**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

Docker pull nginx

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

docker images

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

docker images -q

5. Create a container of your image

docker run --name mynginx -it nginx

6. List all the running containers

docker ps

7. List all the running and stopped containers

[03/09, 13:30] Durge, Shabduli Shrikant

Q7 C:\Users\Shabduli>docker ps -a

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

docker ps -aqf "name=ubuntu" (container name = ubuntu)

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

docker history nginx

10. Get the A docker pull tomcat:7.0.56

pache Tomcat 7 server image from the docker hub.

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

docker create tomcat:7.0.56

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

docker container ls --format "table {{.ID}}\t{{.Names}}\t{{.Ports}}" -a

13. Which Port it is using?

8080

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

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

docker container ls -s

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?

Yes. We can see the specified.

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.

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

Docker pull Ubuntu:xenial

Docker pull Ubuntu:rolling

20. Run the container of ubuntu in attached mode.

Docker attach/start Ubuntu:xenial

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

Docker run –d Ubuntu:rolling

22. Check how many ubuntu containers are running and stopped

Docker ps -a

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

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).

Docker logs Mytomcat

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

Docker run –it Ubuntu:xenial

26. Login as root user in ubuntu container

Docker exec –it [container\_id] /bin/bash

27. Create a file with any name in root directory

Touch test

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

Sudo Apt-get install nginx

Sudo Apt-get install whatever

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.

Docker ps -a

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

Docker run -it Ubuntu:xenial

32. Login as a root user

Docker exec –it [container\_id] /bin/bash

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.

Docker ps –a

No such file and any previously installed software

Docker kill Ubuntu/docker rm –f [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.

done

35. Create an Image out of the existing container.

>docker create --name nginx\_base -p 80:80 nginx:alpine

docker images

docker ps –a

docker commit nginx\_base

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.

Docker ps

Docker images

Docker run –it –name Shabduli shabdulicoursera/Ubuntu:xenial

Ls

exit

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

docker rm -f e9a9462b28e5

docker rm -f 44deb6739064

docker rm -f ecf17224b904

\*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.

• docker run -d -p 8888:80 nginx

• mkdir -p docker-share/html

• ls

• cd html

• touch index.html

• vi index.html

• <h1>This is Tomcat Container</h1>

• docker stop d2c0127704200

• cd ..

• cd ..

• docker run -d -p 8888:80 -v ~/docker-share/html:/usr/share/nginx/html --hostname nginx\_home nginx

39. type below command:-

docker images --help

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

1. -a

2. -f

3. -q

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

1. docker images –a

Output of which is –

<none> <none> 00cd02f061e4 2 hours ago 22.8MB

tomcat latest 02c2632869e8 5 days ago 685MB

ubuntu xenial b6f507652425 6 days ago 135MB

ubuntu rolling e5ec58a9f549 6 days ago 80MB

nginx alpine 1318bf5f63b4 9 days ago 22.8MB

jenkins/jenkins latest 8360c5d104f2 12 days ago 441MB

ubuntu latest 1318b700e415 5 weeks ago 72.8MB

docker/getting-started latest 083d7564d904 2 months ago 28MB

hello-world latest d1165f221234 6 months ago 13.3kB

tomcat 7.0.56 efe1a0718e60 6 years ago 353MB

2. docker images –q

Output of which is-

00cd02f061e4

02c2632869e8

b6f507652425

e5ec58a9f549

1318bf5f63b4

8360c5d104f2

1318b700e415

083d7564d904

d1165f221234

efe1a0718e60

1. docker images -f "dangling=false"
2. REPOSITORY TAG IMAGE ID CREATED SIZE
3. tomcat latest 02c2632869e8 5 days ago 685MB
4. ubuntu xenial b6f507652425 6 days ago 135MB
5. ubuntu rolling e5ec58a9f549 6 days ago 80MB
6. nginx alpine 1318bf5f63b4 9 days ago 22.8MB
7. jenkins/jenkins latest 8360c5d104f2 12 days ago 441MB
8. ubuntu latest 1318b700e415 5 weeks ago 72.8MB
9. docker/getting-started latest 083d7564d904 2 months ago 28MB
10. hello-world latest d1165f221234 6 months ago 13.3kB
11. tomcat 7.0.56 efe1a0718e60 6 years ago 353MB

40. type below command:-

docker ps --help

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

1. -a

2. -f

3. -q

4. -n

5. -l

6. -s

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

docker ps –a

12f3b6e7d913 ubuntu:xenial "/bin/bash" About an hour ago Up About an hour elegant\_fermi

docker ps –l

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

12f3b6e7d913 ubuntu:xenial "/bin/bash" About an hour ago Up About an hour elegant\_fermi

>docker ps --filter status=running

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

12f3b6e7d913 ubuntu:xenial "/bin/bash" About an hour ago Up About an hour elegant\_fermi

2009080fe31f docker/getting-started "/docker-entrypoint.…" 2 hours ago Up 2 hours 0.0.0.0:80->80/tcp, :::80->80/tcp lucid\_volhard

56c514797290 ubuntu "bash" 8 days ago Up About an hour Myubuntu

docker ps --last 2

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

12f3b6e7d913 ubuntu:xenial "/bin/bash" About an hour ago Up About an hour elegant\_fermi

c8f324e94782 nginx:alpine "/docker-entrypoint.…" 2 hours ago Created nginx\_base

docker ps -q

12f3b6e7d913

2009080fe31f

56c514797290

docker ps -s

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES SIZE

12f3b6e7d913 ubuntu:xenial "/bin/bash" 2 hours ago Up 2 hours elegant\_fermi 87.5MB (virtual 222MB)

2009080fe31f docker/getting-started "/docker-entrypoint.…" 3 hours ago Up 3 hours 0.0.0.0:80->80/tcp, :::80->80/tcp lucid\_volhard 1.09kB (virtual 28MB)

56c514797290 ubuntu "bash" 8 days ago Up 2 hours Myubuntu 8B (virtual 72.8MB)

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".

Some example commands are-

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

service Manage services

stack Manage Docker stacks

swarm Manage Swarm

system Manage Docker

trust Manage trust on Docker images

volume Manage volumes

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 – Copies contents of any file/container to any docker host/container

2. create – creates containers from images

3. export- Exports contents

4. history - provides history of containers or images or any task performed by user

5. info – complete information about the docker application in the system

6. login – logs in to dockerhub account

7. logout – logs out of docker hub account

8. rename – renames containers from old to new

9. save – saves the contents of the container to images / saves to tar files

10. stats – all the tasks performed by docker are seen here. Complete usage activity and memory usage

11. top – display running process of container

42. Kill all running container in one liner command.

docker container kill $(docker ps -q)

43. Delete all images in one liner command.

Docker rmi –f $(docker images –q)

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

To build a file - docker build -t myimg1:1.0 .

To check - docker images

To run - docker run a2e6dcac329b

Hello World!!!!!!!! My first docker image

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

2. RUN

3. ENV

4. CMD

5. RUN

6. ENTRYPOINT

7. EXPOSE

8. VOLUME

-> 1. USER >> The USER instruction sets the user name (or UID) and optionally the user group (or GID) to use when running the image and for any RUN, CMD and ENTRYPOINT instructions that follow it in the Dockerfile.

2. RUN >> The RUN instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the Dockerfile.

3. ENV >> Environment variables (declared with the ENV statement) can also be used in certain instructions as variables to be interpreted by the Dockerfile.

4. CMD >> There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect.

6. ENTRYPOINT >> An ENTRYPOINT allows you to configure a container that will run as an executable.

7. EXPOSE >> The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime. You can specify whether the port listens on TCP or UDP, and the default is TCP if the protocol is not specified.

8. VOLUME >> The VOLUME instruction creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.

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.

docker run --name Myubuntuagain1 -it ubuntu:rolling

apt-get update

apt-get install redis

docker images

docker ps –a

docker commit 8a10d3de3e1a redisinubuntu

docker login

>docker login -u shabdulicoursera

docker tag redisinubuntu:latest shabdulicoursera/dockerhub:redisinubuntupush

>docker push shabdulicoursera/dockerhub:redisinubuntupush

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

**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