



TCS iON RIO-125 Internship Report

Internship : TCS iON RIO-125: Secured Docker Based Lab: Enforcing end-to-end security

Name	Harsh Menaria
Mail	harshmenaria114@gmail.com
Starting Date	21/07/2025
Submission Date	30/07/2025
Industry Mentor	NA

Project Brief

Objective

To create an infrastructure to develop highly secure docker based lab environment for running different applications in a secure way and protected manner that confirms security at different levels

Context

Required to create a cloud based infra to dev a secure docker based lab env for app that seals vulnerability of all types

Background

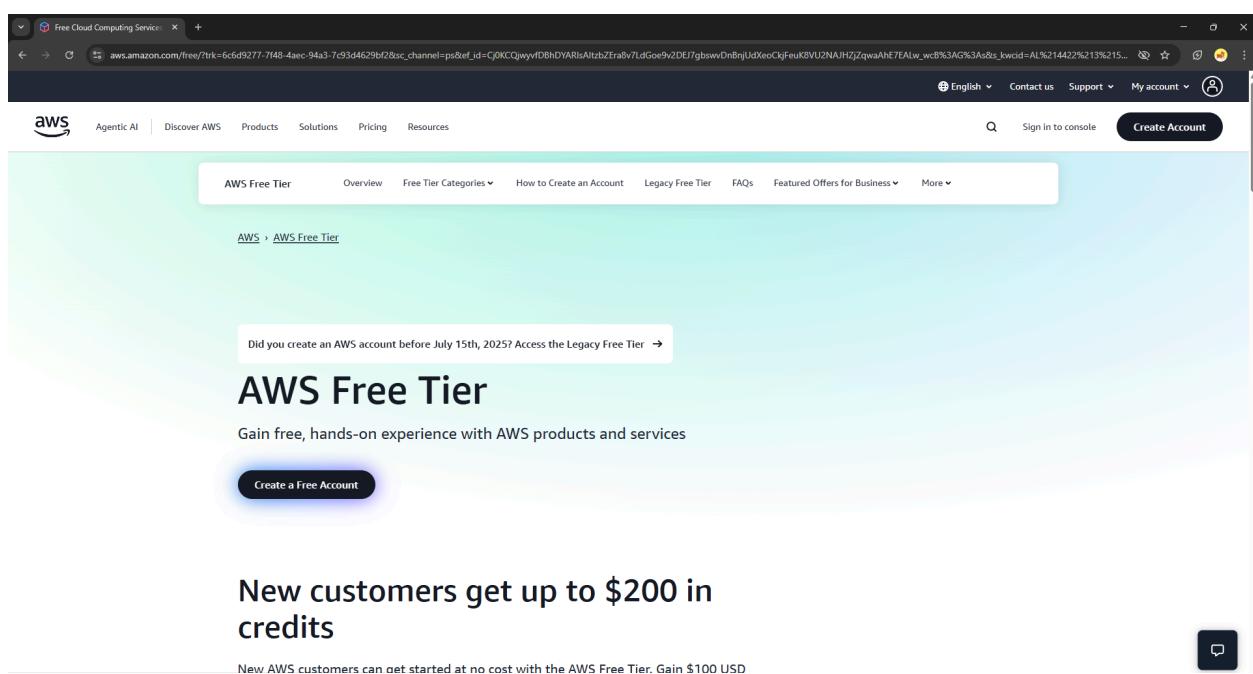
require the technical know-how of cloud based sec and docker component along with security enforcement and testing at different levels

Technologies Involved

1. AWS Cloud instances
2. Docker and its components
3. Nmap , Metasploit, hydra and john

Procedure

1. Creating an AWS Cloud Account
 - a. Visit [aws.amazon.com](https://aws.amazon.com/free/) and create a free tier account



- b. AWS dashboard will look something like this

The screenshot shows the AWS Console Home page. On the left, there's a 'Recently visited' section with links to IAM and Billing and Cost Management. Below it is a 'Welcome to AWS' section with links to Getting started with AWS and Training and certification. To the right, there are three main widgets: 'Applications' (0), 'AWS Health' (Info), and 'Cost and usage'. The 'Applications' widget shows a message to 'Get started by creating an application.' The 'AWS Health' widget shows 0 open issues and 0 scheduled changes. The 'Cost and usage' widget shows credits remaining of \$100.00 USD and 184 days remaining until Jan 19, 2026.

2. Visit EC2 instance page, dashboard will look something like this

The screenshot shows the AWS EC2 Instances page. The left sidebar includes sections for EC2 (Dashboard, EC2 Global View, Events), Instances (Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and Load Balancing. The main content area displays a table for instances, with a search bar at the top. A message states 'No instances' and 'You do not have any instances in this region'. A 'Launch instances' button is located at the bottom of the table.

a. To create an EC2 Instance , change region to location nearest to you

b. In this case that would be Asia Pacific Mumbai

The screenshot shows the AWS EC2 'Launch an instance' wizard. The 'Name and tags' section has 'Securedub' entered in the Name field. The 'Application and OS Images (Amazon Machine Image)' section shows 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' selected from the catalog. Configuration details include 'Image ID: ami-0f518f7e67a3323f0', 'Username: ubuntu', and 'Root device type: ENA Enabled'. On the right, the 'Summary' panel shows 1 instance being launched with the 'Software Image (AMI)' set to 'Ubuntu Server 24.04 LTS (HVM)'. The 'Virtual server type (instance type)' is 't3.micro', and the 'Firewall (security group)' is 'New security group'. A large orange 'Launch instance' button is prominently displayed.

c. Create an Instance with these config

Ubuntu OS 24.04 amd64

Virtual Server Type : t3.micro

Volume : 8GB

Download a *.pem key , necessary for SSH session which will be conducted later

Enable Traffic from anywhere on Internet

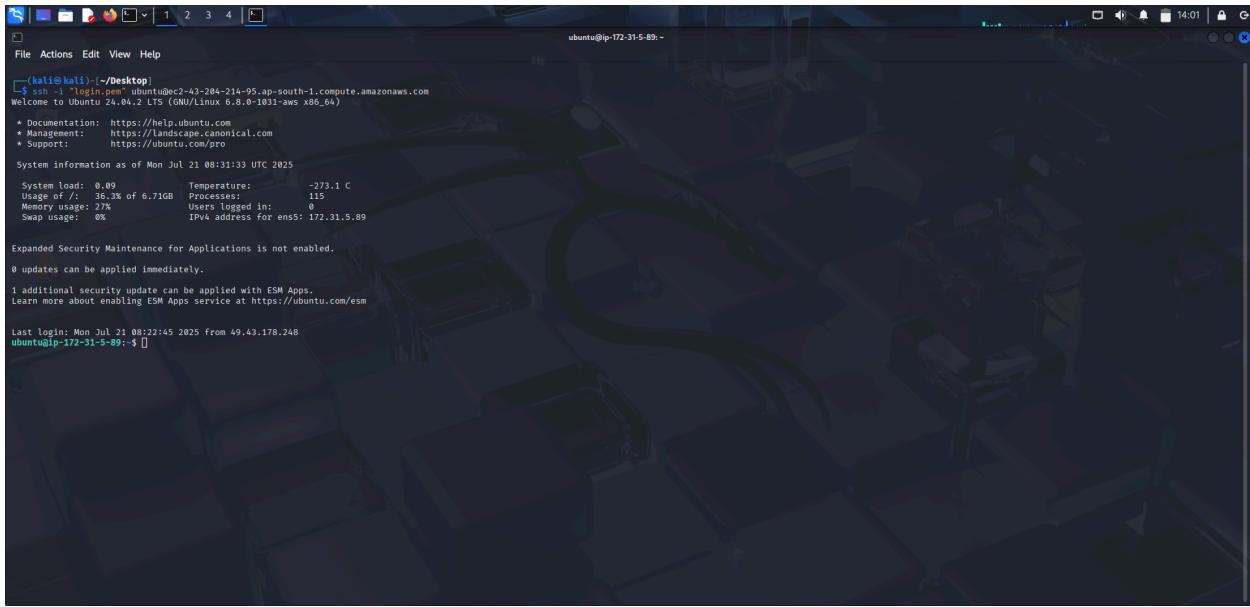
d. Launch the instance and wait for it to initialize and run

Screenshot of the AWS EC2 Instances page in a browser. The instance 'Securedlab' (i-04917b1fb44013406) is listed as running, t3.micro, with 3/3 checks passed. The left sidebar shows navigation options for EC2, Images, Elastic Block Store, Network & Security, and Load Balancing.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
Securedlab	i-04917b1fb44013406	Running	t3.micro	3/3 checks passed	View alarms +	ap-south-1b	ec2-45-204-214-95.ap...	45.204.214.95	-

3. Connecting to EC2 instance with VM , Using the *.pem key that we downloaded earlier from EC2 instance we'll log in via SSH





```
[kali㉿kali: ~] /Desktop
└─$ ssh -l "login.aws" ubuntu@ec2-43-204-214-95.ap-south-1.compute.amazonaws.com
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1031-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Mon Jul 21 08:31:33 UTC 2025

System load: 0.09 Temperature:          -273.1 C
Usage of /: 36.3% of 6.71GB Processes:           115
Memory usage: 0% Users logged in:      0
Swap usage: 0% IPv4 address for ens5: 172.31.5.89

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

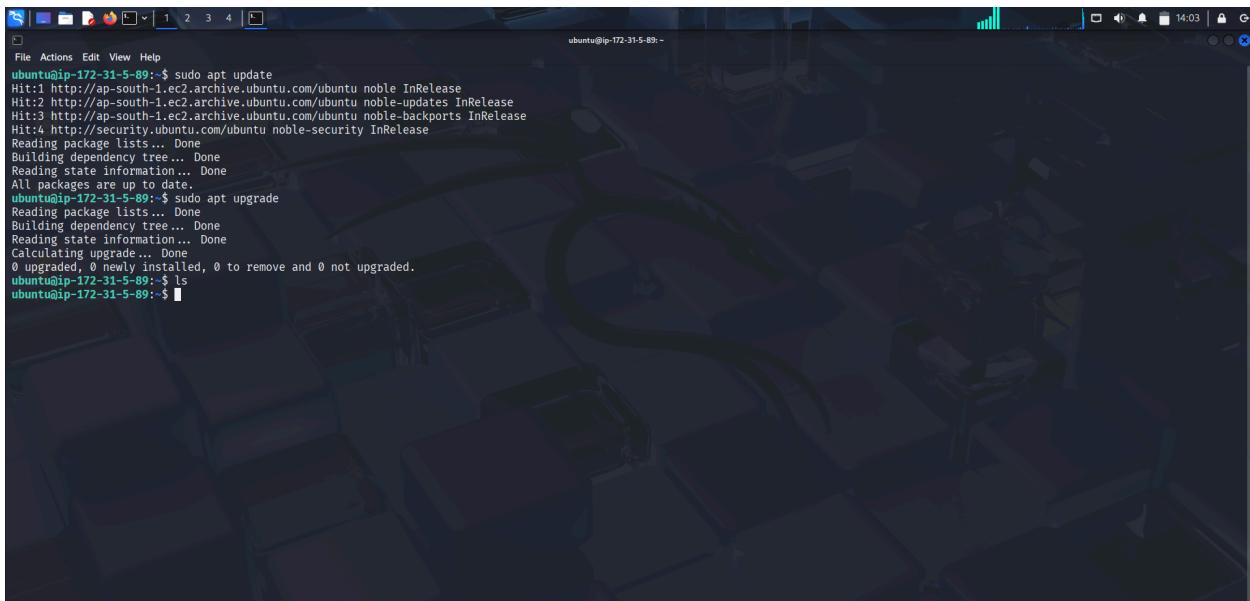
1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Mon Jul 21 08:22:45 2025 From 49.43.178.248
ubuntu@ip-172-31-5-89: ~
```

4. Once logged in , run this

```
sudo apt update -y
sudo apt upgrade -y
```

this will update and upgrade our Ubuntu OS in instance with latest packages and modules



```
[kali㉿kali: ~] /Desktop
└─$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
└─$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
└─$ ls
└─$
```

6. Installing tools required for penetrating testing and vulnerability assessment

```
ubuntu@ip-172-31-5-89:~$ sudo apt update && sudo apt install nmap hydra john traceroute
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
firebird3.0-common firebird3.0-common-doc fontconfig-config fonts-dejavu-core fonts-dejavu-mono john-data libapr1t64 libaprutil1t64 libblas3 libbson-1.0-0t64 libcairo2 libfbclient2 libfontconfig1 libfreerdp2-2t64 libgomp1 libhashkit2t64 libidn12 libjpeg-turbo8 libjpeg8 liblinear4 liblau5-4-0 libmemcached1t64 libmongoc-1.0-0t64 libmongocrypt0 libmysqlclient21 libpixman-1-0 libpq5 libserf-1-1 libsnappy1v5 libssh2-1t64 libsvni libtommath1 libutf8proc3 libwinpr2-2t64 libxcb-render0 libxcb-shm0 libxkbfile1 libxrender1 mysql-common nmap-common
Suggested packages:
hydra-gtk wodlist freerdp2-x11 liblinear-tools liblinear-dev ncdu nndiff zmap
The following NEW packages will be installed:
firebird3.0-common firebird3.0-common-doc fontconfig-config fonts-dejavu-core fonts-dejavu-mono hydra john john-data libapr1t64 libaprutil1t64 libblas3 libbson-1.0-0t64 libcairo2 libfbclient2 libfontconfig1 libfreerdp2-2t64 libgomp1 libhashkit2t64 libidn12 libjpeg-turbo8 libjpeg8 liblinear4 liblau5-4-0 libmemcached1t64 libmongoc-1.0-0t64 libmongocrypt0 libmysqlclient21 libpixman-1-0 libpq5 libserf-1-1 libsnappy1v5 libssh2-1t64 libsvni libtommath1 libutf8proc3 libwinpr2-2t64 libxcb-render0 libxcb-shm0 libxkbfile1 libxrender1
mysql-common nmap-common traceroute
0 upgraded, 44 newly installed, 0 to remove and 0 not upgraded.
Need to get 25.5 MB of archives.
After this operation, 79.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 firebird3.0-common-doc all 3.0.11.33703.ds~ubuntu2 [28,5 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 firebird3.0-common all 3.0.11.33703.ds~ubuntu2 [15.5 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-mono all 2.37-8 [502 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-core all 2.37-8 [835 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 fontconfig-config amd64 2.19.0-1.1ubuntu2 [37,3 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 john-data all 1.9.0-2build1 [9141 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libgomp1 amd64 14.2.0-4ubuntu2-24.04 [148 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 john amd64 1.9.0-2build1 [210 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libapr1t64 amd64 1.7.2-3.1ubuntu0.1 [108 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu0 [91,9 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libblass amd64 3.12.0-3build1.1 [238 kB]
```

7. After successfully updating and upgrading our packages , its time to install docker

```
sudo apt-get install docker.io -y  
sudo systemctl start docker
```

```
File Actions Edit View Help
ubuntu@ip-172-31-5-89:~$ sudo apt-get install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (27.5.1-0ubuntu3~24.04.2).
The following packages were automatically installed and are no longer required:
  libslirp0 python3-compose python3-docker python3-dockerty python3-dotenv python3-exectable python3-websocket slirp4netns
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ip-172-31-5-89:~$ sudo docker --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
ubuntu@ip-172-31-5-89:~$ sudo docker run hello-world

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/
ubuntu@ip-172-31-5-89:~$
```

a. verify installation

```
sudo docker run hello-world
```

```

ubuntu@ip-172-31-5-89:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
    Active: active (running) since Tue 2025-07-22 06:06:00 UTC; 50min ago
      TriggeredBy: docker.socket
        Docs: https://docs.docker.com
  Main PID: 3754 (dockerd)
     Tasks: 12
       CPU: 5.179s
      Memory: 152.8M (peak: 232.1M)
         CPU: 5.179s
   CGroup: /system.slice/docker.service
           └─3754 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jul 22 06:24:44 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:24:44.370259878Z" level=error msg="Not continuing with pull after error: manifest unknown: manifest unknown"
Jul 22 06:30:21 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:30:21.556816118Z" level=error msg="Not continuing with pull after error: errors:\ndenied: requested access to the resource\n"
Jul 22 06:30:21 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:30:21.556854324Z" level=info msg="Ignoring extra error returned from registry: error=unauthorized: authentication required"
Jul 22 06:30:40 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:30:40.800056882Z" level=error msg="Not continuing with pull after error: errors:\ndenied: requested access to the resource\n"
Jul 22 06:30:40 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:30:40.800056882Z" level=info msg="Ignoring extra error returned from registry: error=unauthorized: authentication required"
Jul 22 06:33:34 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:33:34.602407152Z" level=info msg="Ignoring event" container=f18f1160576a27d52185755d960fbcc367604f977c7ba74d36b93ceecd5
Jul 22 06:33:34 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:33:34.677246608Z" level=warning msg="Failed to close stdin: NotFound: task f18f1160576a27d52185755d960fbcc367604f977c7ba74d36b93ceecd5
Jul 22 06:36:17 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:36:17.295554332Z" level=info msg="Ignoring event" container=f18f1160576a27d52185755d960fbcc367604f977c7ba74d36b93ceecd5
Jul 22 06:36:17 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:36:17.310406913Z" level=warning msg="Failed to close stdin: NotFound: task f18f1160576a27d52185755d960fbcc367604f977c7ba74d36b93ceecd5
Jul 22 06:48:57 ip-172-31-5-89 dockerd[3754]: time="2025-07-22T06:48:57.311576204Z" level=info msg="Ignoring event" container=907bb991cf9e49ae3ca5fc9f2ff4c26128bd8670f0824c1983d5c938dbc0ad
lines: 1-22/22 (END)
ubuntu@ip-172-31-5-89:~$
```

b. enable docker service to start automatically when instance starts

```
sudo systemctl enable docker
```

c. version check

```
docker --version
```

d. adding user to docker group , so as to use docker command without sudo

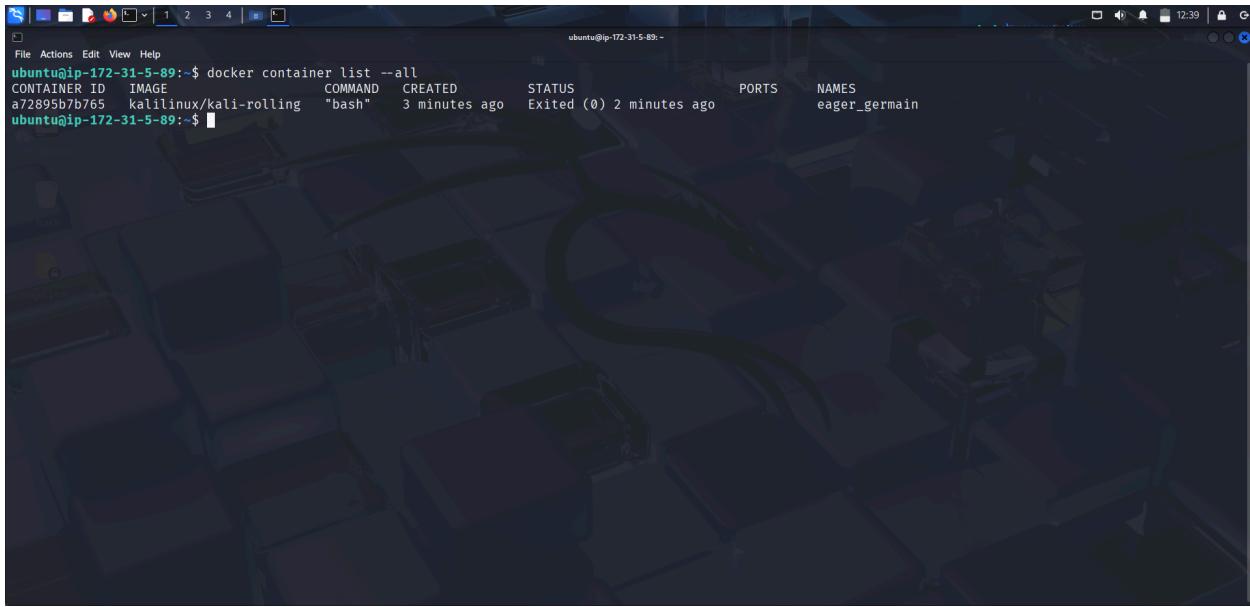
```
sudo usermod -a -G docker $(whoami)
newgrp docker
```

e. restart instance if necessary

```
sudo reboot
```

f. Listing all the containers installed in docker

```
docker container list --all
```



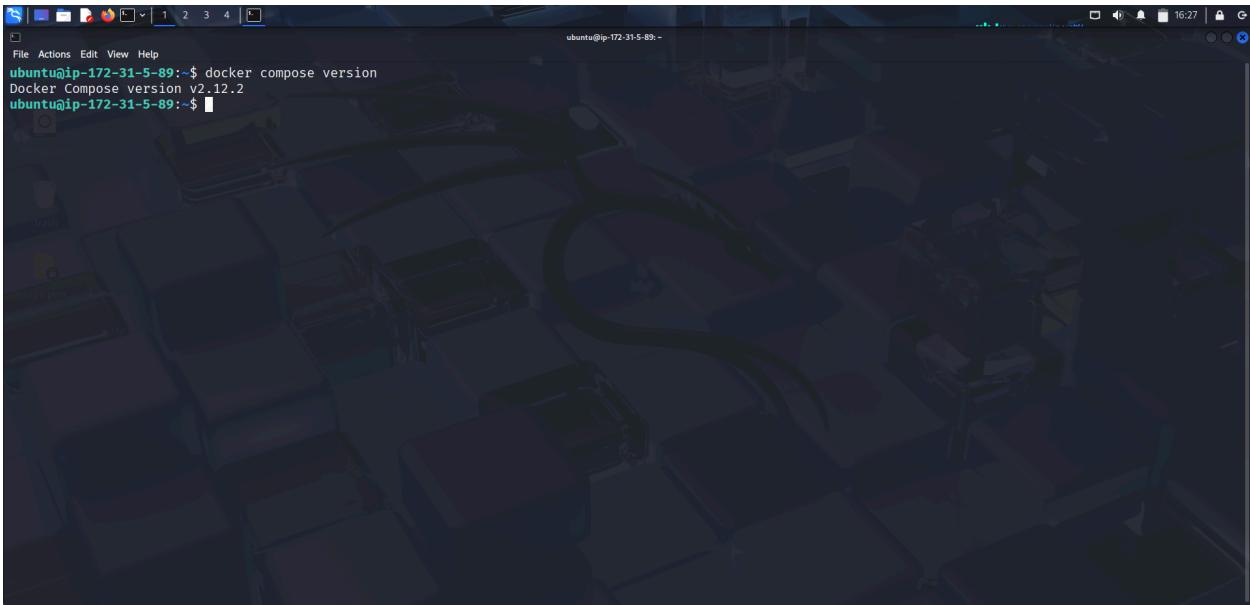
```
ubuntu@ip-172-31-5-89:~$ docker container list --all
CONTAINER ID   IMAGE          COMMAND   CREATED      STATUS      PORTS     NAMES
a72395b7b765   kalilinux/kali-rolling   "bash"    3 minutes ago   Exited (0) 2 minutes ago
ubuntu@ip-172-31-5-89:~$
```

8. Installing docker-compose and its components in instance

```
DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
mkdir -p $DOCKER_CONFIG/cli-plugins
curl -SL https://github.com/docker/compose/releases/download/v2.29.6/docker-compose-linux-x86_64
chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose

# test installation
docker compose version

# if it doesn't install updated version
sudo apt-get update
sudo apt install docker-compose
sudo apt install docker-compose-plugins
```



9. Created a .yaml file for docker-compose in project directory on ec2 instance with Ubuntu OS

- a. file contains
 - i. Ubuntu as base os
 - ii. a configured private network with subnet
 - iii. python and its components
 - iv. java environment
 - v. web-server with static ip address
 - vi. MySQL , Static IP and open ports as required
 - vii. Open ports for services to be exposed to external env

```
version: "3.8"

services:
  ubuntu_base:
    image: ubuntu:20.04
    container_name: ubuntu_base_container
    command: bash -c "apt update && apt install -y curl net-tools iproute2 && tail -f /dev/null"
    networks:
      - mynet

  python_env:
    image: ubuntu:20.04
```

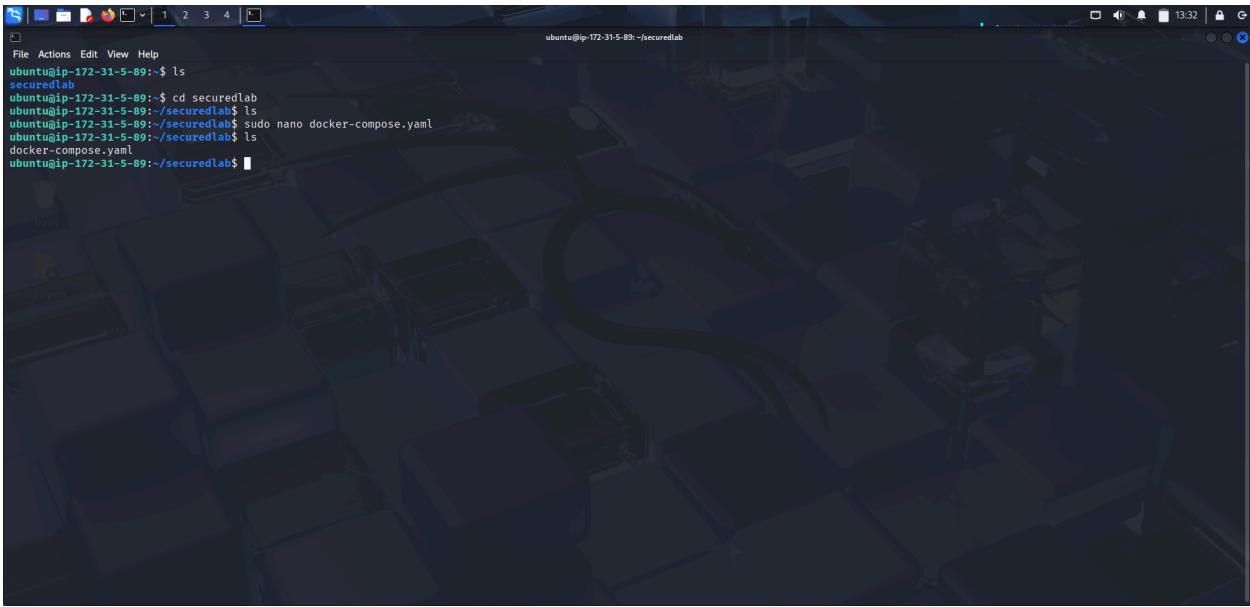
```
container_name: python_container
command: bash -c "apt update && apt install -y python3 python3-pip && tail -f /dev/null"
ports:
- "5000:5000"
networks:
- mynet

java_env:
image: ubuntu:20.04
container_name: java_container
command: bash -c "apt update && apt install -y default-jdk && tail -f /dev/null"
ports:
- "8081:8080"
networks:
- mynet

web_server:
image: nginx:alpine
container_name: web_container
ports:
- "8080:80"
networks:
- mynet

mysql_db:
image: mysql:5.7
container_name: mysql_container
environment:
  MYSQL_ROOT_PASSWORD: rootpass
  MYSQL_DATABASE: demo_db
ports:
- "3306:3306"
networks:
- mynet

networks:
mynet:
  driver: bridge
```



```
File Actions Edit View Help
ubuntu@ip-172-31-5-89:~$ ls
secureLab
ubuntu@ip-172-31-5-89:~$ cd secureLab
ubuntu@ip-172-31-5-89:~/secureLab$ ls
ubuntu@ip-172-31-5-89:~/secureLab$ sudo nano docker-compose.yaml
ubuntu@ip-172-31-5-89:~/secureLab$ ls
docker-compose.yaml
ubuntu@ip-172-31-5-89:~/secureLab$
```

10. Created a directory and made yaml file with the above code

11. Made 5 separate container from one yaml file , listed below are few useful cmds

```
docker compose up -d # to start building group of containers listed in yaml file
docker compose down # to stop the entire group of containers listed in yaml file
docker compose kill # to kill the group of container listed in yaml file
docker rm <container id> # to remove specific container
```

12. executed docker compose up -d and images got pulled and was able to successfully access bash shell of containers

```

ubuntu@ip-172-31-5-89:~/securedlab$ docker container list --all
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
47a74e4sec7ff/   ubuntu:20.04      "bash -c 'apt update..."  16 minutes ago   Up 16 minutes          0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
43adfe1f2ba18/   mysql:5.7        "docker-entrypoint.s..."  16 minutes ago   Up 16 minutes          0.0.0.0:8081->8080/tcp, [::]:8081->8080/tcp
daa01f948de/     ubuntu:20.04      "bash -c 'apt update..."  16 minutes ago   Up 16 minutes          0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
68ef80646ebc/    nginx:alpine     "/docker-entrypoint..."  16 minutes ago   Up 16 minutes          0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
6565de3cela2/    ubuntu:20.04      "bash -c 'apt update..."  16 minutes ago   Up 16 minutes          0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
ubuntu@ip-172-31-5-89:~/securedlab$ docker exec -it ubuntu_base_container bash
root@7a47e4sec7ff:/# "
root@7a47e4sec7ff:/# exit
exit
ubuntu@ip-172-31-5-89:~/securedlab$ docker exec -it python_container bash
root@6565de3cela2:/# exit
exit
ubuntu@ip-172-31-5-89:~/securedlab$ docker exec -it java_container bash
root@daa01f948de:/# exit
exit
ubuntu@ip-172-31-5-89:~/securedlab$ docker exec -it mysql_container bash
bash-4.2# exit
exit
ubuntu@ip-172-31-5-89:~/securedlab$ "

```

13. Since java and python servers don't have default page on browser , only nginx server returned with response

```

ubuntu@ip-172-31-5-89:~/securedlab$ curl http://13.127.55.160:8080/
<!DOCTYPE html>
<html>
<head>
<t&t>Welcome to nginx!</t&t>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p><a href="http://nginx.org/">nginx.org</a> or <a href="http://nginx.com/">nginx.com</a></p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
ubuntu@ip-172-31-5-89:~/securedlab$ "

```

14. Make sure to add ports in inbound rules , EC2 instance → Instance running → details → security → security groups → select the group → inbound rules → edit inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0e47cbfc277623a5	HTTP	TCP	80	Custom	<input type="text"/> 0.0.0.0/0
sgr-067db7d2179009f2c	Custom TCP	TCP	8080	Custom	<input type="text"/> 0.0.0.0/0
sgr-0a587f428c7f4c2c9	MySQL/Aurora	TCP	3306	Custom	<input type="text"/> 0.0.0.0/0
sgr-0c912d5f81a3b3473	Custom TCP	TCP	5000	Custom	<input type="text"/> 0.0.0.0/0
sgr-0020ee64e9bbbc95e	Custom TCP	TCP	8081	Custom	<input type="text"/> 0.0.0.0/0
sgr-03b821125232ceb28	SSH	TCP	22	Custom	<input type="text"/> 0.0.0.0/0
sgr-0e74c6cb524fe3c39	HTTPS	TCP	443	Custom	<input type="text"/> 0.0.0.0/0

[Add rule](#) [Cancel](#) [Preview changes](#) [Save rules](#)

15. Configure awscli

- made a IAM user "securedlab_user" gave it permission "AmazonEC2ReadOnlyAccess"
- downloaded access key and id
- configured it in ec2 instance using aws configure
- described the instance using :

```
$ aws ec2 describe-instances --instance-ids i-04917b1fb44013406
```

16. Pulled linuxserver/wireshark image in docker

```
File Actions Edit View Help
ubuntu@ip-172-31-5-89:~$ docker rm wireshark_test
docker run --name wireshark_test -it linuxserver/wireshark /bin/bash
wireshark_test
[migrations] started
[migrations] no migrations found
usermod: no changes

LSIO
Brought to you by linuxserver.io

To support LSIO projects visit:
https://www.linuxserver.io/donate/

GID/UID

User UID: 911
User GID: 911

.
.
.

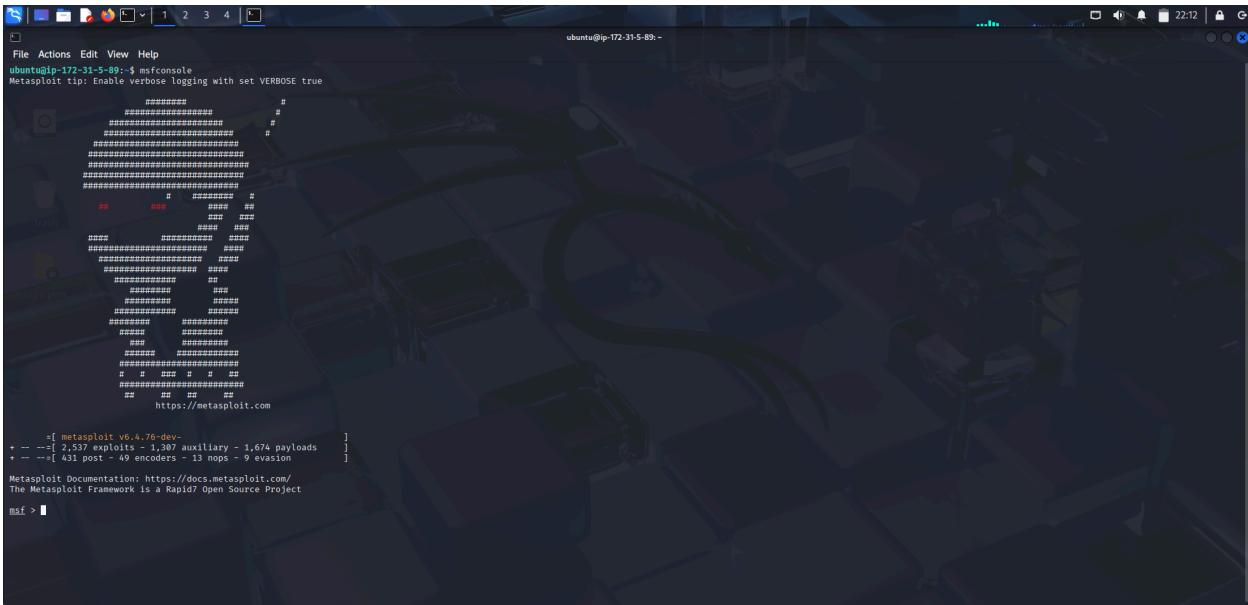
[custom-init] No custom files found, skipping ...
crond[221]: crond (busybox 1.37.0) started, log level 5
crond[221]: userroot entry:@15 * * * * run-parts /etc/periodic/15min
crond[221]: userroot entry:@1 * * * * run-parts /etc/periodic/hourly
crond[221]: userroot entry:@ 2 * * * run-parts /etc/periodic/daily
crond[221]: userroot entry:@ 3 * * 6 run-parts /etc/periodic/weekly
crond[221]: userroot entry:@ 5 1 * * run-parts /etc/periodic/monthly
[3]:1>
[3]:1>ls
[3]:1>cd /tmp
[3]:1>ls
[3]:1>rm -rf x11-socket
[3]:1>mkfifo /tmp/X11-unix
[3]:1>XSERVTransmkdir: ERROR: euid ≠ 0.directory /tmp/X11-unix will not be created.
screen 0 shield 0
INFO: data_websocket:pmflux library found. Audio capture is available.
INFO: data_websocket:pixelflux library found. Striped encoding modes available.
INFO:root:Expected C js_config_t size (from ctype*) 135A bytes
INFO:main:upload directory ensured: /config/Desktop
```

```
docker rm wireshark_test  
docker run --name wireshark_test -it linuxserver/wireshark /bin/bash
```

Vulnerability Scanning

As part of the final assessment of the **Secured Docker-based Lab Environment**, a controlled vulnerability scanning and penetration testing exercise was carried out. The objective was to simulate common attack vectors against the deployed **MySQL service** within the containerized environment hosted on an AWS EC2 instance.

1. Downloaded Metasploit Framework



The screenshot shows a terminal window titled "ubuntu@ip-172-31-5-89: ~" running the Metasploit Framework's msfconsole. The console displays various system and framework information, including the version (msf v6.4.7-dev), exploit count (1,537), auxiliary count (1,087), payloads count (1,674), and evasion encoders count (411). It also shows the URL for the Metasploit documentation (<https://docs.metasploit.com>). The prompt "msf >" is visible at the bottom.

1. Nmap scan

```
$ nmap -sC -sV -Pn 43.204.148.126
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-26 16:24 UTC
Nmap scan report for ec2-43-204-148-126.ap-south-1.compute.amazonaws.com (43.204.1
Host is up (0.00030s latency).

Not shown: 993 filtered tcp ports (no-response)

PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 9.6p1 Ubuntu 3ubuntu13.12 (Ubuntu Linux; protocol 2.0
| ssh-hostkey:
|   256 43:76:21:14:fc:6a:a0:60:30:f7:5c:ca:e7:e8:c2:66 (ECDSA)
|   256 06:eb:b8:22:fe:3c:72:ac:d7:1b:cf:21:44:b3:49:d2 (ED25519)
80/tcp    closed http
443/tcp   closed https
3306/tcp  open  mysql        MySQL 5.7.44
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=MySQL_Server_5.7.44_Auto_Generated_Server_Certificat
| Not valid before: 2025-07-26T08:57:19
|_Not valid after: 2035-07-24T08:57:19
| mysql-info:
|   Protocol: 10
|   Version: 5.7.44
|   Thread ID: 5
|   Capabilities flags: 65535
|   Some Capabilities: Support41Auth, LongPassword, Speaks41ProtocolOld, Speaks41Proto
|   Status: Autocommit
```

```

| Salt: &\x1E\x034Rc*\x01&
| \x12\x1A\x0F-K7s\x1D%=
|_ Auth Plugin Name: mysql_native_password
5000/tcp closed upnp
8080/tcp open http nginx 1.29.0
|_http-title: Welcome to nginx!
|_http-server-header: nginx/1.29.0
|_http-open-proxy: Proxy might be redirecting requests
8081/tcp closed blackice-icecap
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit
Nmap done: 1 IP address (1 host up) scanned in 16.19 seconds

```

b. Metasploit framework vulnerability scanning

MySQL

```

msf > search type:auxiliary mysql

Matching Modules
=====
#  Name                               Disclosure Date Rank Check Description
-  ---
0  auxiliary/server/capture/mysql      .           normal No   Authentication Capt
1  auxiliary/gather/joomla_weblinks_sqli 2014-03-02  normal Yes  Joomla web
2  auxiliary/scanner/mysql/mysql_writable_dirs .           normal No   MYSQL Direct
3  auxiliary/scanner/mysql/mysql_file_enum .           normal No   MYSQL File/Di
4  auxiliary/scanner/mysql/mysql_hashdump .           normal No   MYSQL Passw
5  auxiliary/scanner/mysql/mysql_schemadump .           normal No   MYSQL Sch
6  auxiliary/admin/http/manageengine_pmp_privesc 2014-11-08  normal Yes  Mana
7  auxiliary/scanner/mysql/mysql_authbypass_hashdump 2012-06-09  normal No   M
8  auxiliary/admin/mysql/mysql_enum       .           normal No   MySQL Enumera
9  auxiliary/scanner/mysql/mysql_login    .           normal No   MySQL Login Util
10 auxiliary/admin/mysql/mysql_sql       .           normal No   MySQL SQL Gener
11 auxiliary/scanner/mysql/mysql_version .           normal No   MySQL Server V
12 auxiliary/analyze/crack_databases   .           normal No   Password Cracke
13  \_action: hashcat                 .           .           Use Hashcat
14  \_action: john                   .           .           Use John the Ripper
15 auxiliary/admin/http/rails_devise_pass_reset 2013-01-28  normal No   Ruby on I

```

16 auxiliary/admin/tikiwiki/tikidblib 2006-11-01 normal No TikiWiki Informa

Interact with a module by name or index. For example info 16, use 16 or use auxiliary/admin/t

there are many modules available for configured services on docker in form of containers like mysql, python, java, and webserver

Several modules from the Metasploit Framework were used to simulate real-world attacks against the MySQL service:

- `mysql_version`
- `mysql_login`
- `mysql_sql`
- `exploit/linux/mysql/mysql_udf_payload`

Results:

- The `mysql_login` module failed due to incorrect credentials and account lockout mechanisms.
- Exploits requiring vulnerable UDF or outdated MySQL versions were **not applicable**, as the running instance was **fully patched**.
- The server rejected payloads due to **non-root MySQL execution** within the container — a deliberate hardening choice during configuration.

c. Docker Network

```
$ docker network ls
NETWORK ID     NAME      DRIVER      SCOPE
9460927910d9  bridge    bridge      local
e50ecf055c01  host      host      local
9704d3d205f4  none      null      local
fad096cc7402  securedlab_mynet  bridge      local
```

```
$ docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "9460927910d95fe80d3661c056cddeeb7e676d5370810308b59df0276c4846ab",
    "Created": "2025-07-26T16:11:05.184105118Z",
    "Scope": "local",
```

```

    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
        "Driver": "default",
        "Options": null,
        "Config": [
            {
                "Subnet": "172.17.0.0/16",
                "Gateway": "172.17.0.1"
            }
        ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
        "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {
        "com.docker.network.bridge.default_bridge": "true",
        "com.docker.network.bridge.enable_icc": "true",
        "com.docker.network.bridge.enable_ip_masquerade": "true",
        "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
        "com.docker.network.bridge.name": "docker0",
        "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
}
]

```

```

$ docker network inspect securedlab_mynet
[
{
    "Name": "securedlab_mynet",
    "Id": "fad096cc740256c24c0f71b85704cf7c06df81e02196cae5f53f9ae76abff357",
    "Created": "2025-07-26T08:57:16.140367848Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,

```

```

"IPAM": {
    "Driver": "default",
    "Options": null,
    "Config": [
        {
            "Subnet": "172.18.0.0/16",
            "Gateway": "172.18.0.1"
        }
    ]
},
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
    "Network": ""
},
"ConfigOnly": false,
"Containers": {
    "0b4e14012c007866fc33367bdb712e775c10eb221f7c3dffab3fe325941b3f9f": {
        "Name": "mysql_container",
        "EndpointID": "32660025feedab7459ccd4b354383672ceb99eb8c0286f418ee87d",
        "MacAddress": "02:42:ac:12:00:06",
        "IPv4Address": "172.18.0.6/16",
        "IPv6Address": ""
    },
    "26f8ca2a2e93265adc4b885b7183cbc57407844b769c2b58935a5a98fdf1d848": {
        "Name": "ubuntu_base_container",
        "EndpointID": "f03ee1cc6d4698ed85dfb0d8a22d00463b0f32ab20a12a2ad8ddc1k",
        "MacAddress": "02:42:ac:12:00:04",
        "IPv4Address": "172.18.0.4/16",
        "IPv6Address": ""
    },
    "8178c0f8bbda2e4e2a946a3f63fbf686fa01f820ffca8d99c65ac94b425f087d": {
        "Name": "web_container",
        "EndpointID": "3ad71e082c1b0f1cecef079dc6fcb6e20c40b7cc541167a7e16e3df3f0",
        "MacAddress": "02:42:ac:12:00:05",
        "IPv4Address": "172.18.0.5/16",
        "IPv6Address": ""
    },
    "9bacee61540812a4351e22e36646c6c802dcaf692a6d008b69f2327f7ee7b389": {
        "Name": "python_container",
        "EndpointID": "13d69f5c9aaa52cfb4f4b733588188fd1fc209851fd34c7f6db642504"
    }
}

```

```

        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""

    },
    "c123f037fa3afdf50f3f194214b6e766f805b4d3a0b80026f09a9f2d81295fbe": {
        "Name": "java_container",
        "EndpointID": "7c978860d737775a2a545b4354e635b40ce92e25e09a8704cc46c",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""

    }
},
"Options": {},
"Labels": {
    "com.docker.compose.network": "mynet",
    "com.docker.compose.project": "securedlab",
    "com.docker.compose.version": "2.12.2"
}
}
]

```

d. Hydra Brute-force Test

Hydra was launched with a targeted dictionary attack:

```
hydra -l root -P /usr/share/wordlists/rockyou.txt mysql://<ip>
```

Result:

- All brute-force attempts failed.
- The containerized MySQL instance employed:
 - **Strong credentials** generated with high entropy
 - A login delay mechanism to slow down repeated failed attempts
 - No root login allowed remotely (`skip-name-resolve` , `skip-networking` set in `my.cnf`)

e. Analysis and Security Posture

The environment remained resilient throughout the scanning and testing phases. The following factors contributed to its security:

1. Minimal Exposure:

Only required ports were exposed, and all others were closed or filtered. MySQL was not directly exposed to the public internet at any stage.

2. Container Isolation:

Docker containers operated within a **custom bridge network** with restrictive IP tables. Each service was bound to internal IP addresses and accessed via **NAT translation** from the EC2 host.

3. Principle of Least Privilege:

MySQL ran as a non-root user with restricted filesystem access. Dockerfile had been hardened with multi-stage builds, and base images were stripped of unnecessary utilities.

4. Up-to-date Patching:

All base images and services were updated at the time of deployment. No known vulnerabilities (based on the CVE database and Metasploit modules) were exploitable.

5. Authentication Hardening:

Rate limiting, non-default ports in staging, and strong password policies made brute-force attacks ineffective.

The security controls applied to the Docker-based lab were effective in resisting common exploitation attempts. Despite targeted scanning and attack simulations using professional-grade tools, the MySQL database remained uncompromised.

This result underscores the importance of **network segmentation, updated software, secure defaults, and container security best practices** in modern deployment environments.