A CLOUD GURU

Lesson Guide - Using Buildah to Manage Container Images - Part 2

From time to time, we will need to perform various management operations on our container images. In this lesson, we will look at how we can use Buildah to manage container images. Upon completion, you will have a solid understanding of how to manage container images using Buildah.

Resources

Buildah

Building Container Images with Buildah - Red Hat

Buildah and Podman Relationship - podman.io

Instructions

Let’s take a look at the rest of Buildah!

Now that we’ve mastered building custom container images using Buildah, let’s take a look at all the other things we can do. We’re going to look at the other Buildah commands that make the experience complete.

Let’s jump in!

Commands Covered

- `buildah`: tool that facilitates building OCI images

Image Management Using Buildah

The following `buildah` commands are used to manage images.

Commands: `images`/`login`/`logout`/`pull`/`push`/`rmi`/`tag`

If we'd like to pull a list of container images in our local storage, we can use:

```
buildah images
```

We see our container images.

Let's log in to our local container registry:

```
buildah login https://localhost:5000
```

The username is `cloud_user` and password is `registry`.

We're going to push our `my-fedora-httpd:latest` container image to our local container registry:

```
buildah push localhost/my-fedora-httpd:latest docker://localhost:5000/my-
fedora-httpd:latest
```

Checking our work:

```
curl -u cloud_user:registry https://localhost:5000/v2/_catalog
```

We see the repository for `my-fedora-httpd`.

Let's remove the `my-fedora-httpd:latest` image from our local storage:

```
buildah images
```

```
buildah rmi my-fedora-httpd:latest
```

```
buildah images
```

We see that our `my-fedora-httpd:latest` image has been removed.

Let's get it back, from our local container registry:

```
buildah pull docker://localhost:5000/my-fedora-httpd:latest
```

```
buildah images
```

We see the `my-fedora-httpd:latest` image.

Let's add a tag for our `my-fedora-httpd:latest` image:

```
buildah tag localhost:5000/my-fedora-httpd:latest
ourcustomwebserver:latest
```

```
buildah images
```

We see our new tag, `ourcustomwebserver:latest`.

Finally, let's log out of our local container image registry:

```
buildah logout https://localhost:5000
```

We're logged out.

On to container management!

Container Management Using Buildah

These commands are used to manage containers, but not in the way we use containers with `podman`. The `podman` command is used to create containers for regular consumption, while `buildah` creates containers that are used to build container images. We would never run the `buildah` containers in production.

Commands: `containers`/`rename`/`rm`

We can display a list of containers that `buildah` manages using:

```
buildah containers
```

Let's rename our `fedora-working-container` to `my-container`:

```
buildah rename fedora-working-container my-container
```

Checking again:

```
buildah containers
```

Now let's remove the `my-container` container:

```
buildah rm my-container
```

One more check:

```
buildah containers
```

We're cleaned up!

Great work, Cloud Gurus!

Notes

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

Dockerfile:

```
FROM fedora:latest
LABEL maintainer fedora-apache-container <apache@podman.rulez>

RUN dnf install -y httpd && dnf clean all

RUN echo "Test File 1" > /var/www/html/test1.txt
RUN echo "Test File 2" > /var/www/html/test2.txt
RUN echo "Test File 3" > /var/www/html/test3.txt
RUN echo "Test File 4" > /var/www/html/test4.txt
RUN echo "Test File 5" > /var/www/html/test5.txt

EXPOSE 80
CMD mkdir /run/httpd ; /usr/sbin/httpd -D FOREGROUND
```

Environment Setup:

Create your Cloud Playground server and log in.

Install the `container-tools` Application Stream:

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

Copy the contents of the Dockerfile into a file called `Dockerfile`.

Create a container image using your Dockerfile:

```
buildah bud -t my-fedora-httpd:latest
```

Check your work:

```
buildah images
```

Set Up a Container Registry

How to Implement a Simple Personal/Private Linux Container Image Registry for Internal Use - Red Hat

I used `localhost` for the hostname, as I'm going to access the container registry using `localhost`.

Steps to deploy:

Become `root`:

```
sudo -i
```

Install `httpd-tools`:

```
yum install -y httpd-tools
```

Create the directories for our container registry:

```
mkdir -p /opt/registry/{auth,certs,data}
```

Create our container registry credentials:

```
htpasswd -bBc /opt/registry/auth/htpasswd cloud_user registry
```

Generate our SSL certificate. I used `localhost` for the hostname, as I'm going to access the container registry using `localhost`:

```
openssl req -newkey rsa:4096 -nodes -sha256 -keyout
/opt/registry/certs/domain.key -x509 -days 365 -out
/opt/registry/certs/domain.crt
```

Add our certificate:

```
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
```

Update our CA certificates:

```
update-ca-trust
```

Check your work:

```
trust list | grep -i localhost
```

Start your container registry:

```
podman run --name myregistry -p 5000:5000 -v
/opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -e
"REGISTRY_AUTH=htpasswd" -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm"
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd -v
/opt/registry/certs:/certs:z -e
"REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" -e
"REGISTRY_HTTP_TLS_KEY=/certs/domain.key" -e
REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true -d
docker.io/library/registry:latest
```

Test your container registry:

```
curl -u cloud_user:registry https://localhost:5000/v2/_catalog
```

Exit `root`:

```
exit
```

You're ready to go!