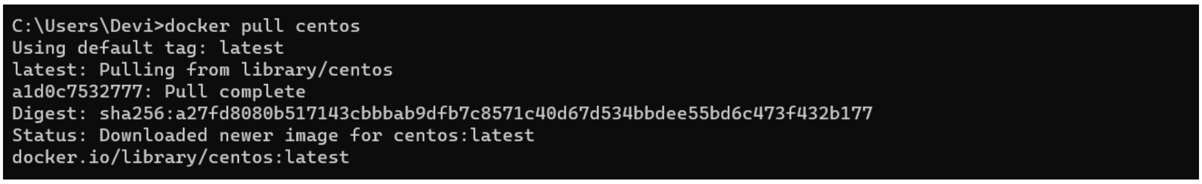
Devops Assignment

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

Pull the image from docker hub.

1.we can pull the image from docker hub using pull command

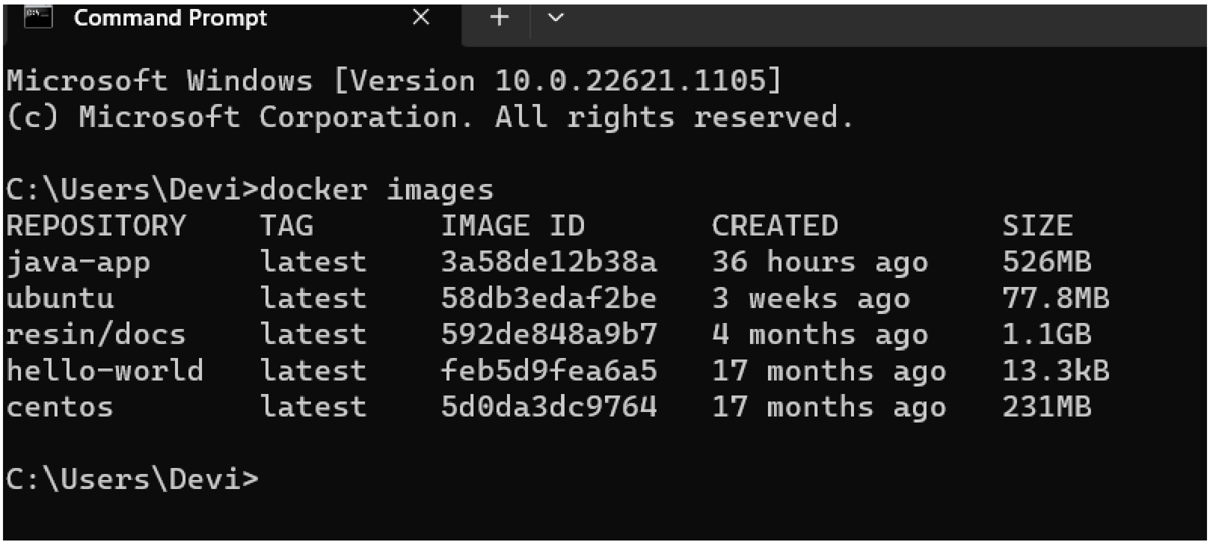
Syntax: docker pull imagename.



```
C:\Users\Devi>docker pull centos
Using default tag: latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
```

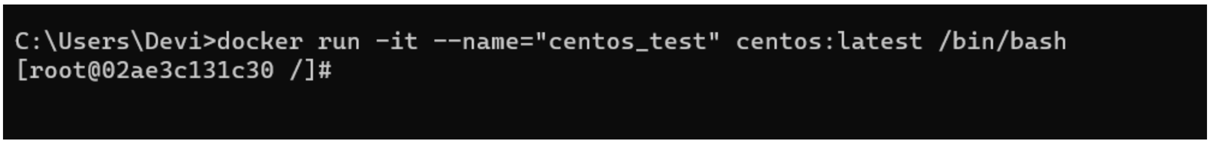
2.We can see the downloaded images by using the below syntax.

Syntax: docker images



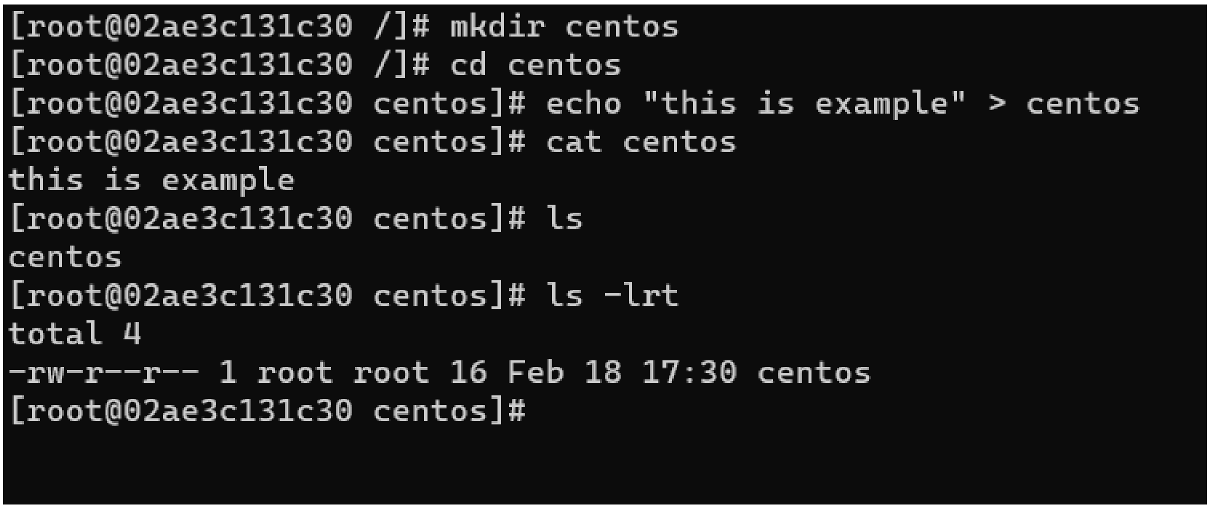
```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Devi>docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
java-app      latest    3a58de12b38a   36 hours ago     526MB
ubuntu        latest    58db3edaf2be   3 weeks ago      77.8MB
resin/docs    latest    592de848a9b7   4 months ago     1.1GB
hello-world   latest    feb5d9fea6a5   17 months ago    13.3kB
centos        latest    5d0da3dc9764   17 months ago    231MB

C:\Users\Devi>
```

3.Once the centos Image is downloaded, we will run docker container based on this image with the name “centos_test”.



```
C:\Users\Devi>docker run -it --name="centos_test" centos:latest /bin/bash
[root@02ae3c131c30 /]#
```

4. Now create a new directory in the container “centos” with a file in it as “centos”. Also add some random text in the centos and execute it.

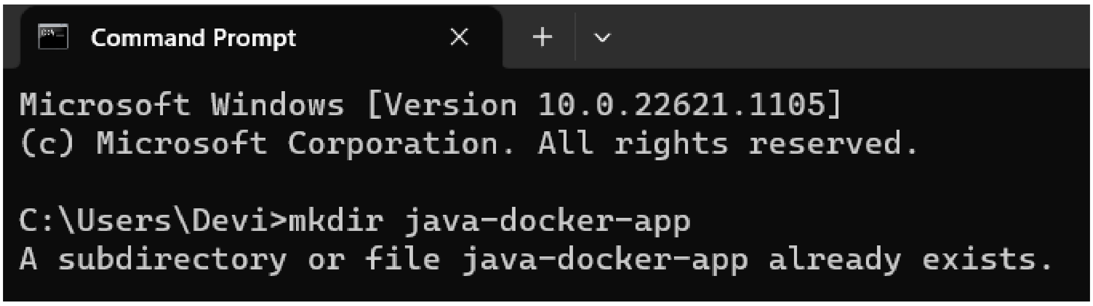


```
[root@02ae3c131c30 /]# mkdir centos
[root@02ae3c131c30 /]# cd centos
[root@02ae3c131c30 centos]# echo "this is example" > centos
[root@02ae3c131c30 centos]# cat centos
this is example
[root@02ae3c131c30 centos]# ls
centos
[root@02ae3c131c30 centos]# ls -lrt
total 4
-rw-r--r-- 1 root root 16 Feb 18 17:30 centos
[root@02ae3c131c30 centos]#
```

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

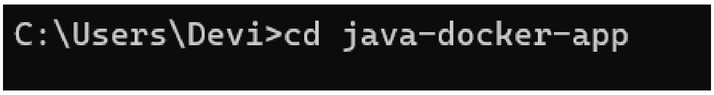
Steps for creating the basic java application.

Step 1) Create a directory



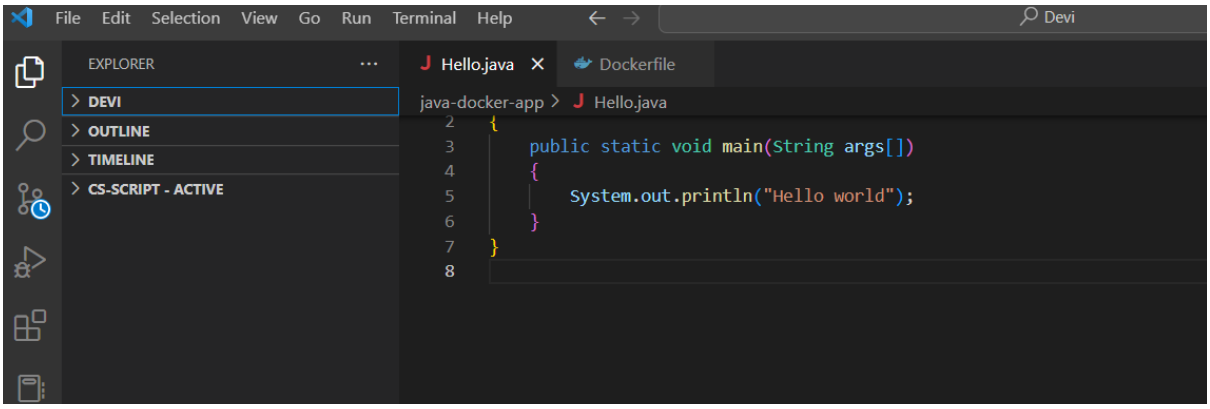
```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Devi>mkdir java-docker-app
A subdirectory or file java-docker-app already exists.
```

Step 2) change the directory that have created.



```
C:\Users\Devi>cd java-docker-app
```

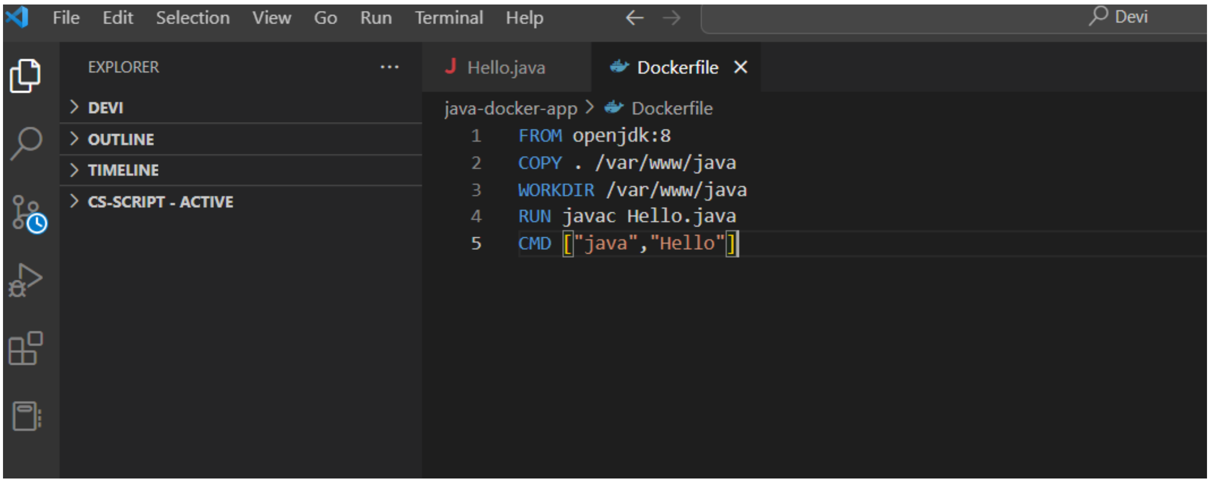
Step 3) create a java file. And write the code in it.


File Edit Selection View Go Run Terminal Help
Devi
EXPLORER
DEVI
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
Hello.java
Dockerfile
java-docker-app > Hello.java

```
{
    public static void main(String args[])
    {
        System.out.println("Hello world");
    }
}
```

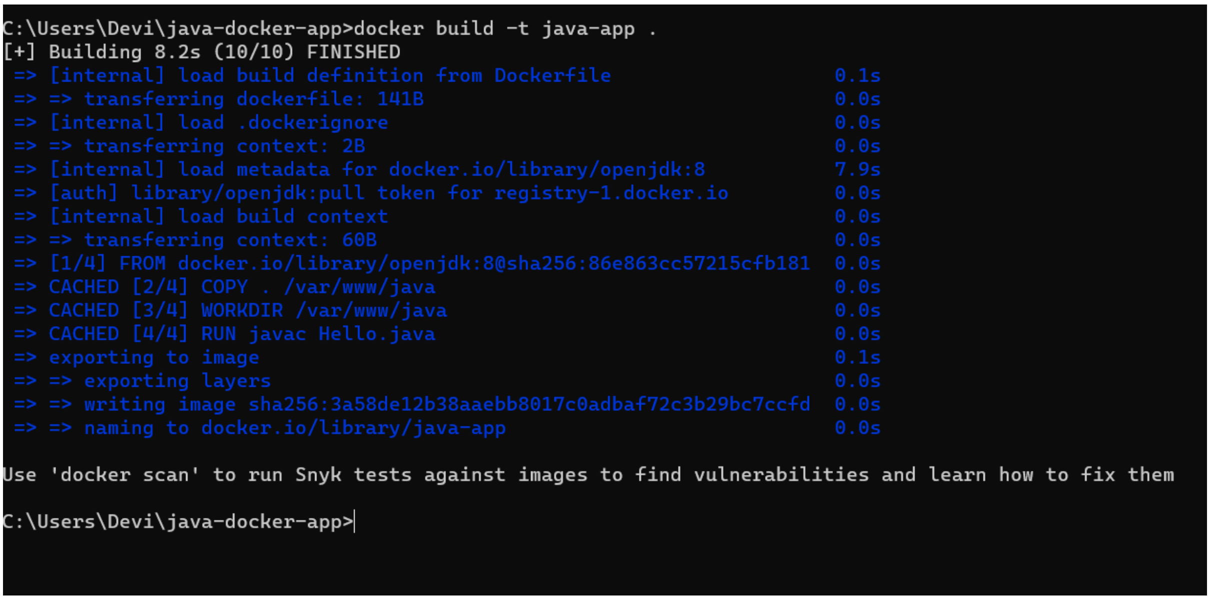
Step 4) create a Dockerfile.

Dockerfile is a simple text file that consists of instructions to build Docker images.


File Edit Selection View Go Run Terminal Help
Devi
EXPLORER
DEVI
OUTLINE
TIMELINE
CS-SCRIPT - ACTIVE
Hello.java
Dockerfile
java-docker-app > Dockerfile

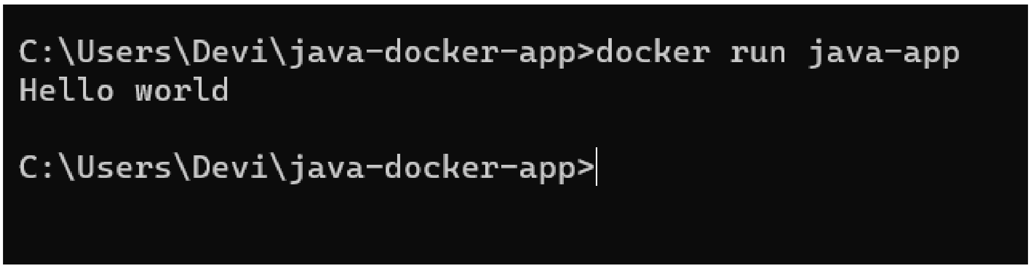
```
FROM openjdk:8
COPY . /var/www/java
WORKDIR /var/www/java
RUN javac Hello.java
CMD ["java","Hello"]
```

Step 5) builds Docker images from a Dockerfile .



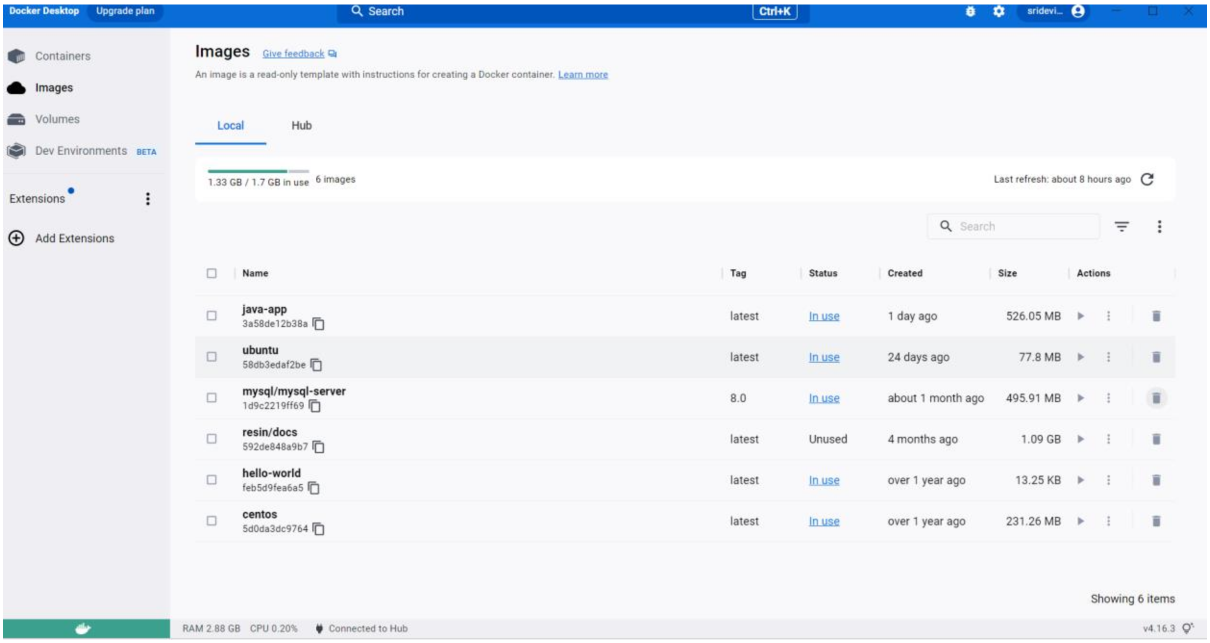
```
C:\Users\Devi\java-docker-app>docker build -t java-app .
[+] Building 8.2s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                   0.1s
 => => transferring dockerfile: 141B                                   0.0s
 => [internal] load .dockerignore                                      0.0s
 => => transferring context: 2B                                        0.0s
 => [internal] load metadata for docker.io/library/openjdk:8           7.9s
 => [auth] library/openjdk:pull token for registry-1.docker.io         0.0s
 => [internal] load build context                                      0.0s
 => => transferring context: 60B                                       0.0s
 => [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181  0.0s
 => CACHED [2/4] COPY . /var/www/java                                  0.0s
 => CACHED [3/4] WORKDIR /var/www/java                                 0.0s
 => CACHED [4/4] RUN javac Hello.java                                  0.0s
 => exporting to image                                                 0.1s
 => => exporting layers                                                0.0s
 => => writing image sha256:3a58de12b38aaebb8017c0adbaf72c3b29bc7ccfd  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

C:\Users\Devi\java-docker-app>
```

Step 6) By using the run command run the Sjava file.



```
C:\Users\Devi\java-docker-app>docker run java-app
Hello world

C:\Users\Devi\java-docker-app>
```

When we open the docker desktop the it shows the java-app image is in use.



Images Give feedback

An image is a read-only template with instructions for creating a Docker container. Learn more

1.33 GB / 1.7 GB in use 6 images

Last refresh: about 8 hours ago

□	Name	Tag	Status	Created	Size	Actions
□	java-app 3a58de12b38a	latest	In use	1 day ago	526.05 MB	
□	ubuntu 58db3edaf2be	latest	In use	24 days ago	77.8 MB	
□	mysql/mysql-server 1d9c2219ff69	8.0	In use	about 1 month ago	495.91 MB	
□	resin/docs 592de848a9b7	latest	Unused	4 months ago	1.09 GB	
□	hello-world feb5d9fea6a5	latest	In use	over 1 year ago	13.25 KB	
□	centos 5d0da3dc9764	latest	In use	over 1 year ago	231.26 MB	

Showing 6 items

RAM 2.88 GB CPU 0.20% Connected to Hub v4.16.3