# Red Hat AMQ 7.1

## Deploying AMQ Broker on OpenShift Container Platform

For Use with AMQ Broker 7.1

Last Updated: 2018-06-14

# Red Hat AMQ 7.1 Deploying AMQ Broker on OpenShift Container Platform

For Use with AMQ Broker 7.1

## Legal Notice

## Abstract

Learn how to install and deploy AMQ Broker on OpenShift Container Platform.

# Table of Contents

# PREFACE

**IMPORTANT**

AMQ Broker 7.1 on OpenShift Container Platform 3.9 is a Technology Preview release only. Technology Preview releases are not supported with Red Hat production service-level agreements (SLAs) and might not be functionally complete, and Red Hat does not recommend using them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information see Red Hat Technology Preview Features Support Scope.

# CHAPTER 1. INTRODUCTION

Red Hat AMQ Broker 7.1 is available as a containerized image that is designed for use with OpenShift Container Platform 3.9. Developers can quickly deploy an AMQ message broker in a hybrid cloud environment.

AMQ Broker, based on Apache ActiveMQ Artemis, is a full-featured, message-oriented middleware broker. It offers specialized queueing behaviors, message persistence, and manageability. Core messaging is provided by Apache ActiveMQ with support for different messaging styles such as publish-subscribe, point-to-point, and store-and-forward. It supports multiple protocols and client languages, so you can use many of your application assets. Lastly, AMQ Broker is supported to work with Red Hat JBoss Enterprise Application Platform.

## 1.1. TECHNOLOGY PREVIEW RELEASE LIMITATIONS FOR AMQ BROKER ON OPENSHIFT CONTAINER PLATFORM

The Technology Preview release of AMQ Broker on OpenShift Container Platform has the following limitations:

- Scaling down and message migration is not available.
- High Availability (HA): Shared store is supported only if the containers have access to a distributed file system.
- External clients do not support HA.
  - Because external Core Protocol JMS clients do not support HA or any type of failover, the connection factories must be configured with **`useTopologyForLoadBalancing=false`**.
  - AMQP clients do not support failover lists.
- The AMQ Core Protocol JMS Client cannot be configured to use an haproxy with TLS passthrough, which requires SNI. This issue occurs because the SNI information is not passed by the Core Client, even when sniHost is set correctly on the URI. To work around this issue, use another method to enable external client access to AMQ Broker on OpenShift Container Platform, such as NodePort.

Scaling up and persistent storage partitioning is available for the Technology Preview release but is not currently documented.

## 1.2. VERSION COMPATIBILITY AND SUPPORT

For details about OpenShift Container Platform 3.9 image version compatibility, see the OpenShift and Atomic Platform Tested Integrations page.

# CHAPTER 2. INSTALLING AND DEPLOYING AMQ BROKER ON OPENSHIFT CONTAINER PLATFORM

## 2.1. INSTALLING THE AMQ BROKER ON OPENSHIFT CONTAINER PLATFORM IMAGE STREAMS AND APPLICATION TEMPLATES

The AMQ Broker on OpenShift Container Platform image is not available in the service catalog, so you must manually install the image.

### Procedure

1. Log in to OpenShift as a cluster administrator (or as a user that has project administrator access to the global openshift project), for example:

```
$ oc login -u system:admin
```

2. At the command line, run the following commands to update the core AMQ Broker on OpenShift Container Platform image stream in the openshift project:

```
$ oc create -n openshift -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-amq-
7-broker-openshift-image/71-1.0.TP/amq-broker-7-image-streams.yaml

$ oc replace -n openshift --force -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-amq-
7-broker-openshift-image/71-1.0.TP/amq-broker-7-image-streams.yaml

$ oc -n openshift import-image amq-broker-71-openshift:1.0
```

**NOTE**

You might receive error messages indicating some image streams already exist after invoking the **`create`** command. A single command to create and replace is not available.

3. Run the following command to update the AMQ Broker templates:

```
$ for template in amq-broker-71-basic.yaml \
amq-broker-71-ssl.yaml \
amq-broker-71-persistence.yaml \
amq-broker-71-persistence-ssl.yaml \
amq-broker-71-statefulset-clustered.yaml;
 do
 oc create -n openshift -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-amq-
7-broker-openshift-image/71-1.0.TP/templates/${template}

 oc replace -n openshift -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-amq-
7-broker-openshift-image/71-1.0.TP/templates/${template}
 done
```

**NOTE**

You might receive "already exists" error messages after invoking the `create` command.

## 2.2. DEPLOYING THE AMQ BROKER ON OPENSHIFT CONTAINER PLATFORM IMAGE

The AMQ Broker on OpenShift Container Platform image requires a service account for deployments. Service accounts are API objects that exist within each project. Three service accounts are created automatically in every project:

- `builder`: This service account is used by build pods. It contains the `system:image-builder` role from which you can push images to any image stream in the project using the internal Docker registry.
- `deployer`: This service account is used by deployment pods. It contains the `system:deployer` role from which you can view and modify replication controllers and pods in the project.
- `default`: This service account is used to run all other pods unless you specify a different service account.

Service accounts can be created or deleted like any other API object. For multiple-node deployments, the service account must have the `view` role enabled so that it can discover and manage the various pods in the cluster. In addition, you must configure SSL to enable connections to AMQ Broker from outside of the OpenShift Container Platform instance; for more information about how to configure SSL see Configuring SSL. The type of discovery protocol that is used for discovering of AMQ Broker mesh endpoints is JGroups with OpenShift.dns ping protocol.

### Procedure

1. Add the `view` role to the service account:

```
$ oc policy add-role-to-user view -z default
```

## 2.3. CONFIGURING SSL

For a minimal SSL configuration to allow connections outside of OpenShift Container Platform, AMQ Broker requires a broker keystore, a client keystore, and a client truststore that includes the broker keystore. The broker keystore is also used to create a secret for the AMQ Broker on OpenShift Container Platform image, which is added to the service account.

The following example commands use Java KeyTool, a package included with the Java Development Kit, to generate the necessary certificates and stores.

### Procedure

1. Generate a self-signed certificate for the broker keystore:

   ```
   $ keytool -genkey -alias broker -keyalg RSA -keystore broker.ks
   ```

2. Export the certificate so that it can be shared with clients:

```
$ keytool -export -alias broker -keystore broker.ks -file
broker_cert
```

3. Generate a self-signed certificate for the client keystore:

```
$ keytool -genkey -alias client -keyalg RSA -keystore client.ks
```

4. Create a client truststore that imports the broker certificate:

```
$ keytool -import -alias broker -keystore client.ts -file
broker_cert
```

5. Export the client’s certificate from the keystore:

```
$ keytool -export -alias client -keystore client.ks -file
client_cert
```

6. Import the client’s exported certificate into a broker SERVER truststore:

```
$ keytool -import -alias client -keystore broker.ts -file
client_cert
```

## 2.4. GENERATING THE AMQ BROKER SECRET

The broker keystore can be used to generate a secret for the namespace, which is also added to the service account so that the applications can be authorized.

### Procedure

1. In a command line, run the following command:

```
$ oc create secret generic <secret-name> --from-file=<broker-
keystore> --from-file=<broker-truststore>
$ oc secrets add sa/<service-account-name> secret/<secret-name>
```

## 2.5. CREATING AN SSL ROUTE

After the AMQ Broker on OpenShift Container Platform image has been deployed, an SSL route needs to be created for the AMQ Broker transport protocol port to allow connections to AMQ Broker outside of OpenShift.

In addition, selecting **Passthrough** for **TLS Termination** relays all communication to AMQ Broker without the OpenShift router decrypting and resending it. Only SSL routes can be exposed because the OpenShift router requires SNI to send traffic to the correct service. For more information see Secured Routes.

The default ports for the various AMQ Broker transport protocols are:

**Table 2.1. Default ports for AMQ Broker transport protocols**

| AMQ Broker transport protocol | Default port |
|---|---|
| All protocols | 61616 |
| All protocols (SSL) | 61617 |
| AMQP | 5672 |
| AMQP (SSL) | 5671 |
| MQTT | 1883 |
| MQTT (SSL) | 8883 |
| STOMP | 61613 |
| STOMP (SSL) | 61612 |

## 2.6. CUSTOMIZING AMQ BROKER CONFIGURATION FILES FOR DEPLOYMENT

If you are using a template from an alternate repository, AMQ Broker configuration files such as **`artemis-users.properties`** can be included. When the image is downloaded for deployment, these files are copied from **`<amq-home>/conf/`** to the **`<broker-instance-dir>/etc/`** directory on AMQ Broker, which is committed to the container and pushed to the OpenShift registry.

**NOTE**

If using this method, ensure that the placeholders in the configuration files (such as **`AUTHENTICATION`**) are not removed, as these placeholders are necessary for building the AMQ Broker on OpenShift Container Platform image.

## 2.7. CONFIGURING CLIENT CONNECTIONS

Clients for the AMQ Broker on OpenShift Container Platform image must specify the OpenShift router port (443) when setting the broker URL for SSL connections. Otherwise, AMQ Broker attempts to use the default SSL port (61617). Including the failover protocol in the URL preserves the client connection in case the pod is restarted or upgraded, or a disruption occurs on the router.

```
...
factory.setBrokerURL("failover://ssl://<route-to-broker-pod>:443");
...
```

**NOTE**

External clients do not support HA.

# CHAPTER 3. TUTORIALS

These tutorials follow on from and assume an OpenShift Container Platform 3.9 instance similar to that created in OpenShift Container Platform 3.9 Getting Started.

In this tutorial you prepare and deploy a multiple-node AMQ Broker instance with persistent storage.

## 3.1. PREPARING THE AMQ BROKER DEPLOYMENT

### Procedure

1. At a command prompt, create a new project:

   ```
   $ oc new-project amq-demo
   ```

2. Create a service account to be used for the AMQ Broker deployment:

   ```
   $ echo '{"kind": "ServiceAccount", "apiVersion": "v1", "metadata":
   {"name": "amq-service-account"}}' | oc create -f -
   ```

3. Add the view role to the service account. The view role enables the service account to view all the resources in the amq-demo namespace, which is necessary for managing the cluster when using the OpenShift dns-ping protocol for discovering the mesh endpoints.

   ```
   $ oc policy add-role-to-user view system:serviceaccount:amq-
   demo:amq-service-account
   ```

4. AMQ Broker requires a broker keystore, a client keystore, and a client truststore that includes the broker keystore. This example uses Java Keytool, a package included with the Java Development Kit, to generate dummy credentials for use with the AMQ Broker installation.

   a. Generate a self-signed certificate for the broker keystore:

      ```
      $ keytool -genkey -alias broker -keyalg RSA -keystore broker.ks
      ```

   b. Export the certificate so that it can be shared with clients:

      ```
      $ keytool -export -alias broker -keystore broker.ks -file
      broker_cert
      ```

   c. Generate a self-signed certificate for the client keystore:

      ```
      $ keytool -genkey -alias client -keyalg RSA -keystore client.ks
      ```

   d. Create a client truststore that imports the broker certificate:

      ```
      $ keytool -import -alias broker -keystore client.ts -file
      broker_cert
      ```

   e. Use the broker keystore file to create the AMQ Broker secret:

      ```
      $ oc secrets new amq-app-secret broker.ks
      ```

f. Add the secret to the service account created earlier:

```
$ oc secrets add sa/amq-service-account secret/amq-app-secret
```

## 3.2. DEPLOYING THE IMAGE AND TEMPLATE

### Procedure

1. Navigate to the OpenShift web console and log in, selecting the **`amq-demo`** project space.
2. Click **Add to Project** > **Browse catalog** to list all of the default image streams and templates.
3. Use the **Filter** search bar to limit the list to those that match **`amq`**. You might need to click **See all** to show the desired application template.
4. Select a template. This example uses the **`amq-broker-71-persistence-ssl.yaml`** template to allow for persistent storage and SSL.

**Table 3.1. Example template**

| Environment variable | Value |
|---|---|
| APPLICATION_NAME | broker |
| AMQ_TRANSPORTS | openwire |
| AMQ_USER | amq-demo-user |
| AMQ_PASSWORD | password |
| VOLUME_CAPACITY | 512Mi |
| AMQ_KEYSTORE_TRUSTSTORE_DIR | /etc/amq-secret-volume |
| AMQ_TRUSTSTORE | broker.ts |
| AMQ_TRUSTSTORE_PASSWORD | password |
| AMQ_KEYSTORE | broker.ks |
| AMQ_KEYSTORE_PASSWORD | password |
| IMAGE_STREAM_NAMESPACE | openshift |

## 3.3. POST-DEPLOYMENT

### 3.3.1. Creating a route

You must create a route for the broker so that clients outside of OpenShift Container Platform can connect using SSL. By default, the OpenWire protocol uses the 61617/TCP port.

**NOTE**

Only one broker can be scaled up. You cannot scale up multiple brokers.

**Procedure**

1. Click **Create a Route** and from the **Service** drop-down menu, click **broker-amq-tcp-ssl**.

2. Select the **Secure route** check box to display the TLS parameters.

3. From the **TLS Termination** drop-down menu, click **Passthrough**. This selection relays all communication to AMQ Broker without the OpenShift router decrypting and resending it.

4. Clients can now connect to the broker by specifying the following in their configuration:

```
factory.setBrokerURL("failover://ssl://broker-amq-
demo.example.com:443");
```

### 3.3.1.1. Monitoring AMQ Broker

This tutorial demonstrates how to monitor AMQ Broker.

**Prerequisite**

You must have created a project and a service account, and added the **`view`** role to the service account for AMQ Broker deployment. For more information see Preparing the AMQ Broker deployment.

**Procedure**

1. At the command line, go to your project:

```
$ oc project monitoramq
```

2. Deploy a new broker instance to the **`monitoramq`** project, using the **`amq-broker-71-basic`** template from the **`openshift`** namespace:

```
$ oc process openshift//amq-broker-71-basic -p
APPLICATION_NAME=broker -p AMQ_USER=admin -p AMQ_PASSWORD=admin -p
AMQ_QUEUES=TESTQUEUE -n monitoramq | oc create -f -

services "broker-amq-amqp" created
services "broker-amq-mqtt" created
services "broker-amq-stomp" created
services "broker-amq-tcp" created
deploymentconfigs "broker-amq" created
```

3. Get the list of running pods:

```
$ oc get pods

NAME                    READY     STATUS    RESTARTS    AGE
broker-amq-1-ftqmk      1/1       Running   0           14d
```

4. Run the **`oc logs`** command:

```
oc logs -f broker-amq-1-ftqmk

Running /amq-broker-71-openshift image, version 1.3-5
INFO: Loading '/opt/amq/bin/env'
INFO: Using java '/usr/lib/jvm/java-1.8.0/bin/java'
INFO: Starting in foreground, this is just for debugging purposes
(stop process by pressing CTRL+C)
...
INFO | Listening for connections at: tcp://broker-amq-1-ftqmk:61616?
maximumConnections=1000&wireFormat.maxFrameSize=104857600
INFO | Connector openwire started
INFO | Starting OpenShift discovery agent for service broker-amq-tcp
transport type tcp
INFO | Network Connector
DiscoveryNetworkConnector:NC:BrokerService[broker-amq-1-ftqmk]
started
INFO | Apache ActiveMQ 5.11.0.redhat-621084 (broker-amq-1-ftqmk,
ID:broker-amq-1-ftqmk-41433-1491445582960-0:1) started
INFO | For help or more information please see:
http://activemq.apache.org
WARN | Store limit is 102400 mb (current store usage is 0 mb). The
data directory: /opt/amq/data/kahadb only has 9684 mb of usable
space - resetting to maximum available disk space: 9684 mb
WARN | Temporary Store limit is 51200 mb, whilst the temporary data
directory: /opt/amq/data/broker-amq-1-ftqmk/tmp_storage only has
9684 mb of usable space - resetting to maximum available 9684 mb.
```

5. Run your query to monitor your broker for **`MaxConsumers`**:

```
$ curl -k -u admin:admin http://console-
broker.monitoramq.apps.example.com/console/jolokia/read/org.apache.a
ctivemq.artemis:broker=%22broker%22,component=addresses,address=%22T
ESTQUEUE%22,subcomponent=queues,routing-
type=%22anycast%22,queue=%22TESTQUEUE%22/MaxConsumers

{"request":
{"mbean":"org.apache.activemq.artemis:address=\"TESTQUEUE\",broker=\
"broker\",component=addresses,queue=\"TESTQUEUE\",routing-
type=\"anycast\",subcomponent=queues","attribute":"MaxConsumers","ty
pe":"read"},"value":-1,"timestamp":1528297825,"status":200}
```

# CHAPTER 4. REFERENCE

## 4.1. APPLICATION TEMPLATE PARAMETERS

Configuration of the AMQ Broker on OpenShift Container Platform image is performed by specifying values of application template parameters. The following parameters can be configured:

**Table 4.1. Application template parameters**

| Parameter | Description |
|---|---|
| `AMQ_ADDRESSES` | Specifies the addresses available by default on the broker on its startup, in a comma-separated list. |
| `AMQ_ADMIN_PASSWORD` | Specifies the password used for authentication to the broker. If no value is specified, a random password is generated. |
| `AMQ_ADMIN_USERNAME` | Specifies the user name used as an administrator authentication to the broker. If no value is specified, a random user name is generated. |
| `AMQ_ALLOW_ANONYMOUS` | If not specified, or set to `true`, `allow-anonymous` is enabled. Otherwise `require-login` is enabled. |
| `AMQ_CLUSTERED` | Enables clustering. |
| `AMQ_CLUSTER_PASSWORD` | Specifies the password to use for clustering. If no value is specified, a random password is generated. |
| `AMQ_CLUSTER_USER` | Specifies the cluster user to use for clustering. If no value is specified, a random user name is generated. |
| `AMQ_DATA_DIR` | Specifies the directory for the data. Used in stateful sets. |
| `AMQ_EXTRA_ARGS` | Specifies additional arguments to pass to `artemis create`. |
| `AMQ_KEYSTORE` | Specifies the SSL keystore file name. If no value is specified, a random password is generated but SSL will not be configured. |
| `AMQ_KEYSTORE_PASSWORD` | (Optional) Specifies the password used to decrypt the SSL keystore. |
| `AMQ_KEYSTORE_TRUSTSTORE_DIR` | Specifies the directory where the secrets are mounted. The default value is `/etc/amq-secret-volume`. |

| Parameter | Description |
|---|---|
| **`AMQ_MAX_CONNECTIONS`** | For SSL only, specifies the maximum number of connections that an acceptor will accept. |
| **`AMQ_NAME`** | Specifies the name of the broker instance. |
| **`AMQ_PASSWORD`** | Specifies the password used for authentication to the broker. If no value is specified, a random password is generated. |
| **`AMQ_QUEUES`** | Specifies the queues available by default on the broker on its startup, in a comma-separated list. |
| **`AMQ_REPLICATED`** | If set to **`true`**, enables broker replication. |
| **`AMQ_RESET_CONFIG`** | If set to **`true`**, overwrites the configuration at the destination directory. |
| **`AMQ_ROLE`** | Specifies the name for the role created. The default value is **`amq`**. |
| **`AMQ_SLAVE`** | If replication is enabled, sets the pod to be a slave. |
| **`AMQ_TRANSPORTS`** | Specifies the messaging protocols used by the broker in a comma-separated list. Available options are **`amqp`**, **`mqtt`**, **`openwire`**, **`stomp`**, and **`hornetq`**. If none are specified, all available protocols are available. Note that for integration of the image with Red Hat JBoss Enterprise Application Platform, the OpenWire protocol must be specified, while other protocols can be optionally specified as well. |
| **`AMQ_TRUSTSTORE`** | Specifies the SSL truststore file name. If no value is specified, a random password is generated but SSL will not be configured. |
| **`AMQ_TRUSTSTORE_PASSWORD`** | (Optional) Specifies the password used to decrypt the SSL truststore. |
| **`AMQ_USER`** | Specifies the user name used for authentication to the broker. If no value is specified, a random user name is generated. |
| **`APPLICATION_NAME`** | Specifies the name of the application used internally within OpenShift. It is used in names of services, pods, and other objects within the application. |

| Parameter | Description |
|---|---|
| `GLOBAL_MAX_SIZE` | Specifies the maximum amount of memory that message data can consume. If no value is specified, half of the system's memory is allocated. |
| `IMAGE` | Specifies the image. Used in the `persistence`, `persistent-ssl`, and `statefulset-clustered` templates. |
| `IMAGE_STREAM_NAMESPACE` | Specifies the image stream name space. Used in the `ssl` and `basic` templates. |
| `JBOSS_AMQ_VERSION` | Specifies the AMQ release version, which determines which AMQ image is used as a basis for the application, for example, `7.1.0`. |
| `OPENSHIFT_DNS_PING_SERVICE_PORT` | Specifies the port number for the OpenShift DNS ping. |
| `VOLUME_CAPACITY` | Specifies the size of the persistent storage for database volumes. |

## 4.2. SECURITY

Only SSL connections can connect from outside of the OpenShift instance. The non-SSL version of the protocols can only be used inside the OpenShift instance.

For security reasons, using the default keystore and truststore generated by the system is discouraged. Generate your own keystore and truststore and supply them to the image by using the OpenShift secrets mechanism.

## 4.3. LOGGING

In addition to viewing the OpenShift logs, you can troubleshoot a running AMQ Broker on OpenShift Container Platform image by viewing the AMQ logs that are outputted to the container's console.

### Procedure

1. At the command line, run the following command:

```
$ oc logs -f <pass:quotes[<pod-name>]> <pass:quotes[<container-name>]>
```

*Revised on 2018-06-14 10:46:43 EDT*