Red Hat AMQ 7.2

Deploying AMQ Interconnect on OpenShift Container Platform

For Use with AMQ Interconnect 1.2

Last Updated: 2018-10-11

Red Hat AMQ 7.2 Deploying AMQ Interconnect on OpenShift Container Platform

For Use with AMQ Interconnect 1.2

Legal Notice

Abstract

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

Table of Contents

CHAPTER 1. OVERVIEW

You can deploy and manage AMQ Interconnect in OpenShift Container Platform, which enables you to build a scalable messaging network in a hybrid cloud environment.

In this solution, AMQ Interconnect is deployed as a message router running in an OpenShift pod. You can scale up the deployment to add routers, which automatically connect to each other in a router mesh topology. You can use the standard OpenShift methods to manage the pods and deploy new versions.

AMQ Interconnect is a lightweight AMQP message router for building scalable, available, and performant messaging networks. It is based on Dispatch Router from the Apache Qpid™ project.

CHAPTER 2. PREPARING TO DEPLOY AMQ INTERCONNECT ON OPENSHIFT CONTAINER PLATFORM

Before deploying AMQ Interconnect on OpenShift Container Platform, you must complete several preparatory tasks.

2.1. VERIFYING THE AVAILABILITY OF AMQ INTERCONNECT TEMPLATES

You must verify that the AMQ Interconnect application templates are available in the OpenShift Container Platform Catalog. Some versions of OpenShift Container Platform include the templates, so you must check to verify that they are available in your environment. If they are not available, you must install them.

Procedure

1. Log in to OpenShift as a cluster administrator:

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

2. Switch to the **openshift** project.

```
$ oc project openshift
```

3. Verify that the latest version of the AMQ Interconnect image stream is available:

```
$ oc get imagestreamtag -n openshift | grep amq-interconnect-1.2-
openshift
```

   If the latest version of the image stream is available in your environment, you should see **amq-interconnect-1.2-openshift:latest** in the command output.

4. Is the latest version of AMQ Interconnect image stream available?

   - If yes, proceed to the next step.
   - If no, install the latest version of the image stream.
     This command imports the latest AMQ Interconnect image stream into the **openshift** namespace:

```
$ oc import-image amq-interconnect-1.2-openshift:latest -n
openshift --from=registry.access.redhat.com/amq-interconnect/amq-
interconnect-1.2-openshift --confirm
```

5. Install the AMQ Interconnect application templates:

```
$ curl https://raw.githubusercontent.com/jboss-container-images/amq-
interconnect-1-openshift-image/amq-interconnect-11-
dev/templates/amq-interconnect-1-basic.yaml | oc create -f -

$ curl https://raw.githubusercontent.com/jboss-container-images/amq-
interconnect-1-openshift-image/amq-interconnect-11-
```

```
dev/templates/amq-interconnect-1-tls-auth.yaml | oc create -f -

$ curl https://raw.githubusercontent.com/jboss-container-images/amq-
interconnect-1-openshift-image/amq-interconnect-11-
dev/templates/amq-interconnect-1-sasldb-auth.yaml | oc create -f -
```

This creates three AMQ Interconnect application templates, which you can use to deploy AMQ Interconnect on OpenShift Container Platform:

`amq-interconnect-1-basic.yaml`

Deploys a router with a basic, default configuration and no security.

`amq-interconnect-1-tls-auth.yaml`

Deploys a router with both inter-router and client traffic secured by TLS authentication.

`amq-interconnect-1-sasldb-auth.yaml`

Deploys a router with inter-router traffic secured by TLS authentication and client traffic secured by SASL user name and password authentication.

2.2. CREATING SECRETS FOR SSL/TLS AUTHENTICATION

OpenShift uses objects called **`Secrets`** to hold sensitive information such as SSL/TLS certificates. If you want to secure inter-router traffic, client traffic, or both, then you must create the SSL/TLS certificates and private keys and provide them to OpenShift as secrets.

Procedure

1. If you do not have an existing certificate authority (CA) certificate for inter-router connections, create one.
   These commands create a self-signed CA certificate for inter-router connections:

```
# Create a new directory for the inter-router certificates.
$ mkdir internal-certs

# Create a private key for the CA.
$ openssl genrsa -out internal-certs/ca-key.pem 2048

# Create a certificate for the CA.
$ openssl req -new -batch -key internal-certs/ca-key.pem -out
internal-certs/ca-csr.pem

# Self sign the certificate.
$ openssl x509 -req -in internal-certs/ca-csr.pem -signkey internal-
certs/ca-key.pem -out internal-certs/ca.crt
```

2. Create a certificate for the router signed by the CA.
   These commands create a private key and a certificate, and sign the certificate using the CA created in the previous step:

```
# Create a private key.
$ openssl genrsa -out internal-certs/tls.key 2048

# Create a certificate for the router.
$ openssl req -new -batch -subj "/CN=amq-
```

```
interconnect.myproject.svc.cluster.local" -key internal-
certs/tls.key -out internal-certs/server-csr.pem

# Sign the certificate using the CA.
$ openssl x509 -req -in internal-certs/server-csr.pem -CA internal-
certs/ca.crt -CAkey internal-certs/ca-key.pem -out internal-
certs/tls.crt -CAcreateserial
```

3. Create a secret containing the private key, router certificate, and CA certificate.
   This command creates the secret using the key and certificates that were created in the previous steps:

```
$ oc create secret generic inter-router-certs --from-
file=tls.crt=internal-certs/tls.crt  --from-file=tls.key=internal-
certs/tls.key  --from-file=ca.crt=internal-certs/ca.crt
```

4. If you want to use SSL/TLS to authenticate client connections (as opposed to authenticating clients using SASL), create a CA certificate for client connections.
   These commands create a self-signed CA certificate for client connections:

```
# Create a new directory for the client certificates.
$ mkdir client-certs

# Create a private key for the CA.
$ openssl genrsa -out client-certs/ca-key.pem 2048

# Create a certificate for the CA.
$ openssl req -new -batch -key client-certs/ca-key.pem -out client-
certs/ca-csr.pem

# Self sign the certificate.
$ openssl x509 -req -in client-certs/ca-csr.pem -signkey client-
certs/ca-key.pem -out client-certs/ca.crt
```

5. Create a certificate for client connections signed by the CA.
   These commands create a private key and a certificate, and then sign the certificate using the CA created in the previous step:

```
# Create a private key.
$ openssl genrsa -out client-certs/tls.key 2048

# Create a certificate for the client connections.
$ openssl req -new -batch -subj "/CN=myclient" -key client-
certs/tls.key -out client-certs/client-csr.pem

# Sign the certificate using the CA.
$ openssl x509 -req -in client-certs/client-csr.pem -CA client-
certs/ca.crt -CAkey client-certs/ca-key.pem -out client-
certs/tls.crt -CAcreateserial
```

6. Create a secret containing the CA certificate used to sign client certificates.
   This command creates the secret using the certificate that was created in the previous steps:

```
$ oc create secret generic client-ca --from-file=ca.crt=client-
certs/ca.crt
```

2.3. CREATING SECRETS FOR SASL AUTHENTICATION

To authenticate clients against user name and password pairs stored in a SASL database, you must create a list containing the user names and passwords, and provide it to OpenShift as a secret.

When using SASL user name and password authentication to connect to a router, you must provide the user name qualified with the domain. For a router running in OpenShift, the domain is the application name (**`amq-interconnect`** by default).

Procedure

1. Create a text file containing the client user names and passwords.
   Use the following syntax: *<username>=<password>*.

   In this example, the user names and passwords are defined for three users:

   users.txt

   ```
   guest=guest
   admin=admin123
   dev=dev123
   ```

2. Create a secret containing the user names and passwords.
   This command creates a secret (**`users`**) containing the **`users.txt`** file created in the previous step.

   ```
   $ oc create secret generic users --from-env-file=./users.txt --dry-
   run=true -o yaml | oc apply -f -
   ```

CHAPTER 3. DEPLOYING AMQ INTERCONNECT ON OPENSHIFT CONTAINER PLATFORM

You can deploy AMQ Interconnect in your OpenShift environment by using the templates that are provided in the OpenShift catalog. Each template deploys one or more AMQ Interconnect routers, which you can scale up to a mesh topology.

NOTE

As an alternative to using the OpenShift web UI, you can use the **oc** command line tool to deploy the templates and set their parameters. For more information, see your OpenShift documentation.

Prerequisites

- Completion of all preparatory tasks.
  For more information, see Chapter 2, *Preparing to deploy AMQ Interconnect on OpenShift Container Platform*.

Procedure

1. In the OpenShift web UI, in your project, select **Add to Project** → **Browse Catalog**.
2. Search for **`interconnect`**, and then select one of the following templates:

   `Red Hat AMQ Interconnect Router 1.x`

   A single router with a basic, default configuration and no security.

   `Red Hat AMQ Interconnect Router 1.x with TLS authentication`

   A router mesh of two routers in which client and inter-router traffic is secured with SSL/TLS authentication.

   `Red Hat AMQ Interconnect Router 1.x with SASL`

   A router mesh of two routers in which the inter-router traffic is secured with SSL/TLS authentication, and the clients are authenticated against user name and password pairs stored in a SASL database.

3. Click **`Next`** and review the configuration.
   You can change the default values for any of the parameters.
4. Click **`Create`** and then confirm the creation.
   AMQ Interconnect is deployed with each router running in a separate pod.
5. Verify that each pod was deployed successfully, and that each router is operational.

   a. Navigate to **Applications** → **Deployments**, and then click the most recent version.
   The deployment details are displayed. The pods should have a status of **`Running`**.

   b. Click **Logs** to verify that each router is operational.
   At the top of log terminal, the log entries indicate that the router is operational:

```
2018-09-12 16:37:50.922166 +0000 SERVER (info) Container Name:
amq-interconnect-1-6pj98
2018-09-12 16:37:50.922346 +0000 ROUTER (info) Router started in
```

```
Interior mode, area=0 id=amq-interconnect-1-6pj98
2018-09-12 16:37:50.922375 +0000 ROUTER (info) Version: Red Hat
AMQ Interconnect 1.2.0 (qpid-dispatch 1.2.0)
2018-09-12 16:37:50.923860 +0000 ROUTER_CORE (info) Router Core
thread running. 0/amq-interconnect-1-6pj98
...
2018-09-12 16:37:51.006921 +0000 SERVER (notice) Operational, 4
Threads Running (process ID 1)
```

CHAPTER 4. MANAGING AMQ INTERCONNECT ON OPENSHIFT CONTAINER PLATFORM

After deploying AMQ Interconnect on OpenShift Container Platform, you can connect messaging endpoints to it, monitor the deployment, add and remove routers, and change the configuration of the routers.

4.1. CREATING ROUTES

In OpenShift Container Platform, routes expose the router mesh's service so that external traffic can reach it (such as clients, message brokers, and routers running in different OpenShift clusters).

Procedure

1. Navigate to **Applications** → **Routes**.
2. Click **`Create Route`**.
3. Enter a **Name** for the route.
   The name will be the first part of the route host name.
4. If necessary, enter a **Hostname** and **Path**.
5. Select the **Target Port**.

To create a route for…	Select this port…
External clients or message brokers to connect to the router mesh without authentication	5672
External clients or message brokers to connect to the router mesh with authentication	5671
External routers to connect to the router mesh	55672
Accessing the web console	8672

6. If necessary, secure the route.
   a. Click **Secure route**.
   b. In the **TLS Termination** drop-down, select **Passthrough**.
      The route is secured using the security certificates that you created and passed to OpenShift as secrets.
7. Click **`Create`**.
   The route is created.

4.2. CONNECTING CLIENTS TO A ROUTER MESH

After deploying AMQ Interconnect on OpenShift Container Platform, the routers begin listening for client connections. Clients running in the same OpenShift cluster as the router mesh, a different cluster, or outside of OpenShift altogether can connect to the router mesh to exchange messages.

Prerequisites

- If the client is in a different OpenShift cluster than the router mesh (or outside of OpenShift altogether), the service must be exposed as a route.
  For more information, see Section 4.1, “Creating routes”.

Procedure

- To connect a client to the router mesh, use the following connection URL syntax:

```
<scheme>://[<username>@]<host>[:<port>]
```

  <scheme>

  For unencrypted TCP, use **amqp**. If you deployed the router mesh with SSL/TLS authentication, use **amqps**.

  <username>

  If you deployed the router mesh with SASL user name/password authentication, you must provide the client’s user name.

  <host>

  If the client is in the same OpenShift cluster as the router mesh, use the OpenShift service IP address. Otherwise, use the host name of the route.

  <port>

  If you are connecting to a route, you must specify the port. Use **80** for unsecured connections, and **443** for secured connections.

  The following table shows some example connection URLs.

URL	Description
amqp://192.0.2.1	The client and router mesh are both in the same OpenShift cluster, so the service IP address is used for the connection URL.
amqps://amq-interconnect-myproject.192.0.2.1.nip.io:443	The client is outside of OpenShift, so the route host name is used for the connection URL. In this case, SSL/TLS authentication is implemented, which requires the **amqps** scheme and port **443**.

4.3. CONNECTING TO A MESSAGE BROKER

You can connect a router mesh running in OpenShift to AMQ Broker brokers so that clients can exchange messages with brokers. You can connect to brokers that are also deployed in OpenShift, or brokers that are running outside of OpenShift.

To connect to a broker, you must add a connector to the router configuration.

Procedure

1. Navigate to **Resources** → **Config Maps**.
   The **amq-interconnect** config map contains the configuration for each router in the router mesh.

2. Click **amq-interconnect**.

3. Select **Actions** → **Edit**.

4. In the **Value** text box, add a **connector** entity.
   A connector defines an outgoing connection from a router to an external AMQP container (in this case, a broker).

   This example defines a connection to a broker:

```
connector {
    name: broker
    role: route-container
    host: 192.0.2.1  (1)
    port: 61616
    saslMechanisms: ANONYMOUS
}
```

   (1) If the broker is running in the same OpenShift cluster as the router mesh, then use the service IP address. If the broker is running in a different OpenShift cluster, then use the fully-qualified domain name of that OpenShift cluster's route.

5. Click **Save**.

6. To apply the changes, trigger a new deployment.

   a. Navigate to **Applications** → **Deployments**.

   b. Click **amq-interconnect**.

   c. Click **Deploy**.
      A new deployment is started and becomes active.

7. In a terminal, run the **qdstat** command from one of the router pods to verify that the routers can connect to the broker.
   This example shows that the router is connected to the broker on the 192.0.2.1 host.

```
$ oc exec amq-interconnect-4-5qkqz -it -- qdstat -c
Connections
  id  host                 container
role            dir  security     authentication  tenant

================================================================================
================================================
  1   192.0.2.1:61616      0.0.0.0
route-container  out  no-security  anonymous-user
  6   203.0.113.10:49806   amq-interconnect-4-mshr4
inter-router     in   no-security  anonymous-user
  4   203.0.113.11:33312   amq-interconnect-4-5qkqz
```

```
inter-router      in    no-security  anonymous-user
  5    203.0.113.3:40484     amq-interconnect-4-ds7lp
inter-router      in    no-security  anonymous-user
  39   127.0.0.1:60678       e3c03c50-4d6d-4962-bc53-65480ab8ae0e
normal            in    no-security  no-auth
```

4.4. CONNECTING ROUTER MESHES RUNNING IN DIFFERENT OPENSHIFT CLUSTERS

You can connect router meshes running in different OpenShift clusters. By doing this, you can create geographically distributed message routing networks that span cloud environments.

This procedure demonstrates how to connect two router meshes running in different OpenShift clusters.

Prerequisites

- If you want the inter-router connections to be secure, both OpenShift clusters must contain a secret containing a private key, a certificate, and CA certificate. You do not need to use the same private key and certificates in both clusters, but they must be signed by the same CA.
  For more information, see Section 2.2, “Creating secrets for SSL/TLS authentication”.

Procedure

1. In the first OpenShift cluster, create a route that is accessible to the second OpenShift cluster. The route should target port 55672, and it should be secured with SSL/TLS passthrough. For more information, see Section 4.1, “Creating routes”.

2. In the second OpenShift cluster, create a connector to the route that you created in the previous step.

   a. Navigate to **Resources** → **Config Maps**.
   The **`amq-interconnect`** config map contains the configuration for each router in the router mesh.

   b. Click **`amq-interconnect`**.

   c. Select **Actions** → **Edit**.

   d. In the **`Value`** text box, add a **`connector`** entity.
   A connector defines an outgoing connection from a router to an external AMQP container (in this case, the routers in the first OpenShift cluster).

   This example defines a connection to a router running in a different OpenShift cluster:

```
connector {
    name: router
    role: inter-router
    host: first-router-mesh-myproject.192.0.2.1.nip.io   1
    port: 443
    sslProfile: inter_router_tls   2
    verifyHostname: no
}
```

1 The host name of the route that you created in the first OpenShift cluster.

2 The name of the SSL/TLS profile that defines the private keys and certificates used to secure inter-router traffic.

e. Click **Save**.

3. Start a new deployment to apply the configuration change to the routers.

a. Navigate to **Applications** → **Deployments**.

b. Click **amq-interconnect**.

c. Click **Deploy**.
A new deployment is started and becomes active. The routers in the second OpenShift cluster connect to the routers in the first OpenShift cluster to form a distributed router mesh that spans OpenShift clusters.

4.5. MONITORING THE ROUTER MESH USING THE WEB CONSOLE

The AMQ Interconnect container image includes the Red Hat AMQ Interconnect Console, which enables you to monitor the status and performance of your router mesh.

Prerequisites

- An OpenShift route that targets port 8672 (HTTP) must exist.
For more information, see Section 4.1, “Creating routes”.

Procedure

1. Navigate to the URL for the HTTP route.
To see a list of routes for your OpenShift cluster, navigate to **Applications** → **Routes**.

The Red Hat AMQ Interconnect Console opens. If you deployed AMQ Interconnect using the **Red Hat AMQ Interconnect Router 1.x with TLS authentication** template, the **Connect** tab is displayed.

2. If necessary, log in to the web console.
If you deployed AMQ Interconnect using the **Red Hat AMQ Interconnect Router 1.x with TLS authentication** template, use the user name and password that you entered when you deployed the template. To find the user name and password, navigate to **Resources** → **Secrets** → **amq-interconnect-users**.

The syntax for the user name is *<user>@<domain>* (the domain is the OpenShift application name; **amq-interconnect** is the default value). For example, **admin@amq-interconnect**.

3. Use the tabs to monitor the router mesh.

This tab…	Provides…
Overview	Aggregate information about routers, addresses, links, connections, and logs.
Entities	Detailed information about each AMQP management entity for each router in the router mesh.

This tab…	Provides…
Topology	A graphical view of the router network. The topology shows how the routers are connected, and how messages are flowing through the network.
Charts	Graphs of the information that is displayed on the **Entities** tab.
Message Flow	A chord diagram showing the real-time message flow by address.
Schema	The management schema that controls each of the routers in the router mesh.

4.6. MONITORING THE ROUTER MESH USING THE CLI

You can use the **qdstat** command line tool from within a router pod to view statistics about the routers in your router mesh. For example, you can view information about the attached links and configured addresses, available connections, and nodes in the router network.

Procedure

- In a terminal, run the **qdstat** command from within a router pod.
  This command displays the router mesh topology from the perspective of router **amq-interconnect-4-5qkqz**.

```
$ oc exec amq-interconnect-4-5qkqz -it -- qdstat -n
Routers in the Network
Last Topology Change: Tuesday Sep 25 20:35:54 2018 GMT
  router-id                 next-hop  link
  ========================================
  amq-interconnect-4-5qkqz  (self)    -
  amq-interconnect-4-8wbfg  -         3
  amq-interconnect-4-ds7lp  -         1
  amq-interconnect-4-mshr4  -         0
```

Additional resources

- Monitoring AMQ Interconnect using qdstat in *Using AMQ Interconnect*
- qdstat man page

4.7. SCALING THE ROUTER MESH

You can scale your deployment to add or remove routers from the router mesh. When you scale up the router mesh, a new pod is deployed with a router, which automatically connects to any other running routers.

Procedure

1. Navigate to the **Overview** page.

2. Do one of the following:

   - To add routers to the mesh, click the up arrow next to the pods diagram.
     A new pod is deployed, with a router running inside of it. The router automatically connects to each router in the mesh to maintain a full mesh topology.

   - To remove routes from the mesh, click the down arrow next to the pods diagram.
     A pod is removed from the deployment, and its router is shut down. Any clients that were connected to the router are disconnected, but can fail over to any of the remaining routers in the mesh.

4.8. CHANGING THE ROUTER CONFIGURATION

The AMQ Interconnect application templates include a basic router configuration that is applied to the router when it is deployed. The configuration is stored in the **`amq-interconnect`** config map. You can edit this config map to change the configuration of the routers in the router mesh.

Procedure

1. Navigate to **Resources** → **Config Maps**.
   The **`amq-interconnect`** config map contains the configuration for each router in the router mesh.

2. Click **amq-interconnect**.

3. Select **Actions** → **Edit**.

4. In the **`Value`** text box, change the router configuration as needed.

5. Click **`Save`**.

6. Navigate to **Applications** → **Deployments**.

7. Click **amq-interconnect**.

8. Click **`Deploy`**.
   A new deployment is started and becomes active. The configuration changes you made are applied to each router in the router mesh.

Additional resources

- For more information about the router configuration file, see Configuration in *Using AMQ Interconnect*.

- For more information about the router configuration entities and attributes, see the qdrouterd.conf man page.

4.9. HOW AMQ INTERCONNECT UPGRADES WORK

The AMQ Interconnect application templates include an **`ImageChange`** trigger. This means that when a new version of the AMQ Interconnect container image is available, the new image is deployed to the router mesh automatically. You do not need to deploy new image versions manually.

Revised on 2018-10-11 19:27:33 UTC