

Experiment No. 8

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

LO No. & Statement: (LO5): Explains the concept of containerization and Analyzes the Containerization of OS images and deployment of applications over Docker.

Theory:

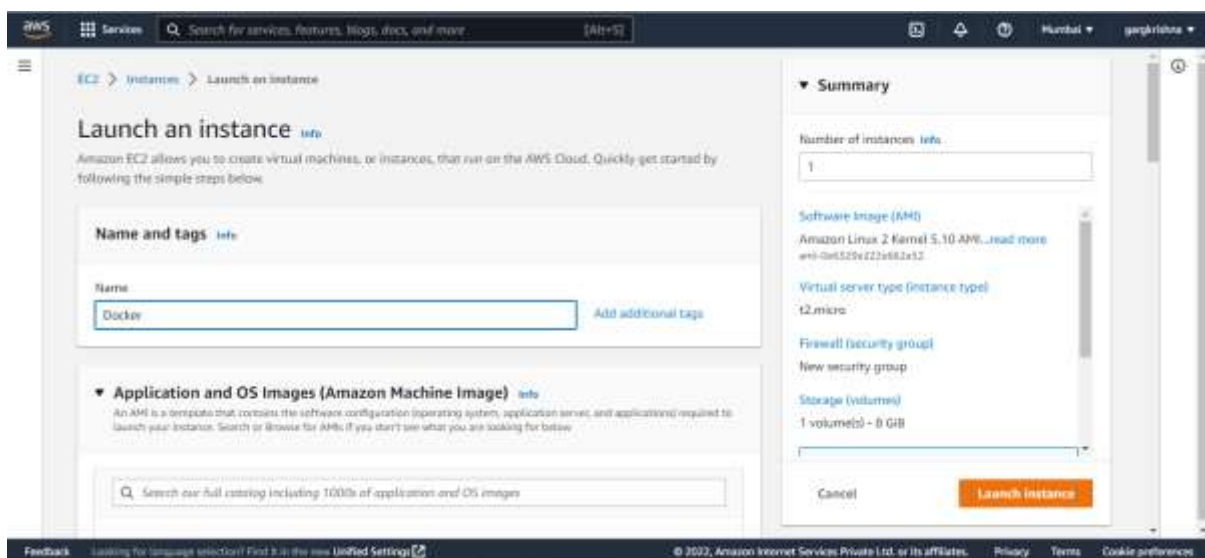
Docker is an open-source centralized platform designed to create, deploy, and run applications. Docker uses containers on the host's operating system to run applications. It allows applications to use the same Linux kernel as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our application works in any environment like development, test, or production.

Docker includes components such as Docker client, Docker server, Docker Machine, Docker hub, Docker composes, etc.

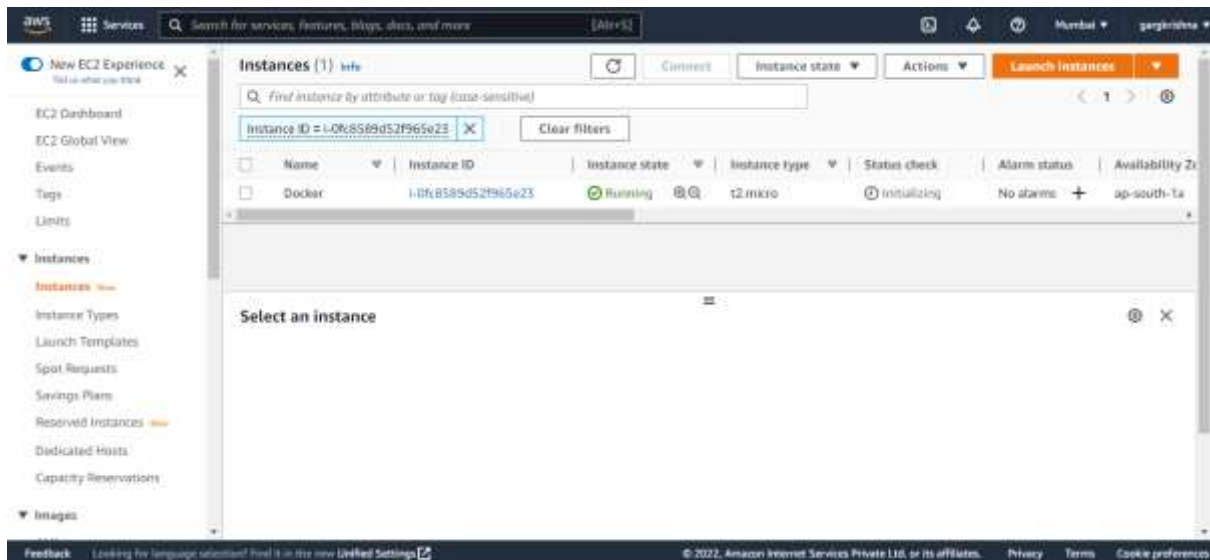
Docker containers are lightweight alternatives to the virtual machine. It allows developers to package up the application with all its libraries and dependencies, and ship it as a single package. The advantage of using a docker container is that you don't need to allocate any RAM and disk space for the applications. It automatically generates storage and space according to the application requirement.

Findings and Requirements:

Launch an Instance



The screenshot shows the AWS Management Console interface for launching an EC2 instance. The page is titled "Launch an instance" and includes a brief description of Amazon EC2. The main configuration area is divided into two columns. The left column contains the "Name and tags" section with a text input field labeled "Name" containing the word "Docker", and the "Application and OS Images (Amazon Machine Image)" section which includes a search bar and a list of AMIs. The right column contains the "Summary" section with a "Number of instances" input field set to "1", and a list of configuration options: "Software Image (AMI)" (Amazon Linux 2 Kernel 5.10 AMI), "Virtual server type (instance type)" (t2.micro), "Firewall (security group)" (New security group), and "Storage (volumes)" (1 volume(s) - 8 GiB). At the bottom of the right column are "Cancel" and "Launch instance" buttons. The footer of the console shows the AWS logo, a search bar, and various navigation links.



Connect the Instance:

```

login as: ec2-user
Authenticating with public key "Krish"

  _| _|_ )
  _| ( _| /   Amazon Linux 2 AMI
 _| \ _| _|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-27-180 ~]$ sudo su
[root@ip-172-31-27-180 ec2-user]# yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
[root@ip-172-31-27-180 ec2-user]# sudo 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

```

```

Installing : runc-1.1.3-1.amzn2.x86_64                1/5
Installing : containerd-1.6.6-1.amzn2.x86_64          2/5
Installing : libcgrou-0.41-21.amzn2.x86_64            3/5
Installing : pigz-2.3.4-1.amzn2.0.1.x86_64            4/5
Installing : docker-20.10.17-1.amzn2.x86_64           5/5
Verifying  : docker-20.10.17-1.amzn2.x86_64           1/5
Verifying  : runc-1.1.3-1.amzn2.x86_64                2/5
Verifying  : pigz-2.3.4-1.amzn2.0.1.x86_64            3/5
Verifying  : containerd-1.6.6-1.amzn2.x86_64          4/5
Verifying  : libcgrou-0.41-21.amzn2.x86_64           5/5

Installed:
  docker.x86_64 0:20.10.17-1.amzn2

Dependency Installed:
  containerd.x86_64 0:1.6.6-1.amzn2      libcgrou.x86_64 0:0.41-21.amzn2
  pigz.x86_64 0:2.3.4-1.amzn2.0.1      runc.x86_64 0:1.1.3-1.amzn2

Complete!

```

```
[root@ip-172-31-27-180 ec2-user]# sudo service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-27-180 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; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-10-04 08:34:38 UTC; 1min 50s ago
     Docs: https://docs.docker.com
   Process: 3470 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Process: 3469 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
```

```
[root@ip-172-31-27-180 ec2-user]# usermod -s -G docker ec2-user
[root@ip-172-31-27-180 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-172-31-27-180 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 syslog
 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
 CPUs: 1
 Total Memory: 965.8MiB
 Name: ip-172-31-27-180.ec2.internal
 ID: 3H5V:7T4Q:SGOQ:YQJB:AENF:DOMT:2RCP:Y5JB:B7I6:WAGT:BFMD:3266
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
```

Conclusion:

In this experiment, Docker was studied, understood, and implemented. Docker is an open-source centralized platform to create, deploy, and run applications that use containers on the host's operating system.

We have achieved LO5 from this experiment.

We have also achieved Program Outcomes PO1, PO2, PO3, PO4, PO5, PO12.