Red Hat Service Interconnect 1.5

Using Service Interconnect

Creating a service network with Skupper CLI and YAML

Last Updated: 2024-04-19

Red Hat Service Interconnect 1.5 Using Service Interconnect

Creating a service network with Skupper CLI and YAML

Legal Notice

Abstract

Red Hat Service Interconnect is a Red Hat build of the open source Skupper project. This Skupper documentation is reproduced for reference.

Table of Contents

CHAPTER 1. USING THE SKUPPER CLI

Using the **skupper** command-line interface (CLI) allows you to create and manage Skupper sites from the context of the current namespace.

A typical workflow is to create a site, link sites together, and expose services to the service network.

1.1. CHECKING THE SKUPPER CLI

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

Procedure

1. Verify the installation.

```
$ skupper version
client version 1.5.2-rh-1
```

1.2. CREATING A SITE USING THE CLI

A service network consists of Skupper sites. This section describes how to create a site in a Kubernetes cluster using the default settings. See link:{podman-link} for information about using the Skupper CLI to create Podman sites.

Prerequisites

- The **skupper** CLI is installed.
- You are logged into the cluster.
- The services you want to expose on the service network are in the active namespace.

Procedure

1. Create a default site:

   ```
   $ skupper init
   ```

   Starting with Skupper release 1.3, the console is not enabled by default. To use the new console, see Using the console.

2. Check the site:

   ```
   $ skupper status
   ```

   The output should look similar to the following:

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

NOTE

The default message above is displayed when you initialize a site on a cluster that does not have a Skupper policy installed. If you install a Skupper policy as described in Securing a service network using policies, the message becomes **Skupper is enabled for namespace "west" in interior mode (with policies)**.

By default, the site name defaults to the namespace name, for example, **west**.

1.3. CUSTOM SITES

The default **skupper init** creates sites that satisfy typical requirements.

Starting with Skupper release 1.3, the console is not enabled by default. To use the new console, see Using the console.

If you require a custom configuration, note the following options:

- Configuring console authentication. There are several **skupper** options regarding authentication for the console:

  --console-auth <authentication-mode>

  Set the authentication mode for the console:

  - **openshift** - Use OpenShift authentication, so that users who have permission to log into OpenShift and view the Project (namespace) can view the console.
  - **internal** - Use Skupper authentication, see the **console-user** and **console-password** options.
  - **unsecured** - No authentication, anyone with the URL can view the console.

  --console-user <username>

  Username for the console user when authentication mode is set to **internal**. Defaults to **admin**.

  --console-password <password>

  Password for the console user when authentication mode is set to **internal**. If not specified, a random passwords is generated.

- Configuring service access

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

NOTE

All sites are associated with a namespace, called the *active namespace* in this procedure.

Services in the active namespace may be accessible to pods in other namespaces on that cluster by default, depending on your cluster network policies. As a result, you can expose services to pods in namespaces not directly connected to the service network. This setting applies a Kubernetes network policy to restrict access to services to those pods in the active namespace.

For example, if you create a site in the namespace **projectA** of **clusterA** and link that site to a service network where the **database** service is exposed, the **database** service is available to pods in **projectB** of **clusterA**.

You can use the **--create-network-policy** option to restrict the **database** service access to **projectA** of **clusterA**.

1.4. LINKING SITES

A service network consists of Skupper sites. This section describes how to link sites to form a service network.

Linking two sites requires a single initial directional connection. However:

- Communication between the two sites is bidirectional, only the initial linking is directional.
- The choice of direction for linking is typically determined by accessibility. For example, if you are linking an OpenShift Dedicated cluster with a CodeReady Containers cluster, you must link from the CodeReady Containers cluster to the OpenShift Dedicated cluster because that route is accessible.

Procedure

1. Determine the direction of the link. If both clusters are publicly addressable, then the direction is not significant. If one of the clusters is addressable from the other cluster, perform step 2 below on the addressable cluster.

2. Generate a token on the cluster that you want to link to:

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

where **<filename>** is the name of a YAML file that is saved on your local filesystem.

This file contains a key and the location of the site that created it.

NOTE

Access to this file provides access to the service network. Protect it appropriately.

For more information about protecting access to the service network, see Using Skupper tokens.

3. Use a token on the cluster that you want to connect from:
To create a link to the service network:

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

where **<filename>** is the name of a YAML file generated from the **skupper token create** command and **<link-name>** is the name of the link.

To check the link:

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

In this example no <link-name> was specified, the name defaulted to **link1**.

To delete a link:

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

where **<link-name>** is the name of the link specified during creation.

1.5. SPECIFYING LINK COST

When linking sites, you can assign a cost to each link to influence the traffic flow. By default, link cost is set to **1** for a new link. In a service network, the routing algorithm attempts to use the path with the lowest total cost from client to target server.

- If you have services distributed across different sites, you might want a client to favor a particular target or link. In this case, you can specify a cost of greater than **1** on the alternative links to reduce the usage of those links.

  NOTE

  The distribution of open connections is statistical, that is, not a round robin system.

- If a connection only traverses one link, then the path cost is equal to the link cost. If the connection traverses more than one link, the path cost is the sum of all the links involved in the path.

- Cost acts as a threshold for using a path from client to server in the network. When there is only one path, traffic flows on that path regardless of cost.

  NOTE

  If you start with two targets for a service, and one of the targets is no longer available, traffic flows on the remaining path regardless of cost.

- When there are a number of paths from a client to server instances or a service, traffic flows on the lowest cost path until the number of connections exceeds the cost of an alternative path. After this threshold of open connections is reached, new connections are spread across the alternative path and the lowest cost path.

Prerequisite

- You have set your Kubernetes context to a site that you want to link *from*.

- A token for the site that you want to link *to*.

Procedure

1. Create a link to the service network:

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

where **<integer-cost>** is an integer greater than 1 and traffic favors lower cost links.

NOTE

If a service can be called without traversing a link, that service is considered local, with an implicit cost of **0**.

For example, create a link with cost set to **2** using a token file named **token.yaml**:

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

2. Check the link cost:

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

The output is similar to the following:

```
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. Observe traffic using the console.
If you have a console on a site, log in and navigate to the processes for each server. You can view the traffic levels corresponding to each client.

NOTE

If there are multiple clients on different sites, filter the view to each client to determine the effect of cost on traffic. For example, in a two site network linked with a high cost with servers and clients on both sites, you can see that a client is served by the local servers while a local server is available.

1.6. EXPOSING SERVICES ON THE SERVICE NETWORK FROM A NAMESPACE

After creating a service network, exposed services can communicate across that network.

The **skupper** CLI has two options for exposing services that already exist in a namespace:

- **expose** supports simple use cases, for example, a deployment with a single service. See Section 1.6.1, "Exposing simple services on the service network" for instructions.
- **service create** and **service bind** is a more flexible method of exposing services, for example, if you have multiple services for a deployment. See Section 1.6.2, "Exposing complex services on the service network" for instructions.

1.6.1. Exposing simple services on the service network

This section describes how services can be enabled for a service network for simple use cases.

Procedure

1. Create a deployment, some pods, or a service in one of your sites, for example:

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

   This step is not Skupper-specific, that is, this process is unchanged from standard processes for your cluster.

2. Create a service that can communicate on the service network:

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

   where

   - **<name>** is the name of your deployment
   - **<selector>** is a pod selector
   - **<statefulsetname>** is the name of a statefulset

   For the example deployment in step 1, you create a service using the following command:

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

   Options for this command include:

   - **--port <port-number>**:: Specify the port number that this service is available on the service network. NOTE: You can specify more than one port by repeating this option.
   - **--target-port <port-number>**:: Specify the port number of pods that you want to expose.
   - **--protocol <protocol>** allows you specify the protocol you want to use, **tcp**, **http** or **http2**

NOTE

If you do not specify ports, **skupper** uses the **containerPort** value of the deployment.

1.6.2. Exposing complex services on the service network

This section describes how services can be enabled for a service network for more complex use cases.

Procedure

1. Create a deployment, some pods, or a service in one of your sites, for example:

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

This step is not Skupper-specific, that is, this process is unchanged from standard processes for your cluster.

2. Create a service that can communicate on the service network:

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

where

- **<name>** is the name of the service you want to create
- **<port>** is the port the service uses

For the example deployment in step 1, you create a service using the following command:

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

3. Bind the service to a cluster service:

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

where

- **<service-name>** is the name of the service on the service network
- **<target-type>** is the object you want to expose, **deployment**, **statefulset**, **pods**, or **service**.
- **<target-name>** is the name of the cluster service

For the example deployment in step 1, you bind the service using the following command:

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

1.6.3. Exposing services from a different namespace to the service network

This section shows how to expose a service from a namespace where Skupper is not deployed.

Skupper allows you expose Kubernetes services from other namespaces for any site. However, if you want to expose workloads, for example deployments, you must create a site as described in this section.

Prerequisites

- A namespace where Skupper is deployed.
- A network policy that allows communication between the namespaces
- cluster-admin permissions if you want to expose resources other than services

Procedure

1. Create a site with cluster permissions if you want to expose a workload from a namespace other than the site namespace:

NOTE

The site does not require the extra permissions granted with the **--enable-cluster-permissions** to expose a Kubernetes service resource.

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

2. To expose a Kubernetes service from a namespace other than the site namespace:

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

- <service> - the name of the service on the service network.
- <namespace> - the name of the namespace where the service you want to expose runs.

For example, if you deployed Skupper in the **east** namespace and you created a **backend** Kubernetes service in the **east-backend** namespace, you set the context to the **east** namespace and expose the service as **backend** on the service network using:

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

3. To expose a workload from a site created with **--enable-cluster-permissions**:

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

- <resource> - the name of the resource.
- <namespace> - the name of the namespace where the resource you want to expose runs.

For example, if you deployed Skupper in the **east** namespace and you created a **backend** deployment in the **east-backend** namespace, you set the context to the **east** namespace and expose the service as **backend** on the service network using:

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

1.7. EXPOSING SERVICES ON THE SERVICE NETWORK FROM A LOCAL MACHINE

After creating a service network, you can expose services from a local machine on the service network.

For example, if you run a database on a server in your data center, you can deploy a front end in a cluster that can access the data as if the database was running in the cluster.

NOTE

This documentation describes creating a gateway from a local host to a cluster site. An alternative approach is to create a site on the local host and link to the cluster site. See link:{podman-link} for information about using the Skupper CLI to create Podman sites.

1.7.1. Exposing simple local services to the service network

This section shows how to expose a single service running locally on a service network.

Prerequisites

- A service network. Only one site is required.
- Access to the service network.

Procedure

1. Run your service locally.
2. Log into your cluster and change to the namespace for your site.
3. Expose the service on the service network:

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

   - <service> - the name of the service on the service network.
   - <port> - the port that runs the service locally.

   > NOTE
   >
   > You can also expose services from other machines on your local network, for example if MySQL is running on a dedicated server (with an IP address of **192.168.1.200**), but you are accessing the cluster from a machine in the same network:
   >
   > ```
   > $ skupper gateway expose mysql 192.168.1.200 3306
   > ```

4. Check the status of Skupper gateways:

   ```
   $ skupper gateway status

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

   This shows that there is only one exposed service and that service is only exposing a single port (BIND). There are no ports forwarded to the local host.

   The URL field shows the underlying communication and can be ignored.

1.7.2. Working with complex local services on the service network

This section shows more advanced usage of skupper gateway.

1. Create a Skupper gateway:

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

   By default a *service* type gateway is created, however you can also specify:

   - **podman**

- **docker**

2. Create a service that can communicate on the service network:

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

   where

   - **<name>** is the name of the service you want to create
   - **<port>** is the port the service uses

   For example:

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

3. Bind the service on the service network:

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

   - <service> - the name of the service on the service network, **mydb** in the example above.
   - <host> - the host that runs the service.
   - <port> - the port the service is running on, **3306** from the example above.

4. Check the status of Skupper gateways:

   ```
   $ skupper gateway status
   ```

   The output looks similar to the following:

   ```
   Gateway Definitions Summary

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

   This shows that there is only one exposed service and that service is only exposing a single port (BIND). There are no ports forwarded to the local host.

   The URL field shows the underlying communication and can be ignored.

   You can create more services in the service network and bind more local services to expose those services on the service network.

5. Forward a service from the service network to the local machine.

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

   where

   - **<service>** is the name of an existing service on the service network.

- **<port>** is the port on the local machine that you want to use.

1.7.3. Creating a gateway and applying it on a different machine

If you have access to a cluster from one machine but want to create a gateway to the service network from a different machine, you can create the gateway definition bundle on the first machine and later apply that definition bundle on a second machine as described in this procedure. For example, if you want to expose a local database service to the service network, but you never want to access the cluster from the database server, you can use this procedure to create the definition bundle and apply it on the database server.

Procedure

1. Log into your cluster from the first machine and change to the namespace for your site.

2. Create a service that can communicate on the service network:

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

where

- **<name>** is the name of the service you want to create
- **<port>** is the port the service uses

For example:

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

3. Create a YAML file to represent the service you want to expose, for example:

```
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 Gateway name, useful for reference only.

② Binding name, useful to track multiple bindings.

③ Name of host providing the service you want to expose.

④ Service name and port on service network. You created the service in a previous step.

5 The protocol you want to use to expose the service, **tcp**, **http** or **http2**.

6 The port on the service network that you want this service to be available on.

7 The port of the service running on the host specified in point 3.

4. Save the YAML file using the name of the gateway, for example, **gateway.yaml**.

5. Generate a bundle that can be applied to the machine that hosts the service you want to expose on the service network:

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

   where:

   - <config-filename> - the name of the YAML file, including suffix, that you generated in the previous step.

   - <destination-directory> - the location where you want to save the resulting gateway bundle, for example ~/**gateways**.

   For example:

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

   This bundle contains the gateway definition YAML and a certificate that allow access to the service network.

6. Copy the gateway definition file, for example, **mylaptop-jdoe.tar.gz** to the machine that hosts the service you want to expose on the service network.

7. From the machine that hosts the service you want to expose:

   ```
   $ mkdir gateway

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

   NOTE

   Use **./launch.py -t podman** or **./launch.py -t docker** to run the Skupper router in a container.

   Running the gateway bundle uses the gateway definition YAML and a certificate to access and expose the service on the service network.

8. Check the status of the gateway service:
   To check a *service* type gateway:

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

   To check a *podman* type gateway:

```
$ podman inspect
```

To check a *docker* type gateway:

```
$ docker inspect
```

NOTE

You can later remove the gateway using **./remove.py**.

9. From the machine with cluster access, check the status of Skupper gateways:

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

This shows that there is only one exposed service and that service is only exposing a single port (BIND). There are no ports forwarded to the local host.

NOTE

If you need to change the gateway definition, for example to change port, you need to remove the existing gateway and repeat this procedure from the start to redefine the gateway.

1.7.4. Gateway YAML reference

The Section 1.7.3, "Creating a gateway and applying it on a different machine" describes how to create a gateway to apply on a separate machine using a gateway definition YAML file.

The following are valid entries in a gateway definition YAML file.

name

Name of gateway

bindings.name

Name of binding for a single host.

bindings.host

Hostname of local service.

bindings.service

Definition of service you want to be available on service network.

bindings.service.address

Address on the service network, name and port.

bindings.service.protocol

Skupper protocol, **tcp**, **http** or **http2**.

bindings.service.ports

A single port that becomes available on the service network.

bindings.service.exposeIngress

(optional) The traffic direction, **ingress** or **egress**.

bindings.service.tlscredentials

(optional) The TLS certificate and key for the service.

bindings.service.tlscertauthority

(optional) The TLS public certificate.

bindings.target_ports

A single port that you want to expose on the service network.

> **NOTE**
>
> If the local service requires more than one port, create separate bindings for each port.

forwards.name

Name of forward for a single host.

forwards.host

Hostname of local service.

forwards.service

Definition of service you want to be available locally.

forwards.service.address

Address on the service network that you want to use locally, name and port.

forwards.service.protocol

Skupper protocol, **tcp**, **http** or **http2**.

forwards.service.ports

A single port that is available on the service network.

forwards.target_ports

A single port that you want to use locally.

> **NOTE**
>
> If the network service requires more than one port, create separate forwards for each port.

qdr-listeners

Definition of skupper router listeners

qdr-listeners.name

Name of skupper router, typically **amqp**.

qdr-listeners.host

Hostname for skupper router, typically **localhost**.

qdr-listeners.port

Port for skupper router, typically **5672**.

1.8. EXPLORING A SERVICE NETWORK

Skupper includes a command to allow you report all the sites and the services available on a service network.

Prerequisites

- A service network with more than one site

Procedure

1. Set your Kubernetes context to a namespace on the service network.
2. Use the following command to report the status of the service network:

```
$ skupper network status
```

For example:

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

❶ The unique identifier of the site associated with the current context, that is, the **west** namespace

❷ The site name. By default, skupper uses the name of the current namespace. If you want to specify a site name, use **skupper init --site-name <site-name>**.

❸ The version of Skupper running the site. The site version can be different from the current **skupper** CLI version. To update a site to the version of the CLI, use **skupper update**.

❹ The unique identifier of a remote site on the service network.

❺ The sites that the remote site is linked to.

6 The unique identifier of a remote podman site. Podman sites do not have an associated context.

1.9. SECURING A SERVICE NETWORK

Skupper provides default, built-in security that scales across clusters and clouds. This section describes additional security you can configure.

See Securing a service network using policies for information about creating granular policies for each cluster.

1.9.1. Restricting access to services using a Kubernetes network policy

By default, if you expose a service on the service network, that service is also accessible from other namespaces in the cluster. You can avoid this situation when creating a site using the **--create-network-policy** option.

Procedure

1. Create the service network router with a Kubernetes network policy:

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

2. Check the site status:

   ```
   $ skupper status
   ```

   The output should be similar to the following:

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

You can now expose services on the service network and those services are not accessible from other namespaces in the cluster.

1.9.2. Applying TLS to TCP or HTTP2 traffic on the service network

By default, the traffic between sites is encrypted, however the traffic between the service pod and the router pod is not encrypted. For services exposed as TCP or HTTP2, the traffic between the pod and the router pod can be encrypted using TLS.

Prerequisites

- Two or more linked sites
- A TCP or HTTP2 frontend and backend service

Procedure

1. Deploy your backend service.
2. Expose your backend deployment on the service network, enabling TLS.
   For example, if you want to expose a TCP service:

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

Enabling TLS creates the necessary certificates required for TLS backends and stores them in a secret named **skupper-tls-<deployment-name>**.

3. Modify the backend deployment to include the generated certificates, for example:

```
...
  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>
```

Each site creates the necessary certificates required for TLS clients and stores them in a secret named **skupper-service-client**.

4. Modify the frontend deployment to include the generated certificates, for example:

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

5. Test calling the service from a TLS enabled frontend.

1.10. SUPPORTED STANDARDS AND PROTOCOLS

Skupper supports the following protocols for your service network:

- TCP - default

- HTTP1
- HTTP2

When exposing or creating a service, you can specify the protocol, for example:

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

where **<protocol>** can be:

- tcp
- http
- http2

When choosing which protocol to specify, note the following:

- **tcp** supports any protocol overlayed on TCP, for example, HTTP1 and HTTP2 work when you specify **tcp**.
- If you specify **http** or **http2**, the IP address reported by a client may not be accessible.
- All service network traffic is converted to AMQP messages in order to traverse the service network.
  TCP is implemented as a single streamed message, whereas HTTP1 and HTTP2 are implemented as request/response message routing.

1.11. CLI OPTIONS

For a full list of options, see the Skupper Kubernetes CLI reference and Skupper Podman CLI reference documentation.

WARNING

When you create a site and set logging level to **trace**, you can inadvertently log sensitive information from HTTP headers.

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

By default, all **skupper** commands apply to the cluster you are logged into and the current namespace. The following **skupper** options allow you to override that behavior and apply to all commands:

--namespace <namespace-name>

Apply command to **<namespace-name>**. For example, if you are currently working on **frontend** namespace and want to initialize a site in the **backend** namespace:

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

--kubeconfig <kubeconfig-path>

Path to the kubeconfig file - This allows you run multiple sessions to a cluster from the same client. An alternative is to set the **KUBECONFIG** environment variable.

--context <context-name>

The kubeconfig file can contain defined contexts, and this option allows you to use those contexts.

CHAPTER 2. USING SKUPPER PODMAN

Using the **skupper** command-line interface (CLI) allows you to create and manage Skupper sites from the context of the current Linux user. Skupper Podman allows you to create a site using containers, without requiring Kubernetes.

A typical workflow is to create a site, link sites together, and expose services to the service network.

2.1. ABOUT SKUPPER PODMAN

Skupper Podman is available with the following precedence:

skupper --platform podman <command>

Use this option to avoid changing mode, for example, if you are working on Kubernetes and Podman simultaneously.

export SKUPPER_PLATFORM=podman

Use this command to use Skupper Podman for the current session, for example, if you have two terminals set to different contexts. To set the environment to target Kubernetes sites:

```
$ export SKUPPER_PLATFORM=kubernetes
```

skupper switch podman

If you enter this command, all subsequent command target Podman rather than Kubernetes for all terminal sessions.

To determine which mode is currently active:

```
$ skupper switch

podman
```

To switch back to target Kubernetes sites: **skupper switch kubernetes**

NOTE

Services exposed on remote sites are not automatically available to Podman sites. This is the equivalent to Kubernetes sites created using **skupper init --enable-service-sync false**.

To consume an exposed service on a Podman site, check that it exists using **skupper service status** on the original site and use that information to create the service on the Podman site:

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

2.2. CREATING A SITE USING SKUPPER PODMAN

A service network consists of Skupper sites. This section describes how to create a site in on a Linux host using the default settings. See Using the Skupper CLI for information about using the Skupper CLI to create Podman sites.

Prerequisites

- The latest **skupper** CLI is installed.
- Podman is installed, see https://podman.io/
  By default, Podman v4 uses Netavark which works with Skupper. If you are using CNI, for example, if you upgrade from Podman v3, you must also install the **podman-plugins** package. For example, **dnf install podman-plugins** for RPM based distributions.

  NOTE

  CNI will be deprecated in the future in preference of Netavark.

- Podman service endpoint.
  Use **systemctl status podman.socket** to make sure the Podman API Socket is running.

  Use **systemctl --user enable --now podman.socket** to start the Podman API Socket.

  See Podman socket activation for information about enabling this endpoint.

1. Set your session to use Skupper Podman:

   ```
   $ export SKUPPER_PLATFORM=podman
   ```

   To verify the **skupper** mode:

   ```
   $ skupper switch

   podman
   ```

2. Create a Skupper site:
   The simplest Skupper site allows you to link to other sites, but does not support linking *to* the current site.

   ```
   $ skupper init

   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.
   ```

   If you do not require that other sites can link to the site you are creating:

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

   In this guide we assume you have enabled ingress using the first command. This allows you create tokens that allow links from every network interface on the host.

   NOTE

   When creating a token you can specify the ingress host.

You can also restrict ingress to an IP address or hostname when initializing as described in the Skupper Podman CLI reference documentation.

3. Check the status of your site:

```
$ 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.
```

NOTE

You can only create one site per user. If you require a host to support many sites, create a user for each site.

2.3. LINKING SITES USING SKUPPER PODMAN

A service network consists of Skupper sites. This section describes how to link sites to form a service network.

Linking two sites requires a single initial directional connection. However:

- Communication between the two sites is bidirectional, only the initial linking is directional.
- The choice of direction for linking is typically determined by accessibility. For example, if you are linking a virtual machine running in the cloud with a Linux host running behind a firewall, you must link from the Linux host to the cloud virtual machine because that route is accessible.

Procedure

1. Generate a token on one site:

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

   The **metadata** section of the resulting YAML file shows which interface is available for linking.

   To specify the interface, use **ifconfig** to determine the IP address of the interface you want to use. You can then specify that IP address when creating the token:

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

2. Create a link from the other site:

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

After you have linked to a network, you can check the link status:

```
$ skupper link status
```

2.4. SPECIFYING LINK COST

When linking sites, you can assign a cost to each link to influence the traffic flow. By default, link cost is set to **1** for a new link. In a service network, the routing algorithm attempts to use the path with the lowest total cost from client to target server.

- If you have services distributed across different sites, you might want a client to favor a particular target or link. In this case, you can specify a cost of greater than **1** on the alternative links to reduce the usage of those links.

  NOTE

  The distribution of open connections is statistical, that is, not a round robin system.

- If a connection only traverses one link, then the path cost is equal to the link cost. If the connection traverses more than one link, the path cost is the sum of all the links involved in the path.
- Cost acts as a threshold for using a path from client to server in the network. When there is only one path, traffic flows on that path regardless of cost.

  NOTE

  If you start with two targets for a service, and one of the targets is no longer available, traffic flows on the remaining path regardless of cost.

- When there are a number of paths from a client to server instances or a service, traffic flows on the lowest cost path until the number of connections exceeds the cost of an alternative path. After this threshold of open connections is reached, new connections are spread across the alternative path and the lowest cost path.

Prerequisite

- You have set your Kubernetes context to a site that you want to link *from*.
- A token for the site that you want to link *to*.

Procedure

1. Create a link to the service network:

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

   where **<integer-cost>** is an integer greater than 1 and traffic favors lower cost links.

   NOTE

   If a service can be called without traversing a link, that service is considered local, with an implicit cost of **0**.

   For example, create a link with cost set to **2** using a token file named **token.yaml**:

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

2. Check the link cost:

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

The output is similar to the following:

```
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. Observe traffic using the console.
If you have a console on a site, log in and navigate to the processes for each server. You can view the traffic levels corresponding to each client.

> **NOTE**
>
> If there are multiple clients on different sites, filter the view to each client to determine the effect of cost on traffic. For example, in a two site network linked with a high cost with servers and clients on both sites, you can see that a client is served by the local servers while a local server is available.

2.5. EXPOSING SERVICES ON THE SERVICE NETWORK FROM A LINUX HOST

After creating a service network, exposed services can communicate across that network.

The general flow for working with services is the same for Kubernetes and Podman sites.

The **skupper** CLI has two options for exposing services that already exist on a host:

- **expose** supports simple use cases, for example, a host with a single service. See Section 2.5.1, "Exposing simple services on the service network" for instructions.
- **service create** and **service bind** is a more flexible method of exposing services, for example, if you have multiple services for a host. See Section 1.6.2, "Exposing complex services on the service network" for instructions.

2.5.1. Exposing simple services on the service network

This section describes how services can be enabled for a service network for simple use cases.

Prerequisites

- A Skupper Podman site

Procedure

1. Run a server, for example:

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

This step is not Skupper-specific, that is, this process is unchanged from standard processes for your host.

2. Create a service that can communicate on the service network:

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

where

- **<host>** is the name of the host where the server is running. For example, the name of the container if you run the server as a container.
- **<ip>** is the IP address where the server is running

For the example deployment in step 1, you create a service using the following command:

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

Options for this command include:

- **--port <port-number>**:: Specify the port number that this service is available on the service network. NOTE: You can specify more than one port by repeating this option.
- **--target-port <port-number>**:: Specify the port number of pods that you want to expose.
- **--protocol <protocol>** allows you specify the protocol you want to use, **tcp**, **http** or **http2**

3. Create the service on another site in the service network:

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

2.5.2. Consuming simple services from the service network

Services exposed on Podman sites are not automatically available to other sites. This is the equivalent to Kubernetes sites created using **skupper init --enable-service-sync false**.

Prerequisites

- A remote site where a service is exposed on the service network
- A Podman site

Procedure

1. Log into the host as the user associated with the Skupper site.
2. Create the local service:

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

2.6. DELETING A PODMAN SITE

When you no longer want the Linux host to be part of the service network, you can delete the site.

NOTE

This procedure removes all containers, volumes and networks labeled **application=skupper**.

To check the labels associated with running containers:

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

Procedure

1. Make sure you are logged in as the user that created the site:

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

2. Delete the site and all podman resources (containers, volumes and networks) labeled with "application=skupper":

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

CHAPTER 3. USING THE SKUPPER CONSOLE

The Skupper console provides data and visualizations of the traffic flow between Skupper sites.

3.1. ENABLING THE SKUPPER CONSOLE

By default, when you create a Skupper site, a Skupper console is not available.

When enabled, the Skupper console URL is displayed whenever you check site status using **skupper status**.

Prerequisites

- A Kubernetes namespace where you plan to create a site

Procedure

1. Determine which site in your service network is best to enable the console.
   Enabling the console also requires that you enable the flow-collector component, which requires resources to process traffic data from all sites. You might locate the console using the following criteria:

   - Does the service network cross a firewall? For example, if you want the console to be available only inside the firewall, you need to locate the flow-collector and console on a site inside the firewall.

   - Is there a site that processes more traffic than other sites? For example, if you have a *frontend* component that calls a set of services from other sites, it might make sense to locate the flow collector and console on that site to minimize data traffic.

   - Is there a site with more or cheaper resources that you want to use? For example, if you have two sites, A and B, and resources are more expensive on site A, you might want to locate the flow collector and console on site B.

2. Create a site with the flow collector and console enabled:

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

3.2. ACCESSING THE SKUPPER CONSOLE

By default, the Skupper console is protected by credentials available in the **skupper-console-users** secret.

Procedure

1. Determine the Skupper console URL using the **skupper** CLI, for example:

   ```
   $ 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. Browse to the Skupper console URL. The credential prompt depends on how the site was created using **skupper init**:

    - Using the **--console-auth unsecured** option, you are not prompted for credentials.
    - Using the **--console-auth openshift** option, you are prompted to enter OpenShift cluster credentials.
    - Using the default or **--console-user <user> --console-password <password>** options, you are prompted to enter those credentials.

3. If you created the site using default settings, that is **skupper init**, a random password is generated for the **admin** user. To retrieve the password the **admin** user:

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

    JNZWzMHtyg
    ```

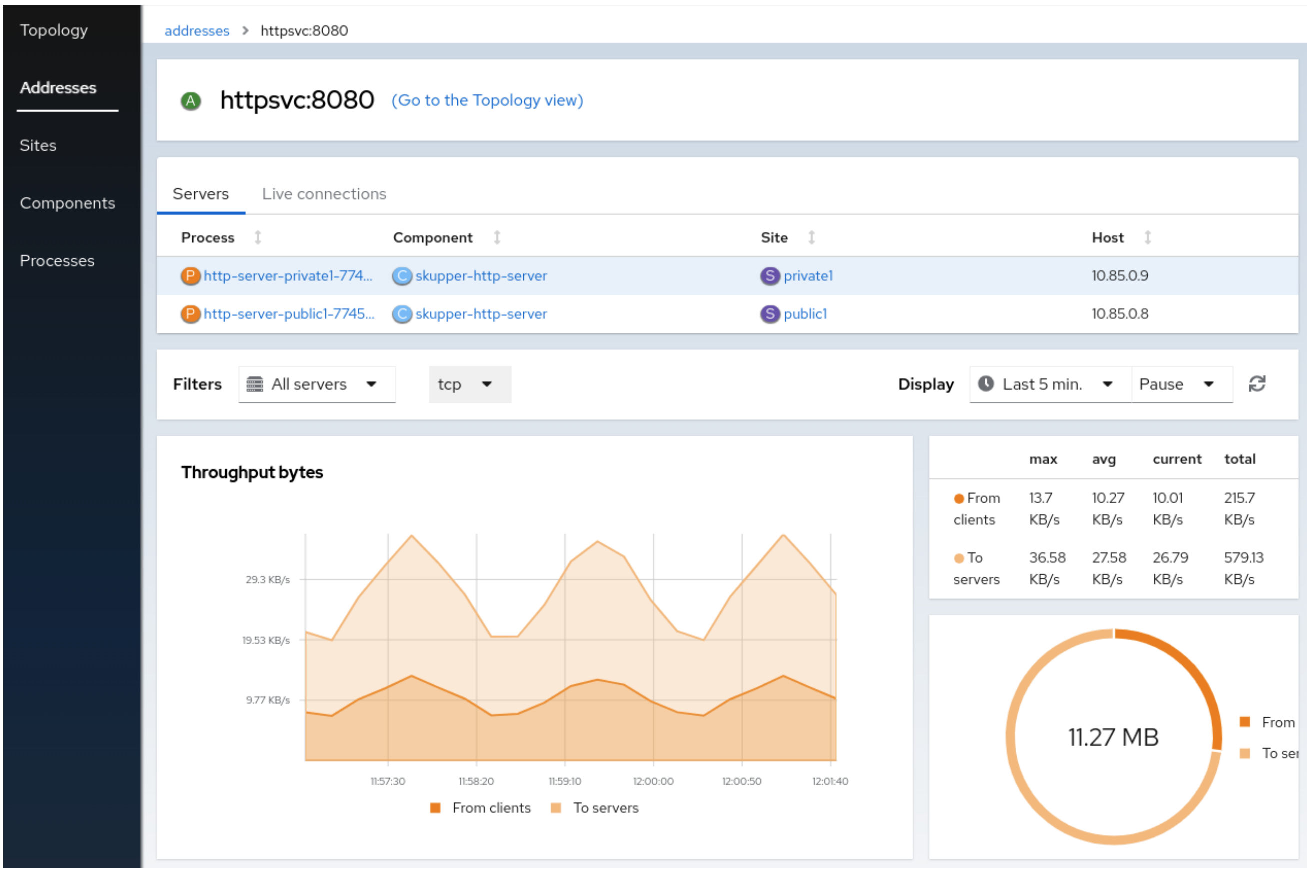
3.3. EXPLORING THE SKUPPER CONSOLE

After exposing a service on the service network, you create an *address*, that is, a service name and port number associated with a site. There might be many replicas associated with an address. These replicas are shown in the Skupper console as *processes*. Not all participants on a service network are services. For example, a *frontend* deployment might call an exposed service named *backend*, but that frontend is not part of the service network. In the console, both are shown so that you can view the traffic and these are called *components*.

The Skupper console provides an overview of the following:

- Topology
- Addresses
- Sites
- Components
- Processes

The Skupper console also provides useful networking information about the service network, for example, traffic levels.



Process	Component	Site	Host
P http-server-private1-774...	C skupper-http-server	S private1	10.85.0.9
P http-server-public1-7745...	C skupper-http-server	S public1	10.85.0.8

	max	avg	current	total
● From clients	13.7 KB/s	10.27 KB/s	10.01 KB/s	215.7 KB/s
● To servers	36.58 KB/s	27.58 KB/s	26.79 KB/s	579.13 KB/s

1. Check the **Sites** tab. All your sites should be listed. See the **Topology** tab to view how the sites are linked.

2. Check that all the services you exposed are visible in the **Components** tab.

3. Click a component to show the component details and associated processes.

4. Click on a process to display the process traffic.

NOTE

The process detail displays the associated image, host, and addresses. You can also view the clients that are calling the process.

5. Click **Addresses** and choose an address to show the details for that address. This shows the set of servers that are exposed across the service network.

TIP

To view information about each window, click the **?** icon.

CHAPTER 4. CONFIGURING SKUPPER SITES USING YAML

Using YAML files to configure Skupper allows you to use source control to track and manage Skupper network changes.

4.1. CREATING A SKUPPER SITE USING YAML

Using YAML files to create Skupper sites allows you to use source control to track and manage Skupper network changes.

Prerequisites

- Skupper is installed in the cluster or namespace you want to target.
- You are logged into the cluster.

Procedure

1. Create a YAML file to define the site, for example, **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"
   ```

   The YAML creates a site with a console and you can create tokens from this site.

   To create a site that has no ingress:

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

2. Apply the YAML file to your cluster:

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

Additional resources

See the Appendix A, *Site ConfigMap YAML reference* section for more reference.

APPENDIX A. SITE CONFIGMAP YAML REFERENCE

Using YAML files to configure Skupper requires that you understand all the fields so that you provision the site you require.

The following YAML defines a Skupper site:

```
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
```

<.> Specifies the site name.

<.> Enables the skupper console, defaults to **false**. NOTE: You must enable **console** and **flow-collector** for the console to function.

<.> Enables the flow collector, defaults to **false**.

<.> Specifies the skupper console authentication method. The options are **openshift**, **internal**, **unsecured**.

<.> Username for the **internal** authentication option.

<.> Password for the **internal** authentication option.

<.> Only accept connections from within the local cluster, defaults to **false**.

<.> Specifies whether an edge site is created, defaults to **false**.

<.> Specifies whether the services are synchronized across the service network, defaults to **true**.

<.> Specifies whether the site supports ingress. If you do not specify a value, the default ingress ('loadbalancer' on Kubernetes, 'route' on OpenShift) is enabled. This allows you to create tokens usable from remote sites.

NOTE

All ingress types are supported using the same parameters as the **skupper** CLI.

CHAPTER 5. USING THE SKUPPER OPERATOR ON KUBERNETES

The Red Hat Service Interconnect Operator creates and manages Skupper sites in Kubernetes.

5.1. CREATING A SITE USING THE SKUPPER OPERATOR

1. Create a YAML file defining the ConfigMap of the site you want to create.
   For example, create **skupper-site.yaml** that provisions a site with a console:

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

   NOTE

   The console is a preview feature and may change before becoming fully supported by skupper.io. Currently, you must enable the console on the same site as you enable the flow collector. This requirement may change before the console is fully supported by skupper.io.

   You can also create a site without a console:

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

2. Apply the YAML to create a ConfigMap named **skupper-site** in the namespace you want to use:

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

3. Verify that the site is created by checking that the Skupper router and service controller pods are running:

   ```
   $ 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
   ```

NOTE

If you deployed the Operator to a single namespace, an additional site controller pod is also running.

CHAPTER 6. SECURING A SERVICE NETWORK USING SKUPPER POLICIES

By default, Skupper includes many security features, including using mutual TLS for all service network communication between sites. You can add extra security features by installing the Skupper policy CRD. By default, applying a Skupper policy CRD to a cluster prevents all service network communication to and from that cluster. You specify granular Skupper policies CRs to permit only the service network communication you require.

NOTE

A Skupper policy is distinct from the Kubernetes network policy, that is the **network-policy** option, which restricts access to Skupper services to the current namespace as described in Using the Skupper CLI.

Each site in a service network runs a Skupper router and has a private, dedicated certificate authority (CA). Communication between sites is secured with mutual TLS, so the service network is isolated from external access, preventing security risks such as lateral attacks, malware infestations, and data exfiltration. A set of Skupper policies adds another layer at a cluster level to help a cluster administrator control access to a service network.

This guide assumes that you understand the following Skupper concepts:

site

A namespace in which Skupper is installed.

token

A token is required to establish a link between two sites.

service network

After exposing services using Skupper, you have created a service network.

6.1. ABOUT SKUPPER POLICIES

After a cluster administrator installs a Skupper policy Custom Resource Definition (CRD), the cluster administrator needs to configure one or more policies to allow *developers* create and use services on the service network.

NOTE

In this guide, *developers* refers to users of a cluster who have access to a namespace, but do not have administrator privileges.

A cluster administrator configures one or more of following items using custom resources (CRs) to enable communication:

Allow incoming links

Use **allowIncomingLinks** to enable developers create tokens and configure incoming links.

Allow outgoing links to specific hosts

Use **allowedOutgoingLinksHostnames** to specify hosts that developers can create links to.

Allow services

Use **allowedServices** to specify which services developers can create or use on the service network.

Allow resources to be exposed

Use **allowedExposedResources** to specify which resources a developer can expose on the service network.

NOTE

A cluster administrator can apply each policy CR setting to one or more namespaces.

For example, the following policy CR fully allows all Skupper capabilities on all namespaces, except for:

- only allows outgoing links to any domain ending in **.example.com**.
- only allows 'deployment/nginx' resources to be exposed on the service network.

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

NOTE

You can apply many policy CRs, and if there are conflicts in the items allowed, the most permissive policy is applied. For example, if you apply an additional policy CR with the line **allowedOutgoingLinksHostnames: []**, which does not list any hostnames, outgoing links to ***.example.com** are still permitted because that is permitted in the original CR.

namespaces

One or more patterns to specify the namespaces that this policy applies to. Note that you can use Label selectors to match the namespaces.

allowIncomingLinks

Specify **true** to allow other sites create links to the specified namespaces.

allowedOutgoingLinksHostnames

Specify one or more patterns to determine which hosts you can create links to from the specified namespaces.

allowedServices

Specify one or more patterns to determine the permitted names of services allowed on the service network from the specified namespaces.

allowedExposedResources

Specify one or more permitted names of resources allowed on the service network from the specified namespaces. Note that patterns are not supported.

TIP

Use regular expressions to create pattern matches, for example:

- **.*\.com$** matches any string ending in **.com**. A double backslash is required to avoid issues in YAML.
- **^abc$** matches the string **abc**.

If you create another Skupper policy CR that allows outgoing links for a specific namespace, a user can create a link from that namespace to join a service network. That is, the logic for multiple policy CRs is **OR**. An operation is permitted if any single policy CR permits the operation.

6.2. INSTALLING THE SKUPPER POLICY CRD

Installing the Skupper policy CRD enables a cluster administrator to enforce policies for service networks.

NOTE

If there are existing sites on the cluster, see Section 6.3, "Installing a Skupper policy CRD on a cluster with existing sites" to avoid service network disruption.

Prerequisites

- Access to a cluster using a **cluster-admin** account
- The Skupper operator is installed

Procedure

1. Log in to the cluster using a **cluster-admin** account.
2. Download the CRD:

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

3. Apply the 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. To verify that a Skupper policy is active, use the **skupper status** command and check that the output includes the following line:

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

6.3. INSTALLING A SKUPPER POLICY CRD ON A CLUSTER WITH EXISTING SITES

If the cluster already hosts Skupper sites, note the following before installing the CRD:

- All existing connections are closed. You must apply a policy CR to reopen connections.
- All existing service network services and exposed resources are removed. You must create those resources again.

Procedure

To avoid disruption:

1. Plan the CRD deployment for an appropriate time.
2. Search your cluster for sites:

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

3. Document each service and resource exposed on the service network.
4. Install the CRD as described in Section 6.2, “Installing the Skupper policy CRD” . This step closes connections and removes all service network services and exposed resources.
5. If Skupper sites that were not created by **cluster-admin** exist in the cluster, you must grant permissions to read Skupper policies to avoid that site being blocked from the service network. For each site namespace:

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

   where **<namespace>** is the site namespace.

6. Create Skupper policy CRs as described in Section 6.4, “Creating Skupper policy CRs”
7. Recreate any services and exposed resources as required.

6.4. CREATING SKUPPER POLICY CRS

Skupper Policy CRs allow a cluster administrator to control communication across the service network from a cluster.

Prerequisites

- Access to a cluster using a **cluster-admin** account.
- The Skupper policy CRD is installed on the cluster.

PROCEDURE

Typically, you create a Skupper policy CR that combines many elements from the steps below. See Section 6.1, “About Skupper policies” for an example CR.

1. Section 6.4.1, “Implement a policy to allow incoming links”
2. Section 6.4.2, “Implement a policy to allow outgoing links to specific hosts”
3. Section 6.4.3, “Implement a policy to allow specific services”
4. Section 6.4.4, “Implement a policy to allow specific resources”

6.4.1. Implement a policy to allow incoming links

Use **allowIncomingLinks** to enable developers create tokens and configure incoming links.

Procedure

1. Determine which namespaces you want to apply this policy to.
2. Create a CR with **allowIncomingLinks** set to **true** or **false**.
3. Create and apply the CR.

For example, the following CR allows incoming links for all namespaces:

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

6.4.2. Implement a policy to allow outgoing links to specific hosts

Use **allowedOutgoingLinksHostnames** to specify hosts that developers can create links to. You cannot create a **allowedOutgoingLinksHostnames** policy to disallow a specific host that was previously allowed.

1. Determine which namespaces you want to apply this policy to.
2. Create a CR with **allowedOutgoingLinksHostnames** set to a pattern of allowed hosts.
3. Create and apply the CR.

For example, the following CR allows links to all subdomains of **example.com** for all namespaces:

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

6.4.3. Implement a policy to allow specific services

Use **allowedServices** to specify which services a developer can create or use on the service network. You cannot create a **allowedServices** policy to disallow a specific service that was previously allowed.

Procedure

1. Determine which namespaces you want to apply this policy to.
2. Create a CR with **allowedServices** set to specify the services allowed on the service network.
3. Create and apply the CR.

For example, the following CR allows users to expose and consume services with the prefix **backend-** for all namespaces:

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

NOTE

When exposing services, you can use the **--address <name>** parameter of the **skupper** CLI to name services to match your policy.

6.4.4. Implement a policy to allow specific resources

Use **allowedExposedResources** to specify which resources a developer can expose on the service network. You cannot create a **allowedExposedResources** policy to disallow a specific resource that was previously allowed.

Procedure

1. Determine which namespaces you want to apply this policy to.
2. Create a CR with **allowedExposedResources** set to specify resources that a developer can expose on the service network.
3. Create and apply the CR.

For example, the following CR allows you to expose an **nginx** deployment for all namespaces:

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

-

NOTE

For **allowedExposedResources**, each entry must conform to the **type/name** syntax.

CHAPTER 7. TROUBLESHOOTING A SERVICE NETWORK

Typically, you can create a service network without referencing this troubleshooting guide. However, this guide provides some tips for situations when the service network does not perform as expected.

See Section 7.7, “Resolving common problems” if you have encountered a specific issue using the **skupper** CLI.

A typical troubleshooting workflow is to check all the sites and create debug tar files.

7.1. CHECKING SITES

Using the **skupper** command-line interface (CLI) provides a simple method to get started with troubleshooting Skupper.

Procedure

1. Check the site status:

```
$ 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.
```

   The output shows:

   - A site exists in the specified namespace.
   - A link exists to two other sites.
   - A service is exposed on the service network and is accessible from this namespace.

2. Check the service network:

```
$ 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
```

NOTE

If the output is not what you expected, you might want to check links before proceeding.

The output shows:

- There are 3 sites on the service network, **vm-user-c3d98**, **east** and **west**.
- Details for each site, for example the namespace names.

3. Check the status of services exposed on the service network:

```
$ skupper service status
Services exposed through Skupper:
└─ backend (tcp port 8080)
  └─ Targets:
    └─ app=backend name=backend
```

The output shows the **backend** service and the related target of that service.

NOTE

The related targets for services are only displayed when the target is available on the current cluster.

4. List the Skupper events for a site:

```
$ 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
```

The output shows sites being linked and a service being exposed on a service network. However, this output is most useful when reporting an issue and is included in the Skupper debug tar file.

5. List the Kubernetes events for a site:

```
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
```

The output shows events relating to Kubernetes resources.

Additional information

- Section 7.2, “Checking links”

7.2. CHECKING LINKS

You must link sites before you can expose services on the service network.

NOTE

By default, tokens expire after 5 minutes and you can only use a token once. Generate a new token if the link is not connected. You can also generate tokens using the **-token-type cert** option for permanent reusable tokens.

This section outlines some advanced options for checking links.

1. Check the link status:

–

```
$ skupper link status --namespace east

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

A link exists from the specified site to another site, meaning a token from another site was applied to the specified site.

NOTE

Running **skupper link status** on a connected site produces output only if a token was used to create a link.

If you use this command on a site where you did not create the link, but there is an incoming link to the site:

```
$ 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. Check the verbose link status:

```
$ 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
```

The output shows detail about the link, including a timestamp of when the link was created and the associated relative cost of using the link.

The status of the link must be **Connected** to allow service traffic.

Additional information

- Section 7.1, “Checking sites”

7.3. CHECKING GATEWAYS

By default, **skupper gateway** creates a service type gateway and these gateways run properly after a machine restart.

However, if you create a docker or podman type gateway, check that the container is running after a machine restart. For example:

1. Check the status of Skupper gateways:

   ```
   $ skupper gateway status

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

   This shows a podman type gateway.

2. Check that the container is running:

   ```
   $ 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
   ```

   This shows the container running.

   NOTE

   To view stopped containers, use **podman ps -a** or **docker ps -a**.

3. Start the container if necessary:

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

7.4. CHECKING POLICIES

As a developer you might not be aware of the Skupper policy applied to your site. Follow this procedure to explore the policies applied to the site.

Procedure

1. Log into a namespace where a Skupper site has been initialized.

2. Check whether incoming links are permitted:

   ```
   $ 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
   ```

   In this example incoming links are not allowed by policy.

3. Check other policies:

   ```
   $ 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
```

As shown, there are commands to check each policy type by specifying what you want to do, for example, to check if you can expose an nginx deployment:

```
$ 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
```

If you allowed an nginx deployment, the same command shows that the resource is allowed and displays the name of the policy CR that enabled it:

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

7.5. CREATING A SKUPPER DEBUG TAR FILE

The debug tar file contains all the logs from the Skupper components for a site and provides detailed information to help debug issues.

1. Create the debug tar file:

```
$  skupper debug dump my-site

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

2. You can expand the file using the following command:

```
$ 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
```

These files can be used to provide support for Skupper, however some items you can check:

versions

See ***versions.txt** for the versions of various components.

ingress

See **skupper-site-configmap.yaml** to determine the **ingress** type for the site.

linking and services

See the **skupper-service-controller-*-events.txt** file to view details of token usage and service exposure.

7.6. IMPROVING SKUPPER ROUTER PERFORMANCE

If you encounter Skupper router performance issues, you can scale the Skupper router to address those concerns.

NOTE

Currently, you must delete and recreate a site to reconfigure the Skupper router.

For example, use this procedure to increase throughput, and if you have many clients, latency.

1. Delete your site or create a new site in a different namespace.
   Note all configuration and delete your existing site:

   ```
   $ skupper delete
   ```

   As an alternative, you can create a new namespace and configure a new site with optimized Skupper router performance. After validating the performance improvement, you can delete and recreate your original site.

2. Create a site with optimal performance CPU settings:

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

3. Recreate your configuration from step 1, recreating links and services.

NOTE

While you can address availability concerns by scaling the number of routers, typically this is not necessary.

7.7. RESOLVING COMMON PROBLEMS

The following issues and workarounds might help you debug simple scenarios when evaluating Skupper.

Cannot initialize skupper

If the **skupper init** command fails, consider the following options:

- Check the load balancer.
  If you are evaluating Skupper on minikube, use the following command to create a load balancer:

  ```
  $ minikube tunnel
  ```

  For other Kubernetes flavors, see the documentation from your provider.

- Initialize without ingress.
  This option prevents other sites from linking to this site, but linking outwards is supported. Once a link is established, traffic can flow in either direction. Enter the following command:

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

  NOTE

  See the Skupper Podman CLI reference documentation for **skupper init**.

Cannot link sites

To link two sites, one site must be accessible from the other site. For example, if one site is behind a firewall and the other site is on an AWS cluster, you must:

1. Create a token on the AWS cluster site.
2. Create the link on the site inside the firewall.

NOTE

By default, a token is valid for only 15 minutes and can only be used once. See Using Skupper tokens for more information on creating different types of tokens.

Cannot access Skupper console

Starting with Skupper release 1.3, the console is not enabled by default. To use the new console, see Using the console.

Use **skupper status** to find the console URL.

Use the following command to display the password for the **admin** user:doctype: article

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

Cannot create a token for linking clusters

There are several reasons why you might have difficulty creating tokens:

Site not ready

After creating a site, you might see the following message when creating a token:

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

Use **skupper status** to verify the site is working and try to create the token again.

No ingress

You might see the following note after using the **skupper token create** command:

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

This output indicates that the site was deployed without an ingress option. For example **skupper init --ingress none**. You must specify an ingress to allow sites on other clusters to link to your site.

You can also use the **skupper token create** command to check if an ingress was specified when the site was created.