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

DOCKER

- It is an application.
- It is a tool of DevOps for deployment
- It is used containerization process.
- Containerization is a software deployment process that bundles an application's code with all the files and libraries it needs to run on any infrastructure. Traditionally, to run any application on your computer, you had to install the version that matched your machine's operating system.
- Android application builds with java and xml

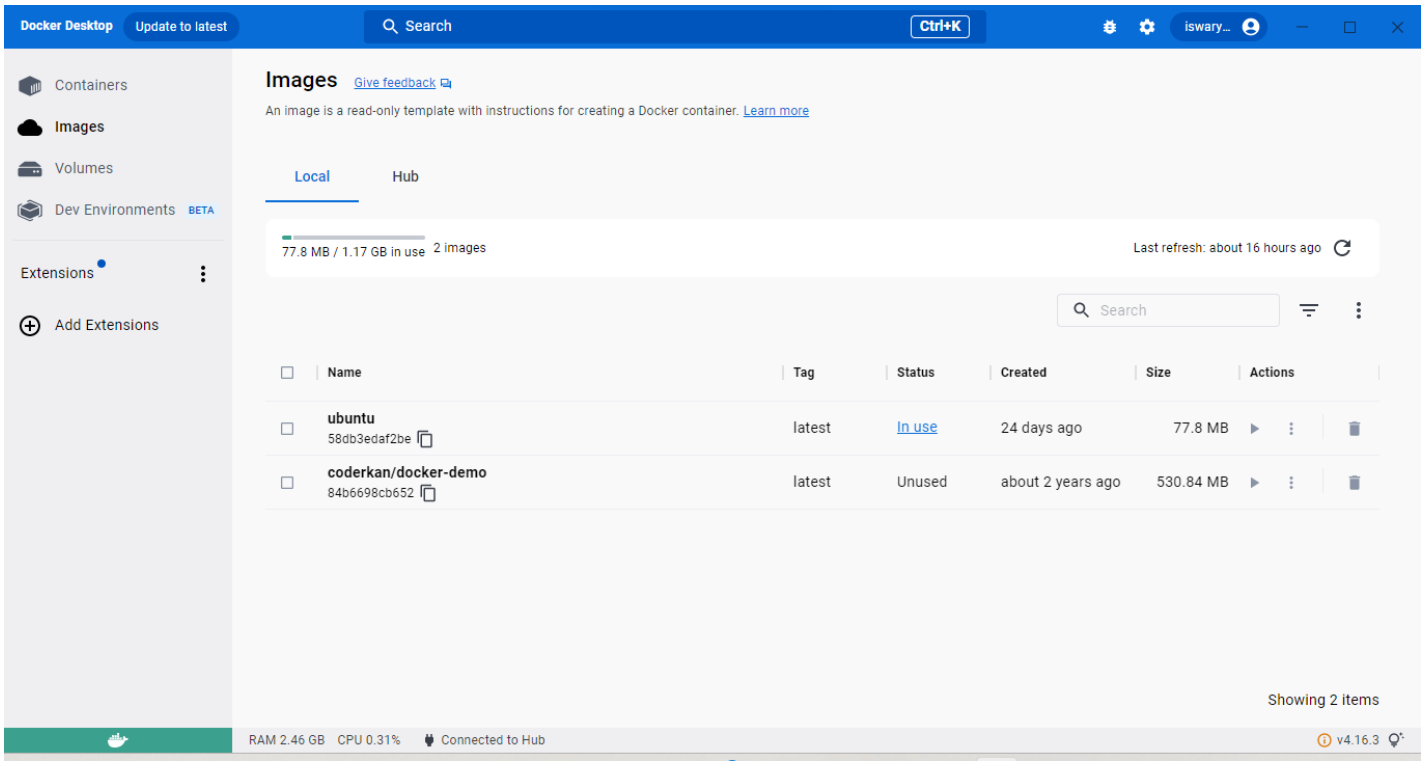
Docker pull image:

docker pull command we create or download the image and it gives the status of the image.

Syntax: docker pull <image name>

When we run the pull command from the command line, it first checks locally or on the host for the images and if the image does not exist locally then the Docker daemon connects to the public registry 'hub.docker.com' if there is no private registry mentioned in the 'daemon.json' file and pulls the Docker image mentioned in the command and if it finds the image locally then it checks for the updates and downloads newer version of the image.

```
C:\Users\durga>docker pull coderkan/docker-demo
Using default tag: latest
latest: Pulling from coderkan/docker-demo
0ecb575e629c: Pull complete
7467d1831b69: Pull complete
feab2c490a3c: Pull complete
f15a0f46f8c3: Pull complete
26cb1dfcbebb: Pull complete
5b224ce6d4ea: Pull complete
c932fe81bb40: Pull complete
b079b2033d71: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:c561be4395468e6da7d4a6397520eb13ba92d599abdb7176834ed46f962aa37d
Status: Downloaded newer image for coderkan/docker-demo:latest
docker.io/coderkan/docker-demo:latest
```



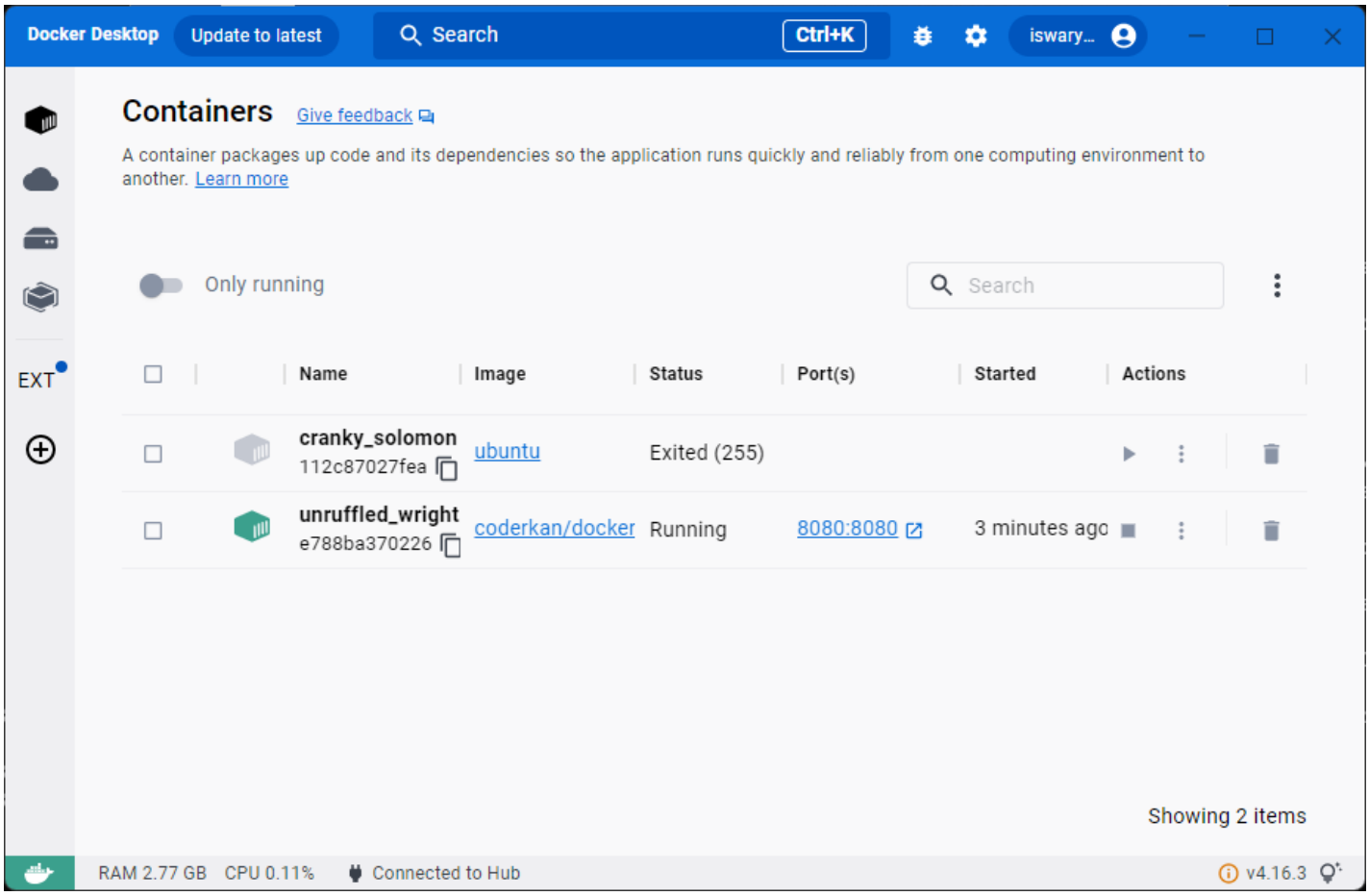
Docker run:

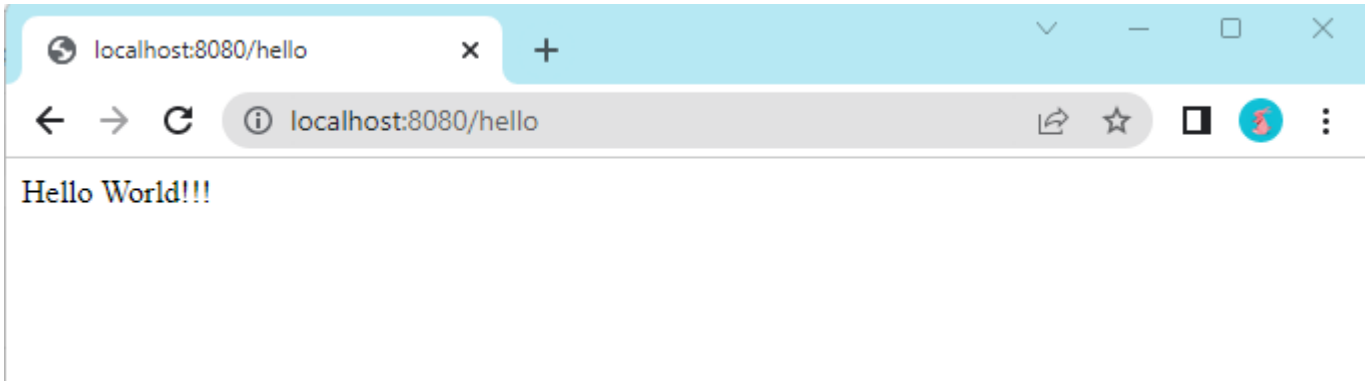
Docker run command is used to create a container and execute or run the container in by using the command **docker run image.**

```
C:\Users\durga>docker run -p 8080:8080 coderkan/docker-demo

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.2)

2023-02-19 06:15:28.059  INFO 1 --- [           main] c.c.dockerdemo.DockerDemoApplication     : Starting DockerDemoApplication v0.0.1-SNAPSHOT usin
g Java 1.8.0_282 on e788ba370226 with PID 1 (/usr/src/my-sample-app.jar started by root in /usr/src)
2023-02-19 06:15:28.068  INFO 1 --- [           main] c.c.dockerdemo.DockerDemoApplication     : No active profile set, falling back to default prof
iles: default
2023-02-19 06:15:30.252  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2023-02-19 06:15:30.273  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-02-19 06:15:30.273  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.41]
2023-02-19 06:15:30.361  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2023-02-19 06:15:30.361  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization complete
d in 2097 ms
2023-02-19 06:15:30.650  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecut
or'
2023-02-19 06:15:31.006  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context
 path ''
2023-02-19 06:15:31.033  INFO 1 --- [           main] c.c.dockerdemo.DockerDemoApplication     : Started DockerDemoApplication in 3.595 seconds (JVM
 running for 4.332)
2023-02-19 06:16:25.375  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherSe
rvlet'
2023-02-19 06:16:25.375  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-02-19 06:16:25.376  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
```





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

Ans: -

Step 1: First create directory using command **mkdir.** It is used to organize the files. To open visual studio code execute **code .**

```
C:\Users\durga\OneDrive\Desktop\Iswarya>mkdir java-docker-app

C:\Users\durga\OneDrive\Desktop\Iswarya>code .
```

Step 2: Create a Hello.java and Dockerfile with the code

Hello.java:

```
class Hello{
   public static void main(String[] args){
   System.out.println("This is java app \n by using Docker");
}
}
```

Dockerfile:

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

Step3: Now build a java application with the help of the command **docker build -t java-app .**

```
PS C:\Users\durga\OneDrive\Desktop\Iswarya\java-docker-app> docker build -t java-app .
[+] Building 2.3s (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                                  2.1s
 => [internal] load build context                                                             0.0s
 => => transferring context: 61B                                                              0.0s
 => [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58b  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                                                                        0.0s
 => => exporting layers                                                                       0.0s
 => => writing image sha256:8a0ca4e4550113b81849c5536c718d8c7b191e0728a428a5f28fd29037877696  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
PS C:\Users\durga\OneDrive\Desktop\Iswarya\java-docker-app>
```

Step 4: Now run the docker image using **docker run java-app**

```
PS C:\Users\durga\OneDrive\Desktop\Iswarya\java-docker-app> docker run java-app
This is java app
 by using Docker
```