**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

ANS:

C:\Users\sneha>docker -v

Docker version 20.10.8, build 3967b7d

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

ANS:C:\Users\sneha>docker pull nginx

Using default tag: latest

latest: Pulling from library/nginx

Digest: sha256:4d4d96ac750af48c6a551d757c1cbfc071692309b491b70b2b8976e102dd3fef

Status: Image is up to date for nginx:latest

docker.io/library/nginx:latest

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

ANS:

C:\Users\sneha>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

myimage1 1.0 ed4b17460dd4 3 days ago 103MB

redis latest ddcca4b8a6f0 2 weeks ago 105MB

nginx latest dd34e67e3371 2 weeks ago 133MB

ubuntu 20.04 1318b700e415 5 weeks ago 72.8MB

ubuntu latest 1318b700e415 5 weeks ago 72.8MB

hello-world latest d1165f221234 6 months ago 13.3kB

jenkins 2.60.3 cd14cecfdb3a 3 years ago 696MB

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

ANS:

C:\Users\sneha>docker images -q

ed4b17460dd4

ddcca4b8a6f0

dd34e67e3371

1318b700e415

1318b700e415

d1165f221234

cd14cecfdb3a

5. Create a container of your image

ANS:

C:\Users\sneha>docker run nginx

/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration

/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/

/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh

10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf

10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf

/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh

/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh

/docker-entrypoint.sh: Configuration complete; ready for start up

2021/09/02 07:13:50 [notice] 1#1: using the "epoll" event method

2021/09/02 07:13:50 [notice] 1#1: nginx/1.21.1

2021/09/02 07:13:50 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)

2021/09/02 07:13:50 [notice] 1#1: OS: Linux 5.10.16.3-microsoft-standard-WSL2

2021/09/02 07:13:50 [notice] 1#1: getrlimit(RLIMIT\_NOFILE): 1048576:1048576

2021/09/02 07:13:50 [notice] 1#1: start worker processes

2021/09/02 07:13:50 [notice] 1#1: start worker process 32

2021/09/02 07:13:50 [notice] 1#1: start worker process 33

2021/09/02 07:13:50 [notice] 1#1: start worker process 34

2021/09/02 07:13:50 [notice] 1#1: start worker process 35

6. List all the running containers

ANS:

C:\Users\sneha>docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

974ae5994cff nginx "/docker-entrypoint.…" 5 minutes ago Up 5 minutes 80/tcp agitated\_tesla

7. List all the running and stopped containers

ANS:

C:\Users\sneha>docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

4144a2276772 redis "docker-entrypoint.s…" 45 seconds ago Up 41 seconds 6379/tcp epic\_ramanujan

547794c5dcdc ubuntu "bash" 4 minutes ago Up 4 minutes intelligent\_lederberg

4247a5b35df8 ubuntu "bash" 5 minutes ago Exited (0) 4 minutes ago relaxed\_saha

ccd0c4cb1481 hello-world "/hello" 6 minutes ago Exited (0) 6 minutes ago laughing\_keldysh

02b8c99b9e6b ubuntu "bash" 8 minutes ago Exited (0) 8 minutes ago ecstatic\_chatterjee

974ae5994cff nginx "/docker-entrypoint.…" 18 minutes ago Exited (0) 9 minutes ago agitated\_tesla

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

ANS:

C:\WINDOWS\system32>docker ps -aq

44de5a08969b

4144a2276772

547794c5dcdc

4247a5b35df8

ccd0c4cb1481

02b8c99b9e6b

974ae5994cff

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

ANS:

C:\Users\sneha>docker history ubuntu

IMAGE CREATED CREATED BY SIZE COMMENT

1318b700e415 5 weeks ago /bin/sh -c #(nop) CMD ["bash"] 0B

<missing> 5 weeks ago /bin/sh -c #(nop) ADD file:524e8d93ad65f08a0… 72.8MB

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

ANS: docker pull tomcat:7.0.109-jdk8-adoptopenjdk-openj9

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

ANS: docker run --name apachetom tomcat:7.0.109-jdk8-adoptopenjdk-openj9

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

ANS: docker inspect 44de5a08969b

docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $INSTANCE\_ID

13. Which Port it is using?

ANS: 8080

docker inspect --format='{{(index (index .NetworkSettings.Ports "8787/tcp") 0).HostPort}}' $INSTANCE\_ID

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

ANS:docker run -p 8888:8080 tomcat:7.0.109-jdk8-adoptopenjdk-openj9

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

ANS: docker ps -a -s

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?

17. Find out which port is exposed for tomcat?

8888

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

docker run -p 8888:8080 tomcat:7.0.109-jdk8-adoptopenjdk-openj9

docker run -p 8000:8080 tomcat:7.0.109-jdk8-adoptopenjdk-openj9

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

docker pull ubuntu:xenial-20210804

docker pull ubuntu:xenial

20. Run the container of ubuntu in attached mode.

docker attach container

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

docker run -d ubuntu:xenial-20210804

22. Check how many ubuntu containers are running and stopped

docker ps -a | grep container

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

yes

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

not able access tomcat's home page

docker logs containeid/name

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

ubuntu conatiner is not running

docker run -it -d ubuntu:xenial

docker exec -it containerid bash

26. Login as root user in ubuntu container

login

27. Create a file with any name in root directory

root@331292f770b3:/# ls

bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var

root@331292f770b3:/# cd root/

root@331292f770b3:~# ls

root@331292f770b3:~# touch random.txt

root@331292f770b3:~# ls

random.txt

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

apt-get install nginx

root@3dcd333691d1:~# apt-get install nginx

Reading package lists... Done

Building dependency tree

Reading state information... Done

E: Unable to locate package nginx

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

exit

30. Check if the ubuntu container is running.

No

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

C:\Users\sneha>docker run -it ubuntu:xenial

32. Login as a root user

login

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.

No file as well as software is not there

C:\Users\sneha>docker run -it ubuntu:xenial

root@8ebda01de127:/# cd root/

root@8ebda01de127:~# ls

root@8ebda01de127:~#

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

yes

35. Create an Image out of the existing container.

docker commit container id snehal11/newimage:version1

C:\Users\sneha>docker commit 8ebda01de127 snehal11/newimage:version1

sha256:08de251b5a5ab56ea900a87125bc6a98ec3f18cb6d1886b00e4e60c4e7aa863c

C:\Users\sneha>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

snehal11/newimage version1 08de251b5a5a 10 seconds ago 135MB

myimage latest b9a51d874af5 46 hours ago 103MB

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.

C:\Users\sneha>docker run -it snehal11/newimage:version1

root@1c0801975440:/#

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

yes

docker kill containerids

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.

39. type below command:-

docker images --help

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

1. -a Show all images (default hides intermediate images)

2. -f Filter output based on conditions provided

3. -q Only show image IDs

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

C:\Users\sneha>docker images -a

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu xenial b6f507652425 4 days ago 135MB

ubuntu xenial-20210804 b6f507652425 4 days ago 135MB

nginx latest dd34e67e3371 2 weeks ago 133MB

tomcat 7.0.109-jdk8-adoptopenjdk-openj9 e4e6da595824 2 months ago 382MB

jenkins 2.60.3 cd14cecfdb3a 3 years ago 696MB

C:\Users\sneha>docker images -f before=nginx

REPOSITORY TAG IMAGE ID CREATED SIZE

tomcat 7.0.109-jdk8-adoptopenjdk-openj9 e4e6da595824 2 months ago 382MB

jenkins 2.60.3 cd14cecfdb3a 3 years ago 696MB

C:\Users\sneha>docker images -q

b6f507652425

b6f507652425

dd34e67e3371

e4e6da595824

cd14cecfdb3a

40. type below command:-

docker ps --help

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

1. -a, --all Show all containers (default shows just running)
2. -f, --filter filter Filter output based on conditions provided

--format string Pretty-print containers using a Go template

1. -n, --last int Show n last created containers (includes all

states) (default -1)

1. -l, --latest Show the latest created container (includes all

states)

--no-trunc Don't truncate output

1. -q, --quiet Only display container IDs
2. -s, --size Display total file sizes

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

C:\Users\sneha>docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f346205545d2 ubuntu:xenial-20210804 "/bin/bash" 18 hours ago Exited (0) 18 hours ago wonderful\_meninsky

44de5a08969b tomcat:7.0.109-jdk8-adoptopenjdk-openj9 "catalina.sh run" 20 hours ago Exited (143) 19 hours ago apachetom

974ae5994cff nginx "/docker-entrypoint.…" 47 hours ago Exited (0) 47 hours ago agitated\_tesla

C:\Users\sneha>docker ps -as

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES SIZE

f346205545d2 ubuntu:xenial-20210804 "/bin/bash" 18 hours ago Exited (0) 18 hours ago wonderful\_meninsky 0B (virtual 135MB)

44de5a08969b tomcat:7.0.109-jdk8-adoptopenjdk-openj9 "catalina.sh run" 20 hours ago Exited (143) 19 hours ago apachetom 8.3kB (virtual 382MB)

974ae5994cff nginx "/docker-entrypoint.…" 47 hours ago Exited (0) 47 hours ago agitated\_tesla 1.09kB (virtual 133MB)

C:\Users\sneha>docker ps -aq

f346205545d2

44de5a08969b

974ae5994cff

41. Type below command:-

docker --help

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

Management Commands:

builder Manage builds

buildx\* Build with BuildKit (Docker Inc., v0.6.1-docker)

compose\* Docker Compose (Docker Inc., v2.0.0-rc.1)

config Manage Docker configs

container Manage containers

context Manage contexts

image Manage images

manifest Manage Docker image manifests and manifest lists

network Manage networks

node Manage Swarm nodes

plugin Manage plugins

scan\* Docker Scan (Docker Inc., v0.8.0)

secret Manage Docker secrets

service Manage services

stack Manage Docker stacks

swarm Manage Swarm

system Manage Docker

trust Manage trust on Docker images

volume Manage volumes

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

C:\Users\sneha>docker service ls

Error response from daemon: This node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this node to swarm and try again.

C:\Users\sneha>docker login

Authenticating with existing credentials...

Login Succeeded

C:\Users\sneha>docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

C:\Users\sneha>docker checkpoint ls

docker checkpoint ls is only supported on a Docker daemon with experimental features enabled

C:\Users\sneha>docker system df

TYPE TOTAL ACTIVE SIZE RECLAIMABLE

Images 4 3 1.346GB 695.7MB (51%)

Containers 3 0 9.393kB 9.393kB (100%)

Local Volumes 1 0 92B 92B (100%)

Build Cache 2 0 30.08MB 30.08MB

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 = Copy files/folders between a container and the local filesystem

2. create = Create a new container

3. export = Export a container's filesystem as a tar file

4. history = Show the history of an image , how many times image is used

5. info = Display system-wide information

6. login = Log in to a Docker registry

7. logout = Logout from a Docker registry

8. rename = renames a container

9. save = Save one or more images to a tar archive (streamed to STDOUT by default)

10. stats = Display a live stream of container(s) resource usage statistics

11. top = Display the running processes of a container

42. Kill all running container in one liner command.

docker container kill $(docker ps -q)

PS C:\Users\sneha> docker container kill $(docker ps -q)

44de5a08969b

974ae5994cff

43. Delete all images in one liner command.

PS C:\Users\sneha> docker images rm $(docker images -a -q)

b6f507652425

b6f507652425

dd34e67e3371

e4e6da595824

cd14cecfdb3a

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

touch Dockerfile

vi Dockerfile

docker build -t imgname placeofdockerfile(. for same dir)

docker images

docker run -p 80:80 imgname

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

1. USER : set username or uid

2. RUN : central executing directive, also can build image

3. ENV : set env variable

4. CMD :executing specific command

6. ENTRYPOINT : overwrite CMD command

7. EXPOSE :active at given port number

8. VOLUME: mounting it on given path

dockerfile

# getting base image ubuntu

FROM ubuntu

USER sneha

RUN apt-get update

ENV SERVER\_WORKS 4

CMD ["echo", "Hello World.....! My first Docker image"]

RUN apt-get install -y ngnix

ADD index.html /usr/share/nginx/html/index.html

ENTRYPOINT ["/usr/sbin/nginx", "-g", "daemon off;"]

EXPOSE 80

VOLUME ["/DockerFiles"]

46. Dockerhub:-

Find a application you care about on docker hub.

Launch the container.

Install another application in it.

Save (commit) the image.

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.

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

**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