



# Red Hat Application Interconnect 1.0

## OpenShift を使用したサービスネットワークの作成

Red Hat Application Interconnect 1.0 で使用する場合 (限定利用)



# Red Hat Application Interconnect 1.0 OpenShift を使用したサービスネットワークの作成

---

Red Hat Application Interconnect 1.0 で使用する場合 (限定利用)

## 法律上の通知

Copyright © 2023 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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> 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<sup>®</sup> 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.

## 概要

このチュートリアルでは、OpenShiftでApplication Interconnectサイトを作成してサービスネットワークを構築する方法を説明します。

---

## 目次

はじめに .....	3
第1章 SKUPPER CLI のインストール .....	4
第2章 ターミナルセッションの設定 .....	5
第3章 両方のクラスターにサービスネットワークルーターをインストールする .....	6
第4章 名前空間を接続してサービスネットワークを作成する .....	7
第5章 フロントエンドサービスの作成 .....	8
第6章 バックエンドサービスを作成し、サービスネットワークで利用できるようにする .....	9
第7章 サービスネットワークの切断 .....	11



## はじめに

このチュートリアルでは、**skupper** コマンドラインインターフェイス (CLI) を使用して、OpenShift クラスタにバックエンドサービスがある OpenShift クラスタでフロントエンドサービスを接続する方法を説明します。

Application Interconnect の概要は、[Application Interconnect の概要](#) を参照してください。

### 前提条件

- 2つの OpenShift クラスタのプロジェクトに対するアクセス権。**cluster-admin** アクセスは必要ありません。
- OpenShift クラスタの1つは、他のクラスタからアドレス可能である必要があります。
- **kubectl** および **oc** CLI。**oc** を使用して多くのコマンドを実行できますが、このチュートリアルでは **kubectl** オプションを示します。

このチュートリアルでは、次の namespace を接続する方法を示します。

- **west** - フロントエンドサービスを実行します。これは通常パブリッククラスタです。
- **east** - バックエンドサービスを実行します。

## 第1章 SKUPPER CLI のインストール

**skupper** コマンドラインインターフェイス (CLI) をインストールすると、Application Interconnect を簡単に使用開始できます。

### 手順

1. サブスクリプションがアクティベートされ、システムが登録されていることを確認します。
2. 必要なリポジトリにサブスクライブします。

#### Red Hat Enterprise Linux 8

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

3. **yum** コマンドまたは **dnf** コマンドを使用して、**skupper** パッケージをインストールします。

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

4. インストールを確認します。

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



## 第2章 ターミナルセッションの設定

この手順では、異なるクラスターで Application Interconnect を設定する際の問題を回避するために、設定を使用するようにターミナルセッションを設定する方法を説明します。

次の表は、ターミナルセッションを設定する方法を示しています。

表2.1ターミナルセッション

west ターミナルセッション	east ターミナルセッション
<pre>\$ kubectl get pods</pre>	<pre>\$ kubectl get pods</pre>

### 前提条件

- OpenShift CLI がインストールされている。**oc** のインストール方法については、[OpenShift CLI](#) のドキュメントを参照してください。



### 注記

OpenShift 4.6 以降では、[Web ターミナル](#) のドキュメントで説明されているように、Web ターミナルを使用して次の手順を実行できます。

### 手順

1. **west** namespace で作業するように端末セッションを開始し、**KUBECONFIG** 環境変数を設定します。

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

このセッションは、後で **west** ターミナルセッションと呼ばれます。

2. **east** namespace で作業するように端末セッションを開始し、**KUBECONFIG** 環境変数を設定します。

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

このセッションは、後で **east** ターミナルセッションと呼ばれます。

3. 各ターミナルセッションで、OpenShift クラスターにログインします。

## 第3章 両方のクラスターにサービスネットワークルーターをインストールする

1. west ターミナルセッションで、以下を行います。

a. **west** プロジェクト (namespace) を作成します。

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

b. サービスネットワークルーターを作成します。

```
$ skupper init
```

c. サイトのステータスを確認します。

```
$ skupper status
```

出力は以下のようになります。

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

2. east ターミナルセッションで、以下を行います。

a. **east** プロジェクト (namespace) を作成します。

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

b. サービスネットワークルーターを作成します。

```
$ skupper init
```

c. サイトのステータスを確認します。

```
$ skupper status
```

出力は以下のようになります。

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

## 第4章 名前空間を接続してサービスネットワークを作成する

サービスネットワークルーターをインストールすると、ルーターを安全に接続し、サービスネットワーク全体でサービスを共有できるようになります。

### 手順

1. west ターミナルセッションで、west namespace への接続を許可する接続トークンを作成します。

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

このコマンドは、ホームディレクトリーに **secret.yaml** ファイルを作成し、セキュアな接続の作成に使用できます。

2. east ターミナルセッションで、トークンを使用して west namespace への接続を作成します。

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

3. west ターミナルセッションからサイトのステータスを確認します。

```
$ skupper status
```

出力は以下のようになります。

```
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'
```

## 第5章 フロントエンドサービスの作成

フロントエンドサービスは、バックエンドアプリケーションのメッセージを表示する簡易な Python アプリケーションです。

### 手順

west ターミナルセッションですべてのタスクを実行します。

1. フロントエンドサービスをデプロイします。

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

2. フロントエンドデプロイメントをクラスターサービスとして公開します。

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

3. フロントエンドのルートを作成します。

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

4. フロントエンドルートを確認します。

- a. ルートの詳細を取得します。

```
$ oc get routes
```

出力は以下のようになります。

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

- b. ブラウザーで **<frontend-url>** の値に移動すると、フロントエンドがバックエンドと通信できないため、以下のようなメッセージが表示されます。

```
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'))
```

この状況を解決するには、バックエンドサービスを作成し、サービスネットワークで利用できるようにする必要があります。

## 第6章 バックエンドサービスを作成し、サービスネットワークで利用できるようにする

バックエンドサービスは **east** namespace で実行し、デフォルトではサービスネットワークで利用できません。**skupper** コマンドを使用して、サービスネットワークのすべての namespace にサービスを公開します。バックエンドアプリケーションは、フロントエンドアプリケーションにメッセージを渡す簡易な Python アプリケーションです。

### 手順

1. east ターミナルセッションにバックエンドサービスをデプロイします。

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

2. east ターミナルセッションからサービスネットワーク上のバックエンドサービスを公開します。

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

3. west ターミナルセッションからサイトのステータスを確認します。

```
$ skupper status
```

出力は以下のようになります。

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

サービスは **east** namespace から公開されます。

4. west ターミナルセッションにフロントエンドルートを確認します。

- a. ルートの詳細を取得します。

```
$ oc get routes
```

出力は以下のようになります。

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

- b. ブラウザーで **<frontend-url>** の値に移動すると、以下のようなメッセージが表示されません。

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

もう一度 **Say hello** をクリックすると、別のバックエンドプロセスが応答し、Application Interconnect が要求のバランスをとる方法を示します。

これは、フロントエンドが別の OpenShift クラスターからサービスネットワークを介してバックエンドサービスを呼び出す方法を示しています。

## 関連情報

- [Skupper コンソールの使用](#)
- [CLI を使用したアプリケーション Interconnect サイトの設定](#)

## 第7章 サービスネットワークの切断

この手順では、作成したサービスネットワークを削除する方法を説明します。

1. west 端末セッションから **west** namespace を削除します。

```
$ kubectl delete namespace west
```

2. east 端末セッションから **east** namespace を削除します。

```
$ kubectl delete namespace east
```

改訂日時: 2022-10-22 20:54:42 +1000