

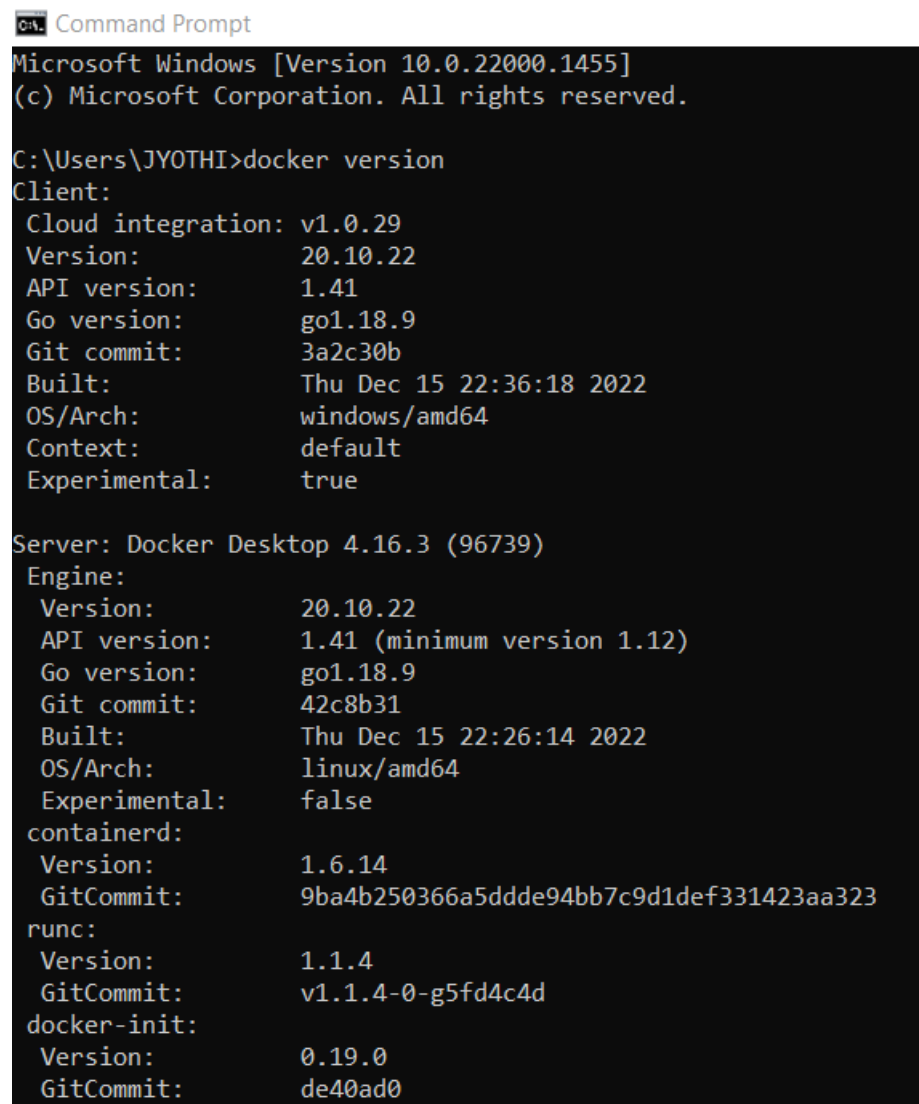
Graded Assignment on Docker:

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 a container to package 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 commands to upload and download images to and from the Docker Hub.

Docker version

Step-1: Verify the Docker version and also log in to Docker Hub. `docker version` `docker login`



```
Command Prompt
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JY0THI>docker version
Client:
 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 22:36:18 2022
 OS/Arch:          windows/amd64
 Context:          default
 Experimental:     true

Server: Docker Desktop 4.16.3 (96739)
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
```

Step-2: Pull the Image from Docker Hub. ...

```
C:\Users\JY0THI>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
```

Step-3: Next, create a new nginx container from the downloaded image and expose it on port 80 using the following command.

```
C:\Users\JY0THI>docker run --name docker-nginx -p 80:80 -d nginx
9c7037bb3a749b0606979c2e9494ec7bef2559caebfe4d2936563c60c1282072

C:\Users\JY0THI>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                    NAMES
9c7037bb3a74   nginx    "/docker-entrypoint. ..."  9 seconds ago Up 8 seconds   0.0.0.0:80->80/tcp      docker-nginx
```

Step-5: Connect to Container Terminal...






```
C:\Users\JY0THI>docker exec -it docker-nginx /bin/bash
root@9c7037bb3a74:/# 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 16s (530 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@9c7037bb3a74:/#
```

Docker Desktop

Containers [Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

☒ Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	Started	Actions
<input type="checkbox"/>	 docker-nginx 9c7037bb3a74 	nginx	Running	80:80	8 minutes ago	  

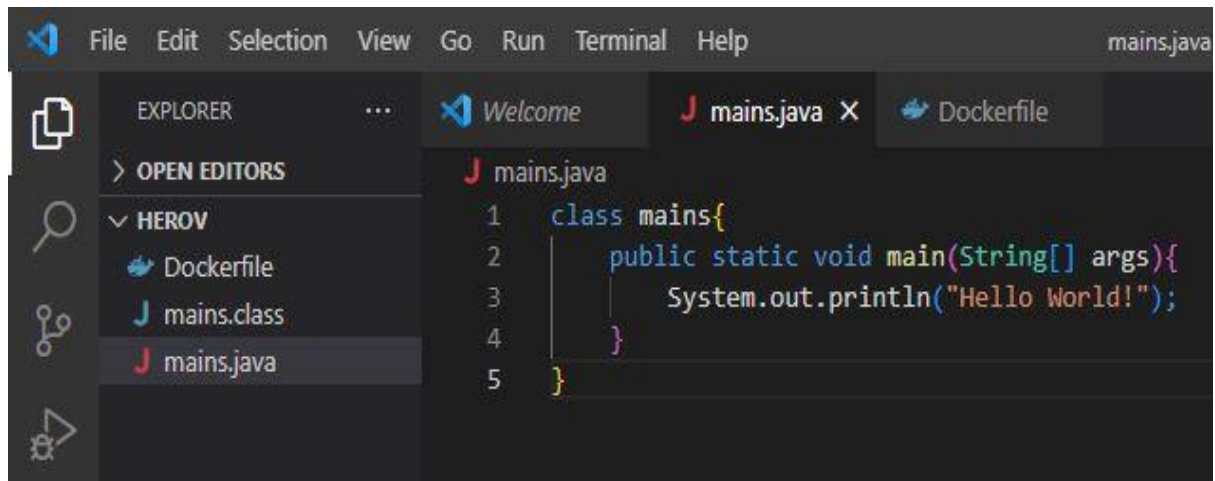
Q2) Create the basic java application, generate its image with necessary files, and execute it with docker. Creating the basic java application.

Step1: Create a folder and two files.

1.mains.java

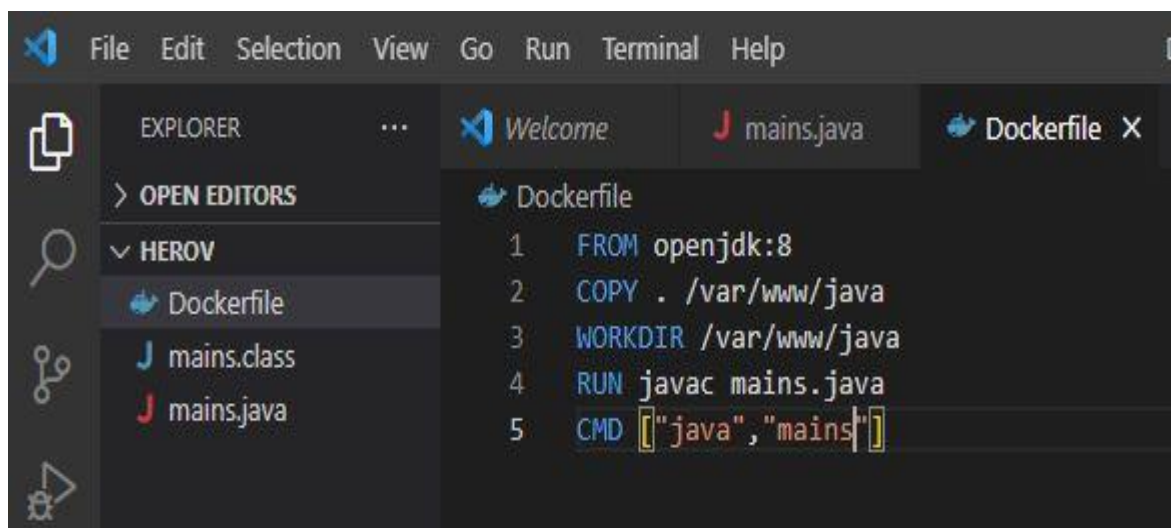
2.Docker

Step2: Create a java file, and save it as a mains.java

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'HEROV' containing 'Dockerfile', 'mains.class', and 'mains.java'. The main editor area has two tabs: 'Welcome' and 'mains.java'. The 'mains.java' tab is active, displaying the following Java code:

```
1 class mains{
2     public static void main(String[] args){
3         System.out.println("Hello World!");
4     }
5 }
```

Step3:Create a Docker file

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows the 'HEROV' project with 'Dockerfile', 'mains.class', and 'mains.java'. The main editor area has two tabs: 'Welcome' and 'Dockerfile'. The 'Dockerfile' tab is active, displaying the following Dockerfile content:

```
1 FROM openjdk:8
2 COPY . /var/www/java
3 WORKDIR /var/www/java
4 RUN javac mains.java
5 CMD ["java","mains"]
```

Step5:Now create an image by following the below command. we must log in as root in order to create an image. In the following command,java-app is the name of the image. We can have any name for our docker image.

```
PS C:\Users\DELL\Desktop\javad> javac mains.java
```

```

PS C:\Users\DELL\Desktop\herov> javac mains.java
PS C:\Users\DELL\Desktop\herov> docker build -t javaimg1 .
[+] Building 8.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 140B 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/openjdk:8 2.7s
=> [auth] library/openjdk:pull token for registry-1.docker.io 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 749B 0.0s
=> CACHED [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 0.0s
=> [2/4] COPY . /var/www/java 0.2s
=> [3/4] WORKDIR /var/www/java 0.2s
=> [4/4] RUN javac mains.java 3.9s
=> exporting to image 0.5s
=> => exporting layers 0.4s
=> => writing image sha256:56e935a3318f640f56499887dab2ca79121249cdf18808092cb5ec7d675e34eb 0.0s
=> => naming to docker.io/library/javaimg1 0.0s

```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

```
PS C:\Users\DELL\Desktop\herov> docker run javaimg1
```

```
Hello World!
```