Red Hat AMQ 2021.Q2

Creating a service network with OpenShift

For Use with AMQ Interconnect 2.0 TECHNOLOGY PREVIEW

Last Updated: 2021-08-16

Red Hat AMQ 2021.Q2 Creating a service network with OpenShift

For Use with AMQ Interconnect 2.0 TECHNOLOGY PREVIEW

Legal Notice

Abstract

This tutorial describes how to create AMQ Interconnect sites on OpenShift to build a service network.

Table of Contents

PREFACE

Making open source more inclusive

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

IMPORTANT

AMQ Interconnect 2.0 Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in 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 about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

CHAPTER 1. CREATING A SERVICE NETWORK WITH OPENSHIFT

This tutorial demonstrates how to connect a frontend service on a OpenShift cluster with a backend service on a OpenShift cluster using the **skupper** command-line interface (CLI).

Prerequisites

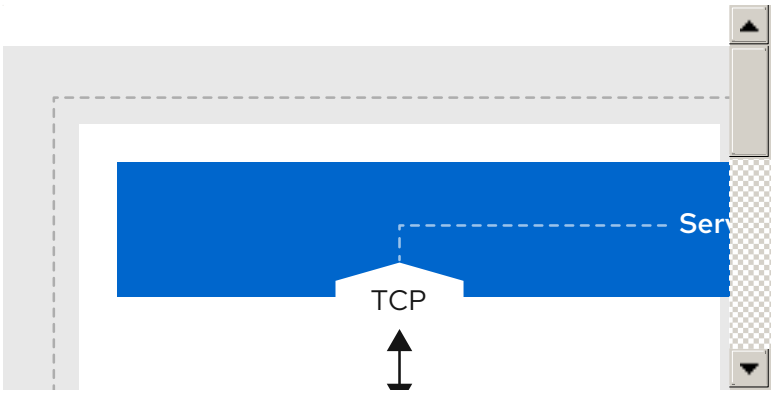
- Access to projects in two OpenShift clusters, **cluster-admin** access is not required.
- One of the OpenShift clusters must be addressable from the other cluster.

This tutorial shows how to connect the following namespaces:

- **west** - runs the frontend service and is typically a public cluster.
- **east** - runs the backend service.

1.1. INTRODUCTION TO AMQ INTERCONNECT 2.0

Interconnect 2.0 introduces a service network, linking services across the hybrid cloud. A service network enables communication between services running in different network locations. It allows geographically distributed services to connect as if they were all running in the same site.



For example, you can deploy your frontend in a public OpenShift cluster and deploy your backend in a private OpenShift cluster, then connect them into a service network.

A service network provides the following features:

- Security by default. All inter-site traffic is protected by mutual TLS using a private, dedicated certificate authority (CA).
- Easy connections between OpenShift clusters, even private clusters.
- A service network supports existing TCP-based applications without requiring modification.
- Monitor your application traffic spread across multiple OpenShift clusters using the service network console.

You deploy and manage a service network using the **skupper** CLI.

1.2. INSTALLING THE SKUPPER CLI

Installing the **skupper** command-line interface (CLI) provides a simple method to get started with AMQ Interconnect.

Procedure

1. Ensure your subscription has been activated and your system is registered.
2. Subscribe to the required repositories:

   Red Hat Enterprise Linux 7

   ```
   $ sudo subscription-manager repos --enable=interconnect-2-for-rhel-7-server-rpms
   ```

   Red Hat Enterprise Linux 8

   ```
   $ sudo subscription-manager repos --enable=interconnect-2-for-rhel-8-x86_64-rpms
   ```

3. Use the **yum** or **dnf** command to install the **skupper** package:

   ```
   $ sudo yum install skupper
   ```

4. Verify the installation.

   ```
   $ skupper version
   client version 0.7.0-redhat-interconnect-2.0.0
   ```

1.3. CONFIGURING TERMINAL SESSIONS

This procedure describes how to configure your terminal sessions to use configurations to avoid problems as you configure AMQ Interconnect on different clusters.

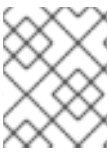
The following table shows how you might set up your terminal sessions.

Table 1.1. Terminal sessions

west terminal session	east terminal session
`$ oc project west`	`$ oc project east`

Prerequisites

- The OpenShift CLI is installed. See the OpenShift CLI documentation for more instructions on how to install **oc**.



NOTE

In OpenShift 4.6 and later, you can use the web terminal to perform the following procedure, as described in the web terminal documentation.

Procedure

1. Start a terminal session to work on the **west** namespace and set the **KUBECONFIG** environment variable:

```
$ export KUBECONFIG=$HOME/.kube/config-west
```

   This session is referred to later as the *west* terminal session.

2. Start a terminal session to work on the **east** namespace and set the **KUBECONFIG** environment variable:

```
$ export KUBECONFIG=$HOME/.kube/config-east
```

   This session is referred to later as the *east* terminal session.

3. In each terminal session, log into the OpenShift cluster, for example:

```
$ oc login
```

1.4. INSTALLING THE SERVICE NETWORK ROUTER IN BOTH CLUSTERS

1. In the west terminal session:
   a. Create the **west** project (namespace):

```
$ oc new-project west
```

   b. Create the service network router:

```
$ skupper init
```

   c. Check the site status:

```
$ skupper status
```

   The output should be similar to the following:

```
Skupper enabled for namespace 'west'. It is not connected to any other sites.
```

2. In the east terminal session:
   a. Create the **east** project (namespace):

```
$ oc new-project east
```

   b. Create the service network router:

```
$ skupper init
```

   c. Check the site status:

```
$ skupper status
```

The output should be similar to the following:

```
Skupper enabled for namespace 'east'. It is not connected to any other sites.
```

1.5. CONNECTING NAMESPACES TO CREATE A SERVICE NETWORK

With the service network routers installed, you can connect them together securely and allow service sharing across the service network.

Procedure

1. In the west terminal session, create a connection token to allow connection to the west namespace:

```
$ skupper token create $HOME/secret.yaml
```

This command creates the **secret.yaml** file in your home directory, which you can use to create the secure connection.

2. In the east terminal session, use the token to create a connection to the west namespace:

```
$ skupper link create $HOME/secret.yaml
```

3. Check the site status from the west terminal session:

```
$ skupper status
```

The output should be similar to the following:

```
Skupper is enabled for namespace "west" in interior mode. It is connected to 1 other site. It has no exposed services.
The site console url is:  https://<skupper-url>
The credentials for internal console-auth mode are held in secret: 'skupper-console-users'
```

1.6. CREATING THE FRONTEND SERVICE

The frontend service is a simple Python application that displays a message from the backend application.

Procedure

Perform all tasks in the west terminal session:

1. Deploy the frontend service:

```
$ oc create deployment hello-world-frontend --image quay.io/skupper/hello-world-frontend
```

2. Expose the frontend deployment as a cluster service:

```
$ oc expose deployment hello-world-frontend --port 8080 --type LoadBalancer
```

3. Create a route for the frontend:

```
$ oc expose svc/hello-world-frontend
```

4. Check the frontend route:

   a. Get the route details:

   ```
   $  oc get routes
   ```

   The output should be similar to the following:

   ```
   NAME                HOST/PORT
   hello-world-frontend   <frontend-url>
   ```

   b. Navigate to the **<frontend-url>** value in your browser, you see a message similar to the following because the frontend cannot communicate with the backend yet:

   ```
   Trouble! HTTPConnectionPool(host='hello-world-backend', port=8080): Max retries exceeded with url: /api/hello (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fbfcdf0d1d0>: Failed to establish a new connection: [Errno -2] Name or service not known'))
   ```

   To resolve this situation, you must create the backend service and make it available on the service network.

1.7. CREATING THE BACKEND SERVICE AND MAKING IT AVAILABLE ON THE SERVICE NETWORK

The backend service runs in the **east** namespace and is not available on the service network by default. You use the **skupper** command to expose the service to all namespaces on the service network. The backend app is a simple Python application that passes a message to the frontend application.

Procedure

1. Deploy the backend service in the east terminal session:

```
$ oc create deployment hello-world-backend --image quay.io/skupper/hello-world-backend
```

2. Expose the backend service on the service network from the east terminal session:

```
$ skupper expose deployment hello-world-backend --port 8080 --protocol tcp
```

3. Check the site status from the west terminal session:

```
$ skupper status
```

The output should be similar to the following:

```
Skupper is enabled for namespace "west" in interior mode. It is connected to 1 other site. It has 1 exposed service.
```

The service is exposed from the **east** namespace.

4. Check the frontend route in the west terminal session:

   a. Get the route details:

   ```
   $ oc get routes
   ```

   The output should be similar to the following:

   ```
   NAME                HOST/PORT
   hello-world-frontend   <frontend-url>
   ```

   b. Navigate to the **<frontend-url>** value in your browser, you see a message similar to the following:

   ```
   I am the frontend.  The backend says 'Hello from hello-world-backend-78cd4d7d8c-plrr9
   (1)'.
   ```

This shows how the frontend calls the backend service over the service network from a different OpenShift cluster.

Additional resources

- Monitoring AMQ Interconnect sites using the console
- Configuring AMQ Interconnect sites using the CLI

1.8. TEARING DOWN THE SERVICE NETWORK

This procedure describes how to remove the service network you created.

1. Delete the **west** namespace from the west terminal session:

   ```
   $ oc delete project west
   ```

2. Delete the **east** namespace from the east terminal session:

   ```
   $ oc delete project east
   ```

Revised on 2021-08-16 14:30:04 UTC