



Red Hat Application Interconnect 1.0

**Creating a service network with OpenShift and
accessing a backend service using a gateway**

For Use with Application Interconnect 1.0 LIMITED AVAILABILITY

Red Hat Application Interconnect 1.0 Creating a service network with OpenShift and accessing a backend service using a gateway

For Use with Application Interconnect 1.0 LIMITED AVAILABILITY

Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

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

Table of Contents

| | |
|---|----|
| PREFACE | 3 |
| CHAPTER 1. INTRODUCTION TO APPLICATION INTERCONNECT 1.0 | 4 |
| CHAPTER 2. INSTALLING THE SKUPPER CLI | 5 |
| CHAPTER 3. CREATING A BACKEND SERVICE | 6 |
| CHAPTER 4. LOGGING INTO CLUSTER | 7 |
| CHAPTER 5. CREATING A SKUPPER SITE | 8 |
| CHAPTER 6. CREATING THE FRONTEND SERVICE | 9 |
| CHAPTER 7. CREATING AND USING A SKUPPER GATEWAY | 10 |
| CHAPTER 8. CHECKING SERVICE ACCESS FROM THE FRONTEND | 11 |
| CHAPTER 9. TEARING DOWN THE SERVICE NETWORK | 12 |

PREFACE

This tutorial describes how to connect a local backend service on a local machine with a frontend service running on a OpenShift cluster.

In this tutorial, the services are the same as used in [Creating a service network with OpenShift](#), however you run the backend service locally and expose the service on the service network using the **skupper** command-line interface (CLI).

Prerequisites

- Access to a projects in a OpenShift cluster, **cluster-admin** access is not required.
- Python on your local machine.

This tutorial shows how to connect the following:

- **west** - a namespace in an accessible OpenShift cluster running the frontend service.
- **hello-world-backend** - a Python service running on a local machine.



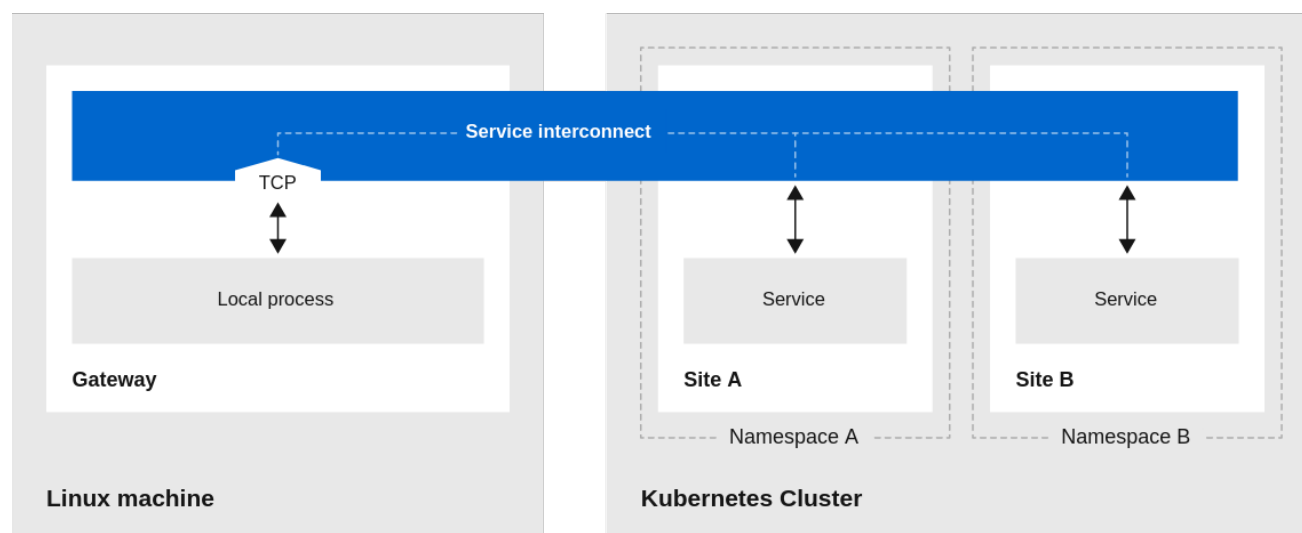
NOTE

Although this tutorial demonstrates exposing a Python service on the service network, a more typical use case would involve a database service, for example, MySQL.

CHAPTER 1. INTRODUCTION TO APPLICATION INTERCONNECT 1.0

Application Interconnect 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.



190_AINQ_002

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

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

Additional resources

- [Introduction to Application Interconnect](#)

CHAPTER 2. INSTALLING THE SKUPPER CLI

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

Procedure

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

Red Hat Enterprise Linux 8

```
$ sudo subscription-manager repos --enable=application-interconnect-1-for-rhel-8-x86_64-rpms
```

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

```
$ sudo yum install skupper-cli
```

4. Verify the installation.

```
$ skupper version
client version 1.0.2-redhat-1
```

CHAPTER 3. CREATING A BACKEND SERVICE

This procedure describes how to create a backend service on your local machine that is accessed from the service network.

Prerequisites

- Python

Procedure

1. Clone the [skupper-example-hello-world](#) repo.
2. Change to the service directory.

```
$ cd skupper-example-hello-world/backend/
```

3. Install the required libraries.

```
$ pip install --user flask starlette uvicorn
```

4. Run the backend service

```
$ python ./main.py
```

The output is similar to the following:

```
INFO: Started server process [107836]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
```

5. Test the service by navigating to the following URL.

```
http://localhost:8080/api/hello
```

The output is similar to the following:

```
Hello from workstation (1)
```

This indicates that the backend service is running and available.

CHAPTER 4. LOGGING INTO CLUSTER

Prerequisites

- The `kubectl` CLI is installed.
The **kubectl** CLI is automatically installed when you install **oc** as described in the [OpenShift CLI](#) 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. Log into the OpenShift cluster.

CHAPTER 5. CREATING A SKUPPER SITE

1. Create the **west** namespace:

```
$ kubectl create namespace west  
$ kubectl config set-context --current --namespace west
```

2. Create the service network site:

```
$ skupper init
```

3. 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.
```

CHAPTER 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:

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

2. Expose the frontend deployment as a cluster service:

```
$ kubectl 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 0x7fbcdf0d1d0>:
Failed to establish a new connection: [Errno -2] Name or service not known'))
```

To resolve this situation, you must make the backend service available on the service network using a gateway.

CHAPTER 7. CREATING AND USING A SKUPPER GATEWAY

This procedure describes how to create a gateway and make a backend service available on the service network.

Prerequisites

- Skupper router is installed on local machine

Procedure

1. Create a gateway:

```
$ skupper gateway init --type podman
```

2. Create a Application Interconnect service:

```
$ skupper service create hello-world-backend 8080
```

3. Bind the local backend service to the Application Interconnect service:

```
$ skupper gateway bind hello-world-backend <backend-ip-address> 8080
```

where the <backend-ip-address> is the address you noted in [Chapter 3, Creating a backend service](#)

4. Check the gateway status:

```
$ skupper gateway status
```

The output should be similar to following:

```
Gateway Definitions:
```

```
└─ <machine>-<user> type: podman version: 1.17.1
```

```
└─ Bindings:
```

```
    └─ hello-world-backend:8080 tcp hello-world-backend:8080 <backend-ip-address> 8080
```

CHAPTER 8. CHECKING SERVICE ACCESS FROM THE FRONTEND

Procedure

1. Get the URL details:

```
$ kubectl get service hello-world-frontend -o jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

Use the output IP address to construct the <frontend-url>:

```
<cluster-ip-address>:8080/
```

2. Navigate to the <frontend-url> value in your browser and click **Say hello**. You see a message similar to the following:

```
Hi, <name>. I am Mathematical Machine (backend-77f8f45fc8-mnrdp).
```

If you click **Say hello** again, a different backend process responds showing how Application Interconnect balances the requests.

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

Additional resources

- [Using the Skupper console](#)
- [Configuring Application Interconnect sites using the CLI](#)

CHAPTER 9. TEARING DOWN THE SERVICE NETWORK

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

1. Delete the gateway:

```
$ skupper gateway delete
```

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

```
$ kubectl delete namespace west
```

Revised on 2022-06-24 16:01:30 UTC