Author:-Dinesh.C.Patel

Infrastructure Support Engineer

(RHCSA,RHCE,DO188,AWS & HashiCorp Terraform Certified)[8766924468]

Email:- dineshcpatel2727@gmail.com

******Docker Full Notes and Commands******

What is OS?

Operating System a type of system software. It basically manages all the resources of the computer.

An operating system acts as an interface between the User and computer hardware.

What is Virtualization?

Virtualization is a technology that allows you to create and run multiple virtual instances of computing resources, such as operating systems, servers, or storage, on a single physical machine.

What is Containerization?

Containerization is a technology that allows you to package an application and all of its dependencies (like libraries, configurations, and binaries) into a **container**.

Key Benefits of Containerization:

Isolation: Each container is isolated from others, so applications in separate containers can't interfere with each other.

Portability: Containers can run on any system that supports the container runtime (Docker, Docker for example), whether it's your local machine, a cloud server, or on-premises hardware.

Scalability: Containers make it easy to scale applications by simply adding or removing containers.

Efficiency: Containers are more lightweight than traditional virtual machines because they share the host system's kernel instead of running their own OS.

What is Container?

Container is combination of OS, Software, libraries and Configuration files

Containers are lightweight a single server or (**Virtual Machine**) VM can run several containers simultaneously. It implements a high-level API to provide these lightweight containers that run processes in isolation.

Popular Container Technologies:

Docker: The most widely used containerization platform.

Podman: A daemonless alternative to Docker.

Containers vs. Virtual Machines (VMs):

Feature	Container	Virtual Machine
OS	Shares host OS kernel	Has its own guest OS
Size	Lightweight	Heavy (includes full OS image)
Startup Time	Seconds	Minutes
Resource Usage	Efficient	More resource-intensive
Isolation	Process-level isolation	Full isolation

What is Podman? (Short for **Pod Manager**)

Red Hat **Podman** is an open-source container management tool developed by **Red Hat**.

It is designed to simplify the creation, management, and deployment of containers and containerized applications. Docker is a lightweight, daemon less container engine that offers functionality similar to Docker.

What is pod?

A Pod is a logical host for one or more containers. Pods are used to run applications in containers.

What is Container file?

A container image is a static file that contains executable code and all the dependencies required to create a container on a computing system.

What is Registry?

A **registry** is a storage location for information related to software or systems.

Container Registry: A storage and distribution system for container images.

Examples:

Docker Hub

Amazon Elastic Container Registry (ECR)

Registry URL:

docker.io , quay.io, registry.access.redhat.com, registry.redhat.io **etc.**

Pull, Create Start, Stop, Restart, Rename & Copy Command in Container

dnf config-manager --add-repo

https://download.docker.com/linux/rhel/docker-ce.repo

dnf install -y dnf-plugins-core

dnf install -y docker-ce docker-ce-cli containerd.io

docker-buildx-plugin docker-compose-plugin

systemctl start docker.service

systemctl enable --now docker.service

systemctl status   docker.service

docker -v (docker version show)

docker image search almalinux (Search registry for image)

docker image pull almalinux (Pull an image from a registry)

docker image ls (to show download images)

docker container create -it almalinux (Create but do not start a container)

docker container ls (to show running containers)

docker container ls -a (to show all start/stop containers)

docker container start crazy_wilson OR id (c9173710e8c2) (to start stop container)

docker container attach crazy_wilson (to connect container)

exit (exit the container but stop container)

ctrl+p+q (without stop exit)

docker container stop crazy_wilson (to stop start container)

docker container restart crazy_wilson (to restart the container)

docker container rename crazy_wilson c1 (to rename the container)

docker container inspect c1 (to show the configuration of a container)

docker container top c1 (to show he running processes of a container)

docker container exec c1touch file1 (to create file in a running container)

docker container exec c1ls (to list in a running container)

docker container cp /etc/fstab c1:/tmp (to copy files to running container)

docker container exec c1 ls /tmp (list)

docker container cp ./. c1:/tmp (to copy all files from current location to container)

docker container cp c1:/file1 /tmp

(Copy a directory on a container to a directory on the host)

docker container cp containerA:/myapp containerB:/newapp (Copy

a directory on a container into a directory on another)

docker container exec c1 rm /tmp/fstab (to delete file in container)

docker container exec c1 ls /tmp (list)

docker container run -itd --name node1 rhel9 (image download and create container)

docker logs c1 (to show container logs)

docker stats (show a live stream of container resource usage statistics)

===== **To Remove Container & Image**=========

docker container stop bc126ff6eea6 (to stop the container)

docker container rm bc126ff6eea6 (to remove stop container)

docker container rm --force bc126ff6eea6 (to remove running container)

docker container stop --all (to stop all containers)

docker container rm --all (to remove all containers)

 # Remove Images

docker image rm docker.io/library/httpd (to remove only image)

docker imag rm --force docker.io/library/almalinux (to remove image with container)

docker image rm --all (to remove all images)

[NOTE: Container Configuration file : **/etc/containers/registries.conf**]

[NOTE: to store all images data in **/var/lib/containers/storage/overlay-images**]

[NOTE: to store all Container in **/var/lib/containers/storage/overlay-containers**]

Port Mapping with Apache Server:

Port mapping in Docker is a technique that allows you to access services running inside a Docker container from outside the container. #

docker container run -d -p <HostPort:containerport> imagename

docker container run -itd --name webserver -p 8080:80 httpd (map port to container)

(i= interactive, t = terminal, d=detach mode, p=port)

docker port webserver (to show Mapping port)

docker port --all (to show all Mapping Ports)

echo *Welcome to Docker Server* > index.html (to create index.html file)

docker container cp index.html webserver:htdocs (copy index files and folder)

docker container exec webserver ls htdocs (list data)

firefox http://192.168.0.165:8080 (to check /Host Machine IP Address)

Mariadb Service

docker run -d --name mariadb-server -e MYSQL_ROOT_PASSWORD=redhat -p 8181:3306 mariadb

docker container ls

yum install mysql -y

mysql -h 192.168.0.165 -P 8181 -u root -p (access mariadb)

Password: redhat

Login done

exit

Managing a Container Network

The chapter provides information about how to communicate Containers.

In Docker, there are two networks

1. **Rootless** networking - the container does not have an IP address.

2. **Rootful** networking - the container has an IP address.

docker network ls

docker network inspect docker (to show default network info)

How to Connect two containers

docker network create mynet (to create network)

docker container run -itd --name con1 almalinux (to create container con1)

docker container run -itd --name con2 almalinux (to create container con2)

docker container ls (to show container)

docker network connect mynet con1 (to connect network to con1 container)

docker network connect mynet con2 (to connect network to con2 container)

docker network inspect mynet (to show network info)

OR

docker container inspect con1 | grep -iwA 10 networks

Testing

docker exec -it con1 bash

[root@36e715907efd /]# ping -c 3 10.89.0.3

docker exec -it con2 bash

[root@36e715907efd /]# ping -c 3 10.89.0.2

exit

[Disconnect the container named con1 from the network named mynet]

docker network disconnect mynet con1

docker network disconnect mynet con1 #

docker network inspect mynet (to verify)

How to Delete Network

docker network rm mynet (to delete networks)

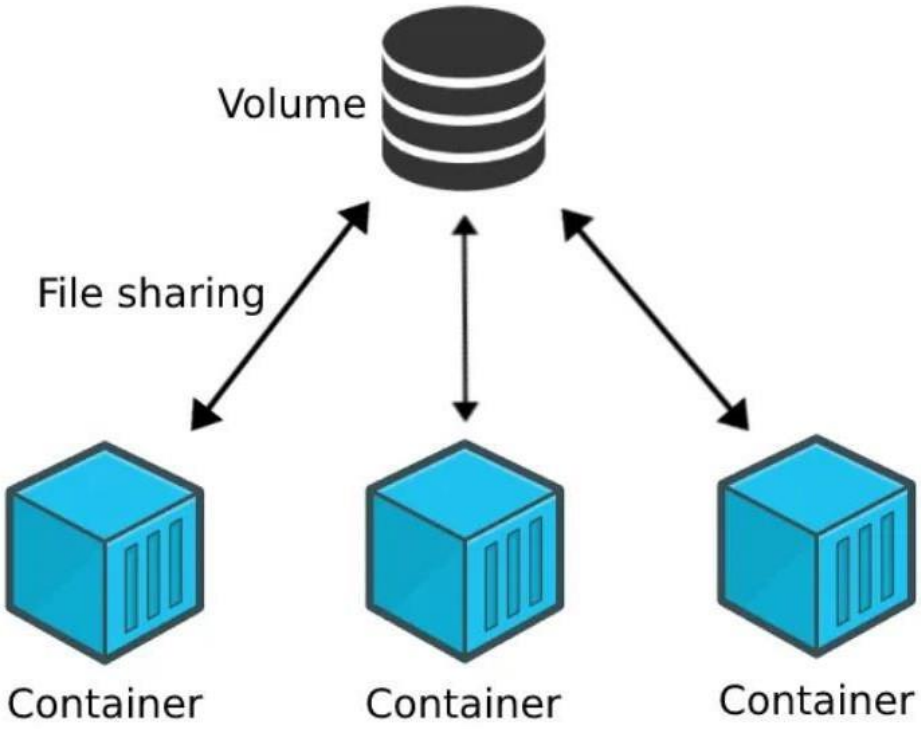
docker network prune (Remove all unused networks)

docker network ls

Volumes (Persisting data)

Volumes are persistent data stores for containers, created and managed by Docker

Mountpoint Path: /var/lib/containers/storage/volumes/vol1/_data



Host to Container Volume Sharing

mkdir /opt/files

touch /opt/files/test{1..3}.txt

docker run -it --name hostcon -v /opt/files:/incoming --privileged=true ubuntu /bin/bash

root@7b0673c18b27:/# ls

root@7b0673c18b27:/# cd incoming/

root@7b0673c18b27:/incoming# ls

test1.txt test2.txt test3.txt (to show files)

root@7b0673c18b27:/incoming# touch test{4..6}.txt (to create files)

root@7b0673c18b27:/incoming# ls

root@7b0673c18b27:/incoming# exit

cd /opt/files

ls (to show files)

Create Volume in container and how to share volume another container

docker run -it --name con1 -v /volume1 ubuntu /bin/bash (create container with folder)

root@cb12886d880f:/# ls (list)

root@cb12886d880f:/# cd volume1/ (go to folder)

root@cb12886d880f:/volume1# touch test{1..5}.html (to create files)

[Open New Terminal or Tab]

docker run -it --name con2 --privileged=true --volumes-from con1 ubuntu /bin/bash (create other container name is con2)

root@45647df0f955:/# cd volume1

root@45647df0f955:/volume1# ls

test1.html test2.html test3.html test4.html test5.html

root@45647df0f955:/volume1# touch update{1..5}.txt

root@45647df0f955:/volume1# ls

[Goto 1st Terminal or Tab]

[Testing]

root@cb12886d880f:/# cd volume1

root@cb12886d880f:/volume1# ls

test1.html test2.html test3.html test4.html test5.html update1.txt update2.txt update3.txt update4.txt update5.txt

root@cb12886d880f:/volume1# exit

DONE root@cb12886d880f:/volume1# ls

test1.html test2.html test3.html test4.html test5.html update1.txt update2.txt update3.txt update4.txt update5.txt

root@cb12886d880f:/volume1# exit

DONE

Create Custom images

docker run -it --name mycontainer ubuntu /bin/bash

touch /tmp/testfile

exit

docker diff mycontainer (to check different)

docker commit mycontainer updateimage (to create image)

docker image ls

Now create container from this image

docker run -it --name newcontainer updateimage /bin/bash

ls

cd tmp

ls (to show testfile)

exit

How to push image on your docker Account

docker login docker.io Username:

Password:

docker push f7d8bafbd9a9 docker.io/redhatwala/httpd-test

Go-to docker hub site, login your account and check

EXPORTING AND IMPORTING CONTAINERS (backup and restore)

you can use the **docker export** command to export a current snapshot of your running container into a tarball on your local machine.

You can use the **docker import** command to import a tarball (revert back to it later) and save it as a filesystem image.

Prerequisites: The container-tools meta-package is installed.

docker run -dt --name myubi rhel9 (to create container)

docker ps -a (to show container)

docker attach myubi (to attach)

[root@1b03f2f8540f /]# echo "hello" > testfile (to create testfile)

Detach from the container with **CTRL+p** and **CTRL+q** (exit)

[Export the file system of the **myubi** as a **myubi-container.tar** on the local machine:]

docker export -o myubi-container.tar 1b03f2f8540f (export the filesytem)

ls -l [List]

docker import myubi-container.tar myubi-imported (to import)

docker image ls

docker run -it --name myubi-imported myubi-imported /bin/bash

ls

cat testfile (read file)

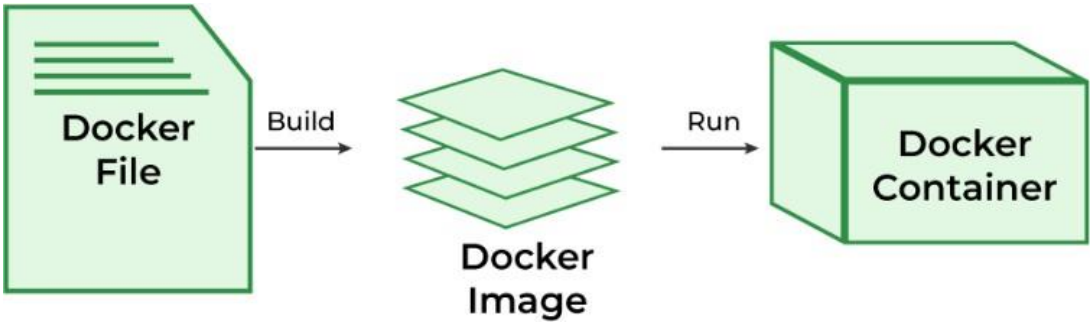
CTRL+p and **CTRL+q** (exit)

DONE

Container File

A Container file is a text file that contains instructions for creating a container image.

Automation of Docker image Creation



Container File Options:

FROM: for base image. This command must be on top of the Containerfile

RUN: to execute commands, it will create a layer in image

MAINTAINER: Author/Owner/Description

COPY: Copy files from local machine we need to provide source, destination. (Only local system)

ADD: Similar to COPY but, is provides a feature to download files from internet, also we extract tarfile at docker image side.

EXPOSE: to expose ports such as port 8080 for Apache.

WORKDIR: to set working directory for a container

CMD: execute commands but during container creation

ENTRYPOINT: similar to CMD, but has higher priority over CMD, 1st commands will be executed by ENTRYPOINT only

ENV: Environment Variables

This example uses the http base image, copies a custom index.html file, and exposes port 80

vim Containerfile

FROM httpd:2.4 [Use an official base image**]**

MAINTAINER Kishor Ahire [Set an Author for the container**]**

COPY ./index.html /usr/local/apache2/htdocs/ [Copy the website content into the container**]**

EXPOSE 80 [Expose port 80**]**

CMD ["httpd-foreground"] [Start the Apache server**]**

:wq

vim index.html

Welcome to Docker Class

:wq

docker build -t httpd . [to create image]

docker run -d --name webserver -p 80:80 httpd [to create container]

curl http://192.168.0.165

**********Thank You!**********