A CLOUD GURU

Lesson Guide - Creating a Container Image Using Buildah Native Commands

One way to create a custom container image is to use Buildah native commands. In this lesson, we will take a look at how to use Buildah and an existing container image to define and create a custom image via Buildah native commands. Upon completion of this lesson, you will be able to create a container image using Buildah native commands.

Resources

Buildah

Getting Started with Buildah - Red Hat

Building with Buildah: Dockerfiles, Command Line, or Scripts - Red Hat - Enable Sysadmin

Building Container Images with Buildah - Red Hat

Instructions

Let's create a custom Apache container image!

Using Buildah native commands, we're going to create our own Apache web server container image, based on the latest Fedora Linux image, and add some content to it. We'll then launch five instances of our custom `my-fedora-httpd` container image and test our work.

Let's go!

Commands Covered

- `buildah images`: lists locally stored images
- `buildah containers`: lists containers which appear to be Buildah working containers
- `buildah from`: creates a new working container, either from scratch or using a specified image as a starting point
- `buildah config`: modifies the configuration values which will be saved to the image
- `buildah run`: runs a specified command using the container's root filesystem as a root filesystem, using configuration settings inherited from the container's image or as specified using previous calls to the config command
- `buildah commit`: writes a new image using the container's read-write layer and, if it is based on an image, the layers of that image
- `buildah rmi`: removes one or more locally stored images
- `podman ps`: displays information about containers and pods
- `podman run`: runs a command in a new container
- `podman stop`: stops one or more containers
- `podman rm`: removes one or more containers from the host

Creating a Custom Container Image Using Buildah Native Commands

Before we start, let's check the status of our Podman environment:

```
podman ps -a
```

```
buildah images
```

```
buildah containers
```

We're going to use the following criteria to build a custom container image:

- Based on the `fedora:latest` container image
- Maintainer = `“buildah@podman.rulez”`
- Install `httpd` in the container
    - Clean up after install
- Create five test text files in `/var/www/html`
    - Contents should be `"Test File <1-5>"`
    - Filenames should be `"test<1-5>.txt"`
- Expose port 80
- Run our `httpd` service using `"/usr/sbin/httpd -D FOREGROUND"`
- Name our image `my-fedora-httpd:latest`

Now that we know what we're going to build, let's build it!

Build Our Custom `my-fedora-httpd` Container Image

Let's create our container for our image:

```
container=$(buildah from fedora:latest)
```

Checking our new working container:

```
buildah containers
```

Checking the value for `$container`:

```
echo $container
```

Add our maintainer information:

```
buildah config --label maintainer="buildah@podman.rulez" $container
```

Next, let's install `httpd` in our container image:

```
buildah run $container dnf install -y httpd
```

We see the installation output scroll by.

Cleaning up after our `dnf` operation in our container:

```
buildah run $container dnf clean all
```

This gives us a smaller container image.

Now we're going to create our test text files:

```
buildah run $container bash -c "echo \"Test File 1\" >
/var/www/html/test1.txt"
```

```
buildah run $container bash -c "echo \"Test File 2\" >
/var/www/html/test2.txt"
```

```
buildah run $container bash -c "echo \"Test File 3\" >
/var/www/html/test3.txt"
```

```
buildah run $container bash -c "echo \"Test File 4\" >
/var/www/html/test4.txt"
```

```
buildah run $container bash -c "echo \"Test File 5\" >
/var/www/html/test5.txt"
```

Test files created!

Finally, we're going to expose port 80 for our web server, set the start command for our `httpd` service and commit our container image to `my-fedora-httpd:latest`:

```
buildah config --port 80 $container
```

```
buildah config --cmd "/usr/sbin/httpd -D FOREGROUND" $container
```

Before we commit, let's check our containers using `buildah`:

```
buildah containers
```

Let's commit our new image:

```
buildah commit --format docker $container my-fedora-httpd:latest
```

Now, checking our `buildah` images and containers:

```
buildah images
```

```
buildah containers
```

We see our new `my-fedora-httpd:latest` container image and our working container as well.

Let's try using our new `my-fedora-httpd` container image!

Running Our `my-fedora-httpd` Containers

```
podman run -d --name my-fedora-httpd-1 -p 8081:80 localhost/my-fedora-
httpd
```

```
podman run -d --name my-fedora-httpd-2 -p 8082:80 localhost/my-fedora-
httpd
```

```
podman run -d --name my-fedora-httpd-3 -p 8083:80 localhost/my-fedora-
httpd
```

```
podman run -d --name my-fedora-httpd-4 -p 8084:80 localhost/my-fedora-
httpd
```

```
podman run -d --name my-fedora-httpd-5 -p 8085:80 localhost/my-fedora-
httpd
```

Checking our work:

```
podman ps -a
```

Testing Our `my-fedora-httpd` Containers

```
curl -s http://localhost:8081/test1.txt
```

```
curl -s http://localhost:8082/test2.txt
```

```
curl -s http://localhost:8083/test3.txt
```

```
curl -s http://localhost:8084/test4.txt
```

```
curl -s http://localhost:8085/test5.txt
```

Mixing it up a little:

```
curl -s http://localhost:8085/test1.txt
```

We can also test a connection using our web browser and the IP address of our host machine, plus the port of the container we're looking to connect to (8081-8085).

Cleaning Up!

Let's clean up after ourselves!

```
podman stop -a
```

```
podman rm -a
```

```
buildah rmi -a
```

```
buildah images
```

```
podman ps -a
```

Congratulations! You just created a custom container image using Buildah native commands!

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!