**Container Networking Basics**

**Objectives**

**• Describe how containers communicate with each other.**

**Container Networking Basics**

Podman comes with a network called podman. By default, containers are attached to this network

and can use it to communicate with one another.

However, you might need to create a new Podman network to better suit the increased

communication needs of most applications. For example, the containers running an application

API and database can use a separate Podman network to isolate their communication from

other containers. Similarly, that same API container can use yet another network to isolate

communication with a third container that hosts the application UI.

Figure 2.8: Example of isolated communication by using Podman networks

In the preceding example diagram, the UI and API containers are attached to the ui-network

Podman network. The API and database containers are attached to the api-network Podman

network.

Managing Podman Networks

Podman network management is done via the podman network subcommand. This

subcommand includes the following operations:

***podman network create***

Creates a new Podman network. This command accepts various options to configure

properties of the network, including gateway address, subnet mask, and whether to use IPv4

or IPv6.

***podman network ls***

Lists existing networks and a brief summary of each. Options for this command include various

filters and an output format to list other values for each network.

***podman network inspect***

Outputs a detailed JSON object containing configuration data for the network.

***podman network rm***

Removes a network.

***podman network prune***

Removes any networks that are not currently in use by any running containers.

***podman network connect***

Connects an already running container to or from an existing network. Alternatively, connect

containers to a Podman network on container creation by using the --net option. The

disconnect command disconnects a container from a network.

For example, the following command creates a new Podman network called example-net:

[user@host ~]$ **podman network create example-net**

To connect a new container to this Podman network, use the --net option. The following example

command creates a new container called my-container, which is connected to the examplenet network.

[user@host ~]$ **podman run -d --name my-container --net example-net container-image:latest**

When you create new containers, you can connect them to multiple networks by specifying

network names in a comma-separated list. For example, the following command creates a new

container called double-connector that connects to both the postgres-net and redis-net

networks.

[user@host ~]$ **podman run -d --name double-connector --net postgres-net,redis-net \ container-image:latest**

Alternatively, if the my-container container is already running, then run the following command

to connect it to the example-net network:

[user@host ~]$ **podman network connect example-net my-container**

Important

If a network is not specified with the podman run command, then the container

connects to the default network. The default network uses the slirp4netns

network mode, and the networks that you create with the podman network

create command use the bridge network mode. If you try to connect a bridge

network to a container by using the slirp4netns network mode, then the

command fails.

Enabling Domain Name Resolution

When you use the default Podman network, the domain name system (DNS) for other containers

in that network is disabled. To enable DNS resolution between containers, create a Podman

network and connect your containers to that network.

When using a network with DNS enabled, a container's hostname is the name assigned to the

container. For example, if a container is started with the following command, then the other

containers on the test-net network can make requests to the first container by using the

basic-container hostname. The basic-container hostname resolves to the current IP

address of the basic-container container.

[user@host ~]$ **podman run --net test-net --name basic-container example-image**

Connecting Containers

You can connect containers to one or more Podman networks. After a container connects to

a network, the container can communicate with other containers on that network. However,

even though the containers are reachable to one another, other components might prevent

connections. For example, firewall rules might block a connection coming from another container.

By default, a container is available within any network that the container connects to.

For example, consider a running container called nginx-host that uses the example-net

network. The container exposes an HTTP server on port 8080. Within another container that uses

the example-net network, the following curl command resolves to the root of the HTTP server.

[user@host ~]$ **curl http://nginx-host:8080**

**References**

Basic Networking Guide for Podman

https://github.com/containers/podman/blob/main/docs/tutorials/basic\_networking.md

Podman 4.0's new network stack: What you need to know

https://www.redhat.com/sysadmin/podman-new-network-stack