcut in the right panel.
4. Click on the green Create New **App** button.
5. In the dialog that opens, select the **SAML 2.0** option, then click the green **Create** button
6. In Step 1 General Settings, enter the application name (e.g. HostName SAML Application) in App name field, then click the green Next button.
7. In Step 2 Configure SAML Paste the URLs like those given below into the Single Sign On URL field: (the sample urls are for internal podium install)

   http://hostname-qdc.qlik.com:8080/qdc/saml/SSO

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

   http://hostname-qdc.qlik.com:8080/qdc/saml/metadata
8. In Step 3 Feedback click the checkbox next to the text This is an internal application that we created then click the green **Finish** button.
9. You will now see the Sign On section of your newly created Spring Security SAML application
10. Copy the Identity Provider metadata link and paste it in the core_env.properties saml.metadata.provider.

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

11. You can also add additional users in the **People** section. All these users will need to be added to 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 QDC application.

There are now multiple ways to log into Qlike Data Catalyst using Okta SAML Provider

1. Log in to http://hostname-qdc.corp.qlik.com:8080/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! The user with the same username must exist in podium as well.

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

3. If you have already authenticated using Okta using some other application, you can directly send a POST request to http://hostname-qdc.qlik.com:8080/qdc/saml/SSO with the SAMLResponse parameter containing the value of the response from IDP.

4.2 Tomcat SSL Configuration

4.2.1 Local

Configure Tomcat to Support SSL (HTTPS) as Local Web Server

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

   Example:
   [root@hostname bin]# keytool -genkey -alias qdc -keyalg RSA –keystore /home/hostname/qdckeystore

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

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

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

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

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

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

   Add following code (after the comments searched above)

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

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

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

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

Where 8080 is the http port and 8443 is the https port:

1. Start the server

2. Browse https://localhost:8443

   If the application doesn't redirect to https, add the following code to web.xml (before web-app tag ends):
   <security-constraint>
   <web-resource-collection>
   <web-resource-name>securedapp</web-resource-name>
   <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <user-data-constraint>
   <transport-guarantee>CONFIDENTIAL</transport-guarantee>
   </user-data-constraint>
   </security-constraint>

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

   RESOLUTION:

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

<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true
maxThreads="150" scheme="https" secure="true
clientAuth="false" sslProtocol="TLS"
keystoreFile="path/to/keystore"
keystoreType="JKS"
keystorePass="pass4keystore"
keyPass="pass4key"
ciphers="TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_RC4_128_SHA,

TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA256,
TLS_RSA_WITH_AES_256_CBC_SHA,SSL_RSA_WITH_RC4_128_SHA"/>

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

4.2.2 Distributed

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

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

   Example:
   [root@hostname bin]# keytool -genkey -alias qdc -keyalg RSA –keystore /home/hostname/qdckeystore

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

(RETURN if same as keystore password):
Re-enter new password:

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

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

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

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

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

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

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

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

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

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

Important:

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

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

4.3 Configuring Impersonation

To enable impersonation on a QDC Single Node system:

1. Set the following properties in the ***core_env.properties*** file. The file is found at <tomcat home>/conf/core_env.properties.
   - enable.impersonation=true
   - hive.admin.user=podium_dist (Any user with super user rights)
   - authorization.mode=SINGLE_SERVER
   - authorization.component=ALL

2. Grant the podium_dist role superuser permission within Postgres
   - Run the following command:

     ```
     psql postgres -U postgres -c "ALTER ROLE podium_dist WITH SUPERUSER;"
     ```

4.4 Implementation Setup for Security Policy Sync with Impersonation

Prerequisites:
Same User name must exist on the machine as created on Qlik Data Catalyst platform
Same Group name must exist on the machine as created on Qlik Data Catalyst platform

core_env.properties setup:
[required to enable impersonation on single server] enable.impersonation=true
[required to enable impersonation on single server] hive.admin.user=podium_dist (Any user with super user rights)
authorization.mode=SINGLE_SERVER
authorization.component=ALL (Set authorization.component 'ALL' to secure both Distribution and File System. Use 'HIVE' to create.distribution.on.sync=true

Distribution policy: Roles for every group and user are created in Qlik Data Catalyst and then the group role is assigned to the user role. For example, if QDC has a group 'test_group' and user 'test_user' then two roles will be created in distribution i.e. 'test_user' which is a user role and 'qdc_grp_test_group' for group role and 'test_user' role would be associated with 'qdc_grp_qdc' role. Schema and tables in the schema are added into the group role and group role is assigned to user role so ultimately the user gets access to all tables defined in group role.

File system policy: Access Control Lists (ACL) are generated for an entity and it's parent source and then the ACL generated for source is applied on source path and the ACL generated for entity is applied on entity directory and sub-directories and files in these directories. ACL looks like: user::rwx,group::rwx,other::---,group:test_group:rwx

4.5 Configuration Recommendations for Qlik Data Catalyst Installations

Single Node Settings for QVDs	
Hardware	
Memory	128G
CPU	32
core-env.properties file	
External.job.runner.spawn	true
external.job.runner.spawn.count	8
external.job.runner.enable	true
external.job.runner.process.rotation.period.hours	24
max.pool.size	96
hadoop.job.poolsize*	5
use.single.receiving.mapper	true

Single Node Settings for non-QVDs	
Hardware	
Memory	128G
CPU	32
core-env.properties file	
External.job.runner.spawn**	true
External.job.runner.spawn.count**	8
external.job.runner.enable	true
external.job.runner.process.rotation.period.hours	24
max.pool.size	96
hadoop.job.poolsize*	32
use.single.receiving.mapper	true

*If ingesting both QVDs and non-QVDs hadoop.job.poolsize must be set to 5
**These settings are mandatory for single-node which only runs in local mode; they are required for multi-node when using Spark for Prepare (due to user context being shared).

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