A CLOUD GURU

Lesson Guide - Managing Containers Using Podman

There's more to managing containers than just running them. In this lesson, we're going to expand our Podman container horizons and take a look at some additional container management operations. Upon completion of this lesson, you will be able to manage different operational states of containers, execute commands in a running container, and interact with a running container.

Resources

Building, Running, and Managing Linux Containers on Red Hat Enterprise Linux 8 - Red Hat

Instructions

We need to get these containers under control!

Now that we've run our first container, and have a grasp on the basics of how to run containers, it's time to dig deeper. We're going to look at a few ways we can change the state of a container — running or not — and also at some ways to manage running containers.

Let's go!

Commands Covered

- `podman container`: the high-level command used to manage containers
- `podman create`: create a new container, but don't start it
- `podman exec`: execute a command in a running container
- `podman kill`: kill one or more running containers with a specific signal
- `podman pause/unpause`: pause/unpause one or more running containers
- `podman run`: run a command in a new container, using a given image
- `podman stop/start/restart`: stop/start/restart one or more containers

Changing the State of a Container

First, let's check for containers:

```
podman ps -a
```

Next, we're going to run a `pms-docker` Plex Media Server container:

```
podman run -dt docker.io/plexinc/pms-docker
```

Then, check for the container:

```
podman ps -a
```

Note that the container has been assigned a container ID.

We're going to go ahead and stop our `pms-docker` container:

```
podman stop <container_ID>
```

Check the container:

```
podman ps -a
```

Our container is stopped.

Let's start it back up:

```
podman start <container_ID>
```

Check the container:

```
podman ps -a
```

Our container is running again!

Let's try restarting our container:

```
podman restart <container_ID>
```

Check the container:

```
podman ps -a
```

We can see that our container has been restarted.

Occasionally, our container may stop behaving, and we may need to kill it. We can do that using the `podman kill` command:

```
podman kill <container_ID>
```

Check the container:

```
podman ps -a
```

Sometimes, we want to create a `nginx` container, but not start it:

```
podman create -t --name mynginx docker.io/library/nginx
```

Check the container:

```
podman ps -a
```

We see our `nginx` container, with the name `mynginx`, ready for us to call it to action. Let's start it:

```
podman start mynginx
```

Check the container:

```
podman ps -a
```

The `mynginx` container is up and running!

There are a couple of operations we can only run on `root` containers. Let's try them using `sudo`.

Let's start a `ubi8` container as root, using `sudo`:

```
sudo podman run -dt --name rootubi8 registry.access.redhat.com/ubi8
```

Check the container:

```
sudo podman ps -a
```

We can see the `rootubi8` container, running as `root`.

We can pause the `rootubi8` container:

```
sudo podman pause rootubi8
```

Check the container:

```
sudo podman ps -a
```

The `rootubi8` container is paused!

Let's try unpausing the container, using `podman unpause` with `sudo`:

```
sudo podman unpause rootubi8
```

Check the container:

```
sudo podman ps -a
```

The `rootubi8` container is running again!

Working With Running Containers

Now that we've explored many ways we can change the state of containers, let's explore how we can work with running containers.

The "main" `podman` command to manage containers is the `podman container` command. Let's take a look:

```
podman container --help | more
```

Many of the commands we've covered so far have a "twin" command under `podman container`.

For example, we used `podman start` to start our `pms-docker` container earlier. Another way to accomplish this is the `podman container start` command:

```
podman container start <container_ID>
```

We get the same result as with `podman start`.

It pays to use the `podman --help` functionality to drill down and explore the various `podman` commands, or you can use the `man pages` as well.

Sometimes we want to interact with our container. One way is to use the `podman exec` command to run a `bash` shell:

```
podman exec -it mynginx /bin/bash
```

This gives us an interactive shell. Let's leave the shell:

```
exit
```

Maybe we want to run a command inside a container, but not interact. We can also use `podman exec` for this.

Let's take a look at the contents of `/etc/redhat-release` in the `rootubi8` container, using `sudo`:

```
sudo podman exec rootubi8 cat /etc/redhat-release
```

Wrapping It Up

We've got a solid handle on basic container management, so we're going to wrap things up for now. We've got more container management coming up in the next few lessons, so don't worry!

Let's clean up our containers:

```
podman stop -a
```

```
podman rm -a
```

```
podman ps -a
```

Let's clean up the `root` containers:

```
sudo podman stop -a
```

```
sudo podman rm -a
```

```
sudo podman ps -a
```

Let's clean up our images:

```
podman images
```

```
podman rmi -a
```

```
podman images
```

Don't forget the `root` images:

```
sudo podman images
```

```
sudo podman rmi -a
```

```
sudo podman images
```

All cleaned up!

Our work here is complete!

Notes

Recording - Environment used: Cloud Playground - Medium 3 unit RHEL 8 Cloud Server

Environment Setup:

Create your Cloud Playground server and log in.

Install the `container-tools` Application Stream:

```
sudo yum -y module install container-tools
```

You're ready to go!