Containers

Docker and Podman commands

Buildah

Basic howto for building a container image with Buildah.

This example set will build the GNU hello application on a Fedora image. To follow along, download the hello source from here: http://mirrors.kernel.org/gnu/hello/

Start with a base image.

```
buildah from fedora:latest
```

Get a list of buildah containers

```
buildah containers
```

```
CONTAINER ID  BUILDER  IMAGE ID     IMAGE NAME                       CONTAINER NAME
4df293490fa6     *     b81e86a2cb9a                                   ubi8-working-container
52a1835eea0d     *     2ef880c78ce8                                   2ef880c78ce8df33284cb56ab10
e3cc371f1df4     *     750037c05cfe registry.fedoraproject.org/fe...  fedora-working-container
```

Set a container variable. (Reduces typing, great for scripting)

```
container=fedora-working-container
```

Copy the source to the container

```
buildah copy $container hello-2.14.tar.gz /tmp/
```

Install required packages into the container, then clear the package cache.

```
buildah run $container dnf install -y gzip tar gcc make automake
buildah run $container dnf clean all
```

Untar the source to /opt

```
buildah run $contaner tar zxvf /tmp/hello-2.12.tar.gz -C /opt
```

Set the working directory

```
buildah config --workingdir /opt/hello-2.12 $container
```

Build the software as required.

```
buildah run $container ./configure
buildah run $container autoconf
buildah run $container make
buildah run $container cp hello /usr/local/bin
```

Check the build and that the binary is in the correct location.

```
buildah run $container hello --version
```

Add an ENTRYPOINT

```
buildah config --entrypoint /usr/local/bin/hello $container
```

Commit the build container to an image.

```
buildah commit --format docker $container hello:latest
```

```
Getting image source signatures
Copying blob c550c8e0f355 skipped: already exists
Copying blob c15d843d546e done
Copying config 3d1cc0ca89 done
Writing manifest to image destination
Storing signatures
3d1cc0ca8946cfe2000a47ee4c33133e23abc4c05dc4f959fbb0a383f2c2175c
```

Run the image.

```
podman run hello
```

Remove the build directory

```
buildah rm $container
```

Dockerfile and container images

Building container images using a Dockerfile.

Here's a Dockerfile example that runs a simple nmap.

Dockerfile

```
# Start with a base image
FROM redhat/ubi8
# Maintainer information
LABEL org.opencontainers.image.authors="mail@clusterapps.com"
LABEL description="Simple Network scan"

# Run commands to build the container
# Do as much with the fewest RUN lines
RUN dnf -y update && \
dnf -y install \
nmap iproute procps-ng && \
bash && \
dnf clean all

# Entrypoint is the command that run when the ccontainer starts
ENTRYPOINT ["/usr/bin/nmap"]
#  The arguments for the entrypoint
CMD ["-sn", "192.168.252.0/24"]
```

What to know about the file:

- FROM

The FROM is the base used for the new image.

- LABEL

LABEL adds metadata to an image

- RUN

The run command is that steps taken to build the image. Each RUN command will build an additional layer to the image. It is best to use the fewest RUN entries as possible.

- ENTRYPOINT

This is the what the container will run when it first starts. This might be a binary or a script that starts jobs or services.

- CMD

These are the arguments to the ENTRYPOINT. The CMD can be overwritten on the command line.

See https://docs.docker.com/engine/reference/builder/ for more detais.

Build, run, tag, and push

Build with a tag.

```
podman build -t nmap:latest .
```

Run the image with the built in CMD.

```
podman run nmap
```

Run with different CMD

```
podman run nmap -sT -Pn -p80 192.168.252.210
```

Tag for a repository

```
podman tag localhost/nmap:latest  quay.io/clearyme/nmap:latest
```

Push to repository

```
podman push quay.io/clearyme/nmap:latest
```

Podman

Basics

Registry file: */etc/containers/registries.conf*

Login to a registry

```
podman login registry.access.redhat.com
```

Search for images

```
podman search mariadb
```

Inspect images without downloading

```
skopeo inspect docker://registry.access.redhat.com/rhscl/mariadb-102-rhel7
```

Download images

```
podman pull registry.access.redhat.com/rhscl/mariadb-102-rhel7
```

List local images

```
podman images
```

Start a container based on an image ID. Get the ID from docker images.

```
podman run --name apache bitnami/apache
```

control-c will stop the container for all of the run commands.

Start an image based on a tag detached

```
podman run -d --name apache bitnami/apache:2.4.52
```

Start the an app with port forwarding

```
podman run -it -p 8080:8080 bitnami/apache
```

Get running images

```
podman ps
```

Get all images

```
podman ps -all
```

Enter container in interactive shell

```
podman exec -it container-name /bin/bash
```

Commit changes to running image

```
podman commit container-name image-name
```

Check container logs

```
podman logs <Container Name>
podman logs -f <Container Name> # Follow the logs
podman logs --tail=25 <Container Name> # Last n lines
```

Stop a running image. The container ID will be in the `podman ps` output.

```
podman kill <Container ID>
```

Remove an image. The container ID will be in the `podman ps` output.

```
podman rm <Container ID>
```

Remove all images.

```
podman rmi --all --force
```

Export image

```
podman save image-name > /path/to/image.tar
```

Restore/Load image

```
podman load -i /path/to/image.tar
```

Parameters and Volumes

Create a container mount point

```
sudo mkdir /srv/mariadb
sudo chown -R 27:27 /srv/mariadb # UID found in podman inspect
sudo semanage fcontext -a -t container_file_t "/srv/mariadb(/.*)"
sudo restorecon -Rv /srv/mariadb
```

Run image
-d detached
-e per variable
-p local_port:container_port
-v local/path:/path/in/pod

```
podman run -d -e MYSQL_USER=user \
-e MYSQL_PASSWORD=pass -e MYSQL_DATABASE=db \
-p 33306:3306 rhscl/mariadb-102-rhel7 \
-v /srv/mariadb:/var/lib/mysql:Z # :Z isn't needed if SELinux manually configured
```

Pods

Create a pod for rootless containers with a specific name and map ports needed. This example creates a Wordpress pod with a dedicated MySQL database using the Bitnami Wordpress image and a MySQL image from Red Hat.

Create a storage area.

```
sudo mkdir /srv/pods/wordpress/database
sudo mkdir /srv/pods/wordpress/sitedata
sudo chown -R poduser:poduser /srv/pods/wordpress # Host user running the pod
sudo semanage fcontext -a -t container_file_t "/srv/pods/wordpress(/.*)"
sudo restorecon -Rv /srv/pods/wordpress
```

Create the pod with port maps for 8443.

```
podman pod create --name press -p 8443:8443
```

Deploy the MySQL container

```
podman run -d --pod press --name mysql \
-e MYSQL_ROOT_PASSWORD=ThereIsAWordHere \
-e MYSQL_USER=wordpress \
-e MYSQL_PASSWORD=presswords \
-e MYSQL_DATABASE=wordpress \
-v /machines/pods/wordpress/database:/var/lib/mysql:Z \
mysql-80-rhel7
```

Deploy the wordpress container.

```
podman run -d --name words --pod press \
-e WORDPRESS_DATABASE_HOST=press \
-e WORDPRESS_DATABASE_USER=wordpress \
-e WORDPRESS_DATABASE_NAME=wordpress \
-e WORDPRESS_DATABASE_USER=wordpress \
-e WORDPRESS_DATABASE_PORT_NUMBER=3306 \
-e WORDPRESS_DATABASE_PASSWORD=presswords \
-v /machines/pods/wordpress/site:/bitnami/wordpress:Z \
bitnami/wordpress
```

Log in to Wordpress at `https://hostname:8443`

Systemd

Create system .service files.

To create systemd files for the above Wordpress pod:

```
podman generate systemd --files --name press
```

Creates: `container-mysql.service  container-words.service  pod-press.service`

Copy the generated file to the user's systemd directory and reload the deamons.

```
cp * ~/.config/systemd/user/
systemctl --user daemon-reload
```

Enable the service at boot time

```
systemctl --user enable pod-press.service
```

Pod scripts

Here are a few quick scripts to get pods up and running quickly on Podman.

Gitea

Gitea is a self-hosted git server.

```
podman pod create --name gitea -p3000:3000 -p 2222:22
podman run -d --pod gitea --name gitea_db -e POSTGRES_PASSWORD=gitea -e POSTGRES_USER=gitea -e POSTGRES_DB=gitea -v /srv/gitea/pgdata:/var/lib/postgresql/data:Z docker.io/library/postgres
podman run -d --pod gitea --name gitea_srv -e USER_ID=1000 -e USER_GID=1000 -e DB_TYPE=postgres -e DB_HOST=gitea:5432 -e DB_NAME=gitea -e DB_PASSWD=gitea -e DB_USER=gitea -v /srv/gitea/data:/data:Z docker.io/gitea/gitea:latest
```

Wiki.js

```
export DATA=$PWD
export PASSWD=zo03gaeCi1
mkdir -p $DATA/wiki/pgdata
mkdir -p $DATA/wiki/data
chown -R 1000:1000
podman pod create --name wiki -p 3001:3000
podman run -d --pod wiki --name wiki_db -e POSTGRES_PASSWORD=$PASSWD -e POSTGRES_USER=wiki -e POSTGRES_DB=wiki -v $DATA/wiki/pgdata:/var/lib/postgresql/data:Z docker.io/library/postgres
podman run -d --pod wiki --name wiki_srv -e DB_TYPE=postgres -e DB_HOST=wiki -e DB_PORT=5432 -e DB_USER=wiki -e DB_PASS=$PASSWD -e DB_NAME=wiki -v $DATA/wiki/data:/wiki/data:Z ghcr.io/requarks/wiki:2
```

MediaWiki

```
#!/bin/bash
export DATA=$PWD
mkdir -p $DATA/wiki/mediawiki
mkdir -p $DATA/wiki/mariadb
podman pod create --name wiki -p 8888:8080 -p 8889:8443
```

```
podman run -d --name mariadb --pod wiki \
  -e MARIADB_ROOT_PASSWORD=aeneinei9Wai \
  -e MARIADB_USER=wikiuser \
  -e MARIADB_PASSWORD=ieshuB7Oozie \
  -e MARIADB_DATABASE=wiki \
  -v $DATA/wiki/mariadb:/bitnami/mariadb:Z \
  docker.io/bitnami/mariadb
podman run -d --name mediawiki --pod wiki \
  -e MEDIAWIKI_DATABASE_HOST=wiki \
  -e MEDIAWIKI_DATABASE_USER=wikiuser \
  -e MEDIAWIKI_DATABASE_PASSWORD=ieshuB7Oozie \
  -e MEDIAWIKI_DATABASE_NAME=wiki \
  -e MEDIAWIKI_HOST="thefed.manor.one" \
  -e MEDIAWIKI_EXTERNAL_HTTP_PORT_NUMBER=8888 \
  -v $DATA/wiki/mediawiki:/bitnami/mediawiki:Z \
  docker.io/bitnami/mediawiki:latest
```