Name: Mrunal Vaidya
Roll : 68
XIE ID: 202003060
Batch: C

Experiment 9

Aim: To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.

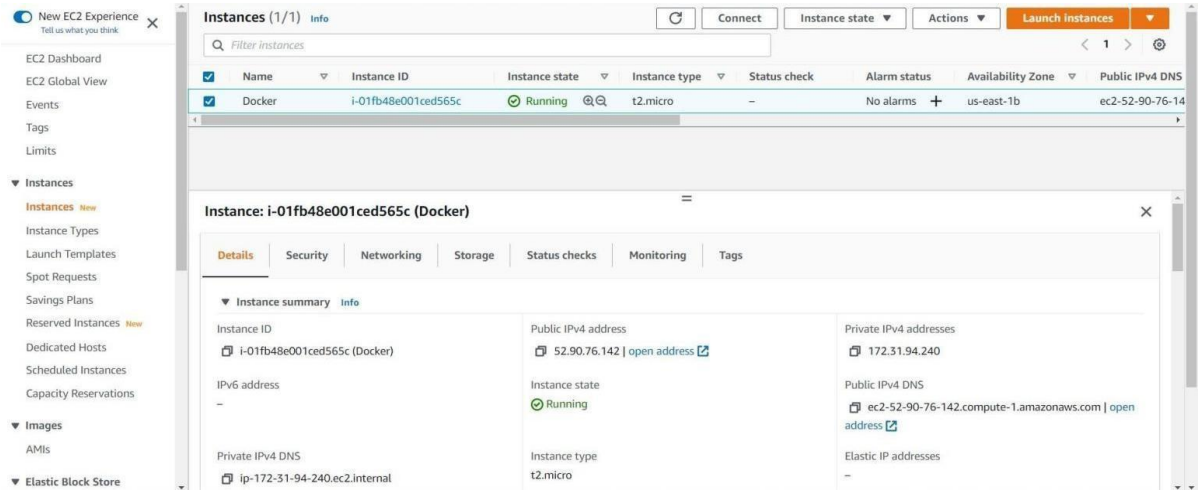
Theory:

A dockerfile contains a set of instructions that are executed step by step when you use the docker build command to build the docker image. It contains certain instructions and commands that decides the structure of your image, the amount of time taken to build the image, contains instructions related to docker build context, contains information related to the packages and libraries to be installed in the container and many more. Hence, it becomes very important to create an efficient, reusable, clean dockerfile as it contains the blueprint of the image that you will build.
Some Dockerfile instructions:

- Run: A RUN instruction is used to run specified commands. You can use several RUN instructions to run different commands. But it is an efficient approach to combine all the RUN instructions into a single one. Each RUN command creates a new cache layer or an intermediate image layer and hence chaining all of them into a single line, becomes efficient. However, chaining multiple RUN instructions could lead to cache bursts as well.
- Pull: A pull instruction is used to pull an image from the daemon.
- Stop: A stop instruction is used to stop an image.

Output:

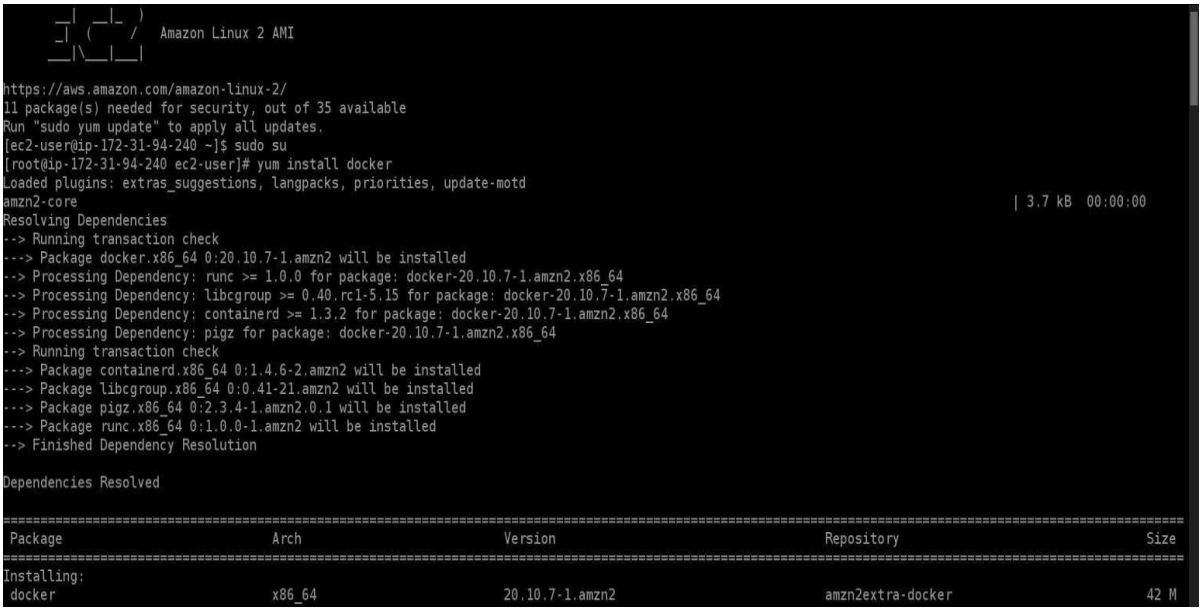
1: Launching an Ec2 linux aws instance



Name: Mrunal Vaidya XIE ID: 202003060
Roll : 68 Batch: C

2: Connecting to the ec2 instance

3: Installing docker using yum install docker



4: Starting Docker service and checking its version

```
[root@ip-172-31-94-240 ec2-user]# systemctl start docker
[root@ip-172-31-94-240 ec2-user]# docker version
Client:
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.15.14
 Git commit:        f0df350
 Built:             Tue Aug 17 16:01:45 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server:
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.15.14
  Git commit:       b0f5bc3
  Built:            Tue Aug 17 16:02:23 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.6
  GitCommit:        d71fcd7d8303cbf684402823e425e9dd2e99285d
 runc:
  Version:          1.0.0
  GitCommit:        %runc_commit
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
[root@ip-172-31-94-240 ec2-user]#
```

5: Running a hello-world image from library after pulling it

```
[root@ip-172-31-94-240 ec2-user]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:393b81f0ea5a98a7335d7ad44be96fe76ca8eb2eaa76950eb8c989ebf2b78ec0
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[root@ip-172-31-94-240 ec2-user]#
```

6: checking all Images

```
[root@ip-172-31-94-240 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
hello-world   latest    feb5d9fea6a5   4 days ago    13.3kB
[root@ip-172-31-94-240 ec2-user]#
```

7: pulling Ubuntu image

```
[root@ip-172-31-94-240 ec2-user]# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
35807b77a593: Pull complete
Digest: sha256:9d6a8699fb5c9c39cf08a0871bd6219f0400981c570894cd8cbea30d3424a31f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[root@ip-172-31-94-240 ec2-user]#
```

8: Ubuntu image pulled and visible in installed images

```
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   4 days ago     13.3kB
ubuntu        latest    fb52e22af1b0   4 weeks ago    72.8MB
[root@ip-172-31-94-240 ec2-user]#
```

9: Pulling alpine image

```
[root@ip-172-31-94-240 ec2-user]# docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
a0d0a0d46f8b: Pull complete
Digest: sha256:e1c082e3d3c45cccac829840a25941e679c25d438cc8412c2fa221cf1a824e6a
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
[root@ip-172-31-94-240 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   4 days ago     13.3kB
ubuntu        latest    fb52e22af1b0   4 weeks ago    72.8MB
alpine        latest    14119a10abf4   4 weeks ago    5.6MB
[root@ip-172-31-94-240 ec2-user]#
```

10: listing all images

```
[root@ip-172-31-94-240 ec2-user]# docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
a0d0a0d46f8b: Pull complete
Digest: sha256:e1c082e3d3c45cccac829840a25941e679c25d438cc8412c2fa221cf1a824e6a
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
[root@ip-172-31-94-240 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   4 days ago     13.3kB
ubuntu        latest    fb52e22af1b0   4 weeks ago    72.8MB
alpine        latest    14119a10abf4   4 weeks ago    5.6MB
[root@ip-172-31-94-240 ec2-user]# docker run -it ubuntu
oot@0098b8ccfaa5: /root@0098b8ccfaa5:/# exit
exit
[root@ip-172-31-94-240 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-94-240 ec2-user]# docker run -it ubuntu
```

11: Listing all the running containers

```
[root@ip-172-31-94-240 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED             STATUS             PORTS     NAMES
1cc87915092a   ubuntu    "bash"    About a minute ago   Up About a minute            frosty_chatelet
[root@ip-172-31-94-240 ec2-user]#
```

12: Displaying all the containers

```
[root@ip-172-31-94-240 ec2-user]# docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
1cc87915092a   ubuntu        "bash"     4 minutes ago    Up 4 minutes                          frosty_chatelet
0098b8ccfaa5   ubuntu        "bash"     8 minutes ago    Exited (0) 6 minutes ago              epic_lewin
fea2d4987465   hello-world   "/hello"   18 minutes ago   Exited (0) 18 minutes ago             recursing_yalow
[root@ip-172-31-94-240 ec2-user]#
```

13: Stopping containers using their Container ID

```
[root@ip-172-31-94-240 ec2-user]# docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
1cc87915092a   ubuntu        "bash"     4 minutes ago    Up 4 minutes                          frosty_chatelet
0098b8ccfaa5   ubuntu        "bash"     8 minutes ago    Exited (0) 6 minutes ago              epic_lewin
fea2d4987465   hello-world   "/hello"   18 minutes ago   Exited (0) 18 minutes ago             recursing_yalow
[root@ip-172-31-94-240 ec2-user]# docker stop 0098b8ccfaa5
0098b8ccfaa5
[root@ip-172-31-94-240 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
1cc87915092a   ubuntu    "bash"    10 minutes ago   Up 10 minutes             frosty_chatelet
[root@ip-172-31-94-240 ec2-user]# docker stop 1cc87915092a
1cc87915092a
[root@ip-172-31-94-240 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-94-240 ec2-user]#
```

Conclusion: From this experiment it is concluded that, we have successfully learned Dockerfile instructions, build an image for a sample web application using Dockerfile. In this experiment we also studied the concept of containerization and analysed the Containerization of OS images and deployment of applications over Docker. And hence, with this experiment we have achieved the Lab Outcome Five (LO5).

POs Achieved: PO1, PO5, PO12.