**Basic Assignment**

NOTE: Do not forget to see manual page using "--help" option in command when searching for options/commands for a particular task.

1. Install Docker, either on your native OS or on a VM. Make sure it runs. type "docker -v" to check if it's installed.

try below commands for help

docker --help ---> This command shows all available options and commands to work with images and containers

docker images --help ---> This command shows all the avaialble options and commands to work with docker images

docker ps --help ---> This command shows all the avaialble options and commands to work with docker containers

2. Find a image from dockerhub of your choice(recommeded: nginx), don't use browser, pull the official image from dockerhub

Ans: docker pull ubuntu

3. List all the available images in your machine/vm, make sure you see recently pulled image in the list.

Ans: docker images

4. Find out the "Full" ImageId of the image that you pulled and write it below.

Ans: docker images -q

5. Create a container of your image

Ans: docker run –it Ubuntu bash

6. List all the running containers

Ans: docker ps

7. List all the running and stopped containers

Ans: docker ps -a

8. Find out the "Full" containerId of the container and write it below.

Ans: docker ps -q

9. Find out how many image layers are used to build this image.

Ans: docker history id

10. Get the Apache Tomcat 7 server image from the docker hub.

Ans: docker pull tomcat

11. Run the Apache Tomcat 7, I mean create a container of Apache Tomcat.

Ans : docker run –it tomcat

12. Find out what is the IP Address of the Apache Tomcat Container that it is running on

Ans: docker inspect <container id>

13. Which Port it is using?

Ans: 8080

14. Try to access the Tomcat's home page from your machine/vm.

Ans: docker run –it - -rm –p 8888:8080 tomcat:8.0

15. What is the disk size of Apache Tomcat image?

Ans: docker system df -v

16. Find out list of all environment variables that is configured for tomcat image, can you see JAVA\_HOME and CATALINA\_HOME? What did you notice about it?

Ans: • C:\Users\Lenovo>docker exec 6d24ba63ca57 env

• PATH=/usr/local/tomcat/bin:/usr/local/openjdk-11/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

• HOSTNAME=6d24ba63ca57

• JAVA\_HOME=/usr/local/openjdk-11

• LANG=C.UTF-8

• JAVA\_VERSION=11.0.12

• CATALINA\_HOME=/usr/local/tomcat

• TOMCAT\_NATIVE\_LIBDIR=/usr/local/tomcat/native-jni-lib

• LD\_LIBRARY\_PATH=/usr/local/tomcat/native-jni-lib

• GPG\_KEYS=48F8E69F6390C9F25CFEDCD268248959359E722B A9C5DF4D22E99998D9875A5110C01C5A2F6059E7 DCFD35E0BF8CA7344752DE8B6FB21E8933C60243

• TOMCAT\_MAJOR=9

• TOMCAT\_VERSION=9.0.52

• TOMCAT\_SHA512=35e007e8e30e12889da27f9c71a6f4997b9cb5023b703d99add5de9271828e7d8d4956bf34dd2f48c7c71b4f8480f318c9067a4cd2a6d76eaae466286db4897b

• HOME=/root

• C:\Users\Lenovo>docker exec 4fae31f8fce0 env

• PATH=/usr/local/tomcat/bin:/usr/local/openjdk-11/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

• HOSTNAME=4fae31f8fce0

• JAVA\_HOME=/usr/local/openjdk-11

• LANG=C.UTF-8

• JAVA\_VERSION=11.0.12

• CATALINA\_HOME=/usr/local/tomcat

• TOMCAT\_NATIVE\_LIBDIR=/usr/local/tomcat/native-jni-lib

• LD\_LIBRARY\_PATH=/usr/local/tomcat/native-jni-lib

• GPG\_KEYS=48F8E69F6390C9F25CFEDCD268248959359E722B A9C5DF4D22E99998D9875A5110C01C5A2F6059E7 DCFD35E0BF8CA7344752DE8B6FB21E8933C60243

• TOMCAT\_MAJOR=9

• TOMCAT\_VERSION=9.0.52

• TOMCAT\_SHA512=35e007e8e30e12889da27f9c71a6f4997b9cb5023b703d99add5de9271828e7d8d4956bf34dd2f48c7c71b4f8480f318c9067a4cd2a6d76eaae466286db4897b

• HOME=/root

17. Find out which port is exposed for tomcat?

Ans: 8888

18. Run multiple conntainers of tomcat on different port and access it's home page.

Ans: docker pull tomcat:jre8

docker run –it –rm –p 8082:8080 tomcat:jre8

docker pull tomcat:jre11

docker run –it –rm –p 8084:8080 tomcat:jre11

19. Pull ubuntu os from dockerhub, try to pull 2 images of ubuntu, Except the latest one.

Ans: docker pull ubuntu:xenial

docker pull ubuntu:rolling

20. Run the container of ubuntu in attached mode.

Ans: docker run ubuntu:xenial

21. Run the container of another ubuntu in detached mode.

Ans: docker run –d ubuntu:rolling

22. Check how many ubuntu containers are running and stopped

Ans: docker ps –a

23. Is the tomcat container running? If no, start one.

Ans: docker start tomcat

24. Check the logs, generated by tomcat container(don't forget to make request to tomcat's home page to see the log).

Ans: docker logs <container id>

25. Check if ubuntu conatiner is running? If no, start one in attached mode to the terminal.

Ans: docker start ubuntu:xenial

26. Login as root user in ubuntu container.

Ans: docker run id Ubuntu:xenial

27. Create a file with any name in root directory

Ans: touch textfile

28. Install software of your choice in ubuntu container using "apt-get install"

Ans: apt-get update

29. Now exit the ubuntu shell, are you back to your host machine, if not, come back to the host machine.

Ans: Exit

30. Check if the ubuntu container is running.

Ans: no

31. Create a new ubuntu container out of the same image as that previous container in attached mode.

Ans: docker run –it Ubuntu:xenial

32. Login as a root user

Ans: docker exec it <Ubuntu container id> bast

33. Check if you can see the file created in previous container, you will not see the file as well as software that you installed in the previous container. Now kill this Container.

Ans: docker kill <container id>

34. Do you have the previous ubuntu container where you created the file and installed the software? If no reapeat step 25 to 29.

Ans: docker start ubuntu:xenial

docker run id Ubuntu:xenial

35. Create an Image out of the existing container.

Ans: Docker container run -it ubuntu bash

Make a file inside the shell and check it using ls

Docker container ls

Docker container commit <containerid> <image name to be created>

Docker image ls

Image is created .

36. Now Create a Container out of this image and login into it to see if you can see the file and software installed by you in the previous container.

Ans: Docker container ls (delete the one existing )

Docker rm -f <containerid>

Docker container run -it <image name which created earlier>

Inside the shell : Ls ; cd tmp ; ls (we get the file created )

37. Do you have running tomcat container? If yes, Stop it and kill all tomcat container.

Ans: docker stop tomcat

docker kill <tomcat container id>

38. Create an index.html file with following code in it:-

<h1>This is Tomcat Container</h1>

Now, Start a ubuntu container, login into the ubuntu, install nginx, access nginx's home page from the container,

Now, come out of the ubuntu container and replace the existing index.html page of nginx with above index.html page that you created, Now access the nginx server from host machine, it should display the index.html that you have created.

Ans:

39. type below command:-

docker images --help

Now, try to run command that proves the concept of following three options:-

Ans:

1. –a : docker images -a , --all show all images

--digests show digests

2. –f : docker images –f , --filter filter output based on condition provided

--format string pretty print images using go templet

3. –q : docker images –q , quiet only show image Ids

write atleast 1 command using each option above and prove their concepts as described in the --help.

40. type below command:-

docker ps --help

Now, try to run command that proves the concept of following six options:-

Ans:

1. –a : docker images -a , --all show all images

--digests show digests

2. –f : docker images –f , --filter filter output based on condition provided

--format string pretty print images using go templet

3. –q : docker images –q , quiet only show image Ids

4. –n : docker ps –n ,--last int Show n last created containers (including all status)

5. –l : docker ps –l , --latest show the latest created container.

--no-trunc don`t truncate output

6. –s : docker ps -s, --size Display total file size

write atleast 1 command using each option above and prove their concepts as described in the --help.

41. Type below command:-

docker --help

you will various sections of commands apart from options like "Managemnet Commands" and "Commands".

Write some texts below describing the use of "Management Commands".

Use each command mentioned below and prove its concepts as described in the --help desription. write what you have understood from the output of the command after its successful execution.

1. cp : $docker cp <container nam:/filename . > (to copy the file in docker )

2. create : $docker run <name> (Create a new container)

3. export : $docker export –output=”<name of the tar file>” <image name | container id> (this command is used to Docker image into a tar file in your local system )

4. history : $docker history <id> (Show the history of an image)

5. info : $docker info (Display system-wide information)

6. login : $docker login (this command is used to login to docker hub repo from command line.)

7. logout :$docker logout ( Log out from a Docker registry )

8. rename: $docker rename <oldname> <newname> (rename the existing file into new name)

9. save : Save one or more images to a tar archive

10. stats :$docker stats <containername|id> ( give container id, name , cpu, input/output, memory location,)

11. top : $docker container top <containerid> Display the running processes of a container

42. Kill all running container in one liner command.

Ans : docker kill $(docker ps –q)

43. Delete all images in one liner command.

Ans: docker rmi –f $(docker images –a –q)

44. Create a simple Dockerfile, build it and run it.

Ans: Step1 : Create a file named Dockerfile

Cd Desktop

mkdir Dockerfiles

Cd DockerFiles

Dir > Dockerfile

Vim dockerfile

Step2 : add instruction in Dockerfile

FROM ubuntu

RUN apt-get update

CMD [“echo,” “hello from Docker file ”]

Step3 : Build dockerfile to create image

Docker build -t myimage1:1.0 .

Docker images

Step4: Run image to create container

Docker run <id>

Output : hello from Docker file

45. Create one or more Dockerfile that demostrate the following commands in Dockerfile (Write a PoC for each in one or more Dockerfile)

1. USER : The trick is to use useradd instead of its interactive wrapper adduser.

RUN useradd -ms /bin/bash newuser

2. RUN : RUN instruction allows **you to install your application and packages requited for it**. It executes any commands on top of the current image and creates a new layer by committing the results. Often you will find multiple RUN instructions in a Dockerfile.

3. ENV: ENV is mainly meant **to provide default values for your future environment variables**. Running dockerized applications can access environment variables.

4. CMD: The CMD command​ specifies the instruction that is to be executed when a Docker container starts. This CMD command is not really necessary for the container to work, as the echo command can be called in a RUN statement as well. The main purpose of the CMD command is **to launch the software required in a container.**

5. ENTRYPOINT : In Dockerfiles, an ENTRYPOINT instruction is **used to set executables that will always run when the container is initiated**. Unlike CMD commands, ENTRYPOINT commands cannot be ignored or overridden—even when the container runs with command line arguments stated.

6. EXPOSE : The EXPOSE instruction **informs Docker that the container listens on the specified network ports at runtime**. EXPOSE does not make the ports of the container accessible to the host.

8. VOLUME : The VOLUME command will specify **a mount point in the container**. This mount point will be mapped to a location on the host that is either specified when the container is created or if not specified chosen automatically from a directory created in /var/lib/docker/volumes.

46. Dockerhub:-

Find a application you care about on docker hub.

Ans: Ngnix

Launch the container.

Ans: docker pull Ubuntu

Docker run –d –it Ubuntu

Docker exec –it <container id> bash

Install another application in it.

Ans: Apt-get install ngnix

Save (commit) the image.

Ans: docker commit container id ,new name (khushbut/ngnix1)

Upload that to docker hub in your account and share it with a colleague, ask them to use your image and run the container out of it in their machine/vm.

Ans: docker push khushbut/ngnix1:latest

------------------------------------------------------------------------------------------------------------------------------------

**Advanced Assignment:-**

Assignment - Dockers and Containers

---Introduction

The goal of this assignment is to get you familiarized with dockers. You will learn about how to dockerize any application by building a simple client and server containers using dockers.

---What is to be done?

You will be building two containers using docker. These containers can be built starting from an official base image. Official Images has a list of different official base images. Start from the image that is suitable for you.

---Server :

Create a volume by name "servervol".

The server container will mount "servervol" in "/serverdata".

This container runs a server application which will create a file of size 1KB with random text data in "/serverdata" and then transfer the file to the client along with the checksum.

The server application itself can be built using any language you are comfortable with. But, the container should include all the packages that are required to run your application. Choose your base image wisely and install only the necessary packages.

The port on which the server runs must be specified as a command line argument when we run docker.

---Client:

Create a volume by name "clientvol".

The client container will mount "clientvol" in "/clientdata".

The client container runs an application that connects to the server, recieves the file that the server sends and saves it in "/clientdata".

Verify that the file is received properly at the clientside by verifying the checksum.

The client application again can be wriiten in any language that you are comfortable with, but the container should include all the necessary packages. Choose your base image wisely and install only necessary packages.

You need to create a user-defined network in docker and run both these containers on the network created. The containers should run these applications by default (i.e, on run command). Mention in README as to what should be done to get into the container shell instead of running the applications by default (i.e, on run command).

NOTE: You should be able to get into the shell of the client container to physically check if the file has been recieved.

---What to submit?

Create a new empty branch in your git repos named "assignment2".

You will be submitting the files that you use to build the server and client containers. Checkout the assignment2 branch and submit the files related to server and client in two folders named "server" and "client" respectively.

As you go about doing the assignment, you will learn that there are multiple ways to create volumes and to have the containers communicate to each other. So submit a README file that expains what method you use.

Create two scripts. one for the server("fileserver.sh") and the other for the client ("fileclient.sh") that has all the commands to build and run server and client containers respectively. These scripts should include the volume creation for client and server respectively. The server script should also include the creation of user-defined network before server runs.

The scripts and the README should be in the root of your branch (outside your server and client directories).

---What is tested?

Run the "fileserver.sh" script on one terminal. It should create the user-defined network, the "servervol", start the server and wait for the client connection.

Run "fileclient.sh" in another terminal. This script should find out the server's IP, create the "clientvol" and run the client container by specifying server's IP and port as command line parameters.

When the client container is run, a connection between the server and client is established and the file is transferred from the server to the client.

Your code and scripts will be tested on creation of volumes, proper running of server and client and file transfer.

You should make sure that the server can transfer any file with random text and the checksum verification succeeds on the client when it receives the file.

---Grade Distribution

Build and run server container - 30 points

Build and run client container - 30 points

Proper communication between the two - 15 points

Scripts to run the containers - 15 points

README file - 10 points

------------------------------------------------------------------------------------------------------------------------------------

**Advanced Assignment 2**

---Introduction

The goal of this assignment is to get you familiarized with CI-CD Pipeline for Creating docker images after successfull build.

You already built 2 app(client and server) from the "Part 1" Assignment.

NOTE: You should be able to get the docker image for both Client and Server after successfull build using Jenkins Pipeline.

Your Client app and Server app must be in the github.

Whenever developer commits changes to client/server app in github, it should trigger a build on Jenkins.

Jenkins Pipeline should do maven clean and package

Jenkins Pipeline should also do the static-code analysis using (checkstyle/sonarqube)

Jenkins pipeline at the end should create docker image

On build failure, Jenkins pipeline should send an email to any emailId about the failure.

---Grade Distribution

Proper Build trigger - 20 points

Proper Declarative Jenkins file - 20 points

Clean, Package and Code Analysis - 20 points

Creating docker image on successful build - 20 points

Sending email on build failure - 20 points