



Red Hat Application Interconnect 1.0

Application Interconnect CLI の使用

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

Red Hat Application Interconnect 1.0 Application Interconnect CLI の使用

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

このガイドでは、Application Interconnect サイトをインストール、設定、および管理して、サービスネットワークを構築する方法について説明します。

目次

はじめに	3
第1章 SKUPPER CLI のインストール	4
第2章 CLI を使ったサイトの作成	5
第3章 カスタムサイト	6
第4章 サイトをリンクする	7
第5章 NAMESPACE からのサービスネットワーク上のサービスの公開	9
5.1. サービスネットワーク上での簡単なサービスの公開	9
5.2. サービスネットワーク上での複雑なサービスの公開	10
第6章 ローカルマシンからのサービスネットワーク上のサービスの公開	12
6.1. サービスネットワークにシンプルなローカルサービスの公開	12
6.2. サービスネットワーク上で複雑なローカルサービスの使用	13
6.3. ゲートウェイを作成し、別のマシンでそれを適用する	14
6.4. ゲートウェイ YAML リファレンス	16
第7章 サービスネットワークの検証	19
第8章 サービスネットワークの保護	21
8.1. ネットワークポリシーを使用したサービスへのアクセスの制限	21
8.2. サービスネットワーク上の HTTP2 トラフィックへの TLS の適用	21
第9章 サポートされる標準およびプロトコル	23
第10章 異なるクラスターを操作するための CLI オプション	24
第11章 SKUPPER トークンの使用	25
11.1. クレームトークンの作成	25
11.2. CERT トークンの作成	26
11.3. サイトへのアクセスの取り消し	26

はじめに

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。



注記

この限定利用リリースは、すべてのお客様が利用できるわけではありません。
Application Interconnect の詳細は、[Red Hat の営業チーム](#) にお問い合わせください。

skupper コマンドラインインターフェイス (CLI) を使用すると、現在の namespace のコンテキストから Application Interconnect サイトを作成および管理できます。

一般的なワークフローは、サイトを作成し、サイト同士をリンクして、サービスをサービスネットワークに公開することです。

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 CLI を使ったサイトの作成

サービスネットワークは、アプリケーションインターコネクトサイトで設定されます。このセクションでは、デフォルト設定を使用してサイトを作成する方法を説明します。

前提条件

- **skupper** CLI がインストールされている。
- クラスターにログインしている。
- サービスネットワーク上で公開したいサービスがアクティブな namespace にある。

手順

1. デフォルトのサイトを作成します。

```
$ skupper init
```

2. サイトを確認します。

```
$ skupper status
```

```
Skupper is enabled for namespace "west" in interior mode. It is not connected to any other sites.
```



注記

上記のデフォルトメッセージは、ポリシーシステムがインストールされていないクラスター上のサイトを初期化するときに表示されます。[Securing a service network using policies](#) で説明されているようにポリシーシステムをインストールした場合、メッセージは **Skupper is enabled for namespace "west" in interior mode (with policies)** となります。

デフォルトのサイト設定は以下の通りです。

- console - Skupper コンソールは、1人のユーザーでプロビジョニングされています。**admin** ユーザーのパスワードは **skupper-console-users** シークレットに保存されます。コンソールの詳細は、[Using the Skupper console](#) を参照してください。
- サイト名 - サイト名はデフォルトで namespace 名に設定されます (例: **west**)。
- ingress - 作成するサイトへのリンクをサポートするために、ingress が作成されます。デフォルトでは、ルートが作成されます。他のサイトをこのサイトにリンクさせたくない場合は、**--ingress none** を指定できます。その他の ingress オプションはサポートされていません。

第3章 カスタムサイト

デフォルトの **skupper init** では、一般的な要件を満たすサイトを作成します。

カスタム設定が必要な場合は、以下のオプションに注意してください。

- コンソールなしでサイトを作成します。

```
$ skupper init --enable-console false
```

- コンソール認証を設定します。コンソールの認証には、**skupper** オプションが複数あります。

--console-auth <authentication-mode>

コンソールの認証モードを設定します。

- **openshift** - OpenShift 認証を使用して、OpenShift へのログイン権限およびプロジェクト (namespace) の表示権限のあるユーザーがコンソールを表示できるようにします。
- **internal** - Application Interconnect 認証を使用します。**console-user** オプションおよび **console-password** オプションを参照してください。
- **unsecured** - 認証なし、URL を持つユーザーがコンソールを表示できます。

--console-user <username>

認証モードが **internal** に設定されている場合のコンソールユーザーのユーザー名。デフォルトは **admin** です。

--console-password <password>

認証モードが **internal** に設定されている場合のコンソールユーザーのパスワード。指定しない場合は、ランダムなパスワードが生成されます。

- サービスアクセスの設定

```
$ skupper init --create-network-policy
```



注記

すべてのサイトは、この手順で **アクティブ namespace** と呼ばれる namespace に関連付けられます。

アクティブ namespace のサービスは、クラスターのネットワークポリシーに応じて、デフォルトでそのクラスターの他の namespace の Pod にアクセスできる場合があります。その結果、サービスネットワークに直接接続されていない namespace にある Pod にサービスを公開することができます。この設定は、アクティブ namespace 内の Pod のサービスへのアクセスを制限するネットワークポリシーを適用します。

たとえば、**clusterA** の namespace **projectA** にサイトを作成し、そのサイトを **database** サービスが公開されているサービスネットワークにリンクすると、**clusterA** の **projectB** の Pod で **database** サービスを利用できます。

--create-network-policy オプションを使用すると、**database** サービスのアクセスを **clusterA** の **projectA** に制限することができます。

第4章 サイトをリンクする

サービスネットワークは、アプリケーションインターコネクトサイトで設定されます。このセクションでは、サイトをリンクしてサービスネットワークを形成する方法を説明します。

2つのサイトをリンクするには、単一の初期方向接続が必要です。ただし、

- 2つのサイト間の通信は双方向であり、初期リンクのみが単一方向です。
- リンクの方法は、一般的にアクセシビリティによって決定されます。たとえば、OpenShift Dedicated クラスターを CodeReady Containers クラスターにリンクする場合、そのルートはアクセス可能なので、CodeReady Containers クラスターから OpenShift Dedicated クラスターにリンクする必要があります。

手順

1. リンクの方法を決定します。両方のクラスターが一般にアドレス指定可能である場合、方向は重要ではありません。一方のクラスターが他方のクラスターからアドレス指定可能な場合は、アドレス指定可能なクラスターに対して以下の手順2を実行します。
2. リンクさせたいクラスターでトークンを生成します。

```
$ skupper token create <filename>
```

<filename> は、ローカルファイルシステムに保存されている YAML ファイルの名前に置き換えます。

このファイルには、キーとそれを作成したサイトの場所が含まれています。



注記

このファイルへのアクセスは、サービスネットワークへのアクセスを提供します。適切に保護してください。

サービスネットワークへのアクセスの保護に関する詳細は、[Using Skupper tokens](#) を参照してください。

3. 接続元となるクラスター上のトークンを使用します。
 - a. サービスネットワークへのリンクを作成します。

```
$ skupper link create <filename> [-name <link-name>]
```

<filename> は、**skupper token create** コマンドから生成された YAML ファイルの名前に、**<link-name>** はリンクの名前に置き換えます。

- b. リンクを確認してください。

```
$ skupper link status
Connection for link1 not active
```

この例では、<link-name> が指定されておらず、デフォルトは **link1** となっています。

4. リンクを削除したい場合は、以下を実行します。

```
$ skupper link delete <link-name>
```

<link-name> は、作成時に指定したリンクの名前に置き換えます。

第5章 NAMESPACE からのサービスネットワーク上のサービスの公開

サービスネットワークを作成した後、公開されたサービスはそのネットワーク全体で通信できます。

skupper CLI には、namespace に存在するサービスを公開するための2つのオプションがあります。

- **expose** は、単一サービスを使用するデプロイメントなど、単純なユースケースをサポートします。手順は「[サービスネットワーク上での簡単なサービスの公開](#)」を参照してください。
- **service create** と **service bind** は、複数のサービスでデプロイメントする場合など、より柔軟にサービスを公開する方法です。手順は「[サービスネットワーク上での複雑なサービスの公開](#)」を参照してください。

5.1. サービスネットワーク上での簡単なサービスの公開

このセクションでは、簡単なユースケースのサービスネットワークでサービスを有効にする方法を説明します。

手順

1. たとえば、[チュートリアル](#) のバックエンドサービスを作成するために、サイトの1つにデプロイメント、いくつかの Pod、またはサービスを作成します。

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

このステップは Application Interconnect 固有のものではありません。つまり、このプロセスはお使いのクラスターの標準プロセスと変わりません。

2. サービスネットワーク上で通信可能なサービスを作成します。

```
$ skupper expose [deployment <name>|pods <selector>|statefulset  
<statefulsetname>|service <name>]
```

ここでは、以下ようになります。

- **<name>** - デプロイメントの名前。
- **<selector>** - Pod セレクター。
- **<statefulsetname>** - ステートフルセットの名前。

手順1のデプロイメント例では、次のコマンドを使用してサービスを作成します。

```
$ skupper expose deployment/hello-world-backend --port 8080
```

このコマンドのオプションは以下の通りです。

- **--port <port-number>**: このサービスがサービスネットワーク上で利用可能なポート番号を指定します。注記: このオプションを繰り返すことで、複数のポートを指定できます。
- **--target-port <port-number>**: 公開したい Pod のポート番号を指定します。

- **--protocol <protocol>** を使用すると、使用するプロトコル、**tcp**、**http**、または **http2** を指定できます。



注記

ポートを指定しないと、**skupper** はデプロイメントの **containerPort** 値を使用します。

5.2. サービスネットワーク上での複雑なサービスの公開

このセクションでは、より複雑なユースケースのサービスネットワークでサービスを有効にする方法を説明します。

手順

1. たとえば、[チュートリアル](#) のバックエンドサービスを作成するために、サイトの1つにデプロイメント、いくつかの Pod、またはサービスを作成します。

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

このステップは Application Interconnect 固有のものではありません。つまり、このプロセスはお使いのクラスターの標準プロセスと変わりません。

2. サービスネットワーク上で通信可能なサービスを作成します。

```
$ skupper service create <name> <port>
```

ここでは、以下ようになります。

- **<name>** - 作成するサービスの名前。
- **<port>** - サービスが使用するポート。

手順1のデプロイメント例では、次のコマンドを使用してサービスを作成します。

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

3. サービスをクラスターサービスにバインドします。

```
$ skupper service bind <service-name> <target-type> <target-name>
```

ここでは、以下ようになります。

- **<service-name>** - サービスネットワーク上のサービスの名前。
- **<target-type>** は、公開するオブジェクト (**deployment**、**statefulset**、**pods**、または **service**) です。
- **<target-name>** - クラスターサービスの名前。
- **--protocol <protocol>** を使用すると、使用するプロトコル、**tcp**、**http**、または **http2** を指定できます。

手順1のデプロイメント例では、次のコマンドを使用してサービスをバインドします。

\$ skupper service bind hello-world-backend deployment hello-world-backend

第6章 ローカルマシンからのサービスネットワーク上のサービスの公開

サービスネットワークの作成後、サービスネットワーク上のローカルマシンからサービスを公開することができます。

たとえば、データセンターのサーバーでデータベースを実行する場合は、データベースがクラスターで実行しているかのようにデータにアクセスできるクラスターにフロントエンドをデプロイすることができます。

6.1. サービスネットワークにシンプルなローカルサービスの公開

本セクションでは、サービスネットワーク上でローカルで実行している単一のサービスを公開する方法を説明します。

前提条件

- サービスネットワーク。必要なサイトは1つだけです。
- サービスネットワークへのアクセス。
- デフォルトの **service** タイプのゲートウェイのための **skrouterd**

手順

1. サービスをローカルに実行します。
2. クラスターにログインし、実際のサイトの namespace に移動します。
3. サービスネットワーク上でサービスを公開します。

```
$ skupper gateway expose <service> localhost <port>
```

- <service> - サービスネットワーク上のサービスの名前。
- <port> - サービスをローカルで実行するポート。



注記

たとえば、MySQL が専用サーバー (IP アドレス **192.168.1.200**) で実行しているが、同じネットワーク内のマシンからクラスターにアクセスしている場合など、ローカルネットワーク上の他のマシンからサービスを公開することもできます。

```
$ skupper gateway expose mysql 192.168.1.200 3306
```

4. Skupper ゲートウェイのステータスを確認します。

```
$ skupper gateway status
```

Gateway Definition:

```
└─ machine-user type:service version:1.18.0
```

```
└─ Bindings:
```

```
└─ mydb:3306 tcp mydb:3306 127.0.0.1 3306
```


-

これは、公開されるサービスが1つだけあり、サービスは単一のポート (BIND) のみを公開することを示しています。ローカルホストへ転送されるポートはありません。

URL フィールドは基礎となる通信を示し、無視できます。

6.2. サービスネットワーク上で複雑なローカルサービスの使用

このセクションでは、skupper ゲートウェイの高度な使用方法を説明します。

1. Skupper ゲートウェイを作成します。

```
$ skupper gateway init --type <gateway-type>
```



注記

デフォルトのサービスタイプのゲートウェイでは、**skrouterd** が実行されている必要があります。

デフォルトでは、サービス タイプのゲートウェイが作成されますが、以下を指定することも可能です。

- **podman**
- **docker**

2. サービスネットワーク上で通信可能なサービスを作成します。

```
$ skupper service create <name> <port>
```

ここでは、以下のようになります。

- **<name>** - 作成するサービスの名前。
- **<port>** - サービスが使用するポート。

以下に例を示します。

```
$ skupper service create mydb 3306
```

3. サービスネットワーク上のサービスをバインドします。

```
$ skupper gateway bind <service> <host> <port>
```

- **<service>** - サービスネットワーク上のサービス名 (上記の例では **mydb**)。
- **<host>** - サービスを実行するホスト。
- **<port>** - サービスが実行中のポート。上記の例の **3306**。

4. Skupper ゲートウェイのステータスを確認します。

```
$ skupper gateway status
Gateway Definitions Summary
```

Gateway Definition:

```
└─ machine-user type:service version:1.18.0
   └─ Bindings:
      └─ mydb:3306 tcp mydb:3306 127.0.0.1 3306
```

これは、公開されるサービスが1つだけあり、サービスは単一のポート (BIND) のみを公開することを示しています。ローカルホストへ転送されるポートはありません。

URL フィールドは基礎となる通信を示し、無視できます。

サービスネットワークに追加のサービスを作成し、さらにローカルサービスをバインドして、これらのサービスをサービスネットワーク上で公開することができます。

5. サービスネットワークからローカルマシンにサービスを転送します。

```
$ skupper gateway forward <service> <port>
```

ここでは、以下のようになります。

- **<service>** は、サービスネットワーク上の既存サービスの名前です。
- **<port>** は、使用するローカルマシンのポートです。

6.3. ゲートウェイを作成し、別のマシンでそれを適用する

1 台のマシンからクラスターにアクセスできるが、別のマシンからサービスネットワークへのゲートウェイを作成したい場合は、この手順で説明するように、最初のマシンでゲートウェイ定義バンドルを作成し、後でその定義バンドルを 2 台目のマシンに適用することができます。たとえば、ローカルのデータベースサービスをサービスネットワークに公開しても、データベースサーバーからクラスターにアクセスしたくない場合は、この手順を使用して定義バンドルを作成し、データベースサーバーに適用することができます。

手順

1. 1 台目のマシンからクラスターにログインし、サイトの namespace に変更します。
2. サービスネットワーク上で通信可能なサービスを作成します。

```
$ skupper service create <name> <port>
```

ここでは、以下のようになります。

- **<name>** - 作成するサービスの名前。
- **<port>** - サービスが使用するポート。

以下に例を示します。

```
$ skupper service create database 5432
```

3. 公開するサービスを表す YAML ファイルを作成します。以下に例を示します。

```
name: database 1
```

```
bindings:
  - name: database ②
    host: localhost ③
    service:
      address: database:5432 ④
      protocol: tcp ⑤
      ports:
        - 5432 ⑥
      target_ports:
        - 5432 ⑦
qdr-listeners:
  - name: amqp
    host: localhost
    port: 5672
```

- ① ゲートウェイ名。参照のみの場合に役立ちます。
- ② バインディング名。複数のバインディングを追跡する際に役立ちます。
- ③ 公開したいサービスを提供しているホストの名前。
- ④ サービスネットワーク上のサービス名とポート。前のステップでサービスを作成しました。
- ⑤ サービスを公開するために使用するプロトコル、**tcp**、**http**、または **http2**。
- ⑥ このサービスを利用できるようにするサービスネットワーク上のポート。
- ⑦ ポイント 3 で指定されたホストで実行されているサービスのポート。

4. ゲートウェイの名前を使用して、YAML ファイルを保存します (例: **gateway.yaml**)。
5. サービスネットワーク上で公開するサービスをホストするマシンに適用できるバンドルを生成します。

```
$ skupper gateway generate-bundle <config-filename> <destination-directory>
```

ここでは、以下ようになります。

- <config-filename> - 前のステップで生成した YAML ファイルの名前 (接尾辞を含む)。
- <destination-directory> - 結果のゲートウェイバンドルを保存する場所 (例: **~/gateways**)。

以下に例を示します。

```
$ skupper gateway generate-bundle database.yaml ./
```

このバンドルには、サービスネットワークへのアクセスを許可するゲートウェイ定義の YAML と証明書が含まれています。

6. ゲートウェイ定義ファイル (**mylaptop-jdoe.tar.gz** など) を、サービスネットワーク上で公開するサービスをホストするマシンにコピーします。
7. 公開したいサービスをホストしているマシンから、以下を実行します。

■

```
$ mkdir gateway
$ tar -xvf <gateway-definition-file> --directory gateway
$ cd gateway
$ sh ./launch.py
```



注記

コンテナ内で Application Interconnect ルーターを実行するには、**./launch.py -t podman** または **./launch.py -t docker** を使用します。

ゲートウェイバンドルを実行すると、ゲートウェイ定義 YAML と証明書を使用して、サービスネットワーク上のサービスにアクセスして公開します。

- ゲートウェイサービスのステータスを確認します。
サービス タイプのゲートウェイを確認するには、以下を実行します。

```
$ systemctl --user status <gateway-definition-name>
```

podman タイプのゲートウェイを確認するには、以下を実行します。

```
$ podman inspect
```

docker タイプのゲートウェイを確認するには、以下を実行します。

```
$ docker inspect
```



注記

後で **./remove.py** を使用してゲートウェイを削除できます。

- クラスターにアクセスできるマシンから、Skupper ゲートウェイのステータスを確認します。

```
$ skupper gateway status
Gateway Definitions Summary

NAME   BINDS  FORWARDS  URL
<machine-name>  1      0         amqp://127.0.0.1:5672
```

これは、公開されるサービスが1つだけあり、サービスは単一のポート (BIND) のみを公開することを示しています。ローカルホストへ転送されるポートはありません。



注記

ポートの変更など、ゲートウェイの定義を変更する必要がある場合は、既存のゲートウェイを削除し、この手順を最初から繰り返してゲートウェイを再定義する必要があります。

6.4. ゲートウェイ YAML リファレンス

「[ゲートウェイを作成し、別のマシンでそれを適用する](#)」では、ゲートウェイ定義 YAML ファイルを使用して、別のマシンで適用するゲートウェイを作成する方法を説明します。

以下は、ゲートウェイ定義 YAML ファイルの有効なエントリです。

name

ゲートウェイの名前。

bindings.name

単一ホストのバインディングの名前。

bindings.host

ローカルサービスのホスト名。

bindings.service

サービスネットワークで利用できるようにしたいサービスの定義。

bindings.service.address

サービスネットワーク上のアドレス、名前、ポート。

bindings.service.protocol

Skupper プロトコル、**tcp**、**http** または **http2**。

bindings.service.ports

サービスネットワーク上で利用可能になる単一のポート。

bindings.service.target_ports

サービスネットワーク上で公開したい単一のポート。



注記

ローカルサービスで複数のポートが必要な場合は、ポートごとに個別のバインディングを作成します。

forwards.name

単一ホストの転送の名前。

forwards.host

ローカルサービスのホスト名。

forwards.service

ローカルで利用できるようにしたいサービスの定義。

forwards.service.address

ローカルで使いたいサービスネットワーク上のアドレス、名前、ポート。

forwards.service.protocol

Skupper プロトコル、**tcp**、**http** または **http2**。

forwards.service.ports

サービスネットワークで利用可能な単一のポート。

forwards.service.target_ports

ローカルで使いたい単一のポート。



注記

ネットワークサービスに複数のポートが必要な場合は、ポートごとに個別の転送を作成します。

qdr-listeners

skupper ルーターのリスナーの定義。

qdr-listeners.name

skupper ルーターの名前 (通常は **amqp**)。

qdr-listeners.host

skupper ルーターのホスト名 (通常は **localhost**)。

qdr-listeners.port

skupper ルーターのポート (通常は **5672**)。

第7章 サービスネットワークの検証

Application Interconnect には、サービスネットワークで利用可能なすべてのサイトとサービスを報告できるコマンドが含まれています。

前提条件

- 複数のサイトを持つサービスネットワーク

手順

1. Kubernetes のコンテキストをサービスネットワーク上の namespace に設定します。
2. 以下のコマンドを使用して、サービスネットワークのステータスを報告します。

```
$ skupper network status
```

たとえば、以下は、[Creating a service network with OpenShift](#) チュートリアルで、**west** namespace から作成されたサービスネットワークの出力を示しています。

```
Sites:
├─ [local] 4dba248 - west ❶
│  └─ URL: 10.96.146.236 ❷
│     └─ name: west ❸
│        └─ namespace: west
│           └─ version: 0.8.6 ❹
│              └─ Services:
│                 └─ name: hello-world-backend ❺
│                    └─ address: hello-world-backend: 8080 ❻
│                       └─ protocol: tcp ❼
├─ [remote] bca99d1 - east ❽
│  └─ URL:
│     └─ name: east
│        └─ namespace: east
│           └─ sites linked to: 4dba248-west ❾
│              └─ version: 0.8.6
│                 └─ Services:
│                    └─ name: hello-world-backend
│                       └─ address: hello-world-backend: 8080
│                          └─ protocol: tcp
│                             └─ Targets:
│                                └─ name: hello-world-backend-7dfb45b98d-mhskw ❿
```

- ❶ 現在のコンテキストに関連付けられたサイトの一意の識別子、つまり **west** namespace
- ❷ サービスネットワークルーターの URL。これは他のサイトがこのサイトに接続するために必要なもので、コンソール URL とは異なります。コンソールの URL が必要な場合は、**skupper status** コマンドを使用して、その URL を表示します。
- ❸ サイト名。デフォルトでは、skupper は現在の namespace の名前を使用します。サイト名を指定する場合は、**skupper init --site-name <site-name>** を使用します。
- ❹ サイトを実行している Application Interconnect のバージョン。サイトのバージョンは、現在の **skupper** CLI バージョンと異なる場合があります。サイトを CLI のバージョンに更

新するには、**skupper update** を使用します。

- ⑤ サービスネットワーク上で公開されているサービスの名前。
- ⑥ サービスネットワーク上で公開されているサービスのアドレス。
- ⑦ サービスネットワーク上で公開されているサービスのプロトコル。
- ⑧ サービスネットワーク上のリモートサイトの一意の識別子。
- ⑨ リモートサイトがリンクしているサイト。
- ⑩ サービスネットワーク上で公開されるローカル Kubernetes オブジェクトの名前。この例では、**hello-world-backend** Pod になります。



注記

east サイトの URL は、そのサイトが以下のコマンドを使用して ingress なしで初期化されたため、値がありません。

```
$ skupper init --ingress none
```


第8章 サービスネットワークの保護

Application Interconnect は、クラスターやクラウドにまたがってスケーリングする、デフォルトのビルトインセキュリティを提供します。このセクションでは、追加で設定可能なセキュリティについて説明します。

各クラスターの詳細なポリシーの作成に関する情報は、[Securing a service network using policies](#) を参照してください。

8.1. ネットワークポリシーを使用したサービスへのアクセスの制限

デフォルトでは、サービスネットワーク上でサービスを公開すると、そのサービスはクラスター内の他の namespace からアクセスできます。**--create-network-policy** オプションを使用してサイトを作成すると、この状況を回避することができます。

手順

1. ネットワークポリシーでサービスネットワークルーターを作成します。

```
$ skupper init --create-network-policy
```

2. 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.
```

サービスネットワーク上でサービスを公開できるようになり、それらのサービスはクラスター内の他の namespace からアクセスできません。

8.2. サービスネットワーク上の HTTP2 トラフィックへの TLS の適用

デフォルトでは、サイト間のトラフィックは暗号化されますが、サービス Pod とルーター Pod 間のトラフィックは暗号化されません。HTTP2 として公開されているサービスでは、Pod とルーター Pod 間のトラフィックを TLS を使用して暗号化することができます。

前提条件

- 2 つ以上のリンクされたサイト
- HTTP2 フロントエンドおよびバックエンドサービス

手順

1. バックエンドサービスをデプロイします。
2. TLS を有効にして、サービスネットワークにバックエンドデプロイメントを公開します。以下に例を示します。

```
$ skupper expose deployment <deployment-name> --port 443 --protocol http2 --enable-tls
```

TLS を有効にすると、TLS バックエンドに必要な証明書が作成され、**skupper-tls-
<deployment-name>** という名前のシークレットに保存されます。

3. 生成された証明書を含めるようにバックエンドデプロイメントを変更します。以下に例を示します。

```
...
spec:
  containers:
    ...
    command:
      ...
      - "/certs/tls.key"
      - "/certs/tls.crt"
    ...
    volumeMounts:
      ...
      - mountPath: /certs
        name: certs
        readOnly: true
  volumes:
    - name: index-html
      configMap:
        name: index-html
    - name: certs
      secret:
        secretName: skupper-tls-<deployment-name>
```

各サイトは、TLS クライアントに必要な証明書を作成し、**skupper-service-client** という名前のシークレットに保存します。

4. 生成された証明書を含めるようにフロントエンドデプロイメントを変更します。以下に例を示します。

```
spec:
  template:
    spec:
      containers:
        ...
        volumeMounts:
          - name: certs
            mountPath: /tmp/certs/skupper-service-client
        ...
      volumes:
        - name: certs
          secret:
            secretName: skupper-service-client
```

5. TLS が有効なフロントエンドからのサービスの呼び出しをテストします。

第9章 サポートされる標準およびプロトコル

Application Interconnect は、お使いのサービスネットワークで以下のプロトコルをサポートしています。

- TCP - デフォルト
- HTTP1
- HTTP2

サービスを公開または作成する際に、プロトコルを指定できます。以下に例を示します。

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

ここで、<protocol> は以下のいずれかになります。

- tcp
- http
- http2

指定するプロトコルを選択するときは、以下の点に注意してください。

- **tcp** は TCP にオーバーレイされたすべてのプロトコルをサポートします。たとえば、**tcp** を指定すると HTTP1 および HTTP2 が機能します。
- **http** または **http2** を指定すると、クライアントから報告された IP アドレスにアクセスできない場合があります。
- すべてのサービスネットワークトラフィックは、サービスネットワークをトラバースするために AMQP メッセージに変換されます。
TCP は単一のストリームメッセージとして実装され、HTTP1 および HTTP2 はリクエスト/レスポンスメッセージルーティングとして実装されています。

第10章 異なるクラスターを操作するための CLI オプション

デフォルトでは、すべての **skupper** コマンドはログインしているクラスターと現在の namespace に適用されます。以下の **skupper** オプションでは、その動作を上書きし、すべてのコマンドに適用できます。

--namespace <namespace-name>

コマンドを **<namespace-name>** に適用します。たとえば、現在 **frontend** namespace で作業しており、**backend** namespace のサイトを初期化する場合は、以下のコマンドを実行します。

```
$ skupper init --namespace backend
```

--kubeconfig <kubeconfig-path>

kubeconfig ファイルへのパス - これにより、同じクライアントからクラスターに対して複数のセッションを実行できます。または、**KUBECONFIG** 環境変数を設定することです。kubeconfig ファイルの使用例については、[チュートリアル](#) を参照してください。

--context <context-name>

kubeconfig ファイルには定義されたコンテキストを含むことができ、このオプションを使用すると、それらのコンテキストを使用できます。

第11章 SKUPPER トークンの使用

Skupper トークンを使用すると、サイト間にリンクを作成できます。一方のサイトでトークンを作成し、もう一方のサイトからそのトークンを使用して、2つのサイト間のリンクを作成します。



注記

リンクのプロセスには方向性がありますが、Skupper リンクは双方向の通信を可能にします。

たとえば、2つのパブリッククラウドなど、両方のサイトに同等にアクセスできる場合は、トークンをどこで作成するかは重要ではありません。ただし、トークンを使用する場合は、リンク元サイトからリンク先サイトにアクセスできなければなりません。たとえば、パブリッククラスターとプライベートクラスターの両方を使用してサービスネットワークを作成する場合は、パブリッククラスターでトークンを作成し、プライベートクラスターからトークンを使用する必要があります。

Skupper トークンには、以下の2つのタイプがあります。

クレームトークン (デフォルト)

クレームトークンは、以下の方法で制限できます。

- 時間 - 指定された期間を過ぎると、トークンの再利用ができなくなります。
- 使用法 - 1つのトークンから複数のリンクを作成することを阻止します。

すべてのサイト間トラフィックは、専用の専用認証局 (CA) を使用した相互 TLS によって保護されます。クレームトークンは証明書ではありませんが、リンクプロセス中に証明書と安全に交換されます。適切な制限を実装 (例: 使い捨てのクレームトークンを作成) することで、証明書が誤って公開されることを回避できます。

Cert トークン

cert トークンを使用して、そのトークンを発行したサイトへのリンクを作成できます。これには、そのサイトからの有効な証明書が含まれています。

すべてのサイト間トラフィックは、専用の専用認証局 (CA) を使用した相互 TLS によって保護されます。cert トークンは、専用 CA が発行する証明書です。適切に保護してください。

11.1. クレームトークンの作成

クレームトークンを使用して、そのトークンを発行したサイトへのリンクを作成できます。そのサイトからの証明書は含まれていませんが、クレームトークンが使用されると、サイトから証明書が渡されます。クレームトークンは、時間または使用法によって制限できます。

手順

1. クラスターにログインします。
2. サイトに関連する namespace に変更します。
3. クレームトークンを作成します。以下に例を示します。

```
$ skupper token create $HOME/secret.yaml --expiry 30m0s --uses 2 -t claim
```



注記

クレームトークンはデフォルトで、コマンドの **-t claim** セクションは不要です。

--expiry

トークンの有効期限を分、秒で指定します。デフォルトは **15m0s** です。

--uses

トークンを使用してリンクを作成できる回数、デフォルトは **1** です。

追加情報

- トークンを使用してリンクを作成する方法の詳細は、[Configuring Application Interconnect sites using the CLI](#) を参照してください。

11.2. CERT トークンの作成

cert トークンを使用すると、さまざまなサイトからサービスネットワークへの多くのリンクを制限なく作成できます。

手順

1. クラスターにログインします。
2. サイトに関連する namespace に変更します。
3. cert トークンを作成します。

```
$ skupper token create $HOME/secret.yaml -t cert
```



注記

cert トークンは常に有効であり、「[サイトへのアクセスの取り消し](#)」で説明されているように、取り消されない限り、無期限に再利用できます。

追加情報

- トークンを使用してリンクを作成する方法の詳細は、[Configuring Application Interconnect sites using the CLI](#) を参照してください。

11.3. サイトへのアクセスの取り消し

トークンが危険にさらされた場合は、サイトから作成されたすべてのトークンを無効にすると、そのトークンの不正使用を防ぐことができます。

このオプションでは、サイトへのすべてのリンクが削除され、サービスネットワークを復元するためにリンクを再作成する必要があります。

1. 手順
2. クラスターにログインします。
3. サイトに関連する namespace に変更します。

4. サイトのステータスを確認します。

```
$ skupper status
Skupper is enabled for namespace "west" in interior mode. It is linked to 2 other sites.
```

5. サイトからの発信リンクを確認します。

```
$ skupper link status
Link link1 is active
```

この場合は、2つのリンクと1つの発信リンクがあります。つまり、1つの着信リンクがあります。

6. 着信リンクからサイトへのアクセスを取り消します。

```
$ skupper revoke-access
```

7. サイトのステータスをチェックして、アクセスの取り消しを確認します。

```
$ skupper status
Skupper is enabled for namespace "west" in interior mode. It is linked to 1 other site.
$ skupper link status
Link link1 is active
```

出力には、**skupper revoke-access** コマンドが着信リンクを取り消しているものの、発信リンクはまだアクティブであることが示されています。

skupper link delete link1 コマンドを使用して、そのリンクを削除することができますが、アクセスを取り消すには、適切なクラスターにログインしている間にこの手順を実行する必要があります。

追加情報

- [Configuring Application Interconnect sites using the CLI](#) を参照してください。

改訂日時: 2022-08-14 10:32:24 +1000