What are Containers?

What is Docker?

Docker utilized LXC containers

Docker has “Community Edition” and “Enterprise Edition”

Why do we need Dockers?

What Dockers can do?

Run Docker Containers

Create a Docker Image

Networks in Docker

Docker Compose

Docker Concepts in Depth

Docker for windows / Mac

Docker Swarm

Docker Vs Kubernetes

Docker Commands

1) ***docker run***  (start a container from the image)

Ex: docker run nginx (runs nginx container, nginx image is first checked on the local machine, if not available then pulls from dockerhub.

This is done only once, subsequent runs will use the same image (cached locally))

2) **docker ps**  (lists all the running containers)

3) docker stop <ContainerID> (to stop the running containers)

4) docker rm <containerID> (to remove the container)

5) docker images (to see the list of images)

6) docker rmi <imagename> (removes image)

7) docker pull <imagename> (just to pull the image and keep it locally)

8) docker run Ubuntu (Ubuntu container will run and exit immediately as there is nothing (no service) to be run on OS)

9) docker run Ubuntu sleep 5 (to keep the ubuntu running for 5 secs, otherwise it will exit soon after it is run, if we check with “docker ps” we can see the container running for 5 secs)

10) docker run –d <SimpleWebAppln> (‘-d’ option is detached mode, to run the container in the background)

Note: To bring the container to the ForeGround, we can always use “attach”  
Ex: docker attach <containerID> ---- this will bring the container to the foreground

11) docker run –it centos bash (runs centos, enters into the os, on the bash prompt, once exited out the os, the container is stopped)

12) docker ps –a (all the containers (exited one and running one) will be listed)

13) docker exec <container id> cat /etc/\*release\* (To run the “/etc/\*release\* cmd INSIDE the container)

14) docker run redis:4.0 (running with specific version (TAG))

15) docker run –i <CONTAINER> (Runs in interactive mode, so that we can provide any inputs (if required) from the host machine to the container)

16) docker run –it <CONTAINER> (i – interactive, t – Terminal)

Note: Every docker will get an IPaddress of its own, which can be used internally

PORT Mapping (To access the containers from the outside world)

17) docker run –p 80:5000 <WebAppln> (80 is the user/view port where in user uses this port to access the website, 5000 is the internal docker port)

Note: we can run as many applications as possible with multiple ports ex:8000:5000, 8001:5000, 8002:5000 etc

Volume Mapping (To persist the data, when we delete the container the data inside the container will be lost, so to persist it, we will have to map it to the external disk)

18) docker run –v /opt/datadir:/var/lib/mysql mysql (“/opt/datadir” is the db in the external disk, “/var/lib/mysql” is the db in the container)

19) docker inspect <containerID> (to get the details of the container in JSON format, including the IPADDRESS of the container etc)

20) docker logs <containerID> (to see the logs of the container, shown on to the standard output)

21) docker run –p 8080:8080 jenkins

22) docker run –p 8080:8080 –v /root/myJenkinsData:/var/Jenkins\_home –u root jenkins (port mapping + volume mapping + authentications)

The above will store all the Jenkins related data Locally(Externally) into “/root/myJenkinsData” , so that even if the Jenkins container is deleted, the data still persists

And we can run another instance of Jenkins container by mapping the volume mapping to the same path (“/root/myJenkinsData”) , then we can get back the same data

23)

**DOCKER IMAGES**

Why Images/Container are required?

If images are not in dockerhub already, or in case of dockerizing the application for the ease of shipping, THEN WE CREATE IMAGES (WE NEED CONTAINERS)

How Images are Created?

Create a docker file (list of commands to be executed, )

Run the cmd “docker build Dockerfile –t <USERNAME / MYAPP>” (t --- for tag)

Run the cmd “docker push <USERNAME>/<MYAPP>” --------------------------- (this pushes into docker registry)

Note: we have to have the account created with the above USERNAME in the <https://hub.docker.com> ,

And then we have to login to the account by following cmd

Run “docker login” (provide “username” and “pwd”)

Note: All Dockerfiles must start with “FROM <BaseImage>” instruction

Setting the “WORKDIR” in the Dockerfile

“COPY” the source code from <source> to <Dest>

“RUN” for running the command

“ENTRYPOINT”—allows us to specify a command, when an image will be run as a container

Note: docker uses cache to store all the steps in the Dockerfile execution, so that whenever there is a failure at a step, next time we don’t have to re run all the steps