Red Hat AMQ 7.6

Deploying AMQ Interconnect on OpenShift

For Use with AMQ Interconnect 1.8

Last Updated: 2020-06-16

Red Hat AMQ 7.6 Deploying AMQ Interconnect on OpenShift

For Use with AMQ Interconnect 1.8

Legal Notice

Abstract

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

Table of Contents

CHAPTER 1. GETTING STARTED WITH AMQ INTERCONNECT ON OPENSHIFT CONTAINER PLATFORM

AMQ Interconnect is a lightweight AMQP 1.0 message router for building large, highly resilient messaging networks for hybrid cloud and IoT/edge deployments. AMQ Interconnect automatically learns the addresses of messaging endpoints (such as clients, servers, and message brokers) and flexibly routes messages between them.

This document describes how to deploy AMQ Interconnect on OpenShift Container Platform by using the AMQ Interconnect Operator and the **Interconnect** Custom Resource Definition (CRD) that it provides. The CRD defines an AMQ Interconnect deployment, and the Operator creates and manages the deployment in OpenShift Container Platform.

NOTE

If you are unable to use the AMQ Interconnect Operator, you can deploy AMQ Interconnect in OpenShift by using the AMQ Interconnect application templates provided in the OpenShift catalog. For more information, see Deploying AMQ Interconnect on OpenShift.

1.1. WHAT OPERATORS ARE

Operators are a method of packaging, deploying, and managing a Kubernetes application. They take human operational knowledge and encode it into software that is more easily shared with consumers to automate common or complex tasks.

In OpenShift Container Platform 4.0, the *Operator Lifecycle Manager (OLM)* helps users install, update, and generally manage the life cycle of all Operators and their associated services running across their clusters. It is part of the Operator Framework, an open source toolkit designed to manage Kubernetes native applications (Operators) in an effective, automated, and scalable way.

The OLM runs by default in OpenShift Container Platform 4.0, which aids cluster administrators in installing, upgrading, and granting access to Operators running on their cluster. The OpenShift Container Platform web console provides management screens for cluster administrators to install Operators, as well as grant specific projects access to use the catalog of Operators available on the cluster.

OperatorHub is the graphical interface that OpenShift Container Platform cluster administrators use to discover, install, and upgrade Operators. With one click, these Operators can be pulled from OperatorHub, installed on the cluster, and managed by the OLM, ready for engineering teams to self-service manage the software in development, test, and production environments.

Additional resources

- For more information about Operators, see the OpenShift documentation.

1.2. PROVIDED CUSTOM RESOURCES

The AMQ Interconnect Operator provides the **Interconnect** Custom Resource Definition (CRD), which allows you to interact with an AMQ Interconnect deployment running on OpenShift Container Platform just like other OpenShift Container Platform API objects.

The **Interconnect** CRD represents a deployment of AMQ Interconnect routers. The CRD provides elements for defining many different aspects of a router deployment in OpenShift Container Platform such as:

- Number of AMQ Interconnect routers
- Deployment topology
- Connectivity
- Address semantics

CHAPTER 2. INSTALLING THE AMQ INTERCONNECT OPERATOR

You use *OperatorHub* to add the following Operators to your OpenShift Container Platform cluster:

AMQ Certificate Manager Operator

A Kubernetes add-on that generates the TLS certificates used by the AMQ Interconnect Operator. This Operator must be installed once for the OpenShift Container Platform cluster. When installed, it is available to all users and projects in the cluster.

AMQ Interconnect Operator

The Operator for deploying AMQ Interconnect router networks. This Operator must be installed separately for each project that uses it.

NOTE

Installing an Operator requires administrator-level privileges for your OpenShift cluster.

2.1. ADDING THE AMQ CERTIFICATE MANAGER OPERATOR

The AMQ Certificate Manager Operator (cert-manager) is a Kubernetes add-on that issues and manages TLS certificates. The AMQ Interconnect Operator uses it to automatically create the TLS certificates needed to secure the router network.

The AMQ Certificate Manager Operator must be installed once for the OpenShift Container Platform cluster. When installed, it is available to all users and projects in the cluster.

Prerequisites

- Access to an OpenShift Container Platform 4.1 cluster using a **cluster-admin** account.

Procedure

1. In the OpenShift Container Platform web console, navigate to **Catalog → OperatorHub**.
2. Choose **AMQ Certificate Manager Operator** from the list of available Operators, and then click **Install**.
3. On the **Create Operator Subscription** page, accept all of the defaults, and then click **Subscribe**.
   This makes the Operator available to all users and projects that use this OpenShift cluster. The **Subscription Overview** page appears displaying the status of the Operator installation.
4. Switch to the **Catalog → Installed Operators** page, and then switch to the **openshift-operators** project.
   The **AMQ Certificate Manager Operator** should be displayed with a status of **InstallSucceeded**.
5. If the installation is not successful, troubleshoot the error:
   a. Switch to the **Catalog → Operator Management** page and inspect the **Operator Subscriptions** and **Install Plans** tabs for any failures or errors in **Status**.

b. Switch to the **Workloads → Pods** page and check the logs in any Pods that are reporting issues.

Additional resources

- For more information about **cert-manager**, see the cert-manager documentation.

2.2. ADDING THE AMQ INTERCONNECT OPERATOR

The AMQ Interconnect Operator creates and manages AMQ Interconnect router networks in OpenShift Container Platform. This Operator must be installed separately for each project that uses it.

Prerequisites

- Access to an OpenShift Container Platform 4.1 cluster using a **cluster-admin** account.
- AMQ Certificate Manager Operator is installed in the OpenShift Container Platform cluster.

Procedure

1. In the OpenShift Container Platform web console, navigate to **Catalog → OperatorHub**.
2. Choose **AMQ Interconnect Operator** from the list of available Operators, and then click **Install**.
3. On the **Create Operator Subscription** page, select the namespace into which you want to install the Operator, and then click **Subscribe**.
   The **Subscription Overview** page appears displaying the status of the Operator installation.
4. Switch to the **Catalog → Installed Operators** page, and verify that the AMQ Interconnect Operator is displayed and its **Status** is **InstallSucceeded**.
5. If the installation is not successful, troubleshoot the error:
   a. Switch to the **Catalog → Operator Management** page and inspect the **Operator Subscriptions** and **Install Plans** tabs for any failures or errors in **Status**.
   b. Switch to the **Workloads → Pods** page and check the logs in any Pods that are reporting issues.

CHAPTER 3. CREATING A ROUTER NETWORK

To create a network of AMQ Interconnect routers, you define a deployment in an **Interconnect** Custom Resource, and then apply it. The AMQ Interconnect Operator creates the deployment by scheduling the necessary Pods and creating any needed Resources.

The procedures in this section demonstrate the following router network topologies:

- Interior router mesh
- Interior router mesh with edge routers for scalability
- Inter-cluster router network that connects two OpenShift clusters

Prerequisites

- The AMQ Interconnect Operator is installed in your OpenShift Container Platform project.

3.1. CREATING AN INTERIOR ROUTER DEPLOYMENT

Interior routers establish connections with each other and automatically compute the lowest cost paths across the network.

Procedure

This procedure creates an interior router network of three routers. The routers automatically connect to each other in a mesh topology, and their connections are secured with mutual SSL/TLS authentication.

1. Create an **Interconnect** Custom Resource YAML file that describes the interior router deployment.

   Sample router-mesh.yaml file

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  deploymentPlan:
    role: interior (1)
    size: 3 (2)
    placement:  Any (3)
```

(1) The operating mode of the routers in the deployment. The Operator will automatically connect interior routers in a mesh topology.

(2) The number of routers to create.

(3) Each router runs in a separate Pod. The placement defines where in the cluster the Operator should schedule and place the Pods. You can choose the following placement options:

Any
: The Pods can run on any node in the OpenShift Container Platform cluster.

Every

The Operator places a router Pod on each node in the cluster. If you choose this option, the **Size** property is not needed - the number of routers corresponds to the number of nodes in the cluster.

Anti-Affinity

The Operator ensures that multiple router Pods do not run on the same node in the cluster. If the size is greater than the number of nodes in the cluster, the extra Pods that cannot be scheduled will remain in a **Pending** state.

2. Create the router deployment described in the YAML file.

```
$ oc apply -f router-mesh.yaml
```

The Operator creates a deployment of interior routers in a mesh topology that uses default address semantics. It also creates a Service through which the routers can be accessed, and a Route through which you can access the web console.

3. Verify that the router mesh was created and the Pods are running.
Each router runs in a separate Pod. They connect to each other automatically using the Service that the Operator created.

```
$ oc get pods
NAME                                    READY   STATUS    RESTARTS   AGE
interconnect-operator-587f94784b-4bzdx   1/1     Running   0          52m
router-mesh-6b48f89bd-588r5              1/1     Running   0          40m
router-mesh-6b48f89bd-bdjc4              1/1     Running   0          40m
router-mesh-6b48f89bd-h6d5r              1/1     Running   0          40m
```

4. Review the router deployment.

```
$ oc get interconnect/router-mesh -o yaml
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
...
spec:
  addresses:  1
  - distribution: closest
    prefix: closest
  - distribution: multicast
    prefix: multicast
  - distribution: closest
    prefix: unicast
  - distribution: closest
    prefix: exclusive
  - distribution: multicast
    prefix: broadcast
  deploymentPlan:  2
    livenessPort: 8888
    placement: Any
    resources: {}
    role: interior
    size: 3
  edgeListeners:  3
  - port: 45672
  interRouterListeners:  4
```

```
- authenticatePeer: true
  expose: true
  port: 55671
  saslMechanisms: EXTERNAL
  sslProfile: inter-router
listeners: 5
- port: 5672
- authenticatePeer: true
  expose: true
  http: true
  port: 8080
- port: 5671
  sslProfile: default
sslProfiles: 6
- credentials: router-mesh-default-tls
  name: default
- caCert: router-mesh-inter-router-tls
  credentials: router-mesh-inter-router-tls
  mutualAuth: true
  name: inter-router
users: router-mesh-users 7
```

1. The default address configuration. All messages sent to an address that does not match any of these prefixes are distributed in a balanced anycast pattern .

2. A router mesh of three interior routers was deployed.

3. Each interior router listens on port **45672** for connections from edge routers.

4. The interior routers connect to each other on port **55671**. These inter-router connections are secured with SSL/TLS mutual authentication. The **inter-router** SSL Profile contains the details of the certificates that the Operator generated.

5. Each interior router listens for connections from external clients on the following ports:

   - **5672** - Unsecure connections from messaging applications.
   - **5671** - Secure connections from messaging applications.
   - **8080** - AMQ Interconnect web console access. Default user name/password security is applied.

6. Using the AMQ Certificate Manager Operator, the AMQ Interconnect Operator automatically creates two SSL profiles:

   - **inter-router** - The Operator secures the inter-router network with mutual TLS authentication by creating a Certificate Authority (CA) and generating certificates signed by the CA for each interior router.
   - **default** - The Operator creates TLS certificates for messaging applications to connect to the interior routers on port **5671**.

7. The AMQ Interconnect web console is secured with user name/password authentication. The Operator automatically generates the credentials and stores them in the **router-mesh-users** Secret.

3.2. CREATING AN EDGE ROUTER DEPLOYMENT

You can efficiently scale your router network by adding an edge router deployment. Edge routers act as connection concentrators for messaging applications. Each edge router maintains a single uplink connection to an interior router, and messaging applications connect to the edge routers to send and receive messages.

Prerequisites

- The interior router mesh is deployed. For more information, see Section 3.1, “Creating an interior router deployment”.

Procedure

This procedure creates an edge router on each node of the OpenShift Container Platform cluster and connects them to the previously created interior router mesh.

1. Create an **Interconnect** Custom Resource YAML file that describes the edge router deployment.

   Sample edge-routers.yaml file

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: edge-routers
spec:
  deploymentPlan:
    role: edge
    placement: Every (1)
  edgeConnectors: (2)
    - host: router-mesh (3)
      port: 45672 (4)
```

   (1) An edge router Pod will be deployed on each node in the OpenShift Container Platform cluster. This placement helps to balance messaging application traffic across the cluster. The Operator will create a DaemonSet to ensure that the number of Pods scheduled always corresponds to the number of nodes in the cluster.

   (2) Edge connectors define the connections from the edge routers to the interior routers.

   (3) The name of the Service that was created for the interior routers.

   (4) The port on which the interior routers listen for edge connections. The default is **45672**.

2. Create the edge routers described in the YAML file:

```
$ oc apply -f edge-routers.yaml
```

   The Operator deploys an edge router on each node of the OpenShift Container Platform cluster, and connects them to the interior routers.

3. Verify that the edge routers were created and the Pods are running.

Each router runs in a separate Pod. Each edge router connects to any of the previously created interior routers.

```
$ oc get pods
NAME                                    READY   STATUS    RESTARTS   AGE
edge-routers-2jz5j                      1/1     Running   0          33s
edge-routers-fhlxv                      1/1     Running   0          33s
edge-routers-gg2qb                      1/1     Running   0          33s
edge-routers-hj72t                      1/1     Running   0          33s
interconnect-operator-587f94784b-4bzdx  1/1     Running   0          54m
router-mesh-6b48f89bd-588r5             1/1     Running   0          42m
router-mesh-6b48f89bd-bdjc4             1/1     Running   0          42m
router-mesh-6b48f89bd-h6d5r             1/1     Running   0          42m
```

3.3. CREATING AN INTER-CLUSTER ROUTER NETWORK

You can create a router network from routers running in different OpenShift Container Platform clusters. This enables you to connect applications running in separate clusters.

Procedure

This procedure creates router deployments in two different OpenShift Container Platform clusters (**cluster1** and **cluster2**) and connects them together to form an inter-cluster router network. The connection between the router deployments is secured with SSL/TLS mutual authentication.

1. In the first OpenShift Container Platform cluster (**cluster1**), create an **Interconnect** Custom Resource YAML file that describes the interior router deployment.
   This example creates a single interior router with a default configuration.

   Sample cluster1-router-mesh.yaml file

   ```
   apiVersion: interconnectedcloud.github.io/v1alpha1
   kind: Interconnect
   metadata:
     name: cluster1-router-mesh
   spec: {}
   ```

2. Create the router deployment described in the YAML file.

   ```
   $ oc apply -f cluster1-router-mesh.yaml
   ```

   The AMQ Interconnect Operator creates an interior router with a default configuration. It uses the AMQ Certificate Manager Operator to create a Certificate Authority (CA) and generate a certificate signed by the CA.

3. Generate an additional certificate for the router deployment in the second OpenShift Container Platform cluster (**cluster2**).
   The router deployment in **cluster2** requires a certificate issued by the CA of **cluster1**.

   a. Create a **Certificate** Custom Resource YAML file to request a certificate.

      Sample certificate-request.yaml file

      ```
      apiVersion: certmanager.k8s.io/v1alpha1
      ```

```
kind: Certificate
metadata:
  name: cluster2-inter-router-tls
spec:
  commonName: cluster1-router-mesh-myproject.cluster2.openshift.com
  issuerRef:
    name: cluster1-router-mesh-inter-router-ca (1)
  secretName: cluster2-inter-router-tls
---
```

(1) The name of the Issuer that created the inter-router CA for **cluster1**. By default, the name of the Issuer is **<*application-name*>-inter-router-ca**.

b. Create the certificate described in the YAML file.

```
$ oc apply -f certificate-request.yaml
```

c. Extract the certificate that you generated.

```
$ mkdir /tmp/cluster2-inter-router-tls
$ oc extract secret/cluster2-inter-router-tls --to=/tmp/cluster2-inter-router-tls
```

4. Log in to the second OpenShift Container Platform cluster (**cluster2**), and switch to the project where you want to create the second router deployment.

5. In **cluster2**, create a Secret containing the certificate that you generated.

```
$ oc create secret generic cluster2-inter-router-tls --from-file=/tmp/cluster2-inter-router-tls
```

6. In **cluster2**, create an **Interconnect** Custom Resource YAML file to describe the router deployment.

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: cluster2-router-mesh
spec:
  sslProfiles:
  - name: inter-cluster-tls (1)
    credentials: cluster2-inter-router-tls
    caCert: cluster2-inter-router-tls
  interRouterConnectors:
  - host: cluster1-router-mesh-port-55671-myproject.cluster1.openshift.com (2)
    port: 443
    verifyHostname: false
    sslProfile: inter-cluster-tls
```

(1) This SSL Profile defines the certificate needed to connect to the router deployment in **cluster1**.

(2) The URL of the Route for the inter-router listener on **cluster1**.

7. Create the router deployment described in the YAML file.

```
$ oc apply -f cluster2-router-mesh.yaml
```

8. Verify that the routers are connected.
   This example displays the connections from the router in **cluster2** to the router in **cluster1**.

```
$ oc exec cluster2-fb6bc5797-crvb6 -it -- qdstat -c
Connections
  id    host                                                container                role         dir
security                        authentication  tenant

================================================================================
================================================================================
=============================================
  1     cluster1-router-mesh-port-55671-myproject.cluster1.openshift.com:443  cluster1-router-
mesh-54cffd9967-9h4vq  inter-router  out  TLSv1/SSLv3(DHE-RSA-AES256-GCM-SHA384)
x.509
```

CHAPTER 4. CONNECTING CLIENTS TO THE ROUTER NETWORK

After creating a router network, you can connect clients (messaging applications) to it so that they can begin sending and receiving messages.

By default, the AMQ Interconnect Operator creates a Service for the router deployment and configures the following ports for client access:

- **5672** for plain AMQP traffic without authentication
- **5671** for AMQP traffic secured with TLS authentication

To connect clients to the router network, you can do the following:

- If any clients are outside of the OpenShift cluster, expose the ports so that they can connect to the router network.
- Configure your clients to connect to the router network.

4.1. EXPOSING PORTS FOR CLIENTS OUTSIDE OF OPENSHIFT CONTAINER PLATFORM

You expose ports to enable clients outside of the OpenShift Container Platform cluster to connect to the router network.

Procedure

1. Start editing the **Interconnect** Custom Resource YAML file that describes the router deployment for which you want to expose ports.

```
$ oc edit -f router-mesh.yaml
```

2. In the **spec.listeners** section, expose each port that you want clients outside of the cluster to be able to access.
   In this example, port **5671** is exposed. This enables clients outside of the cluster to authenticate with and connect to the router network.

   Sample router-mesh.yaml file

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  listeners:
    - port: 5672
    - authenticatePeer: true
      expose: true
      http: true
      port: 8080
    - port: 5671
```

```
      sslProfile: default
      expose: true
  ...
```

The AMQ Interconnect Operator creates a Route, which clients from outside the cluster can use to connect to the router network.

4.2. AUTHENTICATION FOR CLIENT CONNECTIONS

When you create a router deployment, the AMQ Interconnect Operator uses the AMQ Certificate Manager Operator to create default SSL/TLS certificates for client authentication, and configures port **5671** for SSL encryption.

4.3. CONFIGURING CLIENTS TO CONNECT TO THE ROUTER NETWORK

You can connect messaging clients running in the same OpenShift cluster as the router network, a different cluster, or outside of OpenShift altogether so that they can exchange messages.

Prerequisites

- If the client is outside of the OpenShift Container Platform cluster, a connecting port must be exposed. For more information, see Section 4.1, "Exposing ports for clients outside of OpenShift Container Platform".

Procedure

- To connect a client to the router network, use the following connection URL format:

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

<scheme>

Use one of the following:

- **amqp** - unencrypted TCP from within the same OpenShift cluster
- **amqps** - for secure connections using SSL/TLS authentication
- **amqpws** - AMQP over WebSockets for unencrypted connections from outside the OpenShift cluster

<username>

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

<host>

If the client is in the same OpenShift cluster as the router network, use the OpenShift Service host name. Otherwise, use the host name of the Route.

<port>

If you are connecting to a Route, you must specify the port. To connect on an unsecured connection, use port **80**. Otherwise, to connect on a secured connection, use port **443**.

NOTE

To connect on an unsecured connection (port **80**), the client must use AMQP over WebSockets (**amqpws**).

The following table shows some example connection URLs.

URL	Description
amqp://admin@router-mesh:5672	The client and router network are both in the same OpenShift cluster, so the Service host name is used for the connection URL. In this case, user name/password authentication is implemented, which requires the user name (**admin**) to be provided.
amqps://router-mesh-myproject.mycluster.com: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**.
amqpws://router-mesh-myproject.mycluster.com:80	The client is outside of OpenShift, so the Route host name is used for the connection URL. In this case, no authentication is implemented, which means the client must use the **amqpws** scheme and port **80**.

CHAPTER 5. CONNECTING TO EXTERNAL SERVICES

You can connect a router to an external service such as a message broker. The services may be running in the same OpenShift cluster as the router network, or running outside of OpenShift.

Prerequisites

- You must have access to a message broker.

Procedure

This procedure describes how to connect a router to a broker and configure a link route to connect messaging clients to it.

1. Start editing the **Interconnect** Custom Resource YAML file that describes the router deployment that you want to connect to a broker.

```
$ oc edit -f router-mesh.yaml
```

2. In the **spec** section, configure the connection and link route.

Sample router-mesh.yaml file

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  connectors:  1
  - name: my-broker
    host: broker
    port: 5672
    routeContainer: true
  linkRoutes:  2
  - prefix: q1
    direction: in
    connection: my-broker
  - prefix: q1
    direction: out
    connection: my-broker
```

1. The connection to be used to connect this router to the message broker. The Operator will configure this connection from every router defined in this router deployment to the broker. If you only want a single connection between the router network and the broker, then configure a **listener** instead of a connector and have the broker establish the connection.

2. The link route configuration. It defines the incoming and outgoing links and connection to be used to connect messaging applications to the message broker.

3. Verify that the router has established the link route to the message broker.

```
$ oc exec router-mesh-fb6bc5797-crvb6 -it -- qdstat --linkroutes
```

```
Link Routes
  address  dir  distrib      status
  ====================================
  q1       in   linkBalanced  active
  q1       out  linkBalanced  active
```

Additional resources

- For more information about link routes, see Creating link routes.

CHAPTER 6. CONFIGURING THE ADDRESS SPACE FOR MESSAGE ROUTING

AMQ Interconnect provides flexible application-layer addressing and delivery semantics. By configuring addresses, you can route messages in anycast (closest or balanced) or multicast patterns.

6.1. ROUTING MESSAGES BETWEEN CLIENTS

By default, AMQ Interconnect distributes messages in a balanced anycast pattern (each message is delivered to a single consumer, and AMQ Interconnect attempts to balance the traffic load across the network). This means you only need to change the address configuration if you want to apply non-default semantics to an address or range of addresses.

Procedure

This procedure configures an address to use multicast distribution. The router network will distribute a copy of each message sent to this address to every consumer that is subscribed to the address.

1. Start editing the **Interconnect** Custom Resource YAML file that describes the router deployment.

```
$ oc edit -f router-mesh.yaml
```

2. In the **spec** section, define the semantics to be applied to addresses.

Sample router-mesh.yaml file

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  addresses:
  - pattern: */orders  (1)
    distribution: multicast
```

(1) Messages sent to any address that ends with "**orders**" will be distributed in a multicast pattern.

The Operator applies the changes to the router network and restarts each Pod.

3. If you have additional router deployment Custom Resources that define routers in the router network, repeat this procedure for each CR.
Each router in the router network must have the same address configuration.

Additional resources

- For more information about address semantics that you can configure, see Configuring message routing.

6.2. ROUTING MESSAGES THROUGH BROKERS

If you need to store and forward messages, you can route them through a queue on a message broker. In this scenario, message producers send messages to a router, and the router sends the messages to a broker queue. When a consumer connects to the router to receive the messages, the router retrieves them from the broker queue.

You can route messages to brokers running in the same OpenShift cluster as the router network, or to brokers that are running outside of the cluster.

Prerequisites

- You must have access to a message broker.

Procedure

1. Start editing the Interconnect Custom Resource YAML file that describes the router deployment.

```
$ oc edit -f router-mesh.yaml
```

2. In the **spec** section, add a connector to connect to the broker, a waypoint address to point to the broker queue, and autolinks to create the links to the queue.

Sample router-mesh.yaml file

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  addresses:
  - prefix: my-queue (1)
    waypoint: true
  autoLinks: (2)
  - prefix: my-queue
    direction: in
    connection: my-broker
  - prefix: my-queue
    direction: out
    connection: my-broker
  connectors: (3)
  - name: my-broker
    host: broker
    port: 5672
    routeContainer: true
```

(1) The address (or set of addresses) for which messages should be stored on a broker queue.

(2) The autolink configuration. It defines the incoming and outgoing links and connection to be used to send and receive the messages on the broker.

(3) The connection to be used to connect the routers to the message broker.

The Operator applies the changes to the router network and restarts each Pod.

3. Verify that the router has established the autolinks to the message broker.

```
$ oc exec router-mesh-6d6dccb57f-x5cqf -it -- qdstat --autolinks
AutoLinks
  addr      dir  phs  extAddr  link  status  lastErr
  ================================================
  my-queue  in   1            26    active
  my-queue  out  0            27    active
```

4. If you have additional router deployment Custom Resources that define routers in the router network, repeat this procedure for each CR.
   Each router in the router network must have the same address configuration.

Additional resources

- For more information about routing messages to and from broker queues, see Routing Messages through broker queues.

CHAPTER 7. USING PROMETHEUS AND GRAFANA TO MONITOR THE ROUTER NETWORK

Prometheus is container-native software built for storing historical data and for monitoring large, scalable systems such as AMQ Interconnect. It gathers data over an extended time, rather than just for the currently running session.

You use Prometheus and Alertmanager to monitor and store AMQ Interconnect data so that you can use a graphical tool, such as Grafana, to visualize and run queries on the data.

7.1. SETTING UP PROMETHEUS AND GRAFANA

Before you can view AMQ Interconnect dashboards, you must deploy and configure Prometheus, Alertmanager, and Grafana in the OpenShift project in which AMQ Interconnect is deployed. All of the required configuration files are provided in a GitHub repository.

Procedure

1. Clone the **qdr-monitoring** GitHub repository.
   This repository contains the configuration files needed to set up Prometheus and Grafana to monitor AMQ Interconnect.

   ```
   $ git clone https://github.com/interconnectedcloud/qdr-monitoring
   ```

2. Open the **deploy-monitoring.sh** script and set the **NAMESPACE** variable.
   Set **NAMESPACE** to be the name of the project into which you have deployed AMQ Interconnect.

   ```
   #!/bin/bash

   # Change the namespace to that of your project
   NAMESPACE=myproject
   ...
   ```

3. Run the **deploy-monitoring.sh** script.
   This script creates and configures the OpenShift resources needed to deploy Prometheus, Alertmanager, and Grafana in your OpenShift project. It also configures two dashboards that provide metrics for the router network.

   ```
   $ ./deploy-monitoring.sh
   ```

4. Create a Route for the prometheus, alertmanager, and grafana Services.

   ```
   $ oc expose service prometheus

   $ oc expose service alertmanager

   $ oc expose service grafana
   ```

Additional resources

- For more information about Prometheus, see the Prometheus documentation.

- For more information about Grafana, see the Grafana documentation.

7.2. VIEWING AMQ INTERCONNECT DASHBOARDS IN GRAFANA

After setting up Prometheus and Grafana, you can visualize the AMQ Interconnect data on the following Grafana dashboards:

Qpid Dispatch Router

Shows metrics for:

- **Deliveries ingress**
- **Deliveries egress**
- **Deliveries ingress route container**
- **Deliveries egress route container**
- **Deliveries redirected to fallback destination**
- **Dropped presettled deliveries**
- **Presettled deliveries**
- **Auto links**
- **Link routes**
- **Address count**
- **Connection count**
- **Link count**

Qpid Dispatch Router - Delayed Deliveries

Shows metrics for:

- **Cumulative delayed 10 seconds**
- **Cumulative delayed 1 second**
- **Rate of new delayed deliveries**

Procedure

1. In the OpenShift web console, switch to **Networking → Routes**, and click the URL for the **grafana** Route.
   The Grafana Log In page appears.
2. Enter your user name and password, and then click **Log In**.
   The default Grafana user name and password are both **admin**. After logging in for the first time, you can change the password.
3. On the top header, click the dashboard drop-down menu, and then select the **Qpid Dispatch Router** or **Qpid Dispatch Router - Delayed Deliveries** dashboard.

Figure 7.1. Delayed Deliveries dashboard

CHAPTER 8. USING THE AMQ INTERCONNECT WEB CONSOLE TO MONITOR THE ROUTER NETWORK

You can use the AMQ Interconnect web console to monitor the status and performance of your router network. By default, when you create a router deployment, the AMQ Interconnect Operator generates the credentials to access the console and stores them in a Secret.

Procedure

1. In OpenShift, switch to **Networking → Routes**, and click the console Route.
   The web console opens in a new tab.

2. To connect to the web console, complete the following fields:

   Port

   Enter **443**.

   User name

   Enter the user name.
   To find the user name and password for accessing the web console, navigate to **Workloads → Secrets**. The Secret containing the web console credentials is called ***<application-name>*-users** (for example, **router-mesh-users**).

   The syntax for the user name is <*user*>@<*domain*> (the domain is the OpenShift application name, which is the name of the Custom Resource that describes the router deployment). For example, **guest@router-mesh**.

   Password

   Enter the password defined in the ***<application-name>*-users** Secret.

3. Click **Connect**.
   The **Routers** page is displayed showing all of the routers in the router network.

4. Use the web console tabs to monitor the router network.

This tab...	Provides...
Overview	Aggregated information about routers, addresses, links, connections, and logs.
Entities	Detailed information about each AMQP management entity for each router in the router network. Some of the attributes have charts that you can add to the **Charts** tab.
Topology	A graphical view of the router network, including routers, clients, and brokers. The topology shows how the routers are connected, and how messages are flowing through the network.
Charts	Graphs of the information selected on the **Entities** tab.
Message Flow	A chord diagram showing the real-time message flow by address.

This tab...	Provides...
Schema	The management schema that controls each of the routers in the router network.

Revised on 2020-06-16 12:30:57 UTC