Skupper

Version latest, 2023-12-04

This is a preview of skupperproject/skupper-docs: Documentation for the Skupper project

This documentation is built using Antora and contains the following Antora modules:

- ROOT - Metadata like this page and images
- cli - Kubernetes and Podman CLI procedures
- cli-reference - Kubernetes CLI reference (generated)
- console - Console docs
- declarative - Using Skupper with YAML
- overview - Skupper concepts
- operator - Using Skupper with a Kubernetes operator
- policy - Control Skupper usage in a cluster using the policy CRD
- podman-reference - Podman CLI reference (generated)
- troubleshooting - Resolving problems in your service network

This ROOT module is never published but contains partials that are used in various documents.

Skupper - Getting started

Skupper - Examples

Skupper - 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.

Checking the Skupper CLI

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

Procedure

```
 // tag::skupper-io[]
. Follow the instructions for link:/install/index.html[Installing Skupper].
// end::skupper-io[]
.
. Verify the installation.
```

+

```
$ skupper version
client version 1.4
```

Creating a site using the CLI

A service network consists of Skupper sites. This section describes how to create a site using the default settings.

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, which is a preview feature and may change, see https://skupper.io/skupper/latest/console/index.html.

2. Check the site:

```
$ skupper status

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

The default message above is displayed when you initialize a site on a cluster that does not have the policy system installed. If you install the policy system as described in https://skupper.io/skupper/latest/policy/index.html, 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`.

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, which is a preview feature and may change, see https://skupper.io/skupper/latest/console/index.html.

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
```

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 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`.

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.

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

For more information about protecting access to the service network, see https://skupper.io/skupper/latest/cli/tokens.html.

3. Use a token on the cluster that you want to connect from:
   a. 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.

b. Check the link:

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

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

4. If you want to delete a link:

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

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

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.

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.

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.

   

   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

    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.

   

   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.

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 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 Exposing complex services on the service network for instructions.

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`

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

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
   ```

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

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

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

2. Expose the service on the service network:

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

a. If you want 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
```

b. If you want to expose a workload and you created a site 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
```

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.

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.

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
    └── 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.

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
Gateway Definitions Summary

Gateway Definition:
╰── machine-user type:service version:1
    ╰── 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.

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
```

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
```

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
    ╰── 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.

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.

Gateway YAML reference

The 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.

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.

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`.

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] 4dba248 - west  ①
│   URL: 10.96.146.236 ②
│   name: west ③
│   namespace: west
│   version: 0.8.6 ④
│   └── Services:
│       └── name: hello-world-backend ⑤
│           address: hello-world-backend: 8080 ⑥
│           protocol: tcp ⑦
└── [remote] bca99d1 - east ⑧
    URL:
    name: east
    namespace: east
    sites linked to: 4dba248-west ⑨
    version: 0.8.6
    └── Services:
        └── name: hello-world-backend
            address: hello-world-backend: 8080
            protocol: tcp
            └── Targets:
                └── name: hello-world-backend-7dfb45b98d-mhskw ⑩
```

① The unique identifier of the site associated with the current context, that is, the `west` namespace

② The URL of the service network router. This is required for other sites to connect to this site and is different from the console URL. If you require the URL of the console, use the `skupper status` command to display that URL.

③ 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>`.

④ 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`.

⑤ The name of a service exposed on the service network.

⑥ The address of a service exposed on the service network.

⑦ The protocol of a service exposed on the service network.

⑧ The unique identifier of a remote site on the service network.

⑨ The sites that the remote site is linked to.

⑩ The name of the local Kubernetes object that is exposed on the service network. In this example, this is the `hello-world-backend` pod.

The URL for the east site has no value because that site was initialized without ingress using the following command:

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

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 https://skupper.io/skupper/latest/policy/index.html for information about creating granular policies for each cluster.

Restricting access to services using 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 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.

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>`.

1. 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`.

2. 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
```

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

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.

CLI options

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

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.

Skupper - Using Skupper podman

About Skupper podman

Creating a site using Skupper podman

Linking sites using Skupper podman

Working with services using Skupper podman

Skupper - Using the Skupper console

Enabling the Skupper console

Accessing the Skupper console

Exploring the Skupper console

Skupper - Configuring Skupper sites using YAML

Creating a Skupper site using YAML

Linking sites using YAML

Configuring services using annotations

Appendix A: Site ConfigMap YAML reference

Skupper - Troubleshooting a service network

Checking sites

Checking links

Checking gateways

Creating a Skupper debug tar file

Improving Skupper router performance

Resolving common problems

Skupper - Securing a service network using policies

About the policy system

Upgrading on a cluster with existing sites

Creating policies for the policy system

Exploring the current policies for a cluster

Creating a site using the Skupper Operator