Assignment

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

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker version

1. Verify the Docker version and also log in to Docker Hub.
2. Pull the Image from Docker Hub.
3. Next, create a new nginx container from the downloaded image and expose it on port 80 using the following command.
4. Connect to Container Terminal.

```
C:\Users\vamsi>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
```

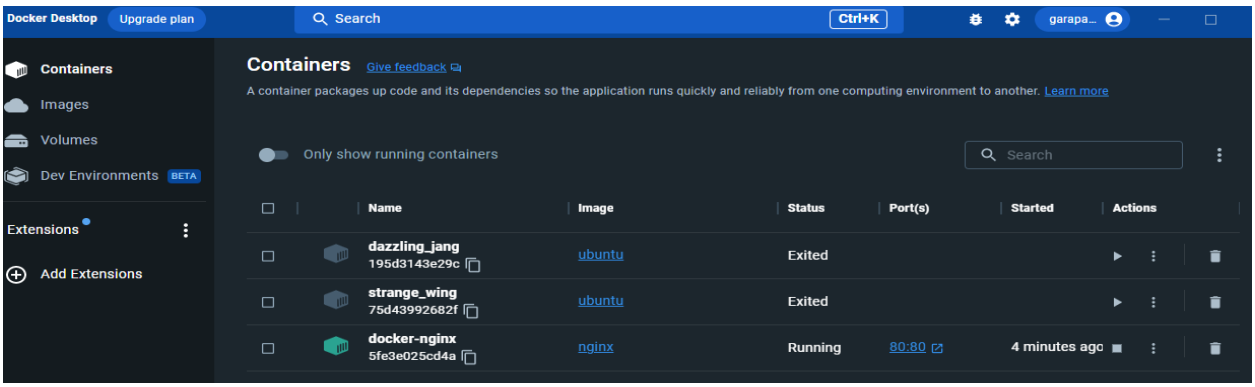
```
C:\Users\vamsi>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
bb263680fed1: Pull complete
258f176fd226: Pull complete
a0bc35e70773: Pull complete
077b9569ff86: Pull complete
3082a16f3b61: Pull complete
7e9b29976cce: Pull complete
Digest: sha256:6650513efd1d27c1f8a5351cbd33edf85cc7e0d9d0fcb4ffb23d8fa89b601ba8
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

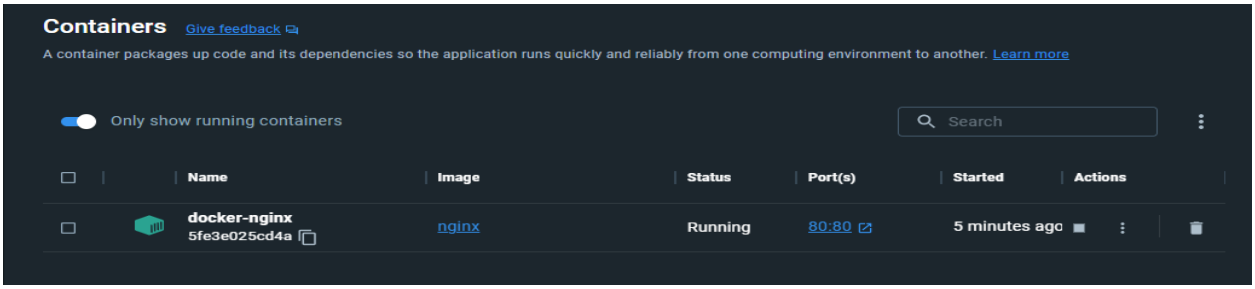
```
C:\Users\vamsi>docker run --name docker-nginx -p 80:80 -d nginx
5fe3e025cd4ad7b9d1b7a60ef653cd743316d2f62623cd22ad59643569a028a7
```

```
C:\Users\vamsi>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                NAMES
5fe3e025cd4a   nginx     "/docker-entrypoint.…"   15 seconds ago   Up 14 seconds   0.0.0.0:80->80/tcp   docker-nginx
```

```
C:\Users\vamsi>docker exec -it docker-nginx /bin/bash
root@5fe3e025cd4a:/# 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 1min 14s (117 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@5fe3e025cd4a:/#
```

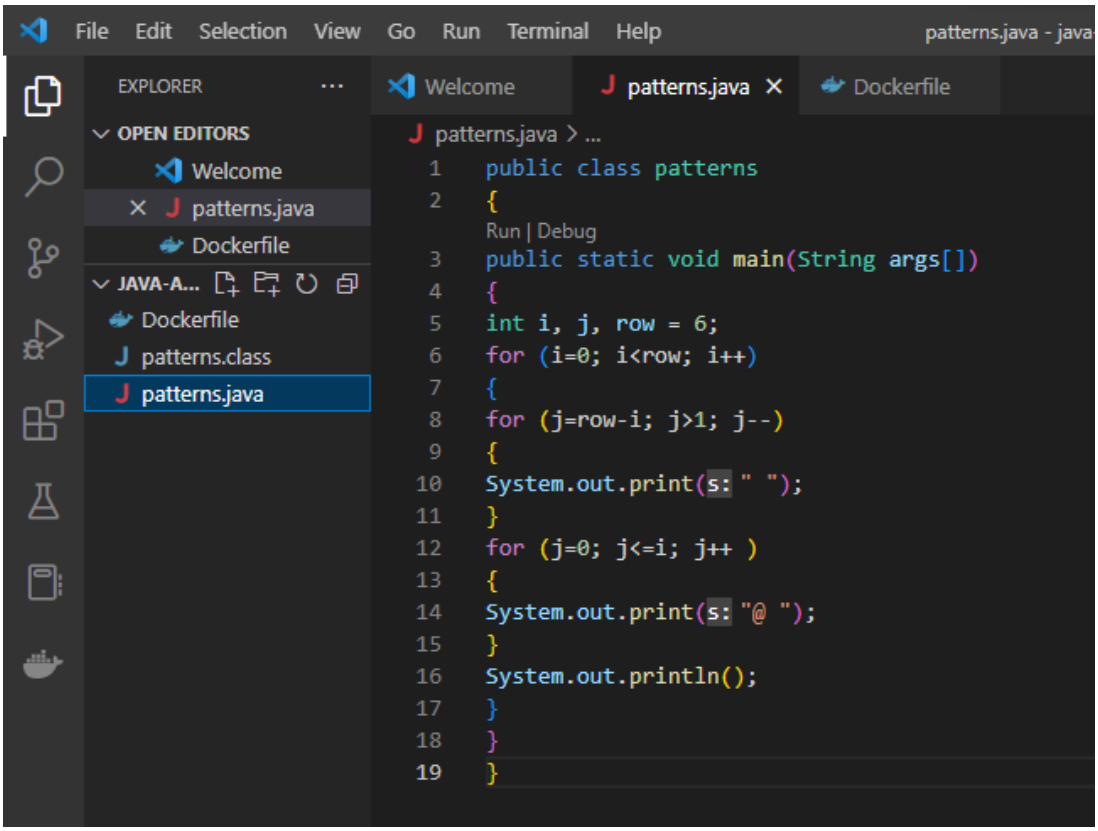
Docker Desktop:

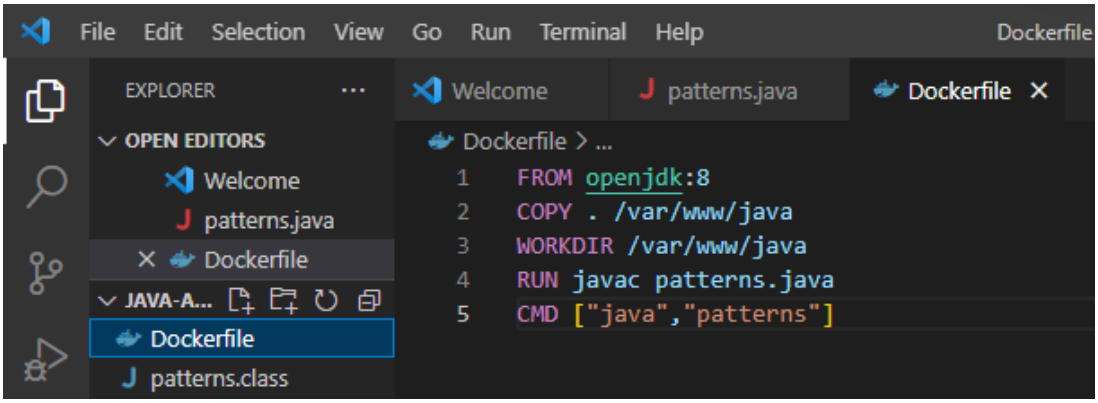


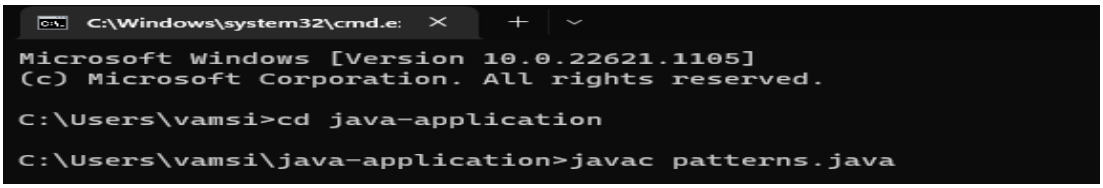


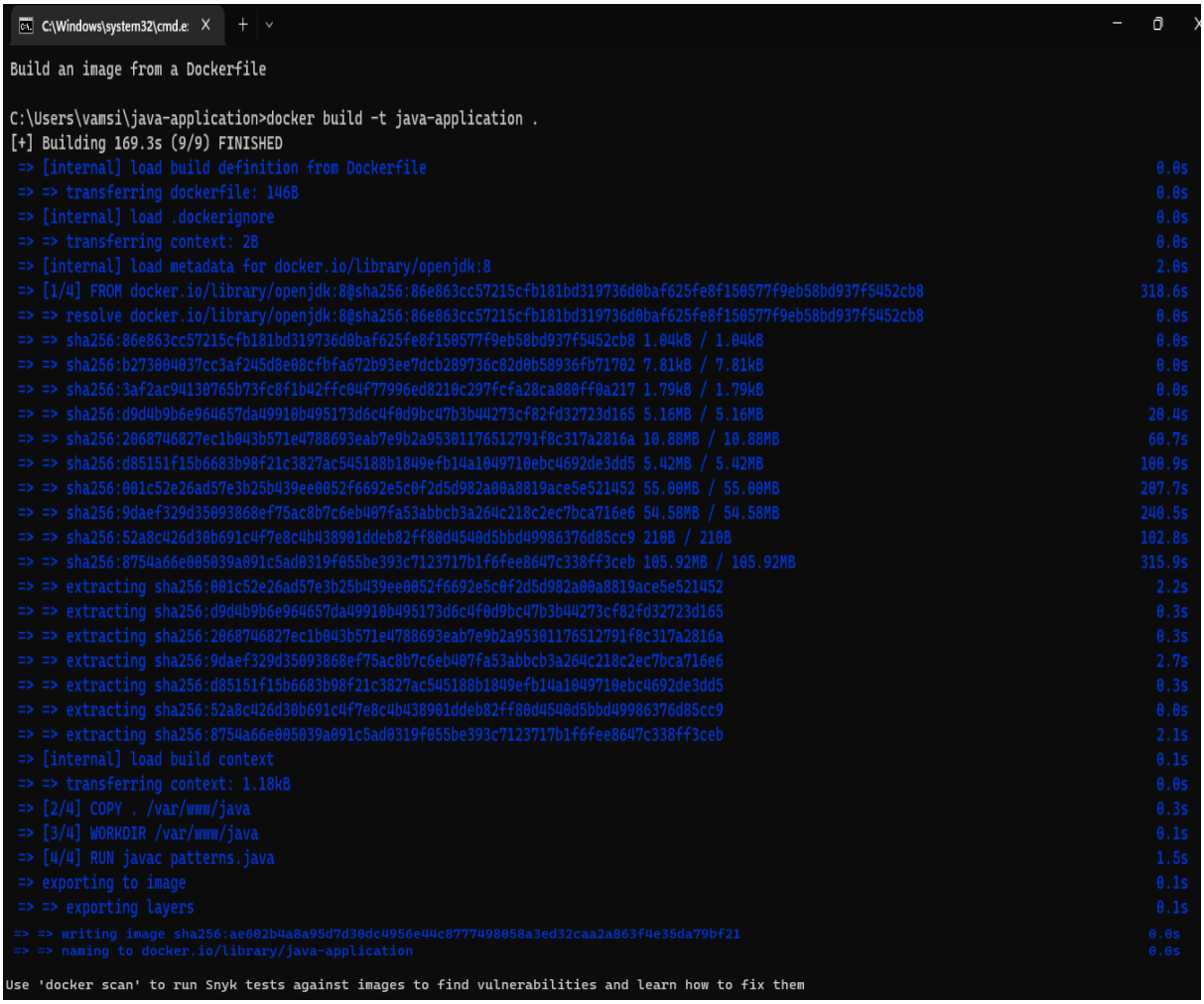
Q2) Create the basic java application, generate its image with necessary files, and execute it with docker. Creating the basic java application.

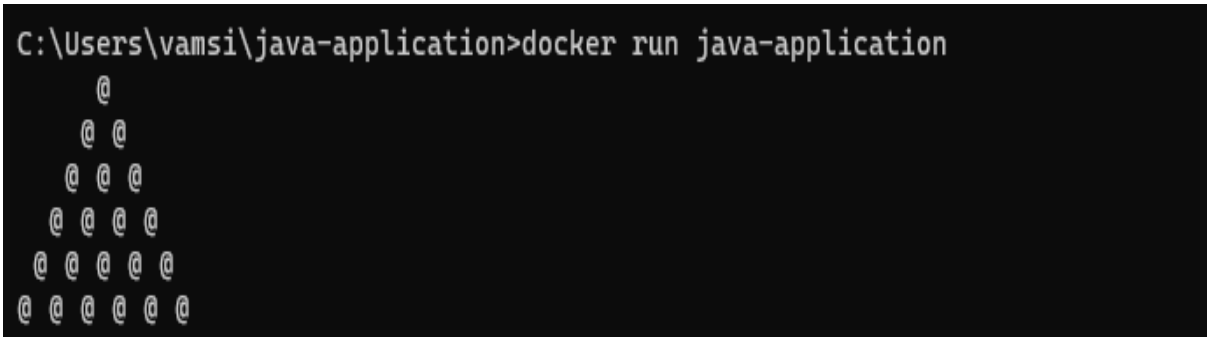
1. Create a folder and two files.
2. Create a java file, and save it as a mains.java
3. Create a Docker file.
4. Now create an image by following the below command. we must log in as root in order to create an image. In the following command,java-app is the name of the image. We can have any name for our docker image.








C:\Windows\system32\cmd.e
Build an image from a Dockerfile
C:\Users\vamsi\java-application>docker build -t java-application .
[+] Building 169.3s (9/9) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 146B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/openjdk:8 2.0s
=> [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 318.6s
=> => resolve docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 0.0s
=> => sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 1.04kB / 1.04kB 0.0s
=> => sha256:b273004037cc3af245d8e08cfbfa672b93ee7dcb289736c82d0b58936fb71702 7.81kB / 7.81kB 0.0s
=> => sha256:3af2ac94130765b73fc8f1b42ffc04f77996ed8210c297fcfa28ca880ff0a217 1.79kB / 1.79kB 0.0s
=> => sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165 5.16MB / 5.16MB 20.4s
=> => sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a 10.88MB / 10.88MB 60.7s
=> => sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5 5.42MB / 5.42MB 100.9s
=> => sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452 55.00MB / 55.00MB 207.7s
=> => sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6 54.58MB / 54.58MB 240.5s
=> => sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff80d4540d5bbd49986376d85cc9 210B / 210B 102.8s
=> => sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb 105.92MB / 105.92MB 315.9s
=> => extracting sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452 2.2s
=> => extracting sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165 0.3s
=> => extracting sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a 0.3s
=> => extracting sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6 2.7s
=> => extracting sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5 0.3s
=> => extracting sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff80d4540d5bbd49986376d85cc9 0.0s
=> => extracting sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb 2.1s
=> [internal] load build context 0.1s
=> => transferring context: 1.18kB 0.0s
=> [2/4] COPY . /var/www/java 0.3s
=> [3/4] WORKDIR /var/www/java 0.1s
=> [4/4] RUN javac patterns.java 1.5s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:ae602b4a8a95d7d30dc4956e44c8777498058a3ed32caa2a863f4e35da79bf21 0.0s
=> => naming to docker.io/library/java-application 0.0s
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them


C:\Users\vamsi\java-application>docker run java-application
@
@ @
@ @ @
@ @ @ @
@ @ @ @ @
@ @ @ @ @ @