A CLOUD GURU

Lesson Guide - Managing Podman Containers Using Cockpit

Cockpit is an easy-to-use, web-based system management tool for Linux servers that can manage Podman containers using the `cockpit-podman` plugin. In this lesson, we will install Cockpit and the `cockpit-podman` plugin, and will use Cockpit to manage Podman containers. Upon completion of this lesson, you will be able to configure a Cockpit installation and use it to manage your Podman containers.

Resources

An Introduction to Cockpit, a Browser-Based Administration Tool for Linux - Red Hat

Managing Linux Servers with Cockpit

Instructions

Self-serve WordPress instances, anyone?

We want to come up with an easy, self-serve set of instructions we can use to provide individual development WordPress instances for a RHEL development environment. We want to minimize command-line requirements, focusing on using the Cockpit web-based system management interface to manage and deploy our WordPress containers.

How do we do this?

Commands Covered

- `yum install`: installs a package or packages on your system
- `yum module install`: installs a module or modules on your system
- `systemctl enable --now`: enables and starts a `systemd` service

Install Cockpit and the Podman Plugin

Install the `container-tools` Application Stream:

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

Note that the `cockpit-podman` package is installed as part of the `container-tools` Application Stream. We don't have to install it separately.

We do have to install `cockpit`, though:

```
sudo yum -y install cockpit
```

Next, enable and start the `cockpit.socket` using `systemctl`:

```
sudo systemctl enable --now cockpit.socket
```

Now we can connect to the Cockpit web console on port `9090`. Let's log in as `cloud_user`.

Let's take a look at the tab for the **Podman containers** plugin.

You'll notice that we get a message that the "Podman service is not active." Let's start it, and also start the *User Podman* service.

Now that we're good to go, we see that we have no containers or images. Let's change that!

First, let's grab some images. We'll grab the following:

- wordpress (from docker://docker.io/library/wordpress)
- mariadb (from docker://docker.io/library/mariadb)

Start a `mariadb` container

- Name the container `wp-db`
- Publish port `3306` in the container to port `3306` on the host
- Use the `mariadb` container image
- Set the following variables
  - MYSQL_ROOT_PASSWORD="dbpass"
  - MYSQL_DATABASE="wp"
  - MYSQL_USER="wordpress"
  - MYSQL_PASSWORD="wppass"

Start a WordPress container

- Name the container `wp-web`
- Publish port `80` in the container to port `8080` on the host
- Use the `wordpress` container image
- Set the following variables
  - WORDPRESS_DB_NAME="wp"
  - WORDPRESS_DB_USER="wordpress"
  - WORDPRESS_DB_PASSWORD="wppass"
  - WORDPRESS_DB_HOST="[private IP address of host]"

Both of our containers are up and running! Let's try connecting to our WordPress installation using the external DNS or IP address of our server and port `8080`. You should get the WordPress setup page.

Now that we're done testing, we'll clean up all the containers and images.

Great work, Cloud Guru! You stood up a WordPress instance using Cockpit and the Podman plugin!

Notes

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

Environment Setup:

Create your Cloud Playground server and log in.

You're ready to go!