A CLOUD GURU

Lesson Guide - Managing Container Storage Using Podman

Containers aren't always islands unto themselves. Sometimes you will want to present existing data or add persistent storage to your container. In this lesson, we will examine how to manage container storage. When you finish this lesson, you will have a solid understanding of how container storage using Podman works.

Resources

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

Instructions

Our NGINX containers need their content!

We want to run a number of `nginx` containers, serving their content from a shared directory or container volume. We're going to see how we can present this persistent storage to our containers so they can serve it.

Let's see how it works!

Commands Covered

- `podman run with -v`: using the `--volume` or `-v` switch with `podman run` to add persistent storage to a container
- `podman volume`: create and manage container volumes
- `podman cp`: copy data from the local filesystem to a container filesystem, or vice versa

Adding Persistent Storage to a Container Using a Shared Directory

We want to run three `nginx` containers, serving content from the `~/html` directory. We're going to create a couple of test text files in the directory and try accessing them using `curl`.

Let's check for containers:

```
podman ps -a
```

We're going to create our `~/html` directory:

```
mkdir ~/html
```

Let's create a text file in our `~/html` directory:

```
echo Testfile! > ~/html/test1.txt
```

Let's look for an `nginx` container image to use:

```
podman search nginx | more
```

Let's use `docker.io/library/nginx`.

Let's start three `nginx` containers, with the names `web1`, `web2` and `web3`. We will attach the `~/html` directory we just created to the `/usr/share/nginx/html/` directory in the container:

```
podman run -dt --name web1 -v ~/html:/usr/share/nginx/html/:z
docker.io/library/nginx
```

Note: You may notice that we used a lowercase `z` to enable SELinux instead of the uppercase `Z` we would normally use with a single container bind. This allows more than one container access to the shared `~/html` directory.

Let's start the other two `nginx` containers:

```
podman run -dt --name web2 -v ~/html:/usr/share/nginx/html/:z
docker.io/library/nginx
```

```
podman run -dt --name web3 -v ~/html:/usr/share/nginx/html/:z
docker.io/library/nginx
```

Let's check for containers again:

```
podman ps -a
```

We see our three containers, running.

Let's use `curl` in our containers to see if we can access the `test1.txt` file via `nginx`:

```
podman exec web1 curl -s http://localhost:80/test1.txt
```

```
podman exec web2 curl -s http://localhost:80/test1.txt
```

```
podman exec web3 curl -s http://localhost:80/test1.txt
```

We can see the contents of our `test1.txt` file!

Let's create another text file called `text2.txt` in `~/html`:

```
echo A second testfile! > ~/html/test2.txt
```

Let's use `curl` in our containers to see if we can access the `test2.txt` file via `nginx`:

```
podman exec web1 curl -s http://localhost:80/test2.txt
```

```
podman exec web2 curl -s http://localhost:80/test2.txt
```

```
podman exec web3 curl -s http://localhost:80/test2.txt
```

We can see the contents of our `test2.txt` file!

Let's clean up our containers:

```
podman stop -a
```

```
podman rm -a
```

```
podman ps -a
```

We're ready to move on!

Adding Persistent Storage to a Container Using a Container Volume

Another way we can share data with containers is by creating a container volume and attaching it to our `nginx` containers.

Let's create a volume:

```
podman volume create webvol
```

Let's list our volumes:

```
podman volume ls
```

Let's start three `nginx` containers, with the names `web1`, `web2` and `web3`. We will attach the `webvol` volume we just created to the `/usr/share/nginx/html/` directory in the container:

```
podman run -dt --name web1 -v webvol:/usr/share/nginx/html/
docker.io/library/nginx
```

```
podman run -dt --name web2 -v webvol:/usr/share/nginx/html/
docker.io/library/nginx
```

```
podman run -dt --name web3 -v webvol:/usr/share/nginx/html/
docker.io/library/nginx
```

Let's take a look at the `/usr/share/nginx/html/` directory in the `web1` container:

```
podman exec web1 ls -al /usr/share/nginx/html/
```

Looks good so far!

Let's copy our text files in `~/html` to our `/usr/share/nginx/html/` directory in the `web1` container:

```
podman cp ~/html/test1.txt web1:/usr/share/nginx/html
```

```
podman cp ~/html/test2.txt web1:/usr/share/nginx/html
```

Let's take a look at the `/usr/share/nginx/html/` directory in all three containers:

```
podman exec web1 ls -al /usr/share/nginx/html/
```

```
podman exec web2 ls -al /usr/share/nginx/html/
```

```
podman exec web3 ls -al /usr/share/nginx/html/
```

We can see our text files in all three containers!

Let's use `curl` in our containers to see if we can access the `test1.txt` file via `nginx`:

```
podman exec web1 curl -s http://localhost:80/test1.txt
```

```
podman exec web2 curl -s http://localhost:80/test1.txt
```

```
podman exec web3 curl -s http://localhost:80/test1.txt
```

We can see the contents of our `test1.txt` file!

Let's use `curl` in our containers to see if we can access the `test2.txt` file via `nginx`:

```
podman exec web1 curl -s http://localhost:80/test2.txt
```

```
podman exec web2 curl -s http://localhost:80/test2.txt
podman exec web3 curl -s http://localhost:80/test2.txt
```

We can see the contents of our `test2.txt` file!

Let's clean up our containers:

```
podman stop -a
```

```
podman rm -a
```

```
podman ps -a
```

Excellent!

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!