Assignment on Docker

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

Image:

An image is a file used to execute code in a docker container . It is like a template that consists of set of instructions to create a container that can be run on docker.

We can pull the images by using the pull command and run it with a single command.

We can also pull the image, create a container and run it separately.

Example:

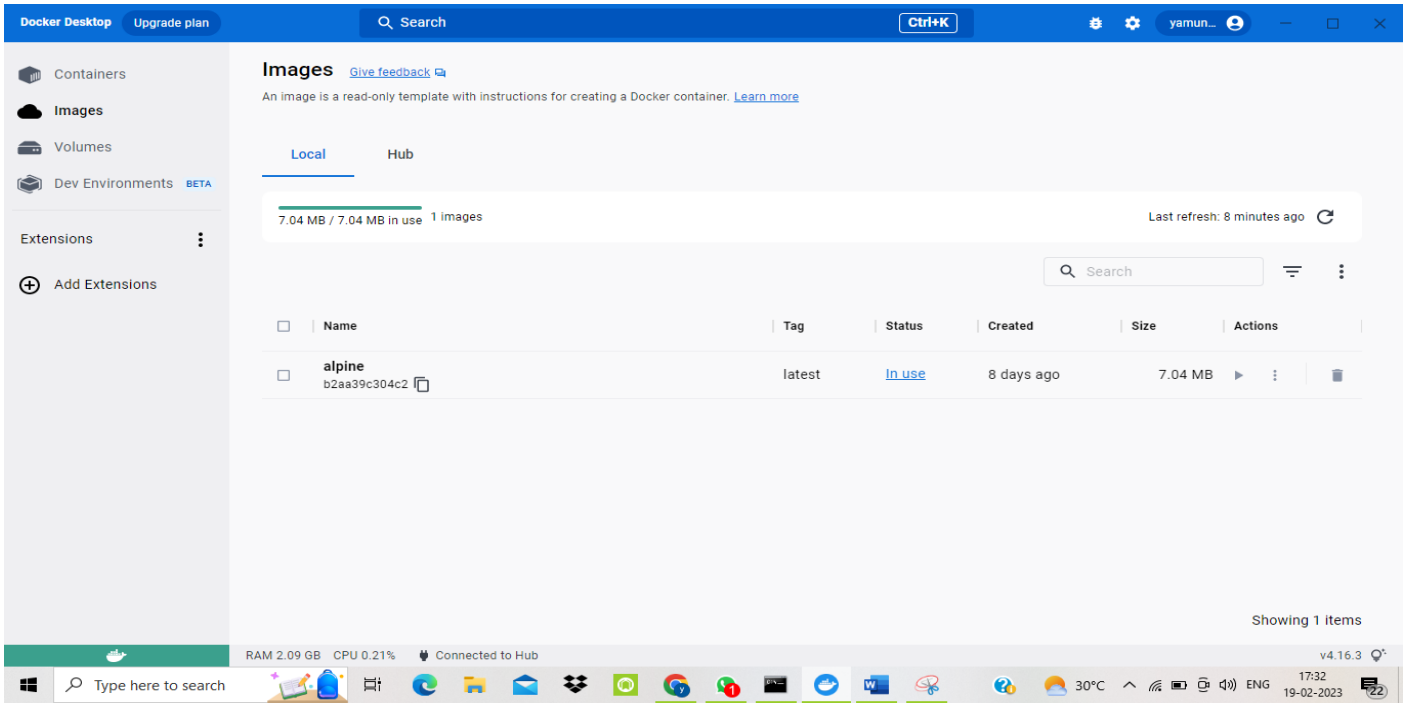
Alpine: Alpine is a linux based image and it is not very user-friendly compared with ubuntu. It is very minimum size. Alpine consists of executable codes, libraries, dependencies and many more.

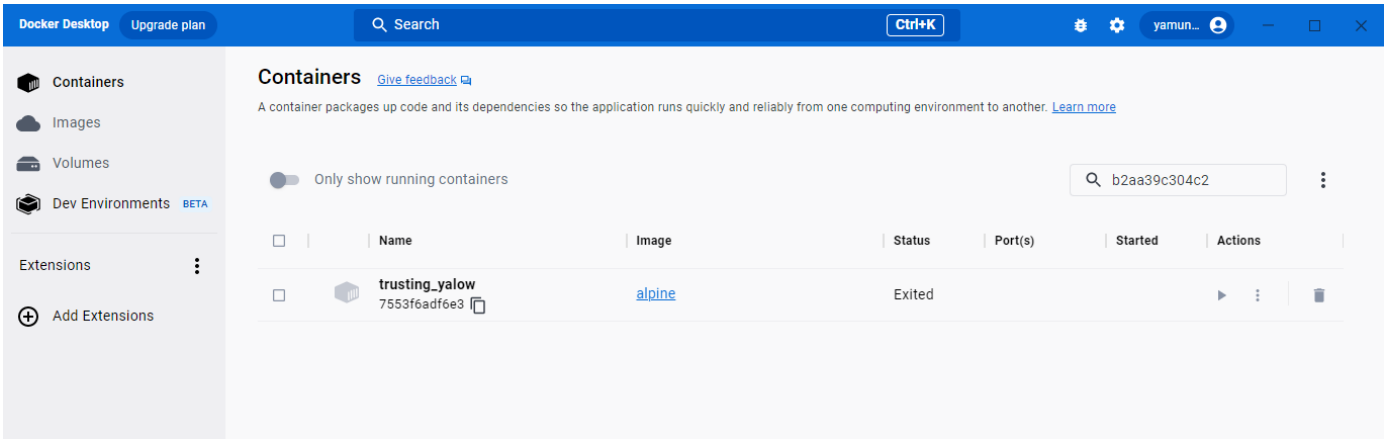
Method-1:

The image can be downloaded and run with a single command along with creating a container.

Command: docker run alpine

```
C:\Users\YAMUNA DEVI>docker run alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
63b65145d645: Pull complete
Digest: sha256:69665d02cb32192e52e07644d76bc6f25abeb5410edc1c7a81a10ba3f0efb90a
Status: Downloaded newer image for alpine:latest
```





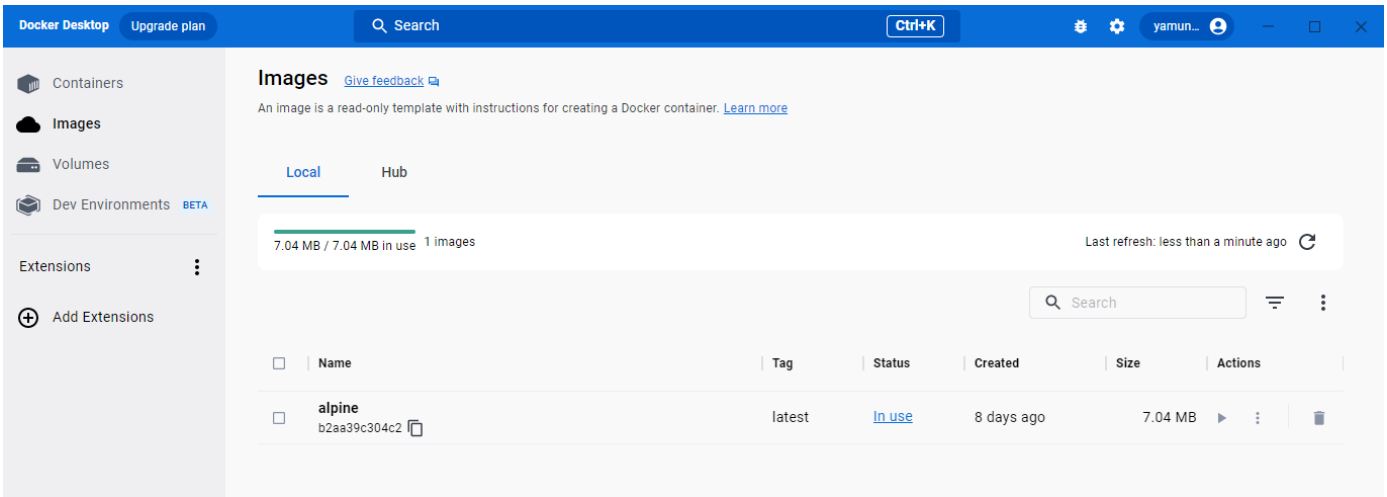
Method-2:

The image can be downloaded i.e., pulled from the docker hub using the command pull

Docker pull alpine

```
C:\Users\YAMUNA DEVI>docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
Digest: sha256:69665d02cb32192e52e07644d76bc6f25abeb5410edc1c7a81a10ba3f0efb90a
Status: Image is up to date for alpine:latest
docker.io/library/alpine:latest

C:\Users\YAMUNA DEVI>
```

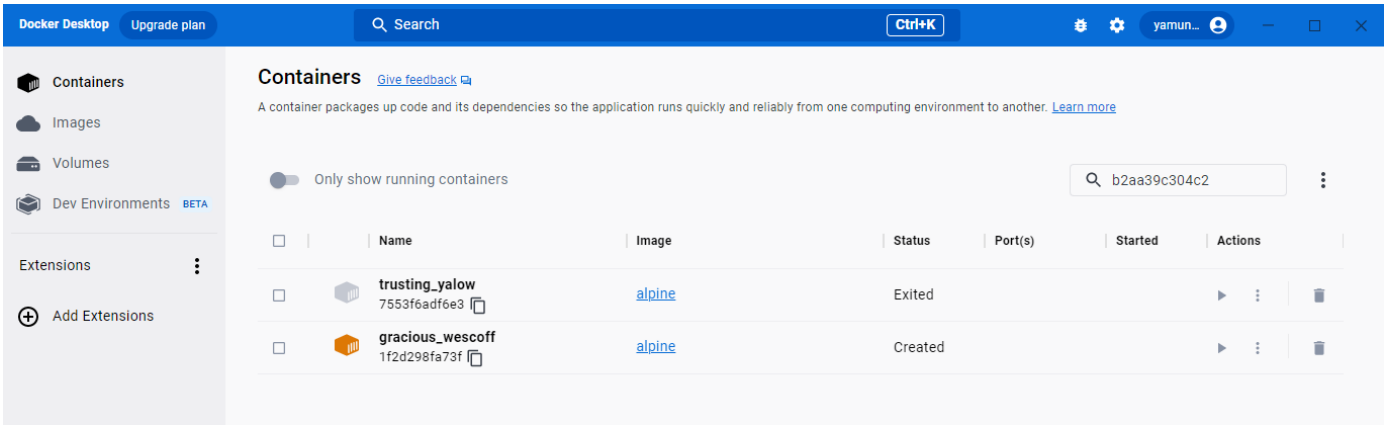


Now, the container can be created using the command create

Docker create <container_name>

```
C:\Users\YAMUNA DEVI>docker create alpine
1f2d298fa73ffc9c1b9b93d23988b21546a60b6be44cf66c0de41ce2c17d3178

C:\Users\YAMUNA DEVI>
```



The container can be run with this command.

Docker start <container_id>

```
C:\Users\YAMUNA DEVI>docker start 1f2d298fa73f
1f2d298fa73f

C:\Users\YAMUNA DEVI>
```

The command ps -a show the containers with all the information regarding them.

```
C:\Users\YAMUNA DEVI>docker ps -a
CONTAINER ID   IMAGE     COMMAND     CREATED              STATUS                      PORTS     NAMES
dccc9a4c7e27   alpine    "/bin/sh"   21 seconds ago       Exited (0) 20 seconds ago             quizzical_ritchie
1f2d298fa73f   alpine    "/bin/sh"   About a minute ago   Created                               gracious_wescoff
7553f6adf6e3   alpine    "/bin/sh"   30 minutes ago       Exited (0) 30 minutes ago             trusting_yalow
```

The command docker list images show the images present in docker

```
C:\Users\YAMUNA DEVI>docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
alpine       latest    b2aa39c304c2   8 days ago   7.05MB
```

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

A java application can be created and executed in docker by following steps:

Step-1: We can create a directory java-app by using mkdir command

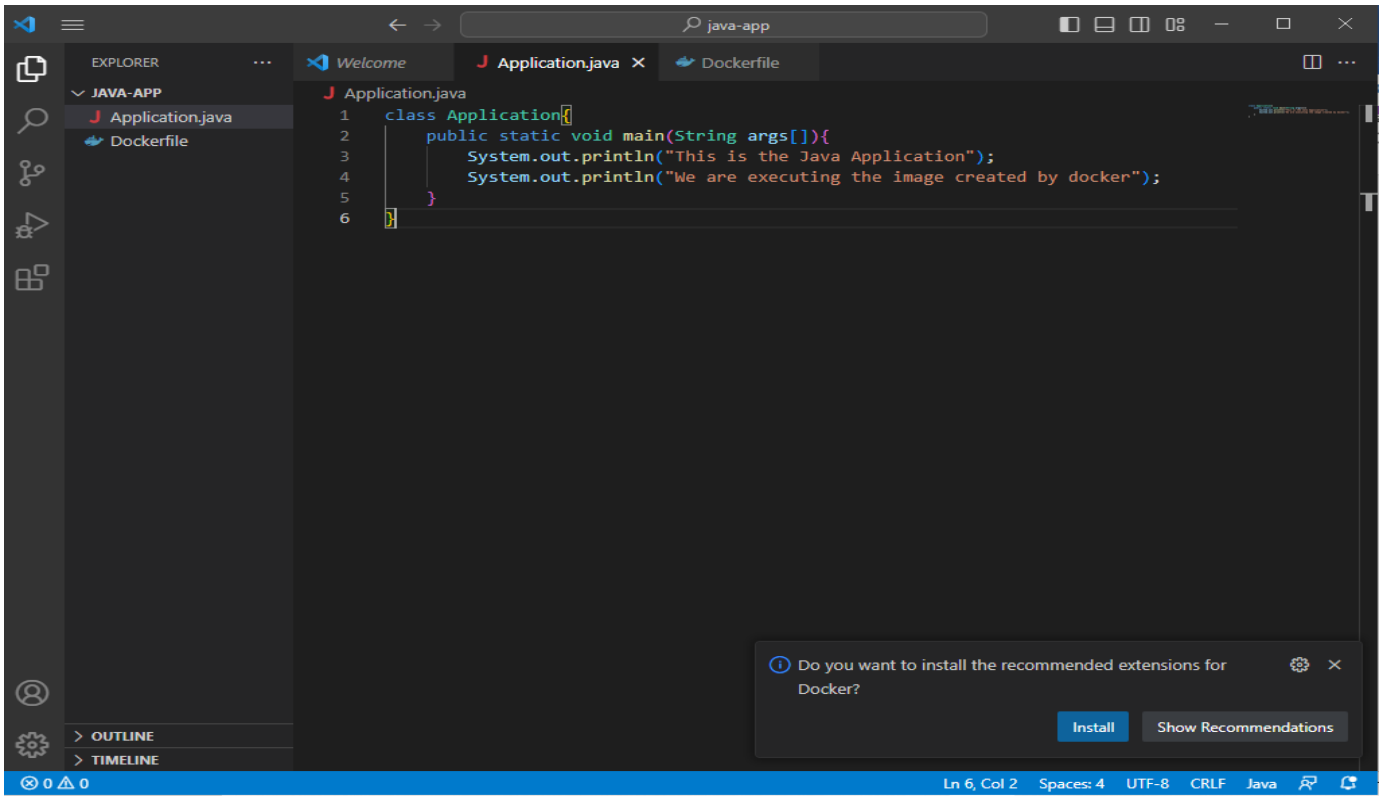
mkdir java-app

```
C:\Users\YAMUNA DEVI>mkdir java-app
```

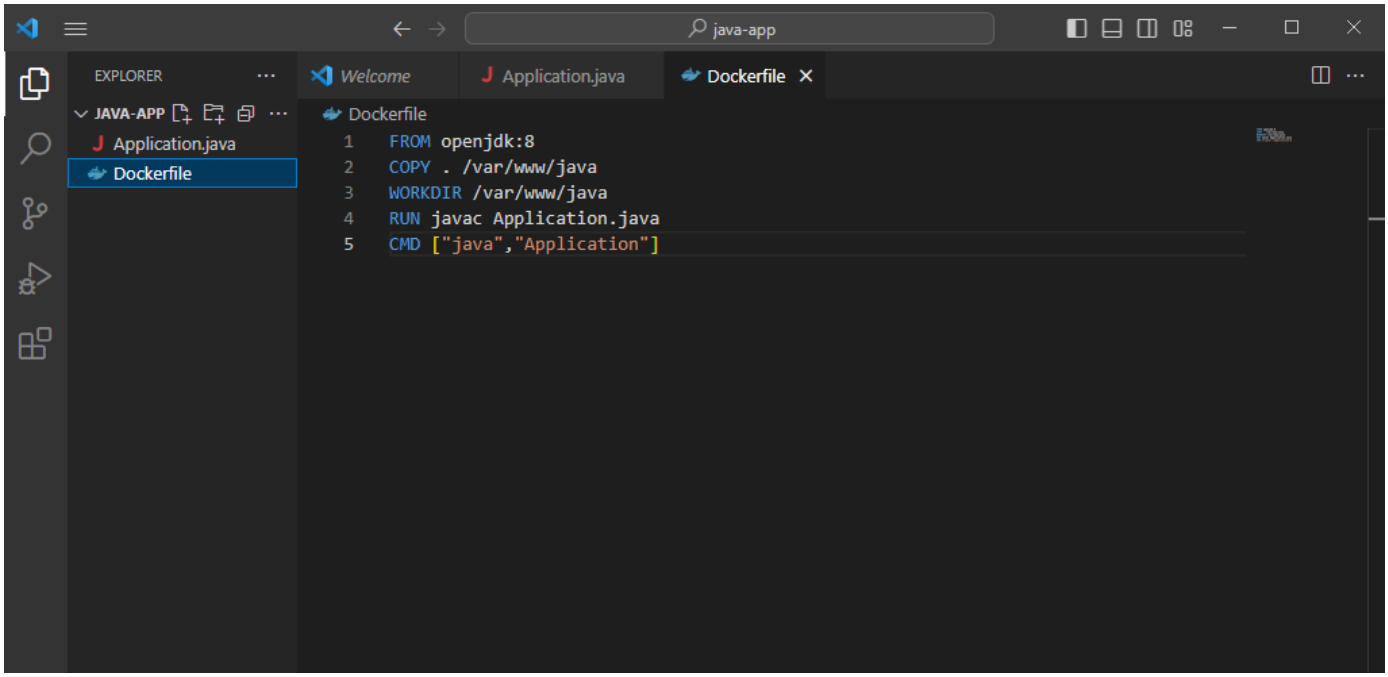
Step-2: Check whether it is created or not bu using dir command.

```
23-01-2023  11:25    <DIR>          help
19-02-2023  17:47    <DIR>          java-app
17-02-2023  10:47    <DIR>          java-docker-app
```

Step-3: Open the java-app folder on VScode and create a file Application.java



Step-4: Create a Dockerfile.



Step-5:

An image can be created using the Dockerfile

docker build -t <application_name>

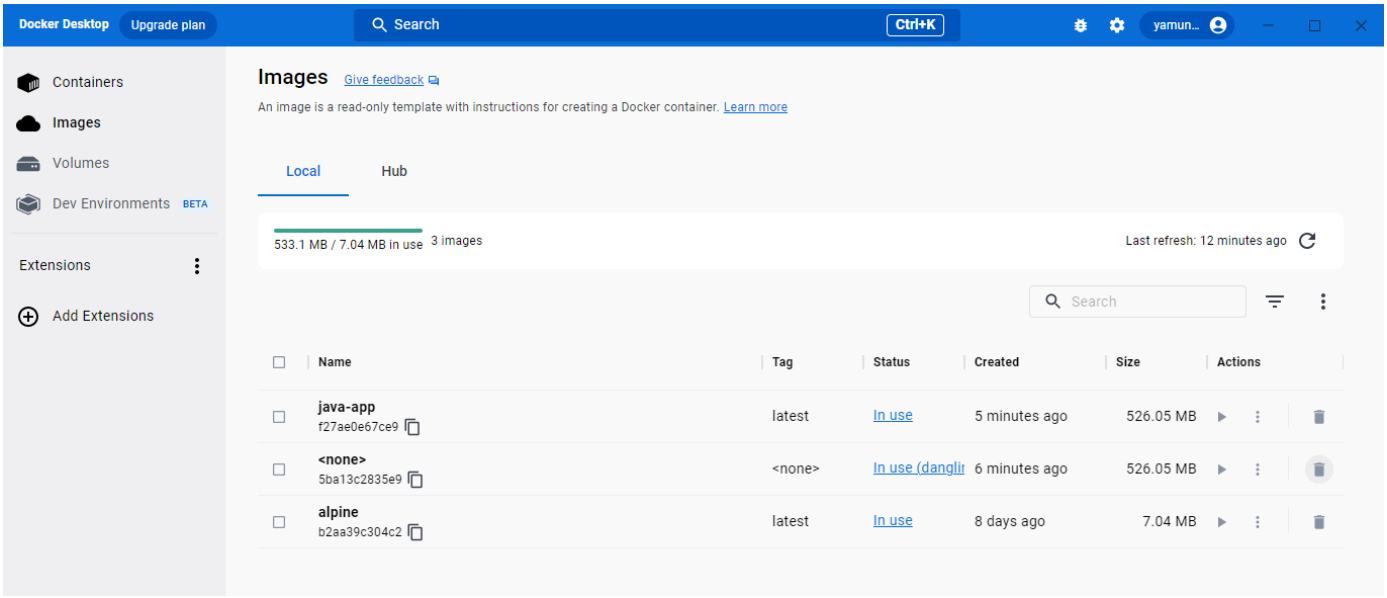
```
C:\Users\YAMUNA DEVI\java-app>docker build -t java-app .
[+] Building 3.4s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 31B                                                                 0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/openjdk:8                                        1.9s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 280B                                                                   0.0s
 => CACHED [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937  0.0s
 => [2/4] COPY . /var/www/java                                                                      0.1s
 => [3/4] WORKDIR /var/www/java                                                                     0.1s
 => [4/4] RUN javac Application.java                                                                1.0s
 => exporting to image                                                                              0.2s
 => => exporting layers                                                                             0.1s
 => => writing image sha256:5ba13c2835e9a98d8c61af477a6d65a807fd7cd0ba6be9da257a1abb4370b352        0.0s
 => => naming to docker.io/library/java-app                                                         0.0s

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

Step-6:

We can check the image created in docker desktop and command prompt alike.

```
C:\Users\YAMUNA DEVI\java-app>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
java-app     latest    f27ae0e67ce9   8 minutes ago   526MB
<none>       <none>    5ba13c2835e9   9 minutes ago   526MB
alpine       latest    b2aa39c304c2   8 days ago      7.05MB
```



Step-7:

Now, we can execute the image created of the java application.

docker run <image>

```
C:\Users\YAMUNA DEVI\java-app>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
java-app     latest    f27ae0e67ce9   8 minutes ago    526MB
<none>       <none>    5ba13c2835e9   9 minutes ago    526MB
alpine       latest    b2aa39c304c2   8 days ago       7.05MB
```

GitHub: https://github.com/Yamuna-Devi-Buradakavi/Assignments