GRADED ASSIGNMENT ON DOCKER

Q1) Pull any image from the docker hub, create its container, and execute it showing the output.

Docker is a software platform to create, test and deploy applications in an isolated environment. Docker uses container to pack up an application with all of the parts it needs including, libraries and dependencies. It allows applications to use the kernel and other resources of the host operating system this will boost the performance and reduce the size of the application. Docker Hub is a centralized repository service that allows you to store container images and share them with your team. You can use Pull and Push command to upload and download images to and from the Docker Hub.

Docker version command

```
priyanka@LAPTOP-9OM8TU8U:~$ docker version
Client: Docker Engine - Community
 Cloud integration: v1.0.29
 Version:           20.10.22
 API version:       1.41
 Go version:        go1.18.9
 Git commit:        3a2c30b
 Built:             Thu Dec 15 22:28:22 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Desktop
 Engine:
  Version:          20.10.22
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.18.9
  Git commit:       42c8b31
  Built:            Thu Dec 15 22:26:14 2022
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.14
  GitCommit:        9ba4b250366a5ddde94bb7c9d1def331423aa323
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Step1:

We can pull the image from the Docker hub using the docker pull image name. Let us download the image called nginx from the Docker hub. Once the nginx image is downloaded, we get the following output.

```
priyanka@LAPTOP-9OM8TU8U:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
bb263680fed1: Pull complete
258f176fd226: Pull complete
a0bc35e70773: Pull complete
077b9569ff86: Pull complete
3082a16f3b61: Pull complete
7e9b29976cce: Pull complete
Digest: sha256:6650513efd1d27c1f8a5351cbd33edf85cc7e0d9d0fcb4ffb23d8fa89b601ba8
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

Step2:

Now create a new nginx container from the downloaded image and expose it on port 80 using the following command.

docker run --name docker-nginx -p 80:80 -d nginx

```
priyanka@LAPTOP-9OM8TU8U:~$ docker run --name docker-nginx -p 80:80 -d nginx
6f2d22fbe3e49b6f3800c748d5013c7667dd3ac3008c8652caa9f5ee835072b7
```

Step3:

We can also verify the nginx container with the below command. docker ps we will get the following output.

It will show the container-id.

```
priyanka@LAPTOP-9OM8TU8U:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                NAMES
6f2d22fbe3e4   nginx     "/docker-entrypoint.…"   15 seconds ago   Up 14 seconds   0.0.0.0:80->80/tcp   docker-nginx
```

Step4:

We can connect to the running container with the following command.

```
priyanka@LAPTOP-9OM8TU8U:~$ docker exec -it docker-nginx /bin/bash
root@6f2d22fbe3e4:/# apt update
```

Now we are connected to the running container.

```
priyanka@LAPTOP-9OM8TU8U:~$ docker exec -it docker-nginx /bin/bash
root@6f2d22fbe3e4:/# apt update
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8183 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [226 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8632 kB in 8s (1095 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
root@6f2d22fbe3e4:/#
```

Q2) Create the basic java application, generate its image with necessary files, and execute it with docker. Creating the basic java application.

Step 1:

Create a directory, it is used to store the files.

Step 2:

Go to the directory that you have created.

```
C:\Users\User>mkdir hero

C:\Users\User>cd hero
```

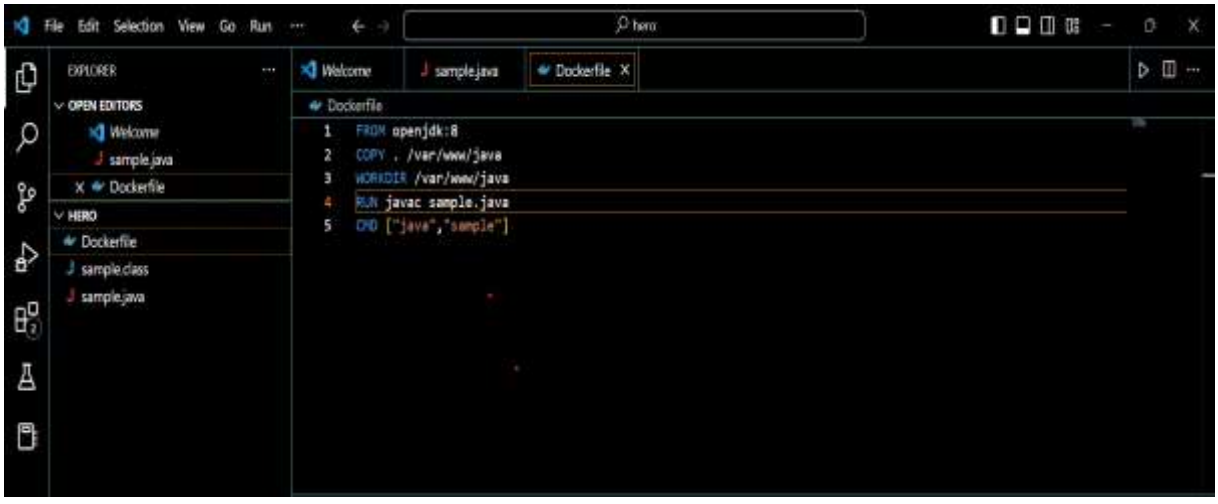
```
C:\Users\User\hero>code
```

Step 3:

Create a java file, save it as sample.java

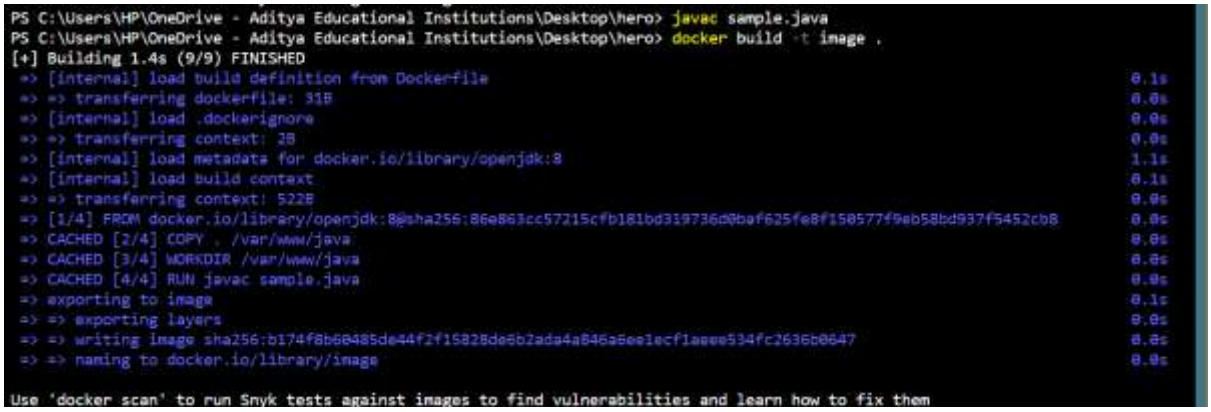
Step 4:

Create a docker file.



Step 5:

Now create an image by following below command. We must login as root in order to create a image. In the following command ,java-app is name of the image. We can have any name for our docker image.



Step 6:

After successfully building the image, now we can run the image by using docker run command.