Red Hat Service Interconnect 1.5

使用 Service Interconnect

使用 CLI 和 YAML 创建服务网络

Last Updated: 2024-06-25

Red Hat Service Interconnect 1.5 使用 Service Interconnect

使用 CLI 和 YAML 创建服务网络

法律通告

摘要

Red Hat Service Interconnect 是一个开源 Skupper 项目的红帽构建。这个 Skupper 文档会被复制以备参考。

目录

第 1 章 使用 SKUPPER CLI

使用 **skupper** 命令行界面(CLI)您可以从当前命名空间上下文创建和管理 Skupper 站点。

典型的工作流是创建一个站点，将站点链接在一起，并向服务网络公开服务。

1.1. 检查 SKUPPER CLI

安装 **skupper** 命令行界面(CLI)提供了开始使用 Skupper 的简单方法。

流程

1. 验证安装。

```
$ skupper version
client version 1.5.3-rh-5
```

1.2. 使用 CLI 创建站点

服务网络由 Skupper 站点组成。本节论述了如何使用默认设置在 Kubernetes 集群中创建站点。有关使用 Skupper CLI 创建 Podman 站点的信息，请参阅使用 Skupper Podman。

先决条件

- 已安装 **skupper** CLI。
- 已登陆到集群。
- 在服务网络中公开的服务位于活跃的命名空间中。

流程

1. 创建默认站点：

```
$ skupper init
```

从 Skupper 版本 1.3 开始，控制台不会被默认启用。要使用新控制台，请参阅使用控制台。

2. 检查网站：

```
$ skupper status
```

输出应类似于如下：

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

注意

当您在集群中初始化没有安装 Skupper 策略的站点时，会显示上面的默认消息。如果您按照使用策略 保护服务网络中的内容安装 Skupper 策略，则该消息将变为 **Skupper is enabled for namespace "west" in interior mode (with policies)**。

默认情况下，站点名称默认为命名空间名称，如 **west**。

1.3. 自定义站点

默认 **skupper init** 创建满足典型要求的站点。

从 Skupper 版本 1.3 开始，控制台不会被默认启用。要使用新控制台，请参阅使用控制台。

如果您需要自定义配置，请注意以下选项：

- 配置控制台身份验证.有关控制台身份验证的几个 **skupper** 选项：

  --console-auth <authentication-mode>

  在控制台中设置身份验证模式：

  - **openshift** - 使用 OpenShift 身份验证，以便有权登录 OpenShift 的用户并查看项目（命名空间）可以查看控制台。
  - **internal** - 使用 Skupper 身份验证，请参阅 **console-user** 和 **console-password** 选项。
  - **unsecured** - 没有身份验证，有 URL 的任何人都可以查看控制台。

  --console-user <username>

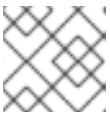
  当将身份验证模式设置为 **internal** 时，控制台用户的用户名。默认为 **admin**。

  --console-password <password>

  当身份验证模式设置为 **internal** 时，控制台用户的密码。如果没有指定，会生成一个随机密码。

- 配置服务访问

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

  注意

  所有站点都与一个命名空间关联。在此过程中被称为*活跃命名空间*。

  活跃命名空间中的服务默认可以被该集群上的其他命名空间中的 pod 访问，具体取决于您的集群网络策略。因此，您可以将服务公开给命名空间中的 pod，而不是直接连接到服务网络。此设置应用 Kubernetes 网络策略，以限制对活跃命名空间中那些 pod 的访问。

  例如，如果您在 **clusterA** 的命名空间 **projectA** 中创建一个站点，并将该站点链接到公开 **database** 服务的服务网络，**database** 服务对于 **clusterA** 的 **projectB** 中的 pod 可用。

  您可以使用 **--create-network-policy** 选项限制对 **clusterA** 的 **projectA** 的 **database** 服务的访问。

1.4. 链接站点

服务网络由 Skupper 站点组成。本节论述了如何将站点链接到构成服务网络。

连接两个站点需要一个初始方向连接。但是：

- 两个站点之间的通信是双向的，只有初始链接是带有方向的。

- 链接的方向通常由可访问性所决定。例如，如果要将 OpenShift Dedicated 集群与 CodeReady Containers 集群相关联，则必须从 CodeReady Containers 集群链接到 OpenShift Dedicated 集群，因为该路由可以访问。

流程

1. 确定链接的方向。如果两个集群都可以被公共访问，则方向并不重要。如果一个集群可以从其他集群寻址，请在可寻址的集群上执行以下步骤 2。
2. 在您要链接的集群中生成令牌：

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

其中 **<filename>** 是保存在本地文件系统中的 YAML 文件的名称。

此文件包含一个密钥以及创建它的站点的位置。

注意

通过访问此文件，可以访问服务网络。对其进行恰当的保护。

有关保护对服务网络的访问的更多信息，请参阅使用 Skupper 令牌。

3. 使用您要从中连接的集群中的令牌：
创建到服务网络的链接：

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

其中，**<filename>** 是由 **skupper token create** 命令创建的 YAML 文件的名称，**<link-name>** 是链接的名称。

检查链接：

```
$ skupper link status
Link link1 not connected
```

在本例中，没有指定 <link-name>，名称默认为 **link1**。

删除链接：

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

其中 **<link-name>** 是创建过程中指定的链接的名称。

第 2 章 指定链接成本

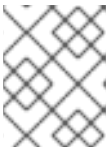
在连接站点时，您可以为每个链接分配一个成本来影响流量流。默认情况下，为新链接将链接成本设置为 **1**。在服务网络中，路由算法尝试使用从客户端到目标服务器的最低总成本的路径。

- 如果您在不同站点间分布服务，您可能需要客户端使用特定目标或链接。在这种情况下，您可以在替代链接中指定大于 **1** 的成本，以减少这些链接的使用。

注意

开放连接的分发是统计数据，即不是循环系统。

- 如果连接只遍历一个链接，则路径成本与链接成本相同。如果连接遍历多个链接，路径成本是路径中涉及的所有链接的总和。
- 成本充当使用从客户端到网络中的服务器的路径的阈值。当只有一个路径时，无论成本如何，该路径上的流量流。



注意

如果您从两个服务的目标开始，并且不再提供其中一个目标，剩余路径上的流量流不再可用，无论成本如何。

- 当多个路径从客户端到服务器实例或服务时，最低成本路径上的流量流，直到连接数量超过替代路径的成本。达到这个打开连接阈值后，新的连接将分散到替代路径和最低的成本路径中。

前提条件

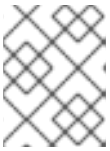
- 您已将 Kubernetes 上下文设置为您要*从其中*链接的站点。
- 要链接*到*的站点的令牌。

流程

1. 创建到服务网络的链接：

```
$ skupper link create <filename> --cost <integer-cost>
```

其中 **<integer-cost>** 是一个大于 1 的整数，流量会优先使用成本链接。



注意

如果可以在不遍历链接的情况下调用服务，该服务被视为本地，且带有隐式成本 **0**。

例如，使用名为 **token.yaml** 的令牌文件创建成本设置为 **2** 的链接：

```
$ skupper link create token.yaml --cost 2
```

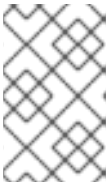
2. 检查链接成本：

```
$ skupper link status link1 --verbose
```

输出结果类似以下：

```
Cost:         2
Created:      2022-11-17 15:02:01 +0000 GMT
Name:         link1
Namespace:    default
Site:         default-0d99d031-cee2-4cc6-a761-697fe0f76275
Status:       Connected
```

3. 使用控制台观察流量。
如果您在站点中有一个控制台，请登录并导航到每台服务器的进程。您可以查看与每个客户端对应的流量级别。



注意

如果不同站点上有多个客户端，请将视图过滤到每个客户端，以确定流量成本的影响。例如，在两个站点网络中与两个站点的服务器和客户端链接在一起，您可以看到客户端在本地服务器可用时由本地服务器提供。

2.1. 从命名空间公开服务网络上的服务

创建服务网络后，公开的服务就可以在该网络间进行通信。

skupper CLI 有两个选项用于公开命名空间中已存在的服务：

- **expose** 支持简单用例，例如，单一服务的部署。具体步骤请查看 第 2.1.1 节 “在服务网络上公开简单的服务”。
- **service create** and **service bind** 是更灵活的公开服务方法，例如，如果您的一个部署有多个服务。具体步骤请查看 第 2.1.2 节 “在服务网络上公开复杂的服务”。

2.1.1. 在服务网络上公开简单的服务

这部分论述了如何为服务网络启用服务用于简单用例。

流程

1. 在其中一个站点中创建部署、一些 pod 或服务，例如：

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

此步骤不是特定于 Skupper，即此过程与集群的标准进程不同。

2. 创建可在服务网络中进行通信的服务：

部署和 pod

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

其中

- **<name** > 是部署的名称

- **<selector>** 是一个 pod 选择器

Kubernetes 服务

使用 **--address** 选项指定生成的服务名称。

```
$ skupper expose service <name> --address <skupper-service-name>
```

其中

- **<** name> 是服务的名称
- **<skupper-service-name** > 是服务网络上共享生成的服务的名称。

StatefulSets

您可以使用以下方法公开 statefulset：

```
$ skupper expose statefulset <statefulsetname>
```

Kubernetes 中的 StatefulSet 通常与无头服务关联，为每个 pod 提供稳定、唯一的网络标识符。如果服务网络上的每个 pod 需要稳定的网络标识符，请使用 **--headless** 选项。

```
$ skupper expose statefulset  --headless
```

注意

当您使用 '--headless" 选项时，服务网络中只有一个 statefulset 才能通过地址 (routing 键)公开。

对于第 1 步中的示例部署，您可以使用以下命令创建服务：

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

expose 命令的选项包括：

- **--port <port-number>**:: 指定该服务在服务网络上提供的端口号。注意：您可以通过重复这个选项来指定多个端口。
- **--target-port <port-number>**:: 指定您要公开的 pod 的端口号。
- **--protocol <protocol>** 允许您指定要使用的协议：**tcp**、**http** 或 **http2**

注意

如果没有指定端口，**skupper** 将使用部署的 **containerPort** 值。

3. 检查服务网络上公开的服务状态(**-v** 仅在 Kubernetes 上可用)：

```
$ skupper service status -v
Services exposed through Skupper:
└─ backend:8080 (tcp)
   └─ Sites:
      ├─ 4d80f485-52fb-4d84-b10b-326b96e723b2(west)
```

```
    policy: disabled
  316fbe31-299b-490b-9391-7b46507d76f1(east)
    policy: disabled
    Targets:
      backend:8080 name=backend-9d84544df-rbzjx
```

2.1.2. 在服务网络上公开复杂的服务

本节描述了如何为服务网络启用服务，以获取更复杂的用例。

流程

1. 在其中一个站点中创建部署、一些 pod 或服务，例如：

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

此步骤不是特定于 Skupper，即此过程与集群的标准进程不同。

2. 创建可在服务网络中进行通信的服务：

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

其中

- **<name>** 是您要创建的服务的名称
- **<port>** 是服务使用的端口

对于第 1 步中的示例部署，您可以使用以下命令创建服务：

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

3. 将服务绑定到集群服务：

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

其中

- **<service-name>** 是服务网络上的服务名称
- **<target-type>** 是您要公开的对象：**deployment**, **statefulset**, **pods**, 或 **service**。
- **<target-name>** 是集群服务的名称

对于第 1 步中的示例部署，您可以使用以下命令绑定该服务：

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

2.1.3. 将不同命名空间中的服务公开给服务网络

本节介绍如何从未部署 Skupper 的命名空间公开服务。

skupper 允许您为任何站点从其他命名空间中公开 Kubernetes 服务。但是，如果要公开工作负载，如部署，您必须创建一个站点，如本节所述。

先决条件

- 部署 Skupper 的命名空间。
- 允许命名空间间通信的网络策略
- 如果要公开服务以外的资源，则 cluster-admin 权限

流程

1. 如果要从站点命名空间以外的命名空间中公开工作负载，请使用集群权限创建站点：

注意

站点不需要通过 **--enable-cluster-permissions** 授予的额外权限来公开 Kubernetes 服务资源。

```
$ skupper init --enable-cluster-permissions
```

2. 从站点命名空间以外的命名空间公开 Kubernetes 服务：

```
$ skupper expose service <service>.<namespace> --address <service>
```

- <service> - 服务网络上的服务名称。
- <namespace> - 要公开服务的命名空间的名称。

例如，如果您在 **east** 命名空间中部署了 Skupper，并在 **east- backend** 命名空间中创建了一个后端 Kubernetes 服务，您可以将上下文设置为 **east** 命名空间，并使用以下方法 **将服务作为** 后端在服务网络上公开：

```
$ skupper expose service backend.east-backend --port 8080 --address backend
```

3. 从使用 **--enable-cluster-permissions** 创建的站点公开工作负载：

```
$ skupper expose <resource> --port <port-number> --target-namespace <namespace>
```

- <resource> - 资源的名称。
- <namespace> - 要公开运行资源的命名空间名称。

例如，如果您在 **east** 命名空间中部署了 Skupper，并在 **east- backend 命名空间中创建了后端** 部署，您可以将上下文设置为 **east** 命名空间，并使用以下方法 **将服务作为** 后端在服务网络上公开：

```
$ skupper expose deployment/backend --port 8080 --target-namespace east-backend
```

第 3 章 从本地机器公开服务网络上的服务

创建服务网络后，您可以从服务网络上的本地机器公开服务。

例如，如果您在数据中心的服务器上运行数据库，您可以在一个集群中部署一个前端，该数据库可以访问数据，就像数据库在集群中运行一样。

注意

本文档描述了从本地主机创建到集群站点的网关。另一种方法是在本地主机上创建站点并链接到集群站点。有关使用 Skupper CLI 创建 Podman 站点的信息，请参阅使用 Skupper Podman。

3.1. 向服务网络公开简单的本地服务

本节介绍如何公开在服务网络中本地运行的单个服务。

先决条件

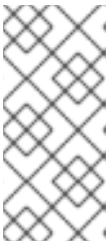
- 服务网络。只需要一个站点。
- 访问服务网络。

流程

1. 在本地运行您的服务。
2. 登录到集群并更改到您的站点的命名空间。
3. 在服务网络中公开服务：

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

   - <service> - 服务网络上的服务名称。
   - <port> - 在本地运行该服务的端口。



注意

您还可以从本地网络上的其他机器公开服务，例如，如果 MySQL 在专用服务器上运行(IP 地址为 **192.168.1.200**)，但您要从同一网络中的机器访问集群：

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

4. 检查 Skupper 网关的状态：

```
$ skupper gateway status

Gateway Definition:
└─ machine-user type:service version:1.5
   └─ Bindings:
      └─ mydb:3306 tcp mydb:3306 localhost 3306
```

这表明只有一个公开的服务，该服务只公开一个端口 (BIND)。没有端口转发到本地主机。

URL 字段显示底层通信，可以忽略。

3.2. 在服务网络上使用复杂的本地服务

本节显示 skupper 网关的更多高级用法。

1. 如果要在 Linux 上创建服务类型网关，则需要路径中的 **skrouterd** 二进制文件。
使用 **yum** 或 **dnf** 命令安装 **skupper-router** 软件包：

```
$ sudo dnf install skupper-router
```

对于 podman 或 docker 类型网关，您可以跳过这一步。

2. 创建 Skupper 网关：

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

默认情况下创建 *service* 类型网关，但您也可以指定：

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

3. 创建可在服务网络中进行通信的服务：

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

其中

- **<name>** 是您要创建的服务的名称
- **<port>** 是服务使用的端口

例如：

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

4. 在服务网络中绑定服务：

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

- <service> - 服务网络上的服务名称，在上例中是 **mydb**。
- <host> - 运行该服务的主机。
- <port> - 服务在上例中运行的端口 **3306**。

5. 检查 Skupper 网关的状态：

```
$ skupper gateway status
```

输出结果类似如下：

```
Gateway Definitions Summary

Gateway Definition:
└─ machine-user type:service version:1.5
   └─ Bindings:
      └─ mydb:3306 tcp mydb:3306 localhost 3306
```

这表明只有一个公开的服务，该服务只公开一个端口 (BIND)。没有端口转发到本地主机。

URL 字段显示底层通信，可以忽略。

您可以在服务网络中创建更多服务，并绑定更多本地服务来公开服务网络上的这些服务。

6. 将服务从服务网络转发到本地计算机。

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

其中

- **<service>** 是服务网络上的现有服务的名称。
- **<port>** 是您要使用的本地机器上的端口。

3.3. 创建网关并在不同的机器上应用它

如果可以从一台机器访问集群，但希望从其他机器创建到服务网络的网关，您可以在第一个机器上创建网关定义捆绑包，稍后将该定义捆绑包应用到第二个机器上，如此过程中所述。例如，如果要将本地数据库服务公开给服务网络，但您不想从数据库服务器访问集群，您可以使用此流程创建定义捆绑包并将其应用到数据库服务器。

流程

1. 从第一个机器登录到集群，并更改到您的站点的命名空间。
2. 创建可在服务网络中进行通信的服务：

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

其中

- **<name>** 是您要创建的服务的名称
- **<port>** 是服务使用的端口

例如：

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

3. 创建一个 YAML 文件来代表您要公开的服务，例如：

```
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. 网关名称，仅适用于参考。
2. 绑定名称，用于跟踪多个绑定。
3. 提供您要公开的服务的主机的名称。
4. 服务网络上的服务名称和端口。您在上一步中创建了服务。
5. 您要用来公开服务的协议：**tcp**, **http** 或 **http2**。
6. 您希望此服务在其中可用的服务网络上的端口。
7. 在点 3 中指定的主机上运行的服务的端口。

4. 使用网关名称保存 YAML 文件，如 **gateway.yaml**。
5. 生成可应用于托管要在服务网络上公开服务的机器的捆绑包：

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

其中：

- <config-filename> - YAML 文件的名称，包括您在上一步中生成的后缀。
- <destination-directory> - 要保存生成的网关捆绑包的位置，如 **~/gateways**。

例如：

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

此捆绑包包含网关定义 YAML 和允许访问服务网络的证书。

6. 将网关定义文件（如 **mylaptop-jdoe.tar.gz** ）复制到托管要在服务网络上公开的服务的机器。
7. 从托管您要公开的服务的机器中：

```
$ mkdir gateway

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

注意

使用 **./launch.py -t podman** 或 **./launch.py -t docker** 来在容器中运行 Skupper 路由器。

运行网关捆绑包使用网关定义 YAML 和证书来访问服务网络上的服务。

8. 检查网关服务的状态：
检查 *service* 类型网关：

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

检查 *podman* 类型网关：

```
$ podman inspect
```

检查 *docker* 类型网关：

```
$ docker inspect
```

注意

之后，您可以使用 **./remove.py** 删除网关。

9. 在具有集群访问权限的机器中，检查 Skupper 网关的状态：

```
$ skupper gateway status
Gateway Definition:
└─ machine-user type:service version:1.5
  └─ Bindings:
    └─ mydb:3306 tcp mydb:3306 localhost 3306
```

这表明只有一个公开的服务，该服务只公开一个端口 (BIND)。没有端口转发到本地主机。

注意

如果您需要更改网关定义，例如要更改端口，您需要删除现有网关，并从开始重复这个过程以重新定义网关。

3.4. 网关 YAML 参考

第 3.3 节 “创建网关并在不同的机器上应用它” 描述如何创建网关以应用到使用网关定义 YAML 文件的独立机器上。

以下是网关定义 YAML 文件中的有效条目：

name

网关的名称

bindings.name

单个主机的绑定名称。

bindings.host

本地服务的主机名。

bindings.service

您希望在服务网络上可用的服务定义。

bindings.service.address

服务网络上的地址、名称和端口。

bindings.service.protocol

Skupper 协议：**tcp**、**http** 或 **http2**。

bindings.service.ports

在服务网络上可用的单个端口。

bindings.service.exposeIngress

（可选）流量方向、**入口** 或 **出口**。

bindings.service.tlscredentials

（可选）服务的 TLS 证书和密钥。

bindings.service.tlscertauthority

（可选）TLS 公共证书。

bindings.target_ports

要在服务网络上公开的单一端口。

注意

如果本地服务需要多个端口，请为每个端口创建单独的绑定。

forwards.name

单个主机的转发名称。

forwards.host

本地服务的主机名。

forwards.service

您希望在本地可用的服务定义。

forwards.service.address

您要在本地、名称和端口的服务网络上的地址。

forwards.service.protocol

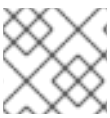
Skupper 协议：**tcp**、**http** 或 **http2**。

forwards.service.ports

服务网络上可用的单个端口。

forwards.target_ports

您要在本地使用的单个端口。



注意

如果网络服务需要多个端口，请为每个端口创建单独的转发。

qdr-listeners

skupper 路由器监听程序的定义

qdr-listeners.name

skupper 路由器的名称，通常为 **amqp**。

qdr-listeners.host

skupper 路由器的主机名，通常为 **localhost**。

qdr-listeners.port

skupper 路由器的端口，通常为 **5672**。

第 4 章 探索服务网络

skupper 包含一个命令，允许您报告服务网络上的所有站点和服务。

先决条件

- 具有多个站点的服务网络

流程

1. 将 Kubernetes 上下文设置为服务网络上的一个命名空间。
2. 使用以下命令报告服务网络的状态：

```
$ skupper network status
```

例如：

```
Sites:
├─ [local] a960b766-20bd-42c8-886d-741f3a9f6aa2(west) 1
│  │ namespace: west
│  │ site name: west 2
│  │ version: 1.5.1 3
│  └─ Linked sites:
│     ├─ 496ca1de-0c80-4e70-bbb4-d0d6ec2a09c0(east)
│     │  direction: outgoing
│     └─ 484cccc3-401c-4c30-a6ed-73382701b18a()
│        direction: incoming
├─ [remote] 496ca1de-0c80-4e70-bbb4-d0d6ec2a09c0(east) 4
│  │ namespace: east
│  │ site name: east
│  │ version: 1.5.1
│  └─ Linked sites:
│     └─ a960b766-20bd-42c8-886d-741f3a9f6aa2(west) 5
│        direction: incoming
└─ [remote] 484cccc3-401c-4c30-a6ed-73382701b18a() 6
   │ site name: vm-user-c3d98
   │ version: 1.5.1
   └─ Linked sites:
      └─ a960b766-20bd-42c8-886d-741f3a9f6aa2(west)
         direction: outgoing
```

1. 与当前上下文关联的站点的唯一标识符，即 **west** 命名空间
2. 站点名称。默认情况下，skupper 使用当前命名空间的名称。如果要指定站点名称，请使用 **skupper init --site-name <site-name>**。
3. 运行站点的 Skupper 版本。站点版本可以与当前的 **skupper** CLI 版本不同。要将站点更新为 CLI 的版本，请使用 **skupper update**。
4. 服务网络上的远程站点的唯一标识符。
5. 远程站点链接到的站点。

6 远程 podman 站点的唯一标识符。Podman 站点没有关联的上下文。

第 5 章 保护服务网络

skupper 提供默认的内置安全性，可在集群和云中扩展。本节论述了您可以配置的其他安全性。

如需有关 为每个集群创建粒度策略的信息，请参阅使用策略保护 服务网络。

5.1. 使用 KUBERNETES 网络策略限制对服务的访问

默认情况下，如果您在服务网络上公开服务，该服务也可以从集群中的其他命名空间访问。您可以使用 **--create-network-policy** 选项创建站点时避免这种情况。

流程

1. 使用 Kubernetes 网络策略创建服务网络路由器：

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

2. 检查站点状态：

```
$ skupper status
```

输出应类似于如下：

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

现在，您可以在服务网络上公开服务，这些服务无法从集群中的其他命名空间访问。

5.2. 将 TLS 应用到服务网络上的 TCP 或 HTTP2 流量

默认情况下，站点之间的流量是加密的，但服务 Pod 和路由器 pod 之间的流量不会被加密。对于作为 TCP 或 HTTP2 公开的服务，pod 和路由器 pod 之间的流量可以使用 TLS 加密。

先决条件

- 两个或多个链接站点
- TCP 或 HTTP2 前端和后端服务

流程

1. 部署后端服务。
2. 在服务网络上公开后端部署，启用 TLS。
   例如，如果要公开 TCP 服务：

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

   启用 TLS 创建 TLS 后端所需的证书，并将其存储在名为 **skupper-tls-<deployment-name>** 的 secret 中。

3. 修改 backend 部署使其包含生成的证书，例如：

```
...
    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>
```

每个站点创建 TLS 客户端所需的证书，并将其存储在名为 **skupper-service-client** 的 secret 中。

4. 修改前端部署使其包含生成的证书，例如：

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

5. 从启用了 TLS 的前端测试调用该服务。

第 6 章 支持的标准和协议

skupper 为您的服务网络支持以下协议：

- TCP - 默认
- HTTP1
- HTTP2

在公开或创建服务时，您可以指定协议，例如：

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

其中 **<protocol>** 可以是：

- tcp
- http
- http2

在选择要指定的协议时，请注意以下几点：

- **tcp** 支持 TCP 上覆盖的任何协议，例如，当您指定 **tcp** 时，HTTP1 和 HTTP2 可以正常工作。
- 如果您指定了 **http** 或 **http2**，客户端报告的 IP 地址可能无法访问。
- 所有服务网络流量将转换为 AMQP 消息，以遍历服务网络。
  TCP 作为单一流消息实施，而 HTTP1 和 HTTP2 则作为请求/响应消息路由实施。

6.1. CLI 选项

有关选项的完整列表，请参阅 Skupper Kubernetes CLI 参考 和 Skupper Podman CLI 参考文档。

警告

当您创建站点并将日志记录级别设置为 **trace** 时，您可以无意记录来自 HTTP 标头的敏感信息。

```
$ skupper init --router-logging trace
```

默认情况下，所有 **skupper** 命令都将应用到您登录的集群和当前命名空间。以下 **skupper** 选项允许您覆盖该行为并适用于所有命令：

--namespace <namespace-name>

将命令应用到 **<namespace-name>**。例如，如果您目前正在处理 **frontend** 命名空间，并希望在 **backend** 命名空间中初始化站点：

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

--kubeconfig <kubeconfig-path>

kubeconfig 文件的路径 - 这允许您从同一客户端运行多个会话。另一种方法是设置 **KUBECONFIG** 环境变量。

--context <context-name>

kubeconfig 文件可以包含定义的上下文，这个选项允许您使用这些上下文。

第 7 章 使用 SKUPPER PODMAN

使用 **skupper** 命令行界面(CLI)可让您从当前 Linux 用户的上下文创建和管理 Skupper 站点。skupper Podman 允许您使用容器创建站点，而无需 Kubernetes。

典型的工作流是创建一个站点，将站点链接在一起，并向服务网络公开服务。

7.1. 关于 SKUPPER PODMAN

skupper Podman 有以下优先级：

skupper --platform podman <command>

使用这个选项以避免更改模式，例如，如果您同时使用 Kubernetes 和 Podman。

export SKUPPER_PLATFORM=podman

使用这个命令将 Skupper Podman 用于当前会话，例如，如果您将两个终端设置为不同的上下文，请使用这个命令。将环境设置为目标 Kubernetes 站点：

```
$ export SKUPPER_PLATFORM=kubernetes
```

skupper switch podman

如果您输入了这个命令，则所有后续命令都会针对所有终端会话为目标 Podman 而不是 Kubernetes。

要确定哪个模式当前处于活跃状态：

```
$ skupper switch

podman
```

要切换回目标 Kubernetes 站点： **skupper switch kubernetes**

注意

在远程站点上公开的服务不会自动提供给 Podman 站点。这等同于使用 **skupper init --enable-service-sync false** 创建的 Kubernetes 站点。

要在 Podman 站点上使用公开的服务，请检查它是否存在原始 **站点上的 skupper** 服务状态，并使用这些信息在 Podman 站点上创建该服务：

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

7.2. 使用 SKUPPER PODMAN 创建站点

服务网络由 Skupper 站点组成。这部分论述了如何使用默认设置在 Linux 主机中创建站点。有关使用 Skupper CLI 创建 Podman 站点的信息，请参阅使用 Skupper CLI。

先决条件

- 已安装最新的 **skupper** CLI。
- podman 已安装，请参阅 https://podman.io/

- **Netavark** 配置为 podman 网络后端。
  默认情况下，Podman v4 使用 Netavark，它可用于 Skupper。

  如果使用 CNI，例如，如果您从 Podman v3 升级，则必须安装 **podman-plugins** 软件包。例如，**dnf install podman-plugins** 用于基于 RPM 的发布。

  注意

  CNI 将在以后的 Netavark 中被弃用。

  检查 **netavark** 是否已配置为 podman network backend:

  ```
  $ podman info | grep networkBackend
  ```

  要在基于 rpm 的 Linux 上安装 **netavark**，如 RHEL8:

  ```
  $ sudo dnf install netavark
  ```

  通过确保 **/etc/containers/containers.conf** 文件中存在以下行，将 podman 配置为使用 **netavark** ：

  ```
  [network]
  network_backend = "netavark"
  ```

- Podman 服务端点。
  使用 **systemctl status podman.socket** 确保 Podman API 套接字正在运行。

  使用 **systemctl --user enable --now podman.socket** 启动 Podman API 套接字。

  有关启用此端点的信息，请参阅 Podman 套接字激活。

流程

1. 将您的会话设置为使用 Skupper Podman：

   ```
   $ export SKUPPER_PLATFORM=podman
   ```

   验证 **skupper** 模式：

   ```
   $ skupper switch

   podman
   ```

2. 创建 Skupper 网站：
   使用以下命令，创建创建令牌以便在任何网络接口上链接的站点：

   ```
   $ skupper init
   ```

   注意

   默认情况下，这个命令会为 podman 站点在 2 分钟后超时。您可以使用 **--timeout** 选项增加时间。

此时会显示以下输出：

```
It is recommended to enable lingering for <username>, otherwise Skupper may not start on boot.
Skupper is now installed for user '<username>'.  Use 'skupper status' to get more information.
```

使用以下命令在系统启动时启动站点服务，并在退出时保留：

```
# loginctl enable-linger <username>
```

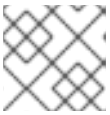
默认情况下，**skupper init** 会尝试包括与本地网络接口关联的所有 IP 地址作为有效的入口主机。您可以使用 **--ingress-host <IP/Hostname&** gt; 将令牌入口限制到特定的网络上下文：

```
$ skupper init --ingress-host my-cloud-vm.example.com
```

如果您不要求其他站点可以链接到您要创建的站点：

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

在本指南中，我们假设您使用第一个命令启用了 ingress。这样，您可以创建令牌，以允许来自主机上每个网络接口的链接。



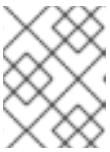
注意

在创建令牌时，您可以指定入口主机。

您还可以在初始化时将入口限制为 IP 地址或主机名，如 Skupper Podman CLI 参考文档中所述。

3. 检查您的站点的状态：

```
$ skupper status
Skupper is enabled for "<username>" with site name "<machine-name>-<username>" in interior mode. It is not connected to any other sites. It has no exposed services.
```



注意

您只能为每个用户创建一个站点。如果您需要主机来支持多个站点，请为每个站点创建一个用户。

7.3. 使用 SKUPPER PODMAN 链接站点

服务网络由 Skupper 站点组成。本节论述了如何将站点链接到构成服务网络。

连接两个站点需要一个初始方向连接。但是：

- 两个站点之间的通信是双向的，只有初始链接是带有方向的。
- 链接的方向通常由可访问性所决定。例如，如果您要将云中运行的虚拟机与防火墙后面的 Linux 主机相关联，则必须从 Linux 主机链接到云虚拟机，因为该路由可以访问。

流程

1. 在一个站点上生成令牌：

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

如果您在没有指定 **ingress-host** 的情况下创建了站点，则令牌对所有网络上下文有效。您可以使用 **--ingress-host <IP/Hostname&** gt; 将令牌入口限制到特定的网络上下文：

```
$ skupper token create <filename> --ingress-host <IP/Hostname>
```

2. 从其他站点创建链接：

```
$ skupper link create <filename>
```

连接到网络后，您可以检查链接状态：

```
$ skupper link status
```

第 8 章 指定链接成本

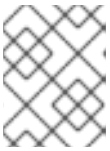
在连接站点时，您可以为每个链接分配一个成本来影响流量流。默认情况下，为新链接将链接成本设置为 **1**。在服务网络中，路由算法尝试使用从客户端到目标服务器的最低总成本的路径。

- 如果您在不同站点间分布服务，您可能需要客户端使用特定目标或链接。在这种情况下，您可以在替代链接中指定大于 **1** 的成本，以减少这些链接的使用。

注意

开放连接的分发是统计数据，即不是循环系统。

- 如果连接只遍历一个链接，则路径成本与链接成本相同。如果连接遍历多个链接，路径成本是路径中涉及的所有链接的总和。
- 成本充当使用从客户端到网络中的服务器的路径的阈值。当只有一个路径时，无论成本如何，该路径上的流量流。



注意

如果您从两个服务的目标开始，并且不再提供其中一个目标，剩余路径上的流量流不再可用，无论成本如何。

- 当多个路径从客户端到服务器实例或服务时，最低成本路径上的流量流，直到连接数量超过替代路径的成本。达到这个打开连接阈值后，新的连接将分散到替代路径和最低的成本路径中。

前提条件

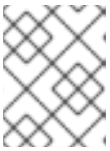
- 您已将 Kubernetes 上下文设置为您要*从其中*链接的站点。
- 要链接*到*的站点的令牌。

流程

1. 创建到服务网络的链接：

```
$ skupper link create <filename> --cost <integer-cost>
```

其中 **<integer-cost>** 是一个大于 1 的整数，流量会优先使用成本链接。



注意

如果可以在不遍历链接的情况下调用服务，该服务被视为本地，且带有隐式成本 **0**。

例如，使用名为 **token.yaml** 的令牌文件创建成本设置为 **2** 的链接：

```
$ skupper link create token.yaml --cost 2
```

2. 检查链接成本：

```
$ skupper link status link1 --verbose
```

输出结果类似以下：

```
Cost:         2
Created:      2022-11-17 15:02:01 +0000 GMT
Name:         link1
Namespace:    default
Site:         default-0d99d031-cee2-4cc6-a761-697fe0f76275
Status:       Connected
```

3. 使用控制台观察流量。
如果您在站点中有一个控制台，请登录并导航到每台服务器的进程。您可以查看与每个客户端对应的流量级别。

注意

如果不同站点上有多个客户端，请将视图过滤到每个客户端，以确定流量成本的影响。例如，在两个站点网络中与两个站点的服务器和客户端链接在一起，您可以看到客户端在本地服务器可用时由本地服务器提供。

8.1. 从 LINUX 主机公开服务网络上的服务

创建服务网络后，公开的服务就可以在该网络间进行通信。

对于 Kubernetes 和 Podman 站点，使用服务的一般流程是相同的。

skupper CLI 有两个选项用于公开主机上已存在的服务：

- **expose** 支持简单的用例，例如，具有单个服务的主机。具体步骤请查看 第 8.1.1 节 “在服务网络上公开简单的服务”。
- **服务创建和 服务绑定** 是一个更灵活的公开服务方法，例如，如果您的主机有多个服务。具体步骤请查看 第 8.1.2 节 “在服务网络上公开复杂的服务”。

8.1.1. 在服务网络上公开简单的服务

这部分论述了如何为服务网络启用服务用于简单用例。

先决条件

- Skupper Podman 网站

流程

1. 运行服务器，例如：

```
$ podman run --name backend-target --network skupper --detach --rm -p 8080:8080
quay.io/skupper/hello-world-backend
```

此步骤不是特定于 Skupper，也就是说，此过程与您的主机的标准进程不同。

2. 创建可在服务网络中进行通信的服务：

```
$ skupper expose [host <hostname|ip>]
```

其中

- **<host** > 是运行服务器的主机的名称。例如，如果您将服务器作为容器运行，则容器的名称。
- **<** IP> 是服务器运行的 IP 地址

对于第 1 步中的示例部署，您可以使用以下命令创建服务：

```
$ skupper expose host backend-target --address backend --port 8080
```

这个命令的选项包括：

- **--port <port-number>**:: 指定该服务在服务网络上提供的端口号。注意：您可以通过重复这个选项来指定多个端口。
- **--target-port <port-number>**:: 指定您要公开的 pod 的端口号。
- **--protocol <protocol>** 允许您指定要使用的协议：**tcp**、**http** 或 **http2**

3. 在服务网络中的另一个站点上创建该服务：

```
$ skupper service create backend 8080
```

8.1.2. 在服务网络上公开复杂的服务

本节描述了如何为服务网络启用服务，以获取更复杂的用例。

先决条件

- Skupper Podman 网站

流程

1. 运行服务器，例如：

```
$ podman run --name backend-target --network skupper --detach --rm -p 8080:8080
quay.io/skupper/hello-world-backend
```

此步骤不是特定于 Skupper，也就是说，此过程与您的主机的标准进程不同。

2. 创建可在服务网络中进行通信的服务：

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

其中

- **<name>** 是您要创建的服务的名称
- **<port>** 是服务使用的端口

对于第 1 步中的示例部署，您可以使用以下命令创建服务：

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

3. 将服务绑定到集群服务：

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

其中

- **<service-name>** 是服务网络上的服务名称
- **<target-type** > 是您要公开的对象， **host** 是唯一有效的值。
- **<target-name>** 是集群服务的名称

对于第 1 步中的示例部署，您可以使用以下命令绑定该服务：

```
$ skupper service bind hello-world-backend host hello-world-backend
```

8.1.3. 从服务网络消耗简单的服务

Podman 站点上公开的服务不自动可供其他站点使用。这等同于使用 **skupper init --enable-service-sync false** 创建的 Kubernetes 站点。

先决条件

- 在服务网络上公开服务的远程站点
- Podman 站点

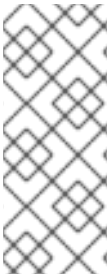
流程

1. 以与 Skupper 站点关联的用户身份登录主机。
2. 创建本地服务：

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

8.2. 删除 PODMAN 站点

当您不再需要 Linux 主机成为服务网络的一部分时，您可以删除该站点。



注意

此流程删除标记为 **application=skupper** 的所有容器、卷和网络。

检查与正在运行的容器关联的标签：

```
$ podman ps -a --format "{{.ID}}  {{.Image}}  {{.Labels}}"
```

流程

1. 确保您已以创建站点的用户身份登录：

```
$ skupper status
Skupper is enabled for "<username>" with site name "<machine-name>-<username>".
```

2. 删除使用 "application=skupper" 标记的站点和所有 podman 资源（容器、卷和网络）：

```
$ skupper delete
Skupper is now removed for user "<username>".
```

第 9 章 使用 SERVICE INTERCONNECT 控制台

Service Interconnect 控制台提供 Skupper 站点之间的流量流的数据和视觉化。

9.1. 启用 SERVICE INTERCONNECT 控制台

默认情况下，当您创建 Skupper 站点时，Service Interconnect 控制台不可用。

启用后，当您使用 **skupper status** 检查站点状态时，会显示 Service Interconnect Console URL。

先决条件

- 计划创建站点的 Kubernetes 命名空间

流程

1. 确定您的服务网络中的哪个站点最适合启用控制台。
   启用控制台还需要启用 flow-collector 组件，这需要资源处理来自所有站点的流量数据。您可以使用以下标准找到控制台：
   - 服务网络是否跨防火墙？例如，如果您希望控制台仅在防火墙中可用，则需要在防火墙内的站点中找到 flow-collector 和控制台。
   - 是否有站点处理超过其他站点的流量？例如，如果您有一个从其他站点调用一组服务的 *frontend* 组件，则在该站点上查找流收集器和控制台，以最小化数据流量。
   - 是否有您要使用的资源或更便宜的网站？例如，如果您有两个站点 A 和 B，且资源在站点 A 上成本更高，您可能希望在站点 B 上找到流收集器和控制台。
2. 创建启用流收集器和控制台的站点：

```
$ skupper init --enable-console --enable-flow-collector
```

9.2. 访问 SERVICE INTERCONNECT 控制台

默认情况下，Service Interconnect 控制台由 **skupper-console-users** secret 中提供的凭证进行保护。

流程

1. 使用 **skupper** CLI 确定 Service Interconnect 控制台 URL，例如：

```
$ skupper status

Skupper is enabled for namespace "west" in interior mode. It is not connected to any other sites. It has no exposed services.
The site console url is:  https://skupper-west.apps-crc.testing
```

2. 浏览到 Service Interconnect 控制台 URL。凭证提示取决于如何使用 **skupper init** 创建站点：
   - 使用 **--console-auth unsecured** 选项时，不会提示您输入凭证。
   - 使用 **--console-auth openshift** 选项时，会提示您输入 OpenShift 集群凭证。

- 使用 default 或 **--console-user <user> --console-password <**password> 选项，会提示您输入这些凭证。

3. 如果您使用默认设置（即 **skupper init** ）创建了站点，则会为 **admin** 用户生成一个随机密码。检索 Kubernetes 站点的 **admin** 用户的密码：

+

```
$ kubectl get secret skupper-console-users -o jsonpath={.data.admin} | base64 -d

JNZWzMHtyg
```

检索 Podman 站点的 **admin** 用户的密码：

+

```
$ cat ~/.local/share/containers/storage/volumes/skupper-console-users/_data/admin

JNZWzMHtyg
```

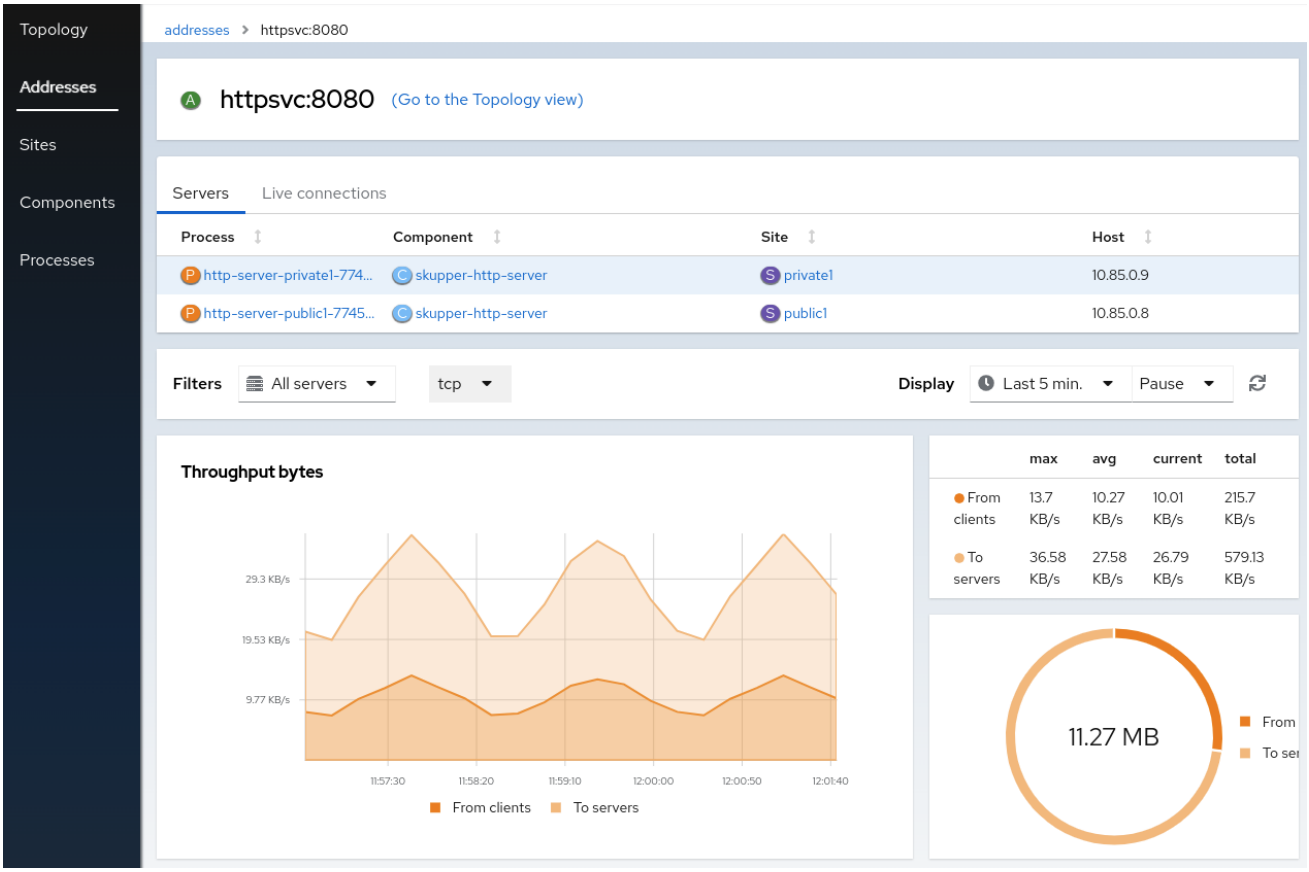
9.3. 探索 SERVICE INTERCONNECT 控制台

在服务网络上公开服务后，您可以创建一个 *地址*，即与站点关联的服务名称和端口号。可能存在很多与地址关联的副本。这些副本在 Service Interconnect 控制台中显示 *为进程*。不是服务网络上的所有参与者都是服务。例如，*前端* 部署可能会调用名为 *backend* 的公开服务，但 frontend 不是服务网络的一部分。在控制台中，会显示这两者，以便您可以查看流量，它们称为 *组件*。

Service Interconnect 控制台提供以下内容概述：

- Topology
- addresses
- Sites
- 组件
- 进程

Service Interconnect 控制台还提供有关服务网络的有用网络信息，如流量级别。



1. 检查 **Sites** 选项卡。所有站点都应列出。请参阅 **Topology** 选项卡来查看站点是如何链接的。
2. 检查您公开的所有服务是否在 **Components** 选项卡中可见。
3. 点一个组件来显示组件详情和相关进程。
4. 单击进程以显示进程流量。

注意

进程详细信息显示关联的镜像、主机和地址。您还可以查看调用进程的客户端。

5. 点 **Addresses** 并选择地址来显示该地址的详情。这显示了在服务网络上公开的一组服务器。

提示

要查看有关每个窗口的信息，请点 **?** 图标。

第 10 章 使用 YAML 配置 SKUPPER 站点

使用 YAML 文件配置 Skupper 允许您使用源控制来跟踪和管理 Skupper 网络更改。

10.1. 使用 YAML 创建 SKUPPER 站点

使用 YAML 文件创建 Skupper 站点，您可以使用源控制跟踪和管理 Skupper 网络更改。

先决条件

- skupper 安装在您要目标的集群或命名空间中。
- 已登陆到集群。

流程

1. 创建一个 YAML 文件来定义站点，如 **my-site.yaml** ：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: skupper-site
data:
  name: my-site
  console: "true"
  console-user: "admin"
  console-password: "changeme"
  flow-collector: "true"
```

YAML 使用控制台创建站点，您可以从此站点创建令牌。

创建没有入口的站点：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: skupper-site
data:
  name: my-site
  ingress: "none"
```

2. 将 YAML 文件应用到集群：

```
kubectl apply -f ~/my-site.yml
```

其他资源

如需了解更多参考，请参阅 第 10.3 节 “站点 ConfigMap YAML 参考” 部分。

10.2. 使用注解配置服务

在创建和连接站点后，您可以使用 Kubernetes 注解来控制服务网络上可用的服务。

10.2.1. 使用注解在服务网络上公开简单的服务

本节提供了 **skupper expose** 命令的替代选择，允许您注解现有资源以在服务网络上公开简单的服务。

先决条件

- 包含您要公开的服务的站点

流程

1. 登录到配置为站点的命名空间中。
2. 在其中一个站点中创建部署、一些 pod 或服务，例如：

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

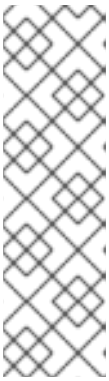
此步骤不是特定于 Skupper，即此过程与集群的标准进程不同。

3. 注解 kubernetes 资源以创建可在服务网络上进行通信的服务，例如：

```
$ kubectl annotate deployment backend "skupper.io/address=backend"
"skupper.io/port=8080" "skupper.io/proxy=tcp"
```

注解包括：

- **skupper.io/proxy** - 要使用的协议： **tcp**、**http** 或 **http2**。这是唯一需要的注解。例如，如果您使用 **skupper.io/proxy=tcp** 注解一个名为 **backend** 的简单部署，该服务会作为后端公开，部署的 **containerPort** 值用作端口号。
- **skupper.io/address** - 服务网络上的服务名称。
- **skupper.io/port** - 服务网络上的服务的一个或多个端口。



注意

在公开服务而非部署等其他资源时，您可以使用 **skupper.io/target** 注解来避免修改原始服务。例如，如果要公开 **后端服务** ：

```
$ kubectl annotate service backend "skupper.io/address=van-backend"
"skupper.io/port=8080" \
"skupper.io/proxy=tcp" "skupper.io/target=backend"
```

这可让您删除并重新创建 **后端服务**，而无需再次应用注解。

4. 检查您是否公开了该服务：

```
$ skupper service status -v
Services exposed through Skupper:
└─ backend:8080 (tcp)
   └─ Sites:
      ├─ 4d80f485-52fb-4d84-b10b-326b96e723b2(west)
      │  policy: disabled
      └─ 316fbe31-299b-490b-9391-7b46507d76f1(east)
```

```
    policy: disabled
    Targets:
      backend:8080 name=backend-9d84544df-rbzjx
```

注意

只有在当前集群中有目标时，才会显示服务的相关目标。

10.2.2. 了解 Skupper 注解

注解允许您公开服务网络上的服务。本节详细介绍了这些注解的范围

skupper.io/address

服务网络上的服务名称。适用于：

- 部署
- StatefulSets
- DaemonSets
- 服务

skupper.io/port

服务网络上的服务端口。适用于：

- 部署
- StatefulSets
- DaemonSets

skupper.io/proxy

要使用的协议： **tcp**、**http** 或 **http2**。适用于：

- 部署
- StatefulSets
- DaemonSets
- 服务

skupper.io/target

要公开的目标服务的名称。适用于：

- 服务

skupper.io/service-labels

以逗号分隔的标签键和值列表。您可以使用此注解为监控公开的服务设置标签。适用于：

- 部署
- DaemonSets

- 服务

10.3. 站点 CONFIGMAP YAML 参考

使用 YAML 文件配置 Skupper 需要了解所有字段，以便置备您需要的站点。

以下 YAML 定义 Skupper 站点：

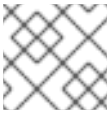
```
apiVersion: v1
data:
  name: my-site
  console: "true"
  flow-collector: "true"
  console-authentication: internal
  console-user: "username"
  console-password: "password"
  cluster-local: "false"
  edge: "false"
  service-sync: "true"
  ingress: "none"
kind: ConfigMap
metadata:
  name: skupper-site
```

name

指定站点名称。

控制台

启用 skupper 控制台，默认为 **false**。



注意

您必须启用 **console** 和 **flow-collector** 才能使控制台正常工作。

flow-collector

启用流收集器，默认为 **false**。

console-authentication

指定 skupper 控制台验证方法。这些选项包括 **openshift**,**internal**,**unsecured**。

console-user

内部身份验证选项 的用户名。

console-password

内部身份验证选项 的密码。

cluster-local

仅接受来自本地集群中的连接，默认为 **false**。

edge

指定是否创建了边缘站点，默认为 **false**。

service-sync

指定服务网络上的服务是否同步，默认为 **true**。

ingress

指定站点是否支持 ingress。如果没有指定值，则启用 Kubernetes 上的默认入口('loadbalancer')，则 OpenShift 上的 'route' 会被启用。这可让您创建可从远程站点使用的令牌。

注意

所有入口类型都支持使用与 **skupper** CLI 相同的参数。

第 11 章 在 KUBERNETES 上使用 SKUPPER OPERATOR

Red Hat Service Interconnect Operator 在 Kubernetes 中创建和管理 Skupper 站点。

11.1. 使用 SKUPPER OPERATOR 创建站点

1. 创建定义您要创建的站点的 ConfigMap 的 YAML 文件。
   例如，创建 **skupper-site.yaml**，以使用控制台置备站点：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: skupper-site
  namespace: my-namespace
data:
  console: "true"
  flow-collector: "true"
  console-user: "admin"
  console-password: "changeme"
```

注意

目前，您必须在与启用流收集器相同的站点中启用控制台。

您还可以在没有控制台的情况下创建站点：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: skupper-site
  namespace: my-namespace
```

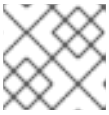
2. 应用 YAML 在您要使用的命名空间中创建一个名为 **skupper-site** 的 ConfigMap：

```
$ kubectl apply -f skupper-site.yaml
```

3. 通过检查 Skupper 路由器和服务控制器 pod 是否正在运行来验证站点是否已创建：

```
$ kubectl get pods

NAME                                          READY   STATUS    RESTARTS   AGE
skupper-router-8c6cc6d76-27562                1/1     Running   0          40s
skupper-service-controller-57cdbb56c5-vc7s2   1/1     Running   0          34s
```



注意

如果您将 Operator 部署到单个命名空间，则还需要运行额外的站点控制器 pod。

第 12 章 使用 SKUPPER 策略保护服务网络

默认情况下，Skupper 包括许多安全功能，包括将 mutual TLS 用于站点之间的所有服务网络通信。您可以通过安装 Skupper 策略 CRD 来添加额外的安全功能。默认情况下，将 Skupper 策略 CRD 应用到集群可防止所有服务网络通信。您可以指定粒度 Skupper 策略 CR，以只允许您需要的服务网络通信。

注意

Skupper 策略与 Kubernetes 网络策略不同，这是 **network-policy** 选项，它限制对当前命名空间的访问 Skupper 服务，如 使用 Skupper CLI 所述。

服务网络中的每个站点都运行 Skupper 路由器，并具有专用证书颁发机构(CA)。站点之间的通信是通过 mutual TLS 保护的，因此服务网络会隔离外部访问，从而防止安全攻击、恶意软件和数据利用的问题。一组 Skupper 策略在集群级别上添加另一个层，以帮助集群管理员控制对服务网络的访问。

本指南假定您了解以下 Skupper 概念：

站点（site）

安装 Skupper 的命名空间。

token

在两个站点间建立链接需要一个令牌。

服务网络（service network）

使用 Skupper 公开服务后，您已创建了服务网络。

12.1. 关于 SKUPPER 策略

集群管理员安装 Skupper 策略自定义资源定义(CRD)后，集群管理员需要配置一个或多个策略，以便开发人员在服务网络上创建和使用服务。

注意

在本指南中，开发人员指的是有权访问命名空间但没有管理员特权的集群用户。

集群管理员使用自定义资源 (CR) 配置一个或多个项目以启用通信：

允许进入的链接

*使用 **allowIncomingLinks** 允许开发人员创建令牌并配置传入的链接。*

允许到特定主机的传出链接

*使用 **allowedOutgoingLinksHostnames** 指定开发人员可以创建链接到的主机。*

允许服务

*使用 **allowedServices** 指定开发人员可以在服务网络上创建和使用的服务。*

允许公开资源

*使用 **allowedExposedResources** 指定开发人员可在服务网络上公开的资源。*

注意

集群管理员可以将每个策略 CR 设置应用到一个或多个命名空间。

例如，以下策略 CR 完全允许所有命名空间上的所有 Skupper 功能，但以下情况除外：

- 仅允许到以 **.example.com** 结尾的任何域的传出链接。
- 仅允许 'deployment/nginx' 资源在服务网络中公开。

```
apiVersion: skupper.io/v1alpha1
kind: SkupperClusterPolicy
metadata:
  name: cluster-policy-sample-01
spec:
  namespaces:
    - "*"
  allowIncomingLinks: true
  allowedExposedResources:
    - "deployment/nginx"
  allowedOutgoingLinksHostnames: [".*\.example.com$"]
  allowedServices:
    - "*"
```

注意

您可以应用许多策略 CR，如果允许的项目中存在冲突，则应用最宽松的策略。例如，如果您使用 **allowedOutgoingLinksHostnames: []** 行（它没有列出任何主机名）应用一个额外的策略 CR，则仍然允许到 ***.example.com** 的传出链接，因为在原始 CR 中允许。

命名空间

指定此策略应用到的命名空间的一个或多个模式。请注意，您可以使用标签选择器来匹配命名空间。

allowIncomingLinks

指定 **true** 以允许其他站点创建到指定命名空间的链接。

allowedOutgoingLinksHostnames

指定一个或多个模式，以确定您可以从指定命名空间创建指向的主机。

allowedServices

指定一个或多个模式，以确定指定命名空间中的服务网络中允许的服务的名称。

allowedExposedResources

指定一个或多个允许的资源名称，以允许来自特定命名空间中的服务网络。请注意，不支持模式。

提示

使用正则表达式来创建模式匹配，例如：

- **ipahealthcheck\.com$** 匹配以 **.com** 结尾的任何字符串。需要使用双引号以避免 YAML 出现问题。
- **^abc$** 匹配字符串 **abc**。

如果您创建另一个 Skupper 策略 CR 以允许特定命名空间的传出链接，用户可以从该命名空间中创建链接来加入服务网络。也就是说，多个策略 CR 的逻辑是 **OR**。如果任何单个策略 CR 允许操作，则允许操作。

12.2. 安装 SKUPPER 策略 CRD

安装 Skupper 策略 CRD 可让集群管理员为服务网络强制执行策略。

注意

如果集群中存在现有站点，请参阅第 12.3 节 "使用现有站点在集群中安装 Skupper 策略 CRD" 避免服务网络中断。

先决条件

- 使用 **cluster-admin** 帐户访问集群
- 已安装 Skupper operator

流程

1. 使用 **cluster-admin** 帐户登录到集群。
2. 下载 CRD：

```
$ wget
https://raw.githubusercontent.com/skupperproject/skupper/1.5/api/types/crds/skupper_cluster_policy_crd.yaml
```

3. 应用 CRD：

```
$ kubectl apply -f skupper_cluster_policy_crd.yaml

customresourcedefinition.apiextensions.k8s.io/skupperclusterpolicies.skupper.io created
clusterrole.rbac.authorization.k8s.io/skupper-service-controller created
```

4. 要验证 Skupper 策略是否活跃，请使用 **skupper status** 命令并检查输出是否包含以下行：

```
Skupper is enabled for namespace "<namespace>" in interior mode (with policies).
```

12.3. 使用现有站点在集群中安装 SKUPPER 策略 CRD

如果集群已经托管 Skupper 站点，请在安装 CRD 前记录以下内容：

- 所有现有的连接都会关闭。您必须应用策略 CR 来重新打开连接。
- 所有现有的服务网络服务和公开资源都将被移除。您必须再次创建这些资源。

流程

避免中断：

1. 为适当的时间规划 CRD 部署。
2. 为站点搜索集群：

```
$ kubectl get pods --all-namespaces --selector=app=skupper
```

3. 记录服务网络上公开的每个服务和资源。

4. 按照 第 12.2 节 “安装 Skupper 策略 CRD” 所述安装 CRD。此步骤关闭连接并删除所有服务网络服务和公开资源。

5. 如果集群中没有由 **cluster-admin** 创建的 Skupper 站点，则必须授予读取 Skupper 策略的权限，以避免该站点被服务网络阻止。
对于每个站点命名空间：

```
$ kubectl create clusterrolebinding skupper-service-controller-<namespace> --clusterrole=skupper-service-controller --serviceaccount=<namespace>:skupper-service-controller
```

其中 **<namespace>** 是站点命名空间。

6. 创建 Skupper 策略 CR，如 所述 第 12.4 节 “创建 Skupper 策略 CR”

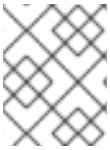
7. 根据需要重新创建任何服务和公开的资源。

12.4. 创建 SKUPPER 策略 CR

skupper Policy CR 允许集群管理员从集群中控制跨服务网络的通信。

先决条件

- 使用 **cluster-admin** 帐户访问集群。
- Skupper 策略 CRD 已安装在集群中。



流程

通常，您可以创建一个 Skupper 策略 CR，它将以下步骤中的多个元素合并。请参阅 第 12.1 节 “关于 Skupper 策略” 中的一个示例 CR。

1. 第 12.4.1 节 “实施策略以允许传入的链接”
2. 第 12.4.2 节 “实施策略以允许到特定主机的传出链接”
3. 第 12.4.3 节 “实施策略以允许特定的服务”
4. 第 12.4.4 节 “实施策略以允许特定资源”

12.4.1. 实施策略以允许传入的链接

使用 **allowIncomingLinks** 允许开发人员创建令牌并配置传入的链接。

流程

1. 决定您要将此策略应用到的命名空间。
2. 创建 CR，并将 **allowIncomingLinks** 设为 **true** 或 **false**。
3. 创建并应用 CR。

例如，以下 CR 允许所有命名空间的传入链接：

```
apiVersion: skupper.io/v1alpha1
kind: SkupperClusterPolicy
metadata:
  name: allowincominglinks
spec:
  namespaces:
    - "*"
  allowIncomingLinks: true
```

12.4.2. 实施策略以允许到特定主机的传出链接

使用 **allowedOutgoingLinksHostnames** 指定开发人员可以创建链接到的主机。您无法创建 **allowedOutgoingLinksHostnames** 策略来禁止之前允许的特定主机。

1. 决定您要将此策略应用到的命名空间。
2. 创建一个 CR，并将 **allowedOutgoingLinksHostnames** 设置为允许的主机模式。
3. 创建并应用 CR。

例如，以下 CR 允许链接到所有命名空间的 **example.com** 的所有子域：

```
apiVersion: skupper.io/v1alpha1
kind: SkupperClusterPolicy
metadata:
  name: allowedoutgoinglinkshostnames
spec:
  namespaces:
    - "*"
  allowedOutgoingLinksHostnames: ['.*\.example\.com']
```

12.4.3. 实施策略以允许特定的服务

使用 **allowedServices** 指定开发人员可以在服务网络上创建和使用的服务。您无法创建 **allowedServices** 策略来禁止之前允许的特定服务。

流程

1. 决定您要将此策略应用到的命名空间。
2. 创建具有 **allowedServices** 设置的 CR，以指定服务网络中允许的服务。
3. 创建并应用 CR。

例如，以下 CR 允许用户公开和使用所有命名空间的前缀 **backend-** 的服务：

```
apiVersion: skupper.io/v1alpha1
kind: SkupperClusterPolicy
metadata:
  name: allowedservices
spec:
  namespaces:
    - "*"
  allowedServices: ['^backend-']
```

注意

在公开服务时，您可以使用 **skupper** CLI 的 **--address <name>** 参数来命名服务以匹配您的策略。

12.4.4. 实施策略以允许特定资源

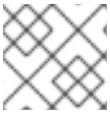
使用 **allowedExposedResources** 指定开发人员可在服务网络上公开的资源。您无法创建 **allowedExposedResources** 策略来禁止之前允许的特定资源。

流程

1. 决定您要将此策略应用到的命名空间。
2. 创建带有 **allowedExposedResources** 设置的 CR，以指定开发人员可以在服务网络上公开的资源。
3. 创建并应用 CR。

例如，以下 CR 允许您为所有命名空间公开 **nginx** 部署：

```
apiVersion: skupper.io/v1alpha1
kind: SkupperClusterPolicy
metadata:
  name: allowedexposedresources
spec:
  namespaces:
    - "*"
  allowedExposedResources: ['deployment/nginx']
```



注意

对于 **allowedExposedResources**，每个条目都必须符合 **type/name** 语法。

第 13 章 服务网络故障排除

通常，您可以创建服务网络，而无需引用此故障排除指南。但是，本指南为服务网络无法按预期工作的情况提供了一些提示。

如果您使用 **skupper** CLI 遇到特定问题，请参阅 第 13.8 节 "解决常见问题"。

典型的故障排除工作流是检查所有站点并创建调试 tar 文件。

13.1. 检查站点

使用 **skupper** 命令行界面(CLI)提供了开始使用 Skupper 故障排除的简单方法。

流程

1. 检查站点状态：

```
$ skupper status --namespace west

Skupper is enabled for namespace "west" in interior mode. It is connected to 2 other sites. It has 1 exposed services.
```

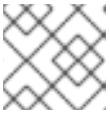
输出显示：

- 一个站点存在于指定命名空间中。
- 其他站点存在链接。
- 服务在服务网络上公开，可从此命名空间中访问。

2. 检查服务网络：

```
$ skupper network status
Sites:
├─ [local] a960b766-20bd-42c8-886d-741f3a9f6aa2(west)
│  │ namespace: west
│  │ site name: west
│  │ version: 1.5.1
│  └─ Linked sites:
│     ├─ 496ca1de-0c80-4e70-bbb4-d0d6ec2a09c0(east)
│     │  direction: outgoing
│     └─ 484cccc3-401c-4c30-a6ed-73382701b18a()
│        direction: incoming
├─ [remote] 496ca1de-0c80-4e70-bbb4-d0d6ec2a09c0(east)
│  │ namespace: east
│  │ site name: east
│  │ version: 1.5.1
│  └─ Linked sites:
│     └─ a960b766-20bd-42c8-886d-741f3a9f6aa2(west)
│        direction: incoming
└─ [remote] 484cccc3-401c-4c30-a6ed-73382701b18a()
   │ site name: vm-user-c3d98
   │ version: 1.5.1
```

```
└─ Linked sites:
  └─ a960b766-20bd-42c8-886d-741f3a9f6aa2(west)
    direction: outgoing
```



注意

如果输出不是预期的，您可能需要在继续前 检查链接。

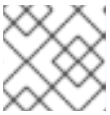
输出显示：

- 服务网络上有 3 个站点 **vm-user-c3d98**、**east** 和 **west**。
- 每个站点的详情，如命名空间名称。

3. 检查服务网络上公开的服务状态(**-v** 仅在 Kubernetes 上可用)：

```
$ skupper service status -v
Services exposed through Skupper:
└─ backend:8080 (tcp)
  └─ Sites:
    ├─ 4d80f485-52fb-4d84-b10b-326b96e723b2(west)
    │  policy: disabled
    └─ 316fbe31-299b-490b-9391-7b46507d76f1(east)
      │ policy: disabled
      └─ Targets:
        └─ backend:8080 name=backend-9d84544df-rbzjx
```

输出显示了 **后端服**务 以及该服务的相关目标。



注意

作为输出每个站点的一部分，会报告该集群上的策略系统状态。

4. 列出站点的 Skupper 事件：

```
$ skupper debug events
NAME                    COUNT                                          AGE
GatewayQueryRequest     3                                              9m12s
                        3    gateway request                           9m12s
SiteQueryRequest        3                                              9m12s
                        3    site data request                         9m12s
ServiceControllerEvent  9                                              10m24s
                        2    service event for west/frontend           10m24s
                        1    service event for west/backend            10m26s
                        1    Checking service for: backend             10m26s
                        2    Service definitions have changed          10m26s
                        1    service event for west/skupper-router     11m4s
DefinitionMonitorEvent  15                                             10m24s
                        2    service event for west/frontend           10m24s
                        1    service event for west/backend            10m26s
                        1    Service definitions have changed          10m26s
                        5    deployment event for west/frontend        10m34s
                        1    deployment event for west/skupper-service-controller  11m4s
ServiceControllerUpdateEvent 1                                         10m26s
```

```
                1    Updating skupper-internal                    10m26s
ServiceSyncEvent        3                                         10m26s
                1    Service interface(s) added backend           10m26s
                1    Service sync sender connection to            11m4s
                     amqps://skupper-router-local.west.svc.cluster.local:5671
                     established
                1    Service sync receiver connection to          11m4s
                     amqps://skupper-router-local.west.svc.cluster.local:5671
                     established
IpMappingEvent          5                                         10m34s
                1    172.17.0.7 mapped to frontend-6b4688bf56-rp9hc    10m34s
                2     mapped to frontend-6b4688bf56-rp9hc          10m54s
                1    172.17.0.4 mapped to                         11m4s
                     skupper-service-controller-6c97c5cf5d-6nzph
                1    172.17.0.3 mapped to skupper-router-547dffdcbf-l8pdc    11m4s
TokenClaimVerification  1                                         10m59s
                1    Claim for efe3a241-3e4f-11ed-95d0-482ae336eb38 succeeded
10m59s
```

输出显示正在链接的站点，以及服务网络上公开的服务。但是，当报告问题时，此输出最有用，并包含在 Skupper debug tar 文件中。

5. *列出站点的 Kubernetes 事件：*

```
kubectl get events | grep "deployment/skupper-service-controller"
10m         Normal    ServiceSyncEvent              deployment/skupper-service-controller
Service sync receiver connection to amqps://skupper-router-
local.private1.svc.cluster.local:5671 established
10m         Normal    ServiceSyncEvent              deployment/skupper-service-controller
Service sync sender connection to amqps://skupper-router-
local.private1.svc.cluster.local:5671 established
10m         Normal    ServiceControllerCreateEvent  deployment/skupper-service-controller
Creating service productcatalogservice
7m59s       Normal    TokenHandler                  deployment/skupper-service-controller
Connecting using token link1
7m54s       Normal    TokenHandler                  deployment/skupper-service-controller
Connecting using token link2
```

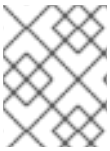
输出显示与 Kubernetes 资源相关的事件。

附加信息

- *第 13.2 节 "检查链接"*

13.2. 检查链接

在服务网络上公开服务前，您必须链接站点。



注意

*默认情况下，令牌在 5 分钟后过期，您只能使用令牌一次。如果链接没有连接，请生成新的令牌。您还可以使用 **-token-type cert** 选项为持久性可重复使用的令牌生成令牌。*

本节概述了检查链接的一些高级选项。

1. 检查链接状态：

```
$ skupper link status --namespace east

Links created from this site:
-----------------------------
Link link1 is connected
```

一个链接存在于指定站点到另一个站点，这意味着另一个站点中的令牌被应用到指定的站点。

注意

仅在连接的站点上运行 **skupper** 链接状态，只有在令牌用于创建链接时才会生成输出。

如果您在没有创建链接的站点中使用这个命令，但站点有一个传入链接：

```
$ skupper link status --namespace west

Links created from this site:
-----------------------------
There are no links configured or connected

Currently connected links from other sites:
-------------------------------------------
A link from the namespace east on site east(536695a9-26dc-4448-b207-519f56e99b71) is connected
```

2. 检查详细链接状态：

```
$ skupper link status link1 --verbose --namespace east

Cost:         1
Created:      2022-10-24 12:50:33 +0100 IST
Name:         link1
Namespace:    east
Site:         east-536695a9-26dc-4448-b207-519f56e99b71
Status:       Connected
```

输出显示该链接的详细信息，包括链接创建时的时间戳，以及使用链接的相关相对成本。

链接的状态必须是 **Connected**，以允许服务流量。

附加信息

- 第 13.1 节 "检查站点"

13.3. 检查网关

默认情况下，**skupper 网关** 会创建一个服务类型网关，这些网关在机器重启后正确运行。

但是，如果您创建 docker 或 podman 类型网关，请检查容器是否在机器重启后运行。例如：

1. 检查 Skupper 网关的状态：

```
$ skupper gateway status

Gateway Definition:
└─ machine-user type:podman version:1.5
   └─ Bindings:
      └─ mydb:3306 tcp mydb:3306 localhost 3306
```

这显示了一个 podman 类型网关。

2. 检查容器是否正在运行：

```
$ podman ps
CONTAINER ID  IMAGE                                COMMAND               CREATED
STATUS            PORTS                   NAMES
4e308ef8ee58  quay.io/skupper/skupper-router:1.5            /home/skrouterd/b...  26 seconds
ago  Up 27 seconds ago                 machine-user
```

这将显示正在运行的容器。

注意

要查看停止的容器，请使用 **podman ps -a** 或 **docker ps -a**。

3. 如果需要，启动容器：

```
$ podman start machine-user
```

13.4. 检查策略

作为开发者，您可能不知道您的站点应用的 Skupper 策略。按照以下步骤探索应用到站点的策略。

流程

1. 登录到初始化 Skupper 站点的命名空间。
2. 检查是否允许传入的链接：

```
$ kubectl exec deploy/skupper-service-controller -- get policies incominglink

ALLOWED POLICY ENABLED ERROR                                          ALLOWED BY
false   true           Policy validation error: incoming links are not allowed
```

在本例中，策略不允许传入的链接。

3. 检查其他策略：

```
$ kubectl exec deploy/skupper-service-controller -- get policies
Validates existing policies

Usage:
  get policies [command]

Available Commands:
```

```
expose       Validates if the given resource can be exposed
incominglink Validates if incoming links can be created
outgoinglink Validates if an outgoing link to the given hostname is allowed
service      Validates if service can be created or imported
```

如上所示，通过指定您要执行的操作来检查每种策略类型的命令，例如，检查是否可以公开 nginx 部署：

```
$ kubectl  exec deploy/skupper-service-controller -- get policies expose deployment nginx
ALLOWED POLICY ENABLED ERROR                                                ALLOWED BY
false   true           Policy validation error: deployment/nginx cannot be exposed
```

如果允许 nginx 部署，则同一命令会显示允许该资源，并显示启用它的策略 CR 的名称：

```
$ kubectl  exec deploy/skupper-service-controller -- get policies expose deployment nginx
ALLOWED POLICY ENABLED ERROR                                                ALLOWED BY
true    true                                                                allowedexposedresources
```

13.5. 创建 SKUPPER 调试 TAR 文件

debug tar 文件包含来自站点的 Skupper 组件的所有日志，并提供了帮助调试问题的详细信息。

1. 创建 debug tar 文件：

```
$  skupper debug dump my-site

Skupper dump details written to compressed archive: `my-site.tar.gz`
```

2. 您可以使用以下命令扩展该文件：

```
$ tar -xvf kind-site.tar.gz

k8s-versions.txt
skupper-versions.txt
skupper-router-deployment.yaml
skupper-router-867f5ddcd8-plrcg-skstat-g.txt
skupper-router-867f5ddcd8-plrcg-skstat-c.txt
skupper-router-867f5ddcd8-plrcg-skstat-l.txt
skupper-router-867f5ddcd8-plrcg-skstat-n.txt
skupper-router-867f5ddcd8-plrcg-skstat-e.txt
skupper-router-867f5ddcd8-plrcg-skstat-a.txt
skupper-router-867f5ddcd8-plrcg-skstat-m.txt
skupper-router-867f5ddcd8-plrcg-skstat-p.txt
skupper-router-867f5ddcd8-plrcg-router-logs.txt
skupper-router-867f5ddcd8-plrcg-config-sync-logs.txt
skupper-service-controller-deployment.yaml
skupper-service-controller-7485756984-gvrf6-events.txt
skupper-service-controller-7485756984-gvrf6-service-controller-logs.txt
skupper-site-configmap.yaml
skupper-services-configmap.yaml
skupper-internal-configmap.yaml
skupper-sasl-config-configmap.yaml
```

这些文件可用于提供对 Skupper 的支持，但您可以检查的一些项目：

版本

有关各种组件的版本，请参阅 "版本.txt "。

ingress

*请参阅 **skupper-site-configmap.yaml** 以确定站点的 **ingress** 类型。*

链接和服务

*请参阅 **skupper-service-controller backed-events.txt** 文件，以查看令牌使用情况和服务暴露的详情。*

13.6. 了解 SKUPPER 大小

2023 年 9 月，会执行大量测试来在不同路由器 CPU 分配时探索 Skupper 性能。您可以查看 大小指南 的结果。

路由器 CPU 和内存的结论如下所示。

路由器 CPU

当您为工作负载扩展 Skupper 时，需要考虑的主要因素是路由器 CPU。（请注意，由于集群入口和连接路由的性质，务必要专注于垂直扩展路由器，而不是水平扩展。）

每个路由器有两个 CPU 内核(2,000 millicore)是一个很好的起点。它包括一些工作室，并为大量工作负载提供低延迟。

如果您的工作负载所需的峰值吞吐量较低，则可以通过路由器 CPU 更少的延迟来实现满意的延迟。

有些工作负载对网络延迟非常敏感。在这些情况下，路由器引入的开销可能会限制难以达到的吞吐量。即，每个路由器需要大于两个内核的 CPU 数值时。

在 flip side 上，一些工作负载接受网络延迟。在这些情况下，一个内核或更少可能足够了。

这些基准测试结果不是最后一词。它们依赖于我们的测试环境的具体内容。要更好地了解 Skupper 在环境中如何执行，您可以自行运行这些基准。

路由器内存

路由器内存通过打开的连接数量进行扩展。通常，良好的起点为 4G。

内存	**并发打开连接**	
512M	8,192	
1G	16,384	
2G	32,768	
4G	**65,536**	
8G	131,072	
16G	262,144	

32G	524,288	
64G	104,8576	

13.7. 提高 SKUPPER 路由器性能

如果您遇到 Skupper 路由器性能问题，您可以扩展 Skupper 路由器来解决这些问题。

注意

目前，您必须删除并重新创建站点来重新配置 Skupper 路由器。

例如，使用此流程提高吞吐量，如果您有多个客户端、延迟。

1. 删除您的站点或在其他命名空间中创建新站点。
   记录所有配置并删除您的现有站点：

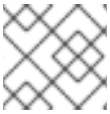
   ```
   $ skupper delete
   ```

   另外，您可以创建新命名空间并配置具有优化 Skupper 路由器性能的新站点。验证性能提升后，您可以删除并重新创建原始站点。

2. 创建具有最佳性能 CPU 设置的站点：

   ```
   $ skupper init --router-cpu 5
   ```

3. 从第 1 步重新创建配置，重新创建链接和服务。



注意

虽然您可以通过扩展路由器数量来解决可用性问题，但通常不需要这样做。

13.8. 解决常见问题

在评估 Skupper 时，以下问题和临时解决方案可能会帮助您调试简单的场景。

无法初始化 skupper

如果 **skupper init** 命令失败，请考虑以下选项：

- 检查负载均衡器。
  如果您要评估 minikube 上的 Skupper，请使用以下命令创建负载均衡器：

  ```
  $ minikube tunnel
  ```

  有关其他 Kubernetes 类型，请查看您的提供程序的文档。

- 在没有 ingress 的情况下初始化。
  这个选项可防止其他站点链接到此站点，但支持链接。建立了链接后，流量就可以在任一方向上流动。输入以下命令：

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

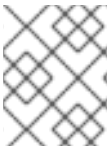
注意

有关 **skupper init**，请参阅 Skupper Podman CLI 参考文档。

无法链接站点

要链接两个站点，必须可以从其他站点访问一个站点。例如，如果一个站点位于防火墙后面，另一个站点位于 AWS 集群中，则必须：

1. 在 AWS 集群站点上创建令牌。
2. 在防火墙内在站点上创建链接。



注意

默认情况下，令牌仅有效 15 分钟，且只能使用一次。有关创建不同类型的 令牌的更多信息，请参阅使用 Skupper 令牌。

无法访问 Skupper 控制台

从 Skupper 版本 1.3 开始，控制台不会被默认启用。要使用新控制台，请参阅使用控制台。

使用 **skupper status** 查找控制台 URL。

使用以下命令显示 **admin** user:doctype 的密码： 文章

```
$ kubectl get secret/skupper-console-users -o jsonpath={.data.admin} | base64 -d
```

无法创建用于链接集群的令牌

一些原因是您在创建令牌时可能遇到困难：

站点未就绪

创建站点后，您可能会在创建令牌时看到以下信息：

```
Error: Failed to create token: Policy validation error: Skupper is not enabled in namespace
```

使用 **skupper status** 验证站点是否正常工作，并尝试再次创建令牌。

没有入口

在使用 **skupper token create** 命令后，您可能会看到以下备注：

```
Token written to <path> (Note: token will only be valid for local cluster)
```

此输出显示站点没有 ingress 选项部署。例如 **skupper init --ingress none**。您必须指定一个入口，以允许其他集群中的站点链接到您的站点。

您还可以使用 **skupper token create** 命令检查在创建站点时是否指定了 ingress。