Name - Manav Pahilwani
Roll No - 37
Class - D11AD

Experiment 8

Aim - To study and Implement Containerization using Docker

Theory - Containerization is a popular approach to application deployment that allows applications to be packaged and run in a lightweight and isolated environment. Docker is a widely used platform for containerization that provides a set of tools and technologies for building, deploying, and managing containerized applications.

At its core, Docker is based on the concept of a container, which is an isolated environment that encapsulates an application and all of its dependencies. Each container runs its own operating system, providing a level of isolation and security that is not possible with traditional virtual machines. Containers can be easily moved between environments, making them ideal for modern, cloud-based applications that require rapid deployment and scaling.

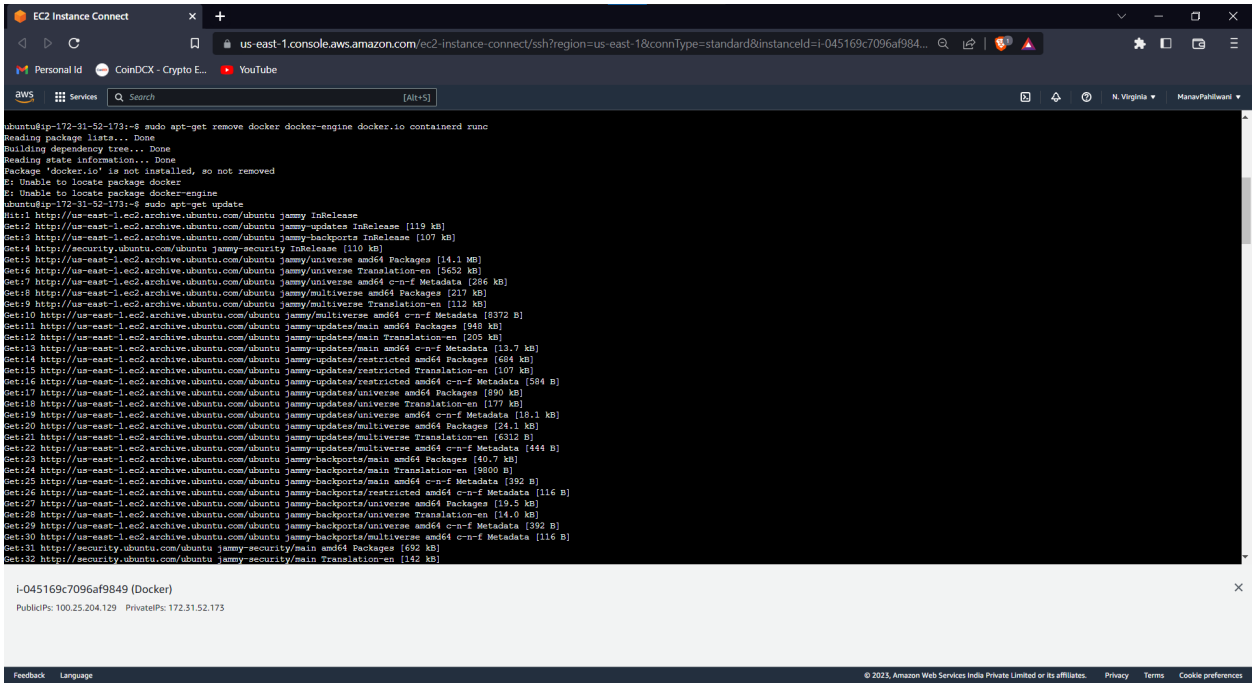
Docker consists of several key components, including the Docker engine, which is responsible for building and running containers, and Docker Hub, which is a central repository for Docker images. Docker images are preconfigured containers that can be used as a starting point for building containerized applications.

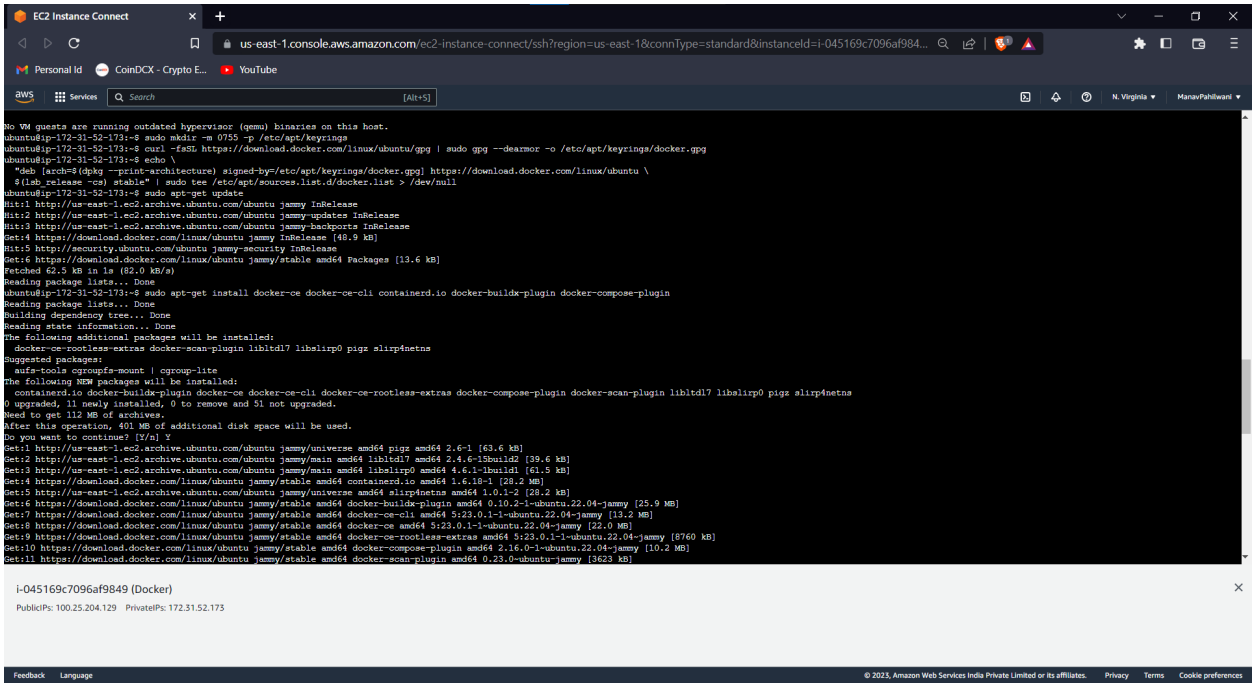
Some key benefits of containerization using Docker include improved application portability, faster deployment times, and better resource utilization. Docker also provides a high degree of flexibility and control, allowing developers to easily manage and scale containerized applications across a variety of environments.

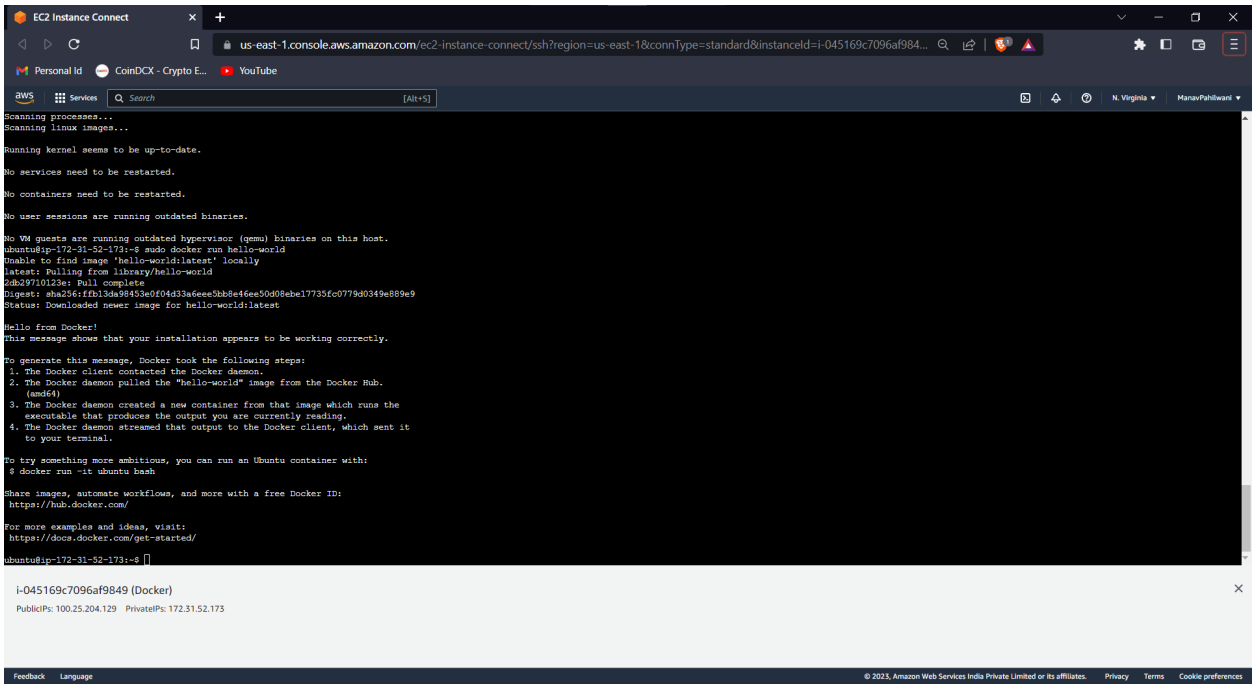
Conclusion - containerization using Docker is a powerful approach to application deployment that provides a range of benefits for modern, cloud-based applications. By leveraging the tools and technologies provided by Docker, developers can build and deploy containerized applications quickly and easily, while also ensuring a high degree of portability, flexibility, and security.

Implementation -

Installing docker







ubuntu@ip-172-31-52-173:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest:
sha256:ffb13da98453e0f04d33a6eee5bb8e46ee50d08ebe17735fc0779d0349e889e9
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/

Exploring Docker Commands :-

```
root@ip-172-31-52-173:/home/ubuntu# docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.10.2
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.16.0
    Path:     /usr/libexec/docker/cli-plugins/docker-compose
  scan: Docker Scan (Docker Inc.)
    Version:  v0.23.0
    Path:     /usr/libexec/docker/cli-plugins/docker-scan

Server:
 Containers: 1
  Running: 0
  Paused: 0
  Stopped: 1
 Images: 1
 Server Version: 23.0.1
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 2
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 runc
 Default Runtime: runc
 Init Binary: docker-init
```

```
root@ip-172-31-52-173:/home/ubuntu# docker version
Client: Docker Engine - Community
 Version:           23.0.1
 API version:       1.42
 Go version:        go1.19.5
 Git commit:        a5ee5b1
 Built:             Thu Feb  9 19:47:01 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          23.0.1
  API version:      1.42 (minimum version 1.12)
  Go version:       go1.19.5
  Git commit:       bc3805a
  Built:            Thu Feb  9 19:47:01 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.18
  GitCommit:        2456e983eb9e37e47538f59ea18f2043c9a73640
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
```

```
ubuntu@ip-172-31-52-173:~$ docker images
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/json": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-52-173:~$ sudo su
root@ip-172-31-52-173:/home/ubuntu# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   18 months ago   13.3kB
root@ip-172-31-52-173:/home/ubuntu#
```

```
root@ip-172-31-52-173:/home/ubuntu# docker ps --all
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
bf0885a26a00   hello-world   "/hello"   23 minutes ago   Exited (0) 23 minutes ago             epic_goldwasser
root@ip-172-31-52-173:/home/ubuntu#
```

```
root@ip-172-31-52-173:/home/ubuntu# docker ps --all
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
bf0885a26a00   hello-world   "/hello"   23 minutes ago   Exited (0) 23 minutes ago             epic_goldwasser
root@ip-172-31-52-173:/home/ubuntu# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
88d153fa1d15   bridge    bridge    local
b88d24b2cbb0   host      host      local
169267e87703   none      null      local
root@ip-172-31-52-173:/home/ubuntu#
```