



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およびゲートウェイを使用したバックエンドサービスへのアクセス

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およびゲートウェイを使用したバックエンドサービスへのアクセス

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[®] 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.

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 APPLICATION INTERCONNECT 1.0 の概要	4
第2章 SKUPPER CLI のインストール	5
第3章 バックエンドサービスの作成	6
第4章 クラスターへのログイン	7
第5章 SKUPPER サイトの作成	8
第6章 フロントエンドサービスの作成	9
第7章 SKUPPER ゲートウェイの作成および使用	10
第8章 フロントエンドからのサービスアクセスの確認	11
第9章 サービスネットワークの切断	12

はじめに

このチュートリアルでは、ローカルマシン上のローカルバックエンドサービスを OpenShift クラスターで実行しているフロントエンドサービスに接続する方法を説明します。

このチュートリアルでは、サービスは [OpenShift を使用したサービスネットワークの作成](#) で使用したものと同じですが、バックエンドサービスをローカルで実行し、**skupper** コマンドラインインターフェイス (CLI) を使用してサービスネットワーク上でサービスを公開します。

前提条件

- OpenShift クラスター内のプロジェクトへのアクセス。**cluster-admin** アクセスは必要ありません。
- ローカルマシン上の Python。

このチュートリアルでは、以下の接続方法を示します。

- **west** - frontend サービスを実行しているアクセス可能な OpenShift クラスターの namespace。
- **hello-world-backend** - ローカルマシンで実行している Python サービス。



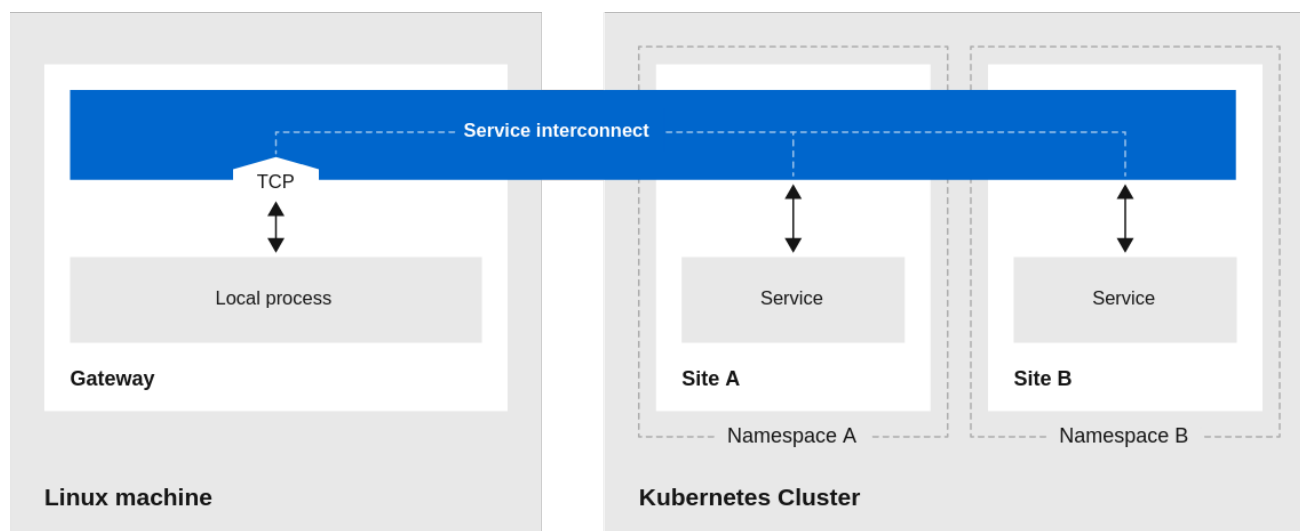
注記

このチュートリアルでは、サービスネットワーク上で Python サービスを公開する方法を示していますが、より一般的なユースケースには、MySQL などのデータベースサービスが含まれます。

第1章 APPLICATION INTERCONNECT 1.0 の概要

Application Interconnect は、ハイブリッドクラウド全体でサービスをリンクするサービスネットワークを導入します。

サービスネットワークにより、異なるネットワークロケーションで実行しているサービス間の通信が有効になります。これにより、地理的に分散したサービスを、すべて同じサイトで実行しているかのように接続できます。



190_AINQ_002

たとえば、フロントエンドをパブリック OpenShift クラスターにデプロイして、バックエンドをローカルネットワークにデプロイしてから、サービスネットワークに接続することができます。

skupper CLI を使用して、ゲートウェイを含むサービスネットワークをデプロイし、管理します。

関連情報

- [Application Interconnect の概要](#)

第2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3章 バックエンドサービスの作成

この手順では、サービスネットワークからアクセスするローカルマシンにバックエンドサービスを作成する方法を説明します。

前提条件

- Python

手順

1. `skupper-example-hello-world` リポジトリのクローンを作成します。
2. サービスディレクトリーに移動します。

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

3. 必要なライブラリーをインストールします。

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

4. バックエンドサービスを実行します。

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

出力は以下の例のようになります。

```
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. 次の URL に移動して、サービスをテストします。

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

出力は以下の例のようになります。

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

これは、バックエンドサービスが実行中であり、利用可能であることを示しています。

第4章 クラスターへのログイン

前提条件

- kubectl CLI がインストールされている。
[OpenShift CLI](#) のドキュメントで説明されているように、**oc** をインストールする時に、**kubectl** CLI が自動的にインストールされます。

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. OpenShift クラスターにログインします。

第5章 SKUPPER サイトの作成

1. **west** namespace を作成します。

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

2. サービスネットワークサイトを作成します。

```
$ skupper init
```

3. 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.
```

第6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。

```
$ oc 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 0x7bfcd0d1d0>: Failed to establish a new connection: [Errno -2] Name or service not known'))
```

この状況を解決するには、ゲートウェイを使用してサービスネットワークでバックエンドサービスを利用できるようにする必要があります。

第7章 SKUPPER ゲートウェイの作成および使用

この手順では、ゲートウェイを作成し、サービスネットワークでバックエンドサービスを利用できるようにする方法を説明します。

前提条件

- Skupper ルーターがローカルマシンにインストールされている。

手順

1. ゲートウェイを作成します。

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

2. アプリケーション相互接続サービスを作成します。

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

3. ローカルバックエンドサービスをアプリケーションインターコネクトサービスにバインドします。

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

ここで、<backend-ip-address> は、[3章バックエンドサービスの作成](#)に記載されているアドレスになります。

4. ゲートウェイのステータスを確認します。

```
$ skupper gateway status
```

出力は以下のようになります。

```
Gateway Definitions:
```

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

```
└─ Bindings:
```

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

第8章 フロントエンドからのサービスアクセスの確認

手順

1. URL の詳細を取得します。

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

出力 IP アドレスを使用して <frontend-url> を作成します。

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

2. ブラウザーで <frontend-url> 値に移動し、**Say hello** をクリックします。次のようなメッセージが表示されます。

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

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 OpenShift クラスターからサービスネットワーク経由でバックエンド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9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. ゲートウェイを削除します。

```
$ skupper gateway delete
```

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

```
$ kubectl delete namespace west
```

改訂日時: 2022-07-03 00:30:43 +1000