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

Experiment 8

Aim: To understand Docker Architecture and Container Life Cycle, install Docker and execute Docker commands to manage images and interact with containers.

Theory:

❖ Docker overview

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.

❖ The Docker platform

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

- Docker provides tooling and a platform to manage the lifecycle of your containers:
- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.

When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

Before learning the Docker architecture, first, you should know about the Docker Daemon.

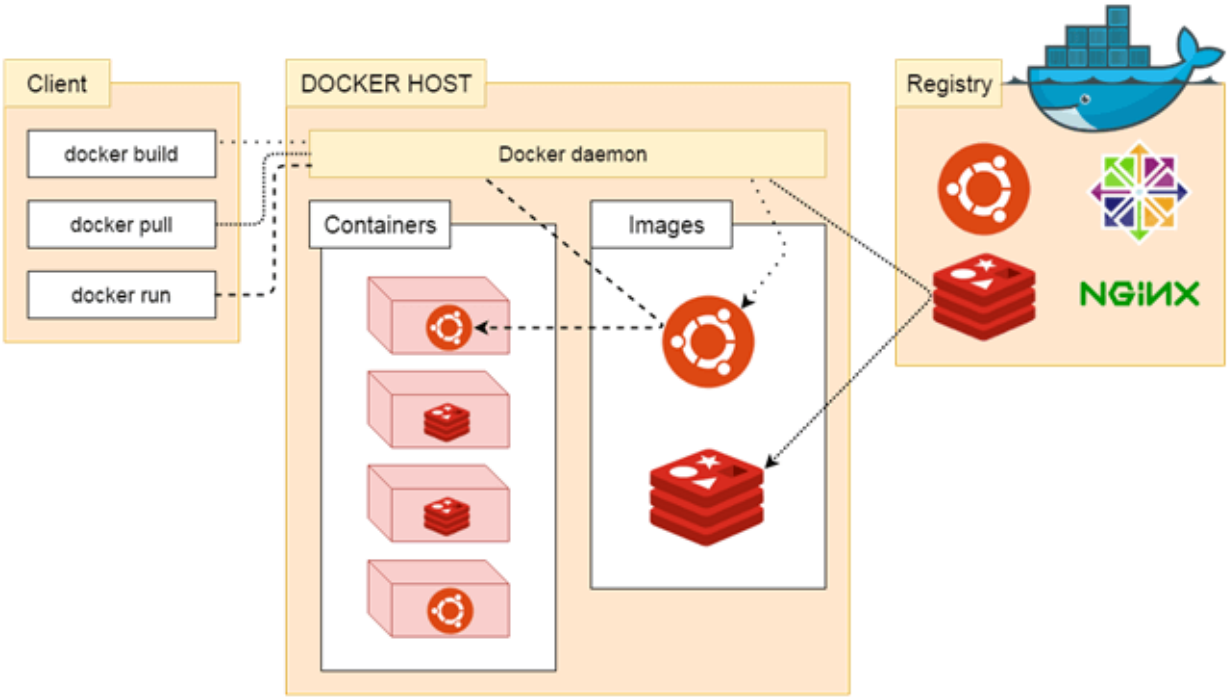
❖ What is Docker daemon?

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

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

❖ Docker architecture

Docker follows Client-Server architecture, which includes the three main components that are **Docker Client**, **Docker Host**, and **Docker Registry**.



1. Docker Client:

Docker client uses **commands** and **REST APIs** to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request. Docker Client uses Command Line Interface (CLI) to run the following commands –

- docker build
- docker pull
- docker run

2. Docker Host:

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

3. Docker Registry:

Docker Registry manages and stores the Docker images.
There are two types of registries in the Docker –

Pubic Registry - Public Registry is also called as **Docker hub**.
Private Registry - It is used to share images within the enterprise.

❖ **Docker Objects**

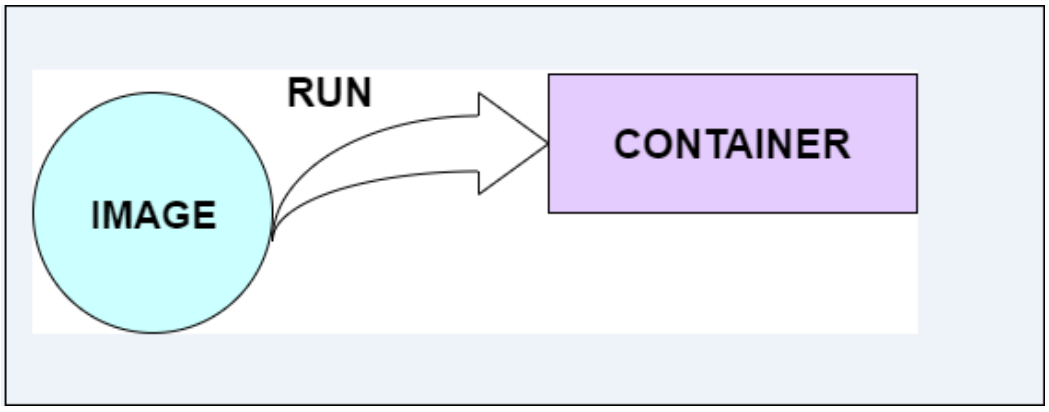
There are the following Docker Objects -

1. **Docker Images:**

Docker images are the **read-only binary templates** used to create Docker Containers. It uses a private container registry to share container images within the enterprise and also uses public container registry to share container images within the whole world. Metadata is also used by docket images to describe the container's abilities.

2. **Docker Containers:**

Containers are the structural units of Docker, which is used to hold the entire package that is needed to run the application. The advantage of containers is that it requires very less resources. In other words, we can say that the image is a template, and the container is a copy of that template.



3. **Docker Networking:**

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

- **Bridge -** Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.
- **Host -** It is used when we don't need for network isolation between the container and the host.
- **None -** It disables all the networking.
- **Overlay -** Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.
- **Macvlan -** Macvlan is used when we want to assign MAC addresses to the containers.

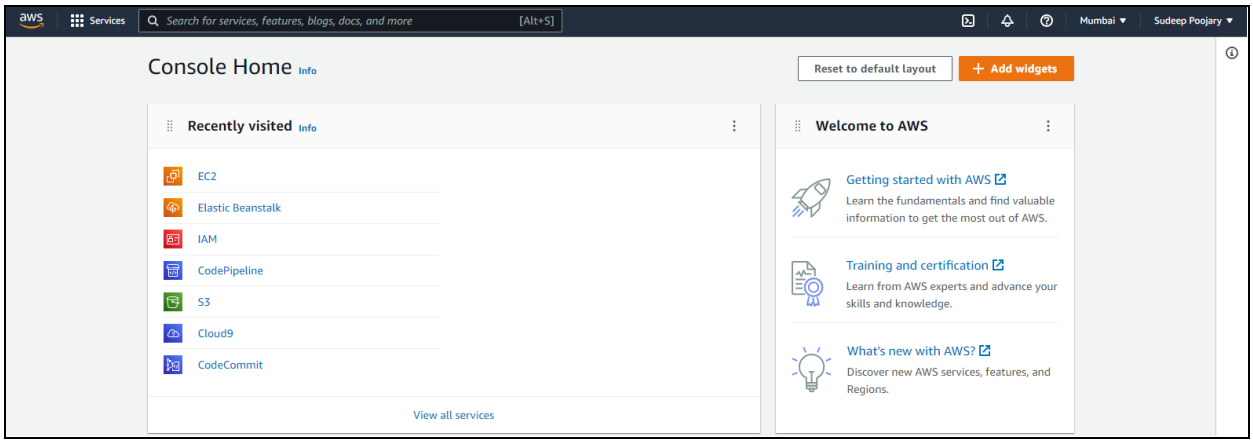
4. Docker Storage:

Docker Storage is used to store data on the container. Docker offers the following options for the Storage -

- **Data Volume -** Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.
- **Directory Mounts -** It is one of the best options for docker storage. It mounts a host's directory into a container.
- **Storage Plugins -** It provides an ability to connect to external storage platforms.

STEPS TO PERFORM THE EXPERIMENT:

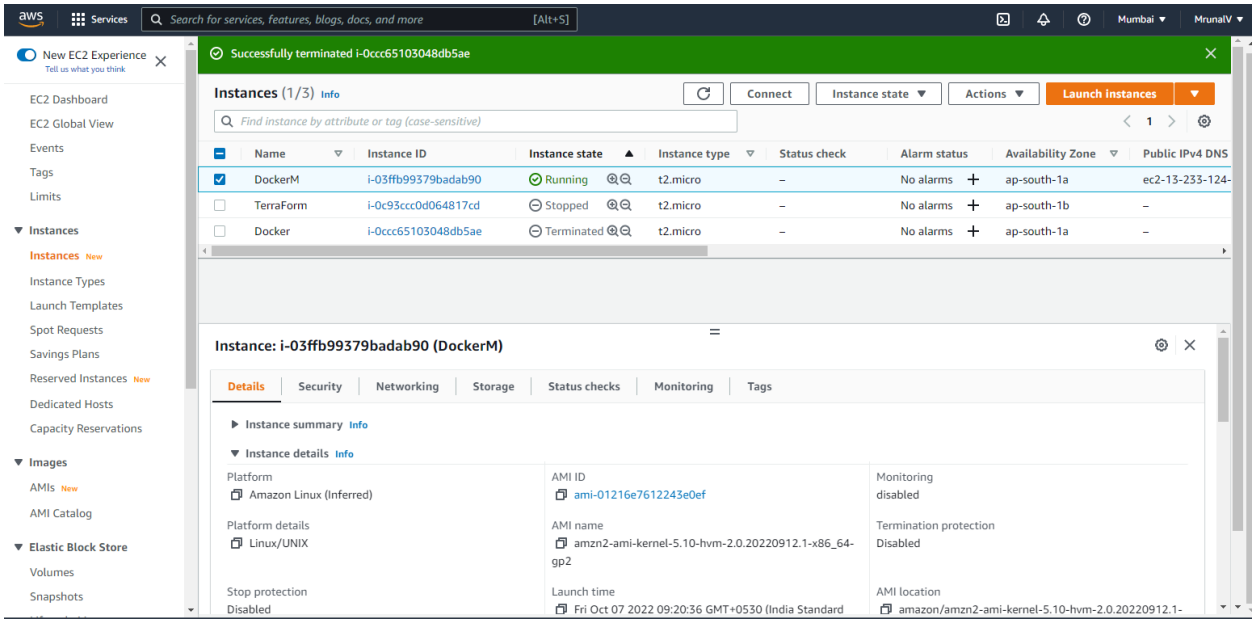
44Login to your AWS Account
Now, Launch an instance and select Amazon Linux

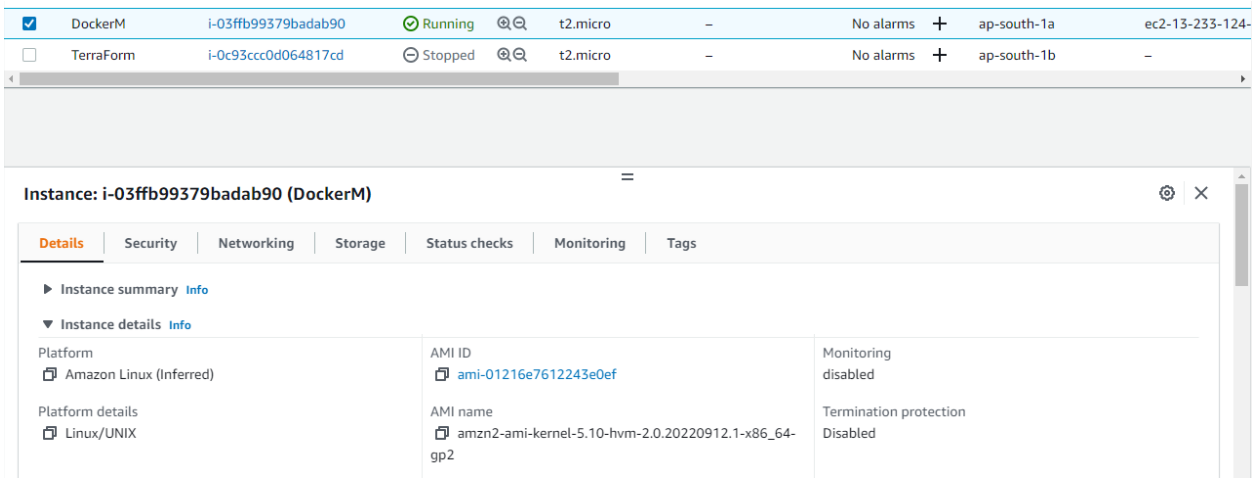


Name: Mrunal Vaidya
Roll : 68

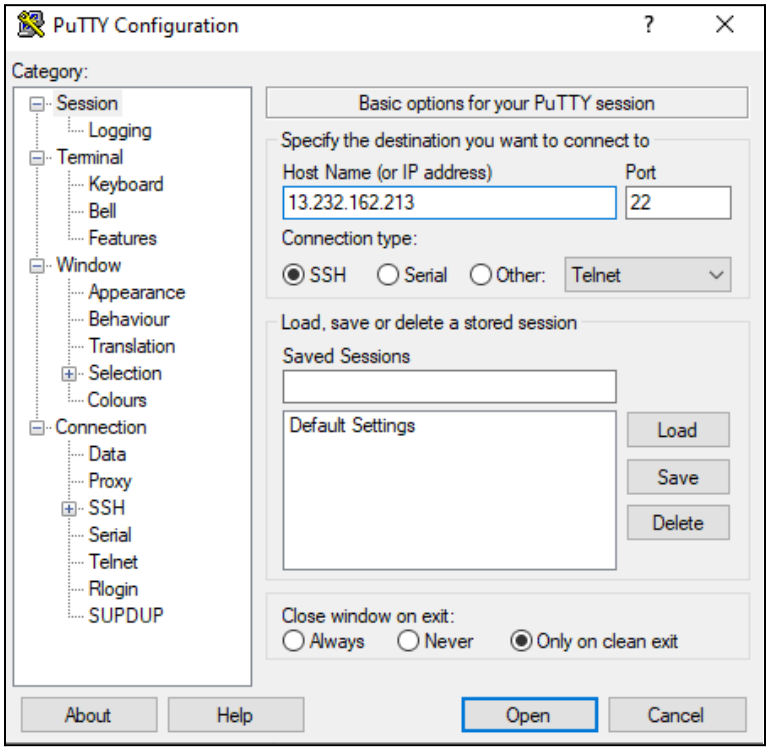
XIE ID: 202003060
Batch: C

3. Go to AWS Management console and Launch a new Instance

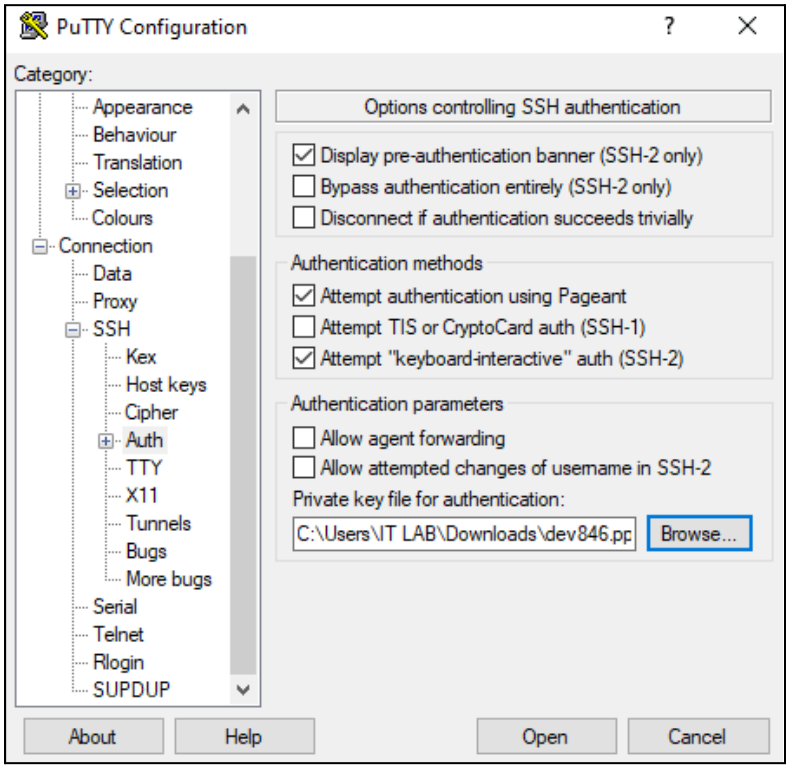




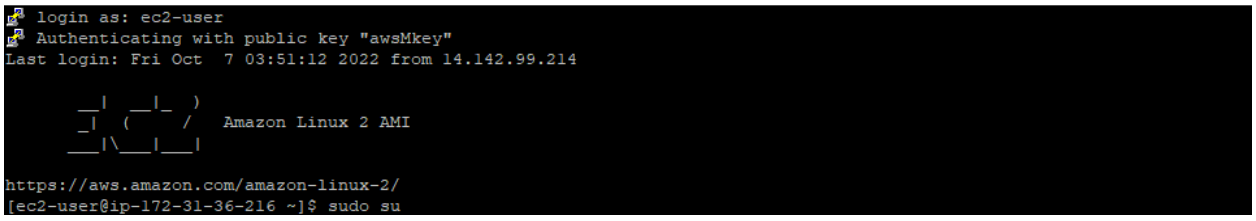
4.Now connect to PuTTY



Select “Auth” and give the path to the .ppk file



Now, execute the commands on the terminal



5.Install Docker

```
[root@ip-172-31-36-216 ec2-user]# amazon-linux-extras install docker -y
Installing docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                          | 3.7 kB     00:00
amzn2extra-docker                                   | 3.0 kB     00:00
amzn2extra-kernel-5.10                              | 3.0 kB     00:00
(1/7): amzn2-core/2/x86_64/group_gz                   | 2.5 kB   00:00
(2/7): amzn2-core/2/x86_64/updateinfo                 | 498 kB   00:00
(3/7): amzn2extra-docker/2/x86_64/updateinfo          | 6.4 kB   00:00
(4/7): amzn2extra-kernel-5.10/2/x86_64/updateinfo     |  18 kB   00:00
(5/7): amzn2extra-docker/2/x86_64/primary_db          |  93 kB   00:00
(6/7): amzn2extra-kernel-5.10/2/x86_64/primary_db     |  11 MB   00:00
(7/7): amzn2-core/2/x86_64/primary_db                 |  65 MB   00:00
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:20.10.17-1.amzn2 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.17-1.amzn2.x8
6_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-20.10.
17-1.amzn2.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.17-1.am
zn2.x86_64
--> Processing Dependency: pigz for package: docker-20.10.17-1.amzn2.x86_64
--> Running transaction check
```

6.Now, Start the Docker Services

```
[root@ip-172-31-36-216 ec2-user]# service docker start
Redirecting to /bin/systemctl start docker.service
```

7.Enable Docker & Check Docker Status

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

```
[root@ip-172-31-36-216 ec2-user]# usermod -a -G docker ec2-user
[root@ip-172-31-36-216 ec2-user]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service
to /usr/lib/systemd/system/docker.service.
[root@ip-172-31-36-216 ec2-user]# service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor prese
t: disabled)
   Active: active (running) since Fri 2022-10-07 04:17:08 UTC; 44s ago
     Docs: https://docs.docker.com
 Main PID: 3588 (dockerd)
   CGroup: /system.slice/docker.service
           └─3588 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont...

Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal systemd[1]: Star...
Oct 07 04:17:08 ip-172-31-36-216.ap-south-1.compute.internal dockerd[3588]: t...
Hint: Some lines were ellipsized, use -l to show in full.
```

8. Check Docker Information

```
[root@ip-172-31-36-216 ec2-user]# docker info
Client:
 Context:    default
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.17
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk s
yslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 10c12954828e7c7c9b6e0ea9b0c02b01407d3ae1
 runc version: 1e7bb5b773162b57333d57f612fd72e3f8612d94
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 5.10.135-122.509.amzn2.x86_64
 Operating System: Amazon Linux 2
 OSType: linux
 Architecture: x86_64
```

Name: Mrunal Vaidya
Roll : 68

XIE ID: 202003060
Batch: C

9. Check Docker Images

```
[root@ip-172-31-36-216 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

10. Pull Images from Ubuntu

```
[root@ip-172-31-36-216 ec2-user]# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
cf92e523b49e: Pull complete
Digest: sha256:35fb073f9e56eb84041b0745cb714eff0f7b225ea9e024f703cab56aaa5c7720
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[root@ip-172-31-36-216 ec2-user]# docker run -it -d ubuntu
c6e7b0a0305d5b4fddd9f45ae732b427c889d1e837e5ccb527ef83d254c6f15d
[root@ip-172-31-36-216 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NA
MES
c6e7b0a0305d   ubuntu    "bash"    17 seconds ago   Up 16 seconds             fe
stive_fermi
```

11. Check for the process

```
[root@ip-172-31-36-216 ec2-user]# docker exec -it c6e7b0a0305d bash
root@c6e7b0a0305d:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@c6e7b0a0305d:/# exit
exit
```

12. Run docker hello-word

```
[root@ip-172-31-36-216 ec2-user]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:62af9efd515a25f84961b70f973a798d2eca956b1b2b026d0a4a63a3b0b6a3f2
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
```

13. Check again docker images

```
[root@ip-172-31-36-216 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
ubuntu        latest    216c552ea5ba   2 days ago      77.8MB
hello-world   latest    feb5d9fea6a5   12 months ago   13.3kB
[root@ip-172-31-36-216 ec2-user]# docker search mysql
NAME                              DESCRIPTION
STARS     OFFICIAL   AUTOMATED
mysql                             MySQL is a widely used, open-source relation…
13234     [OK]
mariadb                           MariaDB Server is a high performing open sou…
5062      [OK]
phpmyadmin                        phpMyAdmin - A web interface for MySQL and M…
640       [OK]
percona                           Percona Server is a fork of the MySQL relati…
588       [OK]
bitnami/mysql                     Bitnami MySQL Docker Image
77                  [OK]
databack/mysql-backup             Back up mysql databases to... anywhere!
70
linuxserver/mysql-workbench
44
linuxserver/mysql                 A Mysql container, brought to you by LinuxSe…
37
ubuntu/mysql                      MySQL open source fast, stable, multi-thread…
36
circleci/mysql                    MySQL is a widely used, open-source relation…
27
google/mysql                      MySQL server for Google Compute Engine
21                  [OK]
rapidfort/mysql                   RapidFort optimized, hardened image for MySQL
13
```

14. Pull alpine image

```
[root@ip-172-31-36-216 ec2-user]# docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
213ec9aee27d: Pull complete
Digest: sha256:bc41182d7ef5ffc53a40b044e725193bc10142a1243f395ee852a8d9730fc2ad
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
[root@ip-172-31-36-216 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
ubuntu        latest    216c552ea5ba   2 days ago      77.8MB
alpine        latest    9c6f07244728   8 weeks ago     5.54MB
hello-world   latest    feb5d9fea6a5   12 months ago   13.3kB
[root@ip-172-31-36-216 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED         STATUS         PORTS     NAME
S
c6e7b0a0305d   ubuntu    "bash"    4 minutes ago   Up 4 minutes             fest
ive_fermi
[root@ip-172-31-36-216 ec2-user]# docker run -it -d alpine
28903f135c010f8a38ec2622af4bdae0ee3af36d8e7fa39ad9d893457e52c037
[root@ip-172-31-36-216 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED         STATUS         PORTS     NA
MES
28903f135c01   alpine    "/bin/sh"   7 seconds ago   Up 5 seconds             in
teresting_hertz
c6e7b0a0305d   ubuntu    "bash"      5 minutes ago   Up 5 minutes             fe
stive_fermi
```

Conclusion:

From the above experiment, we have successfully understood Docker Architecture and Container Life Cycle, installed Docker, and executed Docker commands to manage images and interact with containers. And hence, with this experiment we have achieved the Lab Outcome Five (LO5).

POs Achieved: PO1, PO5, PO12.