AWS Server Management Assignment — Combined README (Tasks 1–6)

This repository contains **all 7 tasks** in one bounded `README.md` file, including **screenshots** (stored under the `images/` folder).

Table of Contents

Docker Workflow (Container Lifecycle) – Explanation

This diagram represents the **Docker Container Lifecycle State Machine**, which explains how a Docker container moves between different states based on Docker commands and runtime events.

Container States

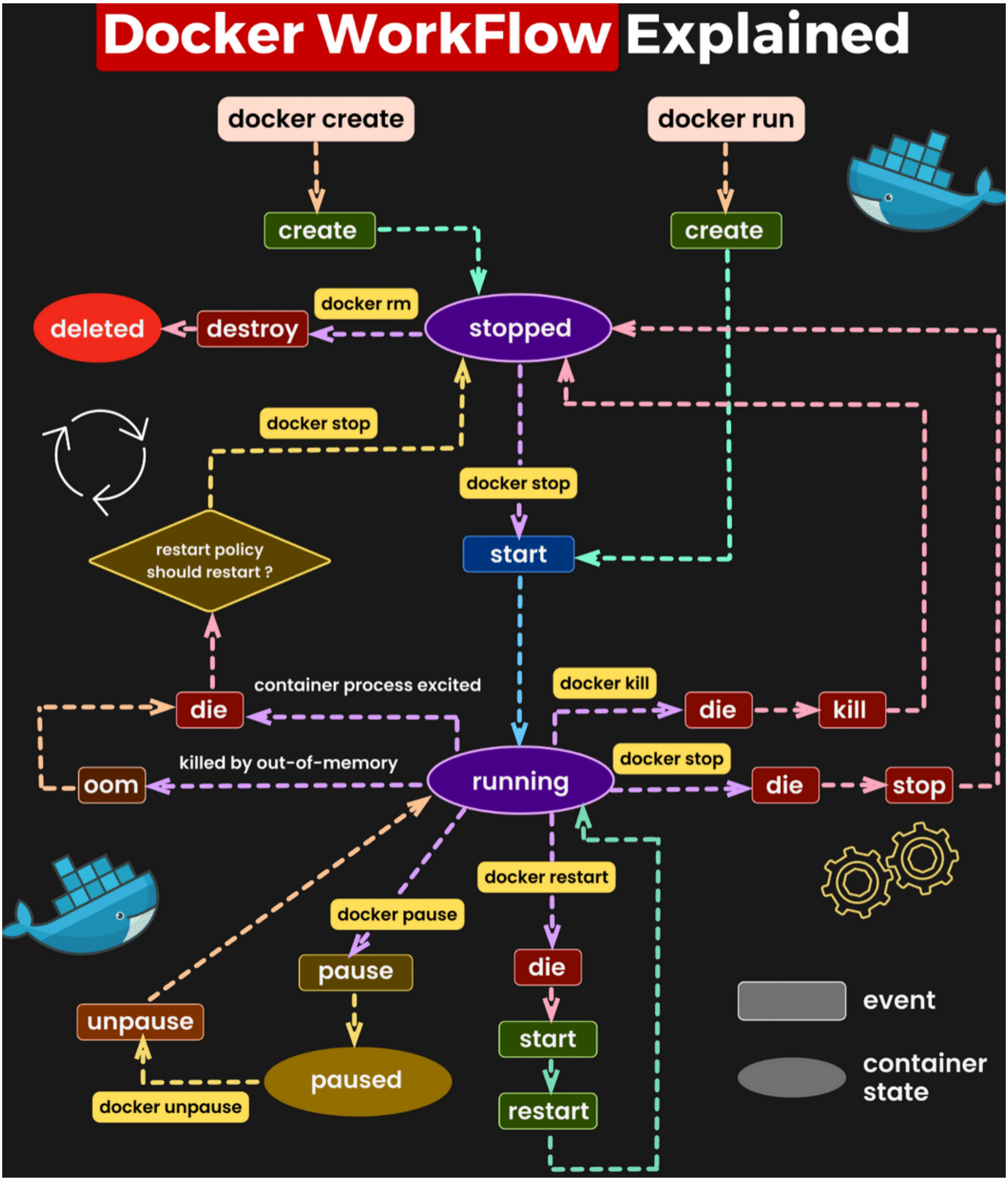
- **created** – Container is created but not started.
- **running** – Container is actively executing.
- **paused** – Container execution is temporarily frozen.
- **stopped** – Container has exited normally.
- **deleted** – Container is removed from the system.
- **die** – Container process has exited.
- **oom** – Container is killed due to Out Of Memory.

Key Docker Commands and Transitions

- `docker run` → Creates and starts a container (`created → running`)
- `docker stop` → Gracefully stops a container (`running → stopped`)
- `docker kill` → Forcefully stops a container (`running → die`)
- `docker start` → Starts a stopped container (`stopped → running`)
- `docker restart` → Stops and starts again (`running → die → running`)
- `docker pause` → Freezes container execution (`running → paused`)
- `docker unpause` → Resumes execution (`paused → running`)
- `docker rm` → Removes a stopped container (`stopped → deleted`)

Restart Policy

- If a container exits unexpectedly (`die` or `oom`), Docker checks the **restart policy**.
- If enabled, the container is automatically restarted.



This workflow shows how Docker manages container states internally and how user commands control container behavior. It helps understand container start, stop, crash handling, restart, and cleanup processes.

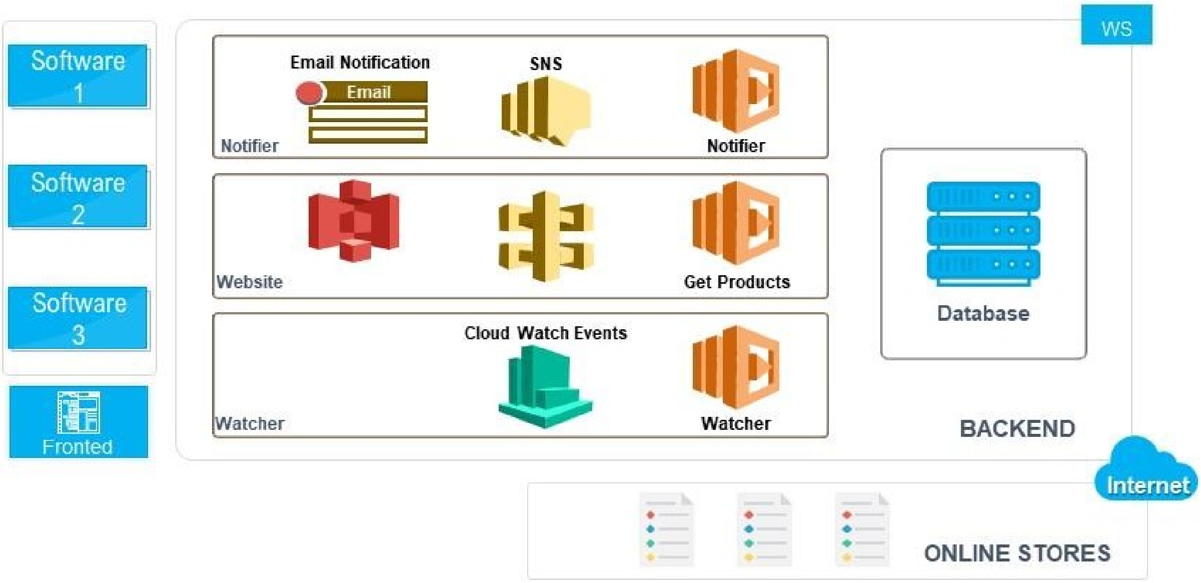
Task 1 — Simple Full Stack Application (Frontend + Backend + MySQL)

Project Overview

This project is a **simple full stack web application** developed as part of **Task 1**.
It demonstrates how a **frontend application communicates with a backend API**, which in turn **connects successfully to a MySQL database**.

The application allows users to **add, view, and delete student records** using a clean UI and REST-style backend.

Architecture



Tech Stack

Frontend

- HTML5
- Bootstrap 5
- JavaScript (Fetch API)

Backend

- PHP (REST-style API)

Database

- MySQL

Tools

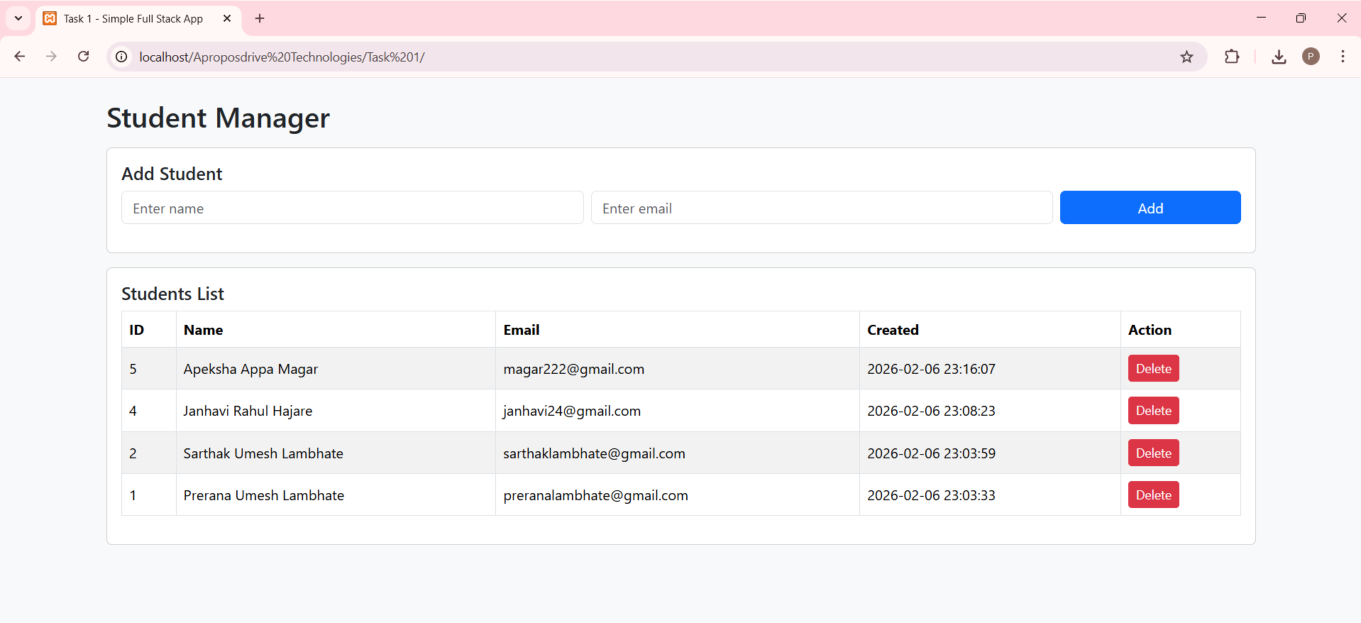
- XAMPP (Apache + MySQL)
- phpMyAdmin
- Web Browser

Application Features

- Responsive frontend using Bootstrap
- Backend API using PHP

- MySQL database integration
- Add student records
- View all student records
- Delete student records
- JSON-based communication
- Database connection validation

User Interface


Task 1 - Simple Full Stack App
localhost/Apropsdrive%20Technologies/Task%201/

Student Manager

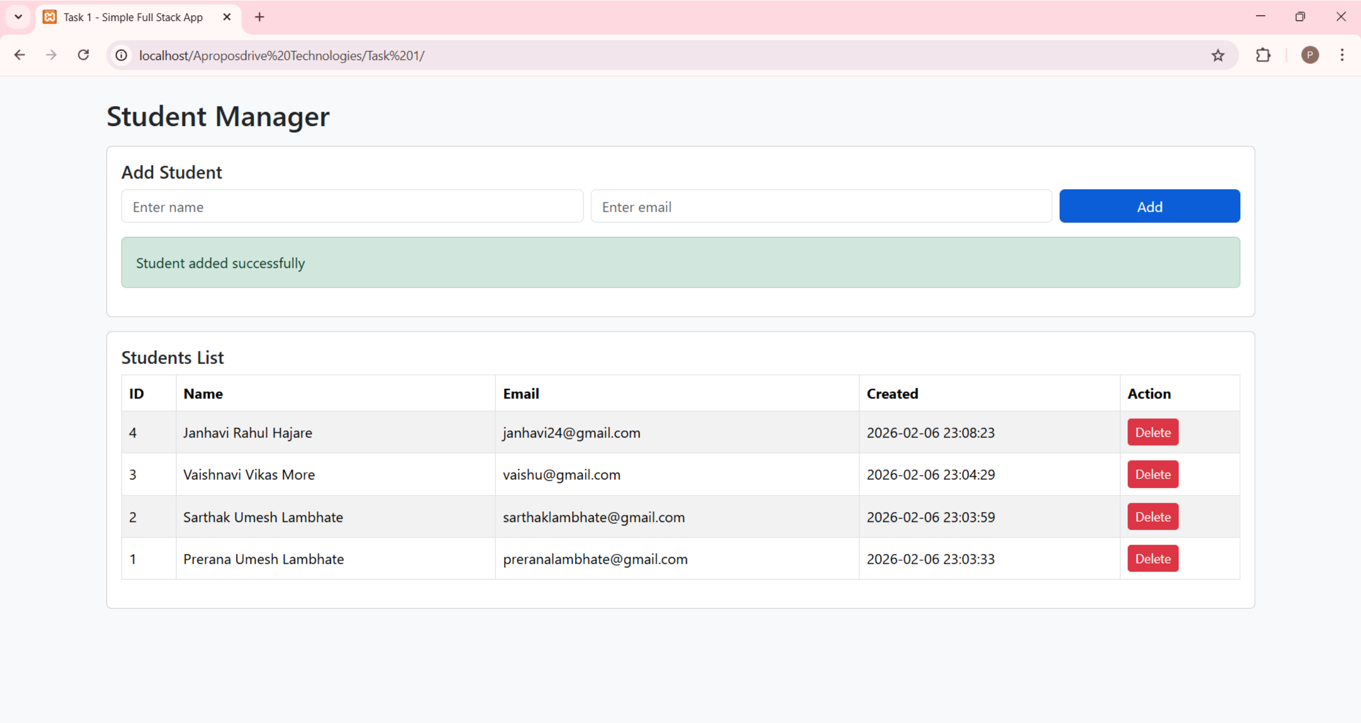
Add Student

Enter name | Enter email | Add

Students List

ID	Name	Email	Created	Action
5	Apeksha Appa Magar	magar222@gmail.com	2026-02-06 23:16:07	Delete
4	Janhavi Rahul Hajare	janhavi24@gmail.com	2026-02-06 23:08:23	Delete
2	Sarthak Umesh Lambhate	sarthaklambhate@gmail.com	2026-02-06 23:03:59	Delete
1	Prerana Umesh Lambhate	preranalambhate@gmail.com	2026-02-06 23:03:33	Delete

Adding Student Successfully


Task 1 - Simple Full Stack App
localhost/Apropsdrive%20Technologies/Task%201/

Student Manager

Add Student

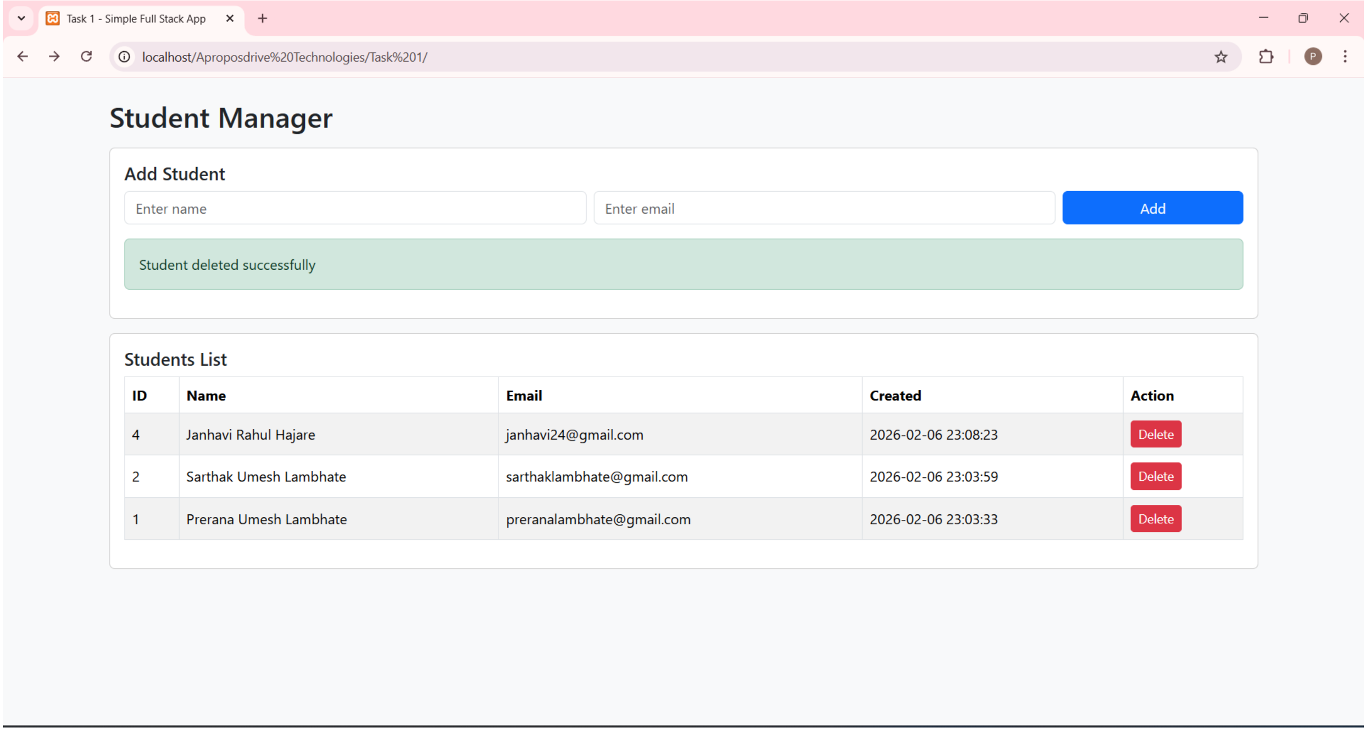
Enter name | Enter email | Add

Student added successfully

Students List

ID	Name	Email	Created	Action
4	Janhavi Rahul Hajare	janhavi24@gmail.com	2026-02-06 23:08:23	Delete
3	Vaishnavi Vikas More	vaishu@gmail.com	2026-02-06 23:04:29	Delete
2	Sarthak Umesh Lambhate	sarthaklambhate@gmail.com	2026-02-06 23:03:59	Delete
1	Prerana Umesh Lambhate	preranalambhate@gmail.com	2026-02-06 23:03:33	Delete

Deleting Student Successfully



Task 1 - Simple Full Stack App
localhost/Aproposdrive%20Technologies/Task%201/

Student Manager

Add Student

Enter name
Enter email
Add

Student deleted successfully

Students List

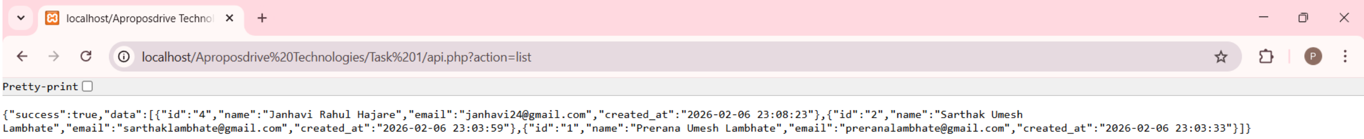
ID	Name	Email	Created	Action
4	Janhavi Rahul Hajare	janhavi24@gmail.com	2026-02-06 23:08:23	Delete
2	Sarthak Umesh Lambhate	sarthaklambhate@gmail.com	2026-02-06 23:03:59	Delete
1	Prerana Umesh Lambhate	preranalambhate@gmail.com	2026-02-06 23:03:33	Delete

Database Design

Database Name : `task1_db`

Table Schema

```
CREATE TABLE students (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(120) NOT NULL UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```



localhost/Aproposdrive Technol
localhost/Aproposdrive%20Technologies/Task%201/api.php?action=list
Pretty-print □

{"success":true,"data":[{"id":"4","name":"Janhavi Rahul Hajare","email":"janhavi24@gmail.com","created_at":"2026-02-06 23:08:23"},{"id":"2","name":"Sarthak Umesh Lambhate","email":"sarthaklambhate@gmail.com","created_at":"2026-02-06 23:03:59"},{"id":"1","name":"Prerana Umesh Lambhate","email":"preranalambhate@gmail.com","created_at":"2026-02-06 23:03:33"}]}

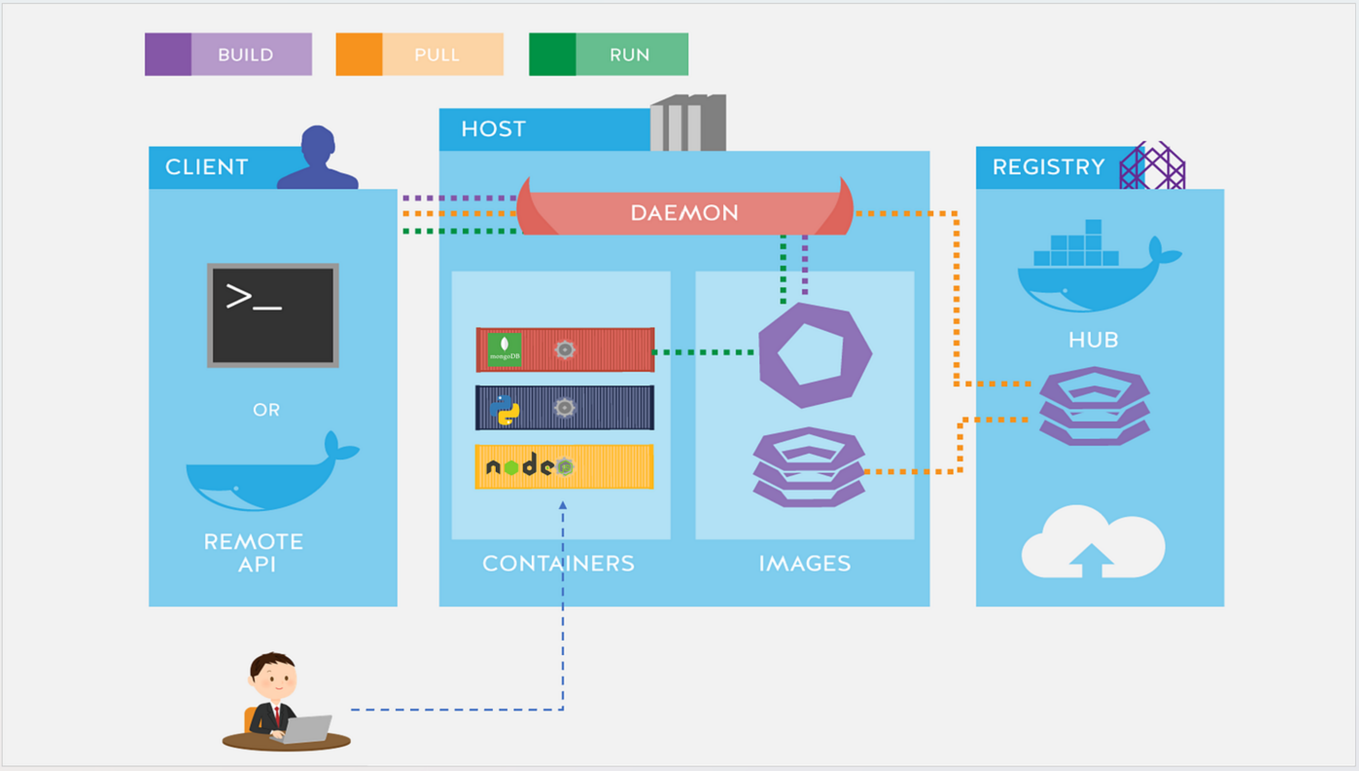
Task 2 — Docker Deployment on AWS EC2 (Amazon Linux)

Objective

- Create Dockerfile(s)
- Run app using Docker containers

- Expose required ports
- Ensure containers auto-start on EC2 reboot

Architecture



Technologies Used

- AWS EC2 (Amazon Linux 2 / Amazon Linux 2023)
- Docker
- Docker Compose
- PHP (Backend)
- MySQL (Database)
- Apache Web Server

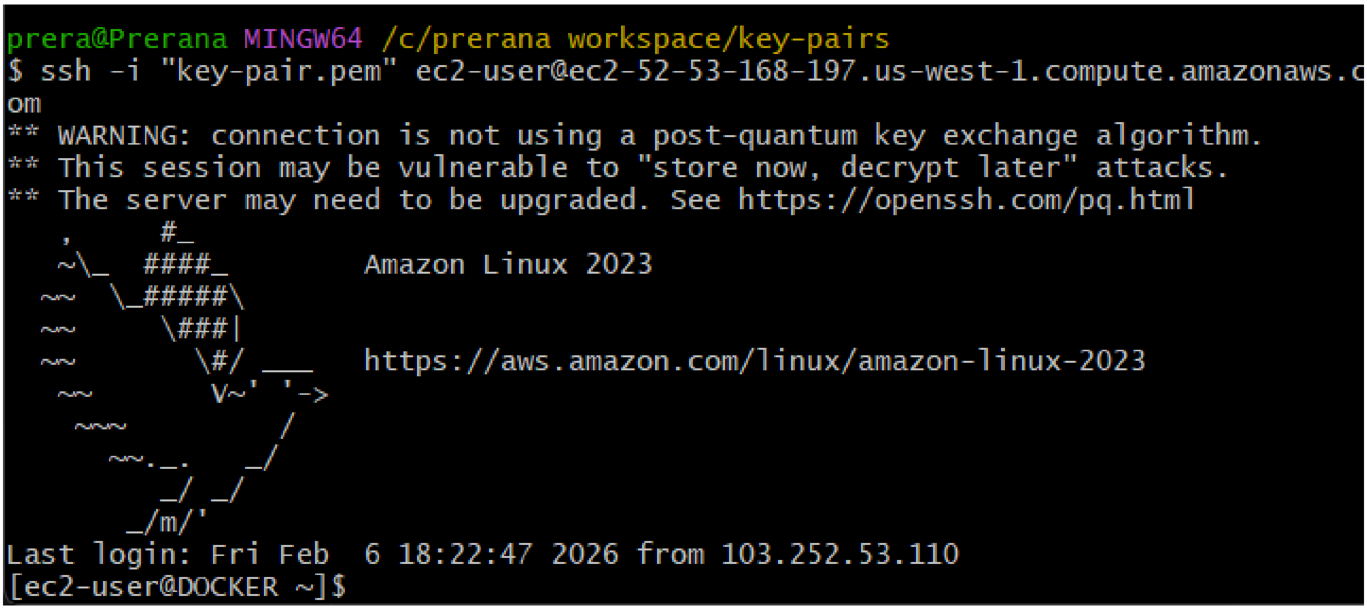
Step 1: Launch AWS EC2 Instance

Instance configuration

- AMI: Amazon Linux 2 / Amazon Linux 2023
- Instance Type: t2.micro
- Security Group Inbound Rules:
  - SSH – Port 22
  - HTTP – Port 80
  - MySQL – Port 3306 (optional, for testing)

Connect

```
ssh -i key.pem ec2-user@<EC2_PUBLIC_IP>
```



```
prera@Prerana MINGW64 /c/prerana workspace/key-pairs
$ ssh -i "key-pair.pem" ec2-user@ec2-52-53-168-197.us-west-1.compute.amazonaws.c
om
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Feb  6 18:22:47 2026 from 103.252.53.110
[ec2-user@DOCKER ~]$
```

Step 2: Install Docker on Amazon Linux

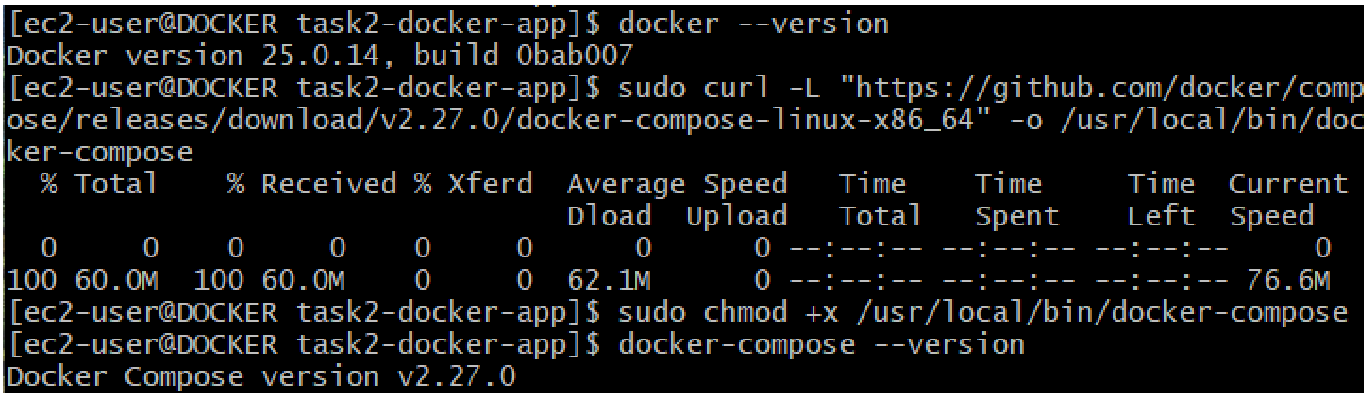
```
sudo yum update -y
sudo yum install docker -y
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker ec2-user
newgrp docker
docker --version
docker ps
```

Step 3: Install Docker Compose (Manual Method)

Amazon Linux may not support `docker-compose-plugin` via yum, so Docker Compose can be installed manually.

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.27.0/docker-compose-linux-x86_64" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

Screenshot



```
[ec2-user@DOCKER task2-docker-app]$ docker --version
Docker version 25.0.14, build 0bab007
[ec2-user@DOCKER task2-docker-app]$ sudo curl -L "https://github.com/docker/comp
ose/releases/download/v2.27.0/docker-compose-linux-x86_64" -o /usr/local/bin/doc
ker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 60.0M  100 60.0M    0     0  62.1M      0 --:--:-- --:--:-- --:--:-- 76.6M
[ec2-user@DOCKER task2-docker-app]$ sudo chmod +x /usr/local/bin/docker-compose
[ec2-user@DOCKER task2-docker-app]$ docker-compose --version
Docker Compose version v2.27.0
```

Step 4: Create Project Directory

```
mkdir task2-docker-app
cd task2-docker-app
mkdir app
```

Step 5: Create Application File (PHP)

Create a PHP file (example: `app/index.php`) using `vim`:

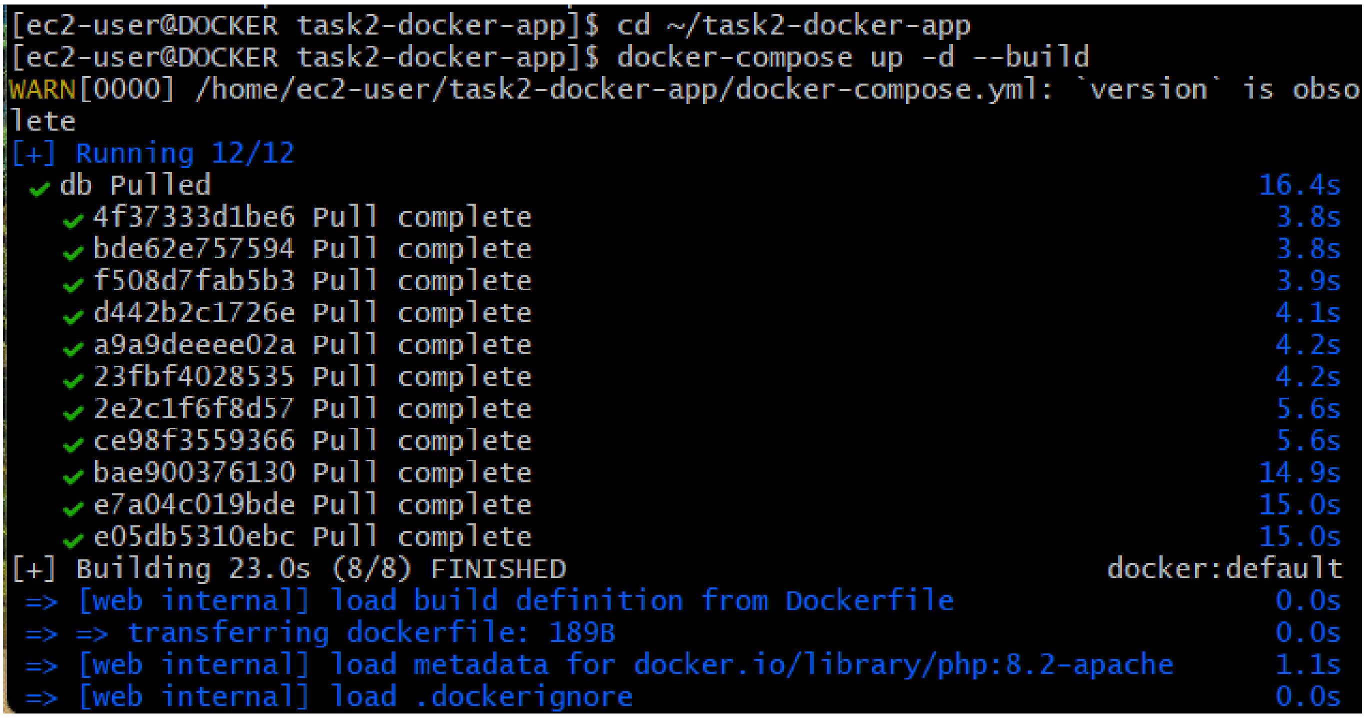
```
<?php
$host = getenv("DB_HOST") ?: "db";
$user = getenv("DB_USER") ?: "root";
$pass = getenv("DB_PASS") ?: "rootpass";
$name = getenv("DB_NAME") ?: "studentdb";

$conn = new mysqli($host, $user, $pass, $name);

if ($conn->connect_error) {
  die("Database connection FAILED: " . $conn->connect_error);
}

echo "<h2>Database Connected Successfully!</h2>";
?>
```

Step 6: Create Dockerfile

Create `Dockerfile`:

```
FROM php:8.2-apache
RUN docker-php-ext-install mysqli
```

```
COPY app/ /var/www/html/
EXPOSE 80
```



```
[ec2-user@DOCKER task2-docker-app]$ cd ~/task2-docker-app
[ec2-user@DOCKER task2-docker-app]$ docker-compose up -d --build
WARN[0000] /home/ec2-user/task2-docker-app/docker-compose.yml: `version` is obso
lete
[+] Running 12/12
 ✓ db Pulled                                                              16.4s
   ✓ 4f37333d1be6 Pull complete                                            3.8s
   ✓ bde62e757594 Pull complete                                            3.8s
   ✓ f508d7fab5b3 Pull complete                                            3.9s
   ✓ d442b2c1726e Pull complete                                            4.1s
   ✓ a9a9deeee02a Pull complete                                            4.2s
   ✓ 23fbf4028535 Pull complete                                            4.2s
   ✓ 2e2c1f6f8d57 Pull complete                                            5.6s
   ✓ ce98f3559366 Pull complete                                            5.6s
   ✓ bae900376130 Pull complete                                           14.9s
   ✓ e7a04c019bde Pull complete                                           15.0s
   ✓ e05db5310ebc Pull complete                                           15.0s
[+] Building 23.0s (8/8) FINISHED                                 docker:default
 => [web internal] load build definition from Dockerfile                   0.0s
 => => transferring dockerfile: 189B                                       0.0s
 => [web internal] load metadata for docker.io/library/php:8.2-apache      1.1s
 => [web internal] load .dockerignore                                      0.0s
```

Step 7: Create docker-compose.yml

Create `docker-compose.yml`:

```
version: "3.9"
services:
  web:
    build: .
    container_name: php_web
    ports:
      - "80:80"
    environment:
      DB_HOST: db
      DB_USER: root
      DB_PASS: rootpass
      DB_NAME: studentdb
    depends_on:
      - db
    restart: always

  db:
    image: mysql:8.0
    container_name: mysql_db
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: studentdb
    volumes:
      - mysql_data:/var/lib/mysql
    restart: always

volumes:
  mysql_data:
```

Step 8: Build and Run Containers

```
docker-compose up -d --build
docker ps
```

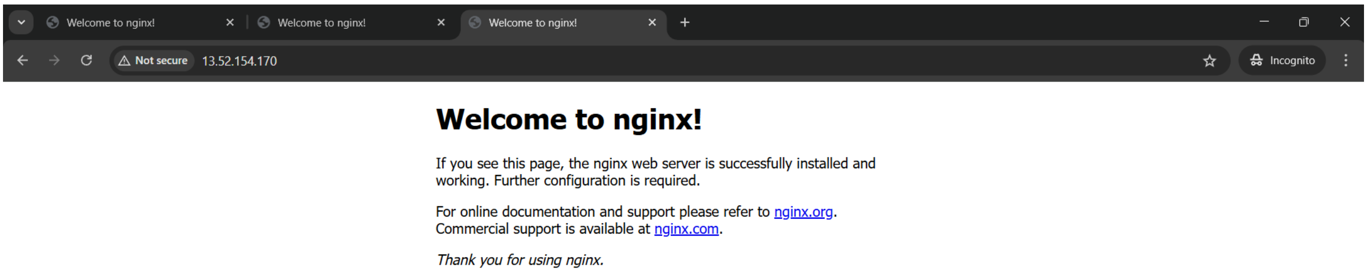
Step 9: Test Application

From EC2:

```
curl http://localhost
```

From browser:

```
http://<EC2_PUBLIC_IP>
```



Welcome to nginx!

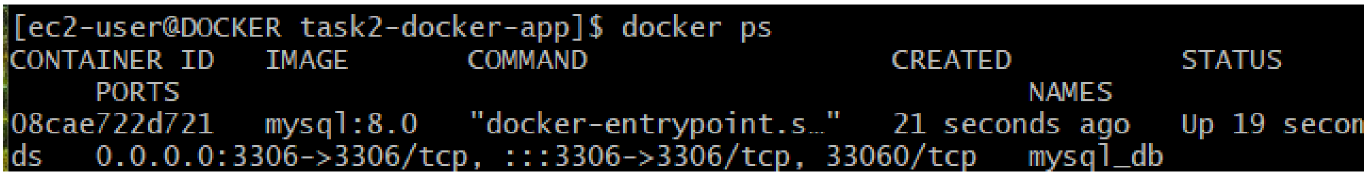
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Step 10: Auto-start Containers on Reboot

```
sudo systemctl enable docker
sudo reboot
# after reconnect
docker ps
```



```
[ec2-user@DOCKER task2-docker-app]$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS
     PORTS                                                 NAMES
08cae722d721   mysql:8.0    "docker-entrypoint.s…"   21 seconds ago   Up 19 secon
ds    0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp    mysql_db
```

Conclusion

This task demonstrates:

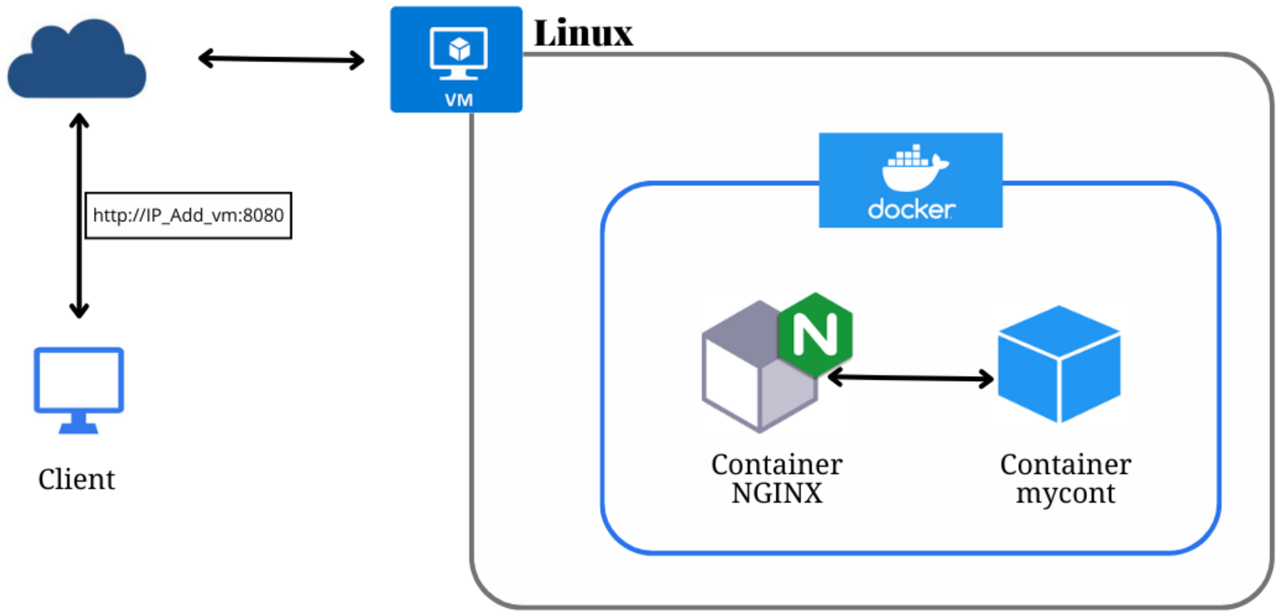
- Docker installation on AWS EC2
- Dockerfile creation
- Multi-container app using Docker Compose
- Port exposure
- Container auto-start on reboot

Task 3 — AWS EC2 Deployment using Docker (Nginx)

Objective

Deploy and run Docker containers on an AWS EC2 instance (Amazon Linux) and access the app publicly.

Architecture



Technologies Used

- AWS EC2
- Amazon Linux

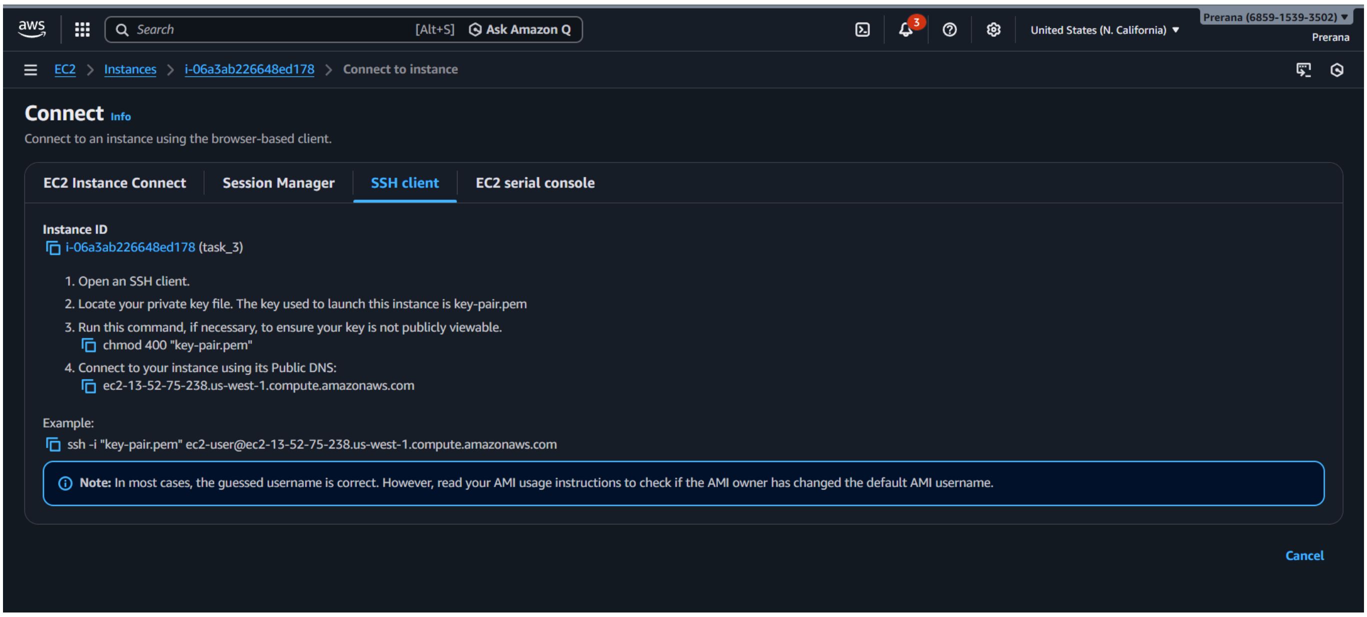
- Docker
- Nginx (Docker Container)

EC2 Instance Configuration

- Instance Type: t2.micro (Free Tier)
- Security Group:
  - SSH (22) – My IP
  - HTTP (80) – 0.0.0.0/0

Steps (with Screenshots)

Step 1: Launch EC2 Instance


aws
Search
[Alt+S]
Ask Amazon Q
3
United States (N. California) ▼
Prerana (6859-1539-3502) ▼
Prerana
EC2 > Instances > i-06a3ab226648ed178 > Connect to instance
Info
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Cancel

Connect

Connect to an instance using the browser-based client.

Instance ID

i-06a3ab226648ed178 (task_3)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is key-pair.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "key-pair.pem"
4. Connect to your instance using its Public DNS:
   ec2-13-52-75-238.us-west-1.compute.amazonaws.com

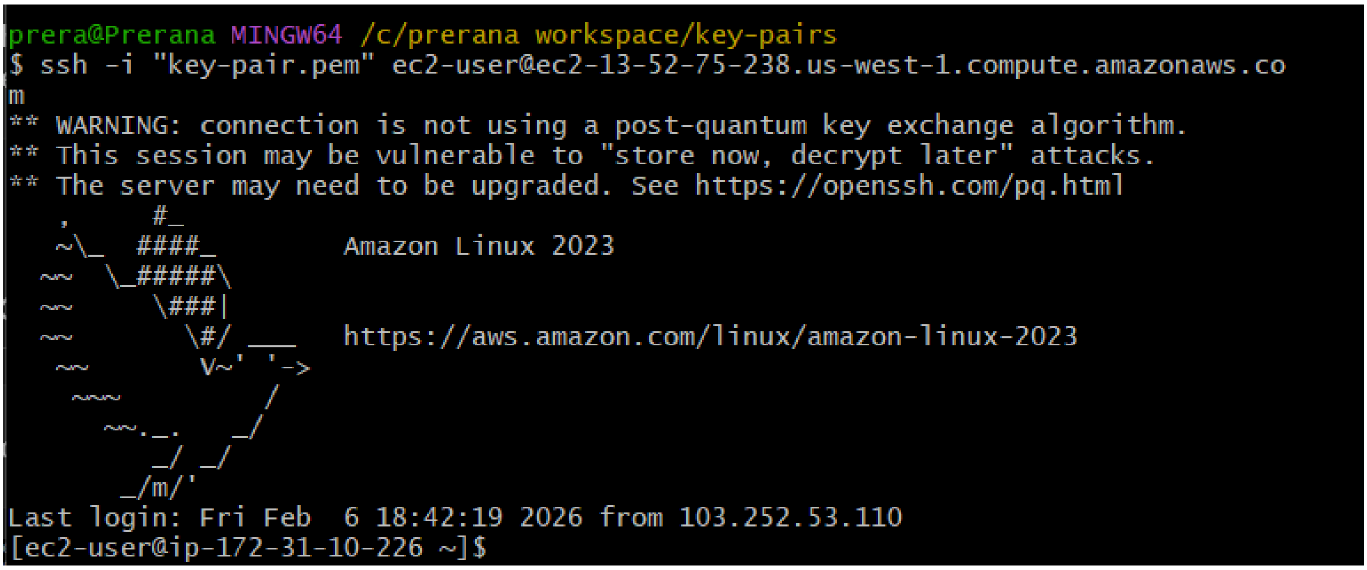
Example:

ssh -i "key-pair.pem" ec2-user@ec2-13-52-75-238.us-west-1.compute.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Step 2: Connect to EC2

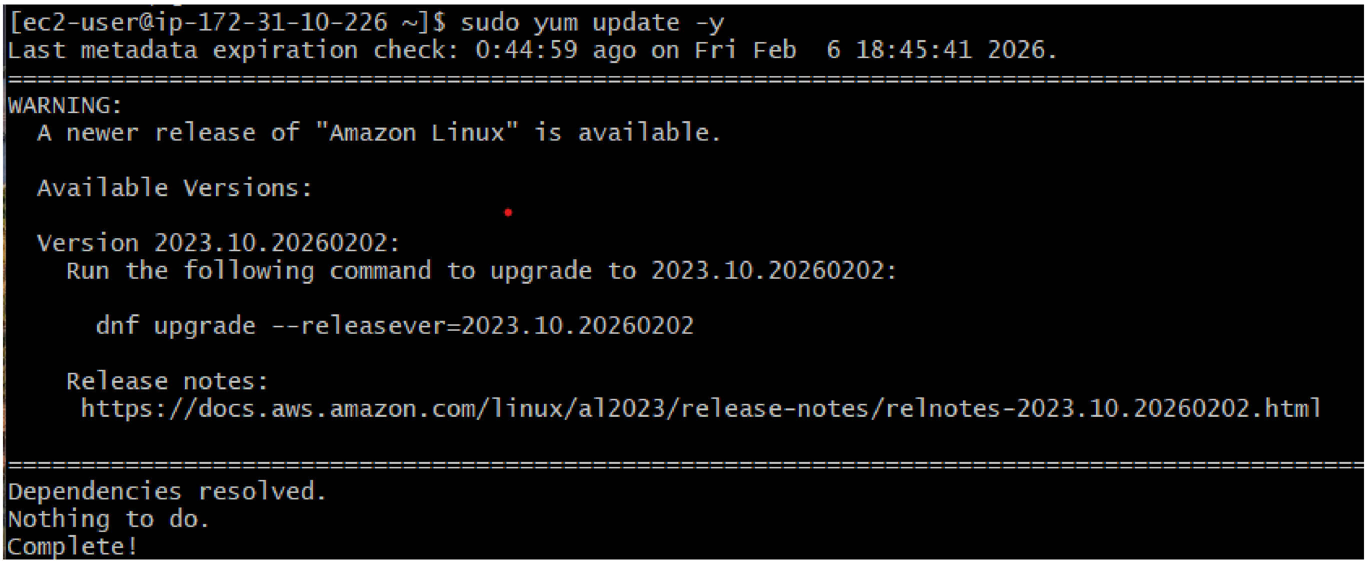
```
ssh -i "your-key.pem" ec2-user@<EC2_PUBLIC_IP>
```



```
prera@Prerana MINGW64 /c/prerana workspace/key-pairs
$ ssh -i "key-pair.pem" ec2-user@ec2-13-52-75-238.us-west-1.compute.amazonaws.co
m
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Feb  6 18:42:19 2026 from 103.252.53.110
[ec2-user@ip-172-31-10-226 ~]$
```

Step 3: Update System Packages

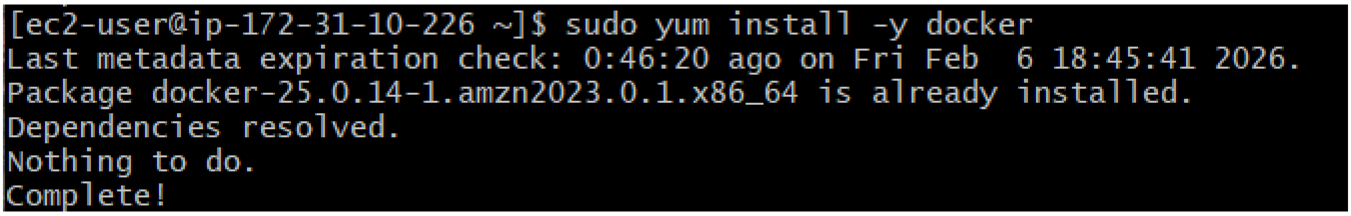
```
sudo yum update -y
```



```
[ec2-user@ip-172-31-10-226 ~]$ sudo yum update -y
Last metadata expiration check: 0:44:59 ago on Fri Feb  6 18:45:41 2026.
============================================================================================
WARNING:
  A newer release of "Amazon Linux" is available.

  Available Versions:

  Version 2023.10.20260202:
    Run the following command to upgrade to 2023.10.20260202:

      dnf upgrade --releasever=2023.10.20260202

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.10.20260202.html

============================================================================================
Dependencies resolved.
Nothing to do.
Complete!
```

Step 4: Install Docker

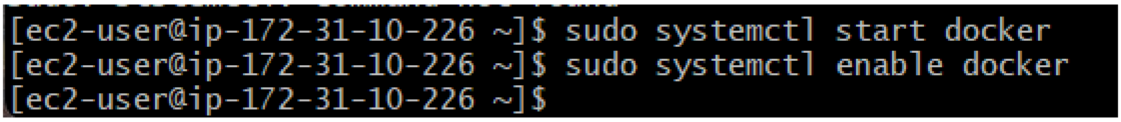
```
sudo yum install -y docker
```



```
[ec2-user@ip-172-31-10-226 ~]$ sudo yum install -y docker
Last metadata expiration check: 0:46:20 ago on Fri Feb  6 18:45:41 2026.
Package docker-25.0.14-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Step 5: Start and Enable Docker

```
sudo systemctl start docker
sudo systemctl enable docker
```



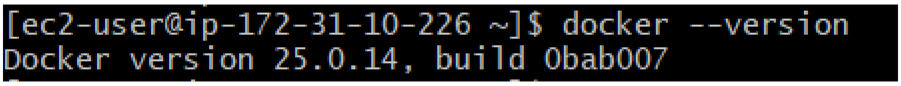
```
[ec2-user@ip-172-31-10-226 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-10-226 ~]$ sudo systemctl enable docker
[ec2-user@ip-172-31-10-226 ~]$
```

Step 6: Add User to Docker Group

```
sudo usermod -aG docker ec2-user
```

Step 7: Verify Docker

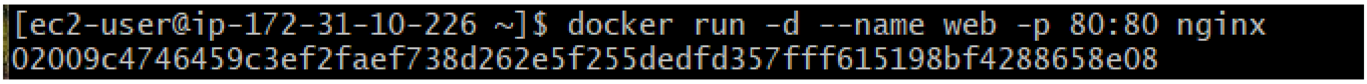
```
docker --version
```



```
[ec2-user@ip-172-31-10-226 ~]$ docker --version
Docker version 25.0.14, build 0bab007
```

Step 8: Run Nginx Container

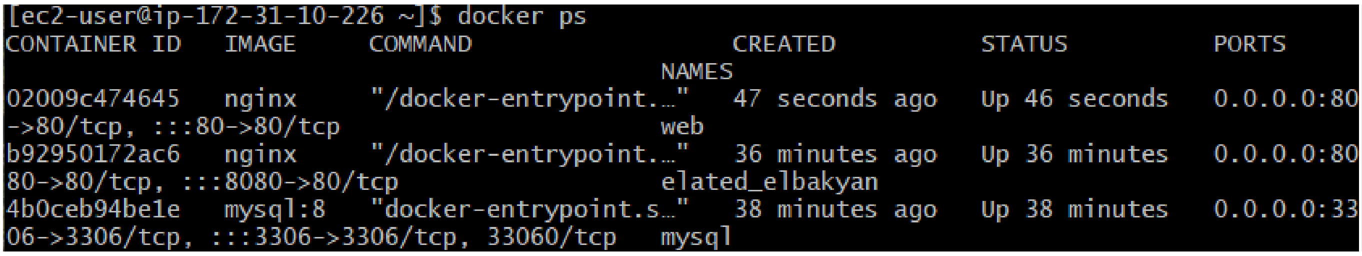
```
docker run -d --name web -p 80:80 nginx
```



```
[ec2-user@ip-172-31-10-226 ~]$ docker run -d --name web -p 80:80 nginx
02009c4746459c3ef2faef738d262e5f255dedfd357fff615198bf4288658e08
```

Step 9: Verify Running Containers

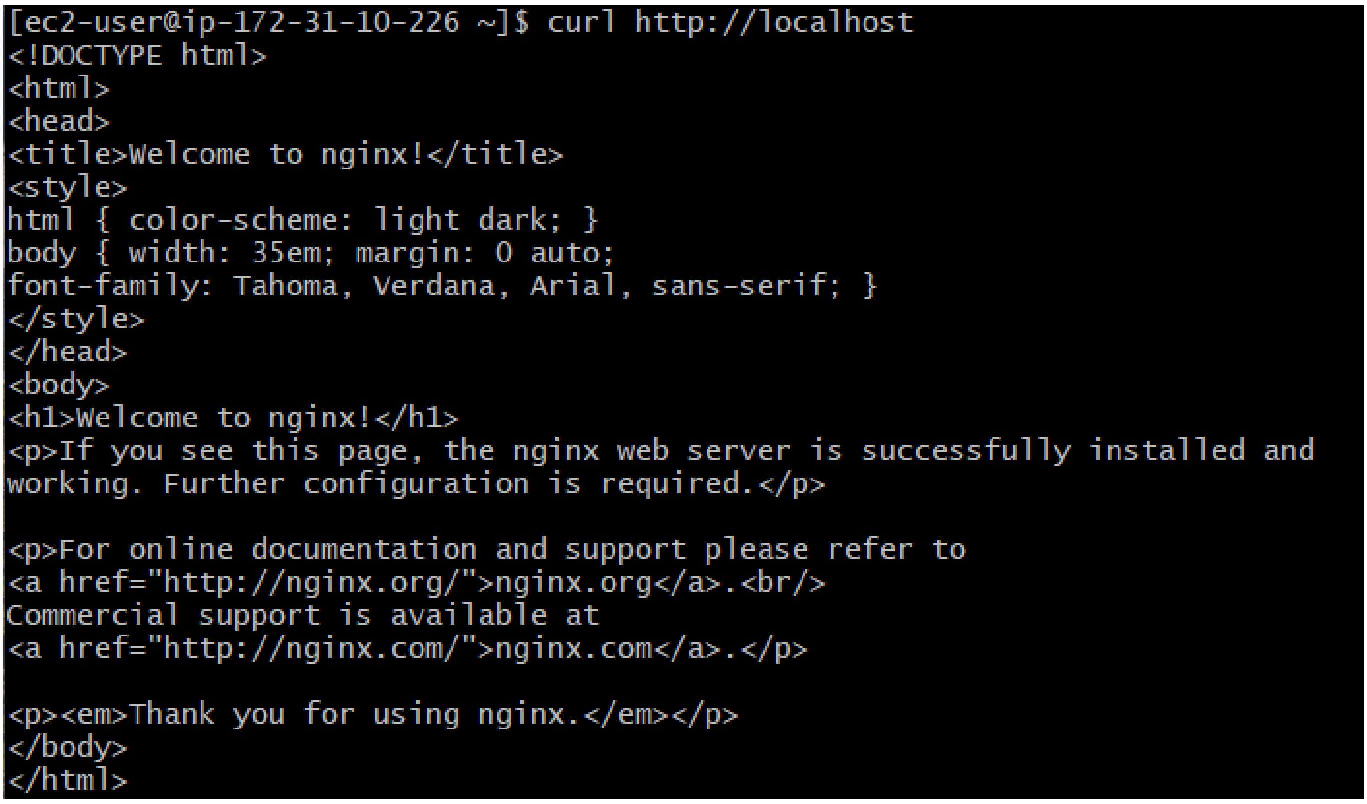
```
docker ps
```



```
[ec2-user@ip-172-31-10-226 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
                                                  NAMES
02009c474645   nginx     "/docker-entrypoint.…"   47 seconds ago   Up 46 seconds   0.0.0.0:80
->80/tcp, :::80->80/tcp                           web
b92950172ac6   nginx     "/docker-entrypoint.…"   36 minutes ago   Up 36 minutes   0.0.0.0:80
80->80/tcp, :::8080->80/tcp                       elated_elbakyan
4b0ceb94be1e   mysql:8   "docker-entrypoint.s…"   38 minutes ago   Up 38 minutes   0.0.0.0:33
06->3306/tcp, :::3306->3306/tcp, 33060/tcp        mysql
```

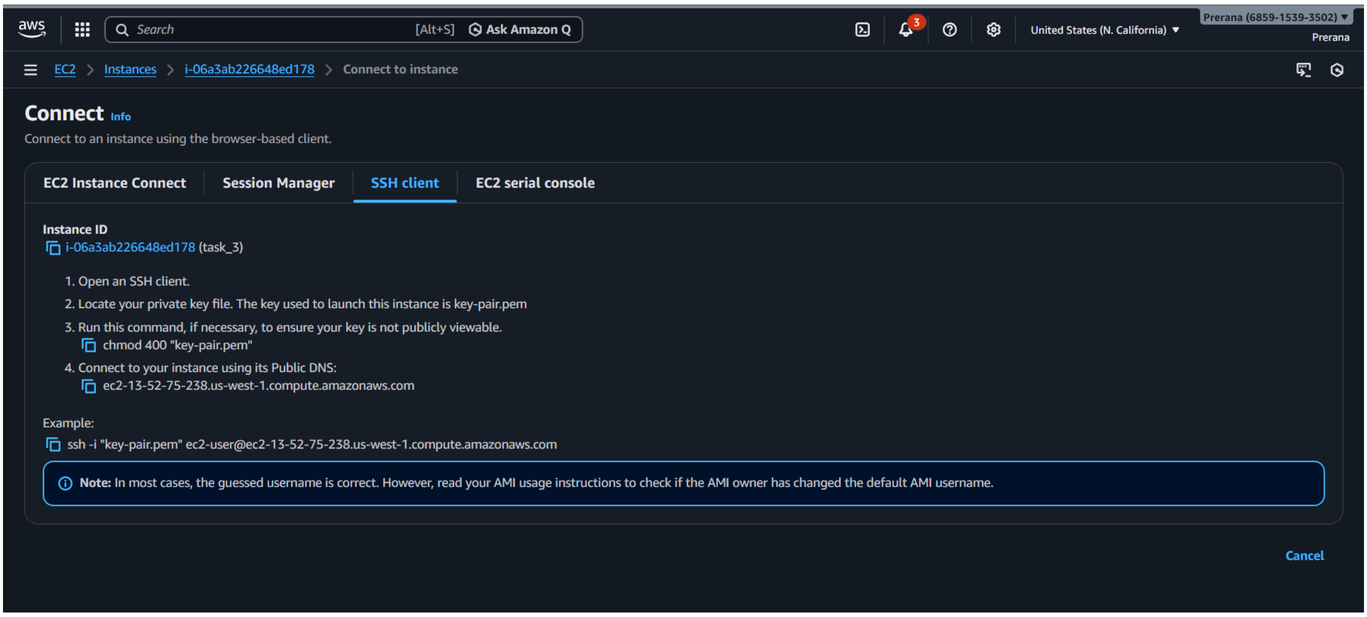
Step 10: Test Inside EC2

```
curl http://localhost
```



```
[ec2-user@ip-172-31-10-226 ~]$ curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<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>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Step 11: Access from Browser



aws
Search
[Alt+S]
Ask Amazon Q
United States (N. California) ▾
Prerana (6859-1539-3502) ▾
Prerana
EC2 > Instances > i-06a3ab226648ed178 > Connect to instance
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Cancel

Connect Info

Connect to an instance using the browser-based client.

Instance ID

i-06a3ab226648ed178 (task_3)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is key-pair.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "key-pair.pem"
4. Connect to your instance using its Public DNS:
   ec2-13-52-75-238.us-west-1.compute.amazonaws.com

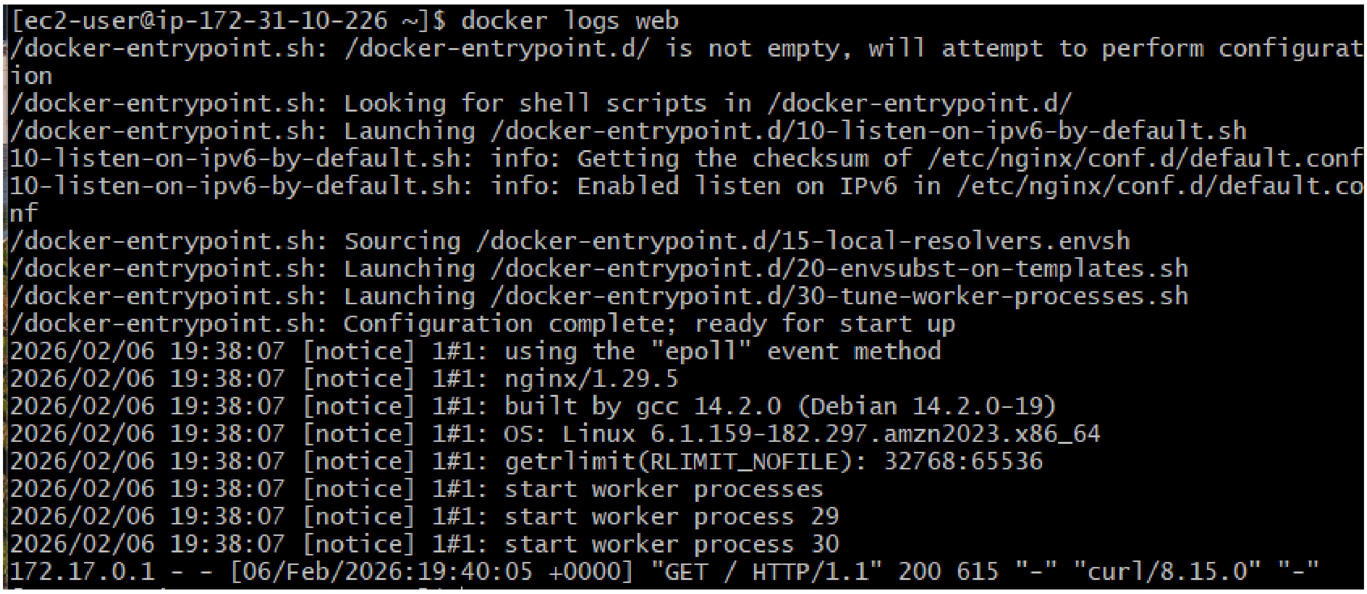
Example:

ssh -i "key-pair.pem" ec2-user@ec2-13-52-75-238.us-west-1.compute.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Step 12: View Logs

```
docker logs web
```



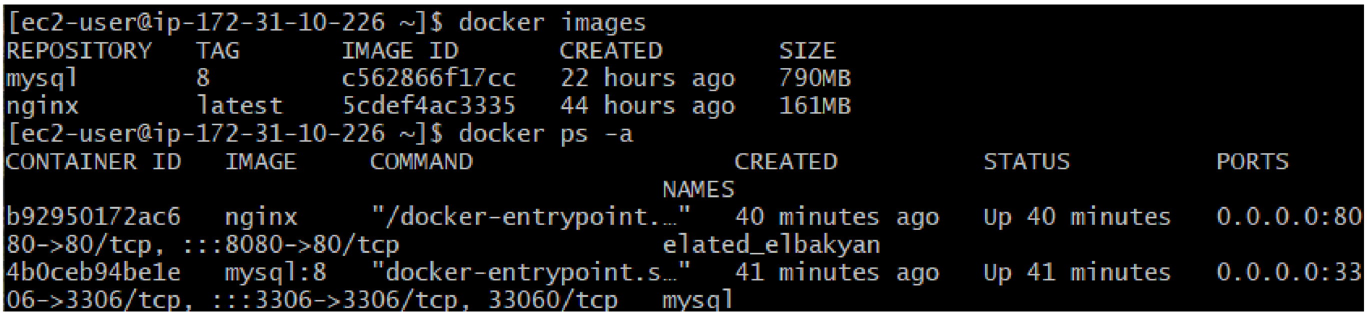
```
[ec2-user@ip-172-31-10-226 ~]$ docker logs web
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configurat
ion
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.co
nf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2026/02/06 19:38:07 [notice] 1#1: using the "epoll" event method
2026/02/06 19:38:07 [notice] 1#1: nginx/1.29.5
2026/02/06 19:38:07 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2026/02/06 19:38:07 [notice] 1#1: OS: Linux 6.1.159-182.297.amzn2023.x86_64
2026/02/06 19:38:07 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 32768:65536
2026/02/06 19:38:07 [notice] 1#1: start worker processes
2026/02/06 19:38:07 [notice] 1#1: start worker process 29
2026/02/06 19:38:07 [notice] 1#1: start worker process 30
172.17.0.1 - - [06/Feb/2026:19:40:05 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.15.0" "-"
```

Step 13: Stop and Remove Container

```
docker stop web
docker rm web
```

Step 14: Cleanup

```
docker images
docker ps -a
```



```
[ec2-user@ip-172-31-10-226 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
mysql        8         c562866f17cc   22 hours ago   790MB
nginx        latest    5cdef4ac3335   44 hours ago   161MB
[ec2-user@ip-172-31-10-226 ~]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
                                            NAMES
b92950172ac6   nginx     "/docker-entrypoint.…"   40 minutes ago   Up 40 minutes   0.0.0.0:80
80->80/tcp, :::8080->80/tcp                 elated_elbakyan
4b0ceb94be1e   mysql:8   "docker-entrypoint.s…"   41 minutes ago   Up 41 minutes   0.0.0.0:33
06->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql
```

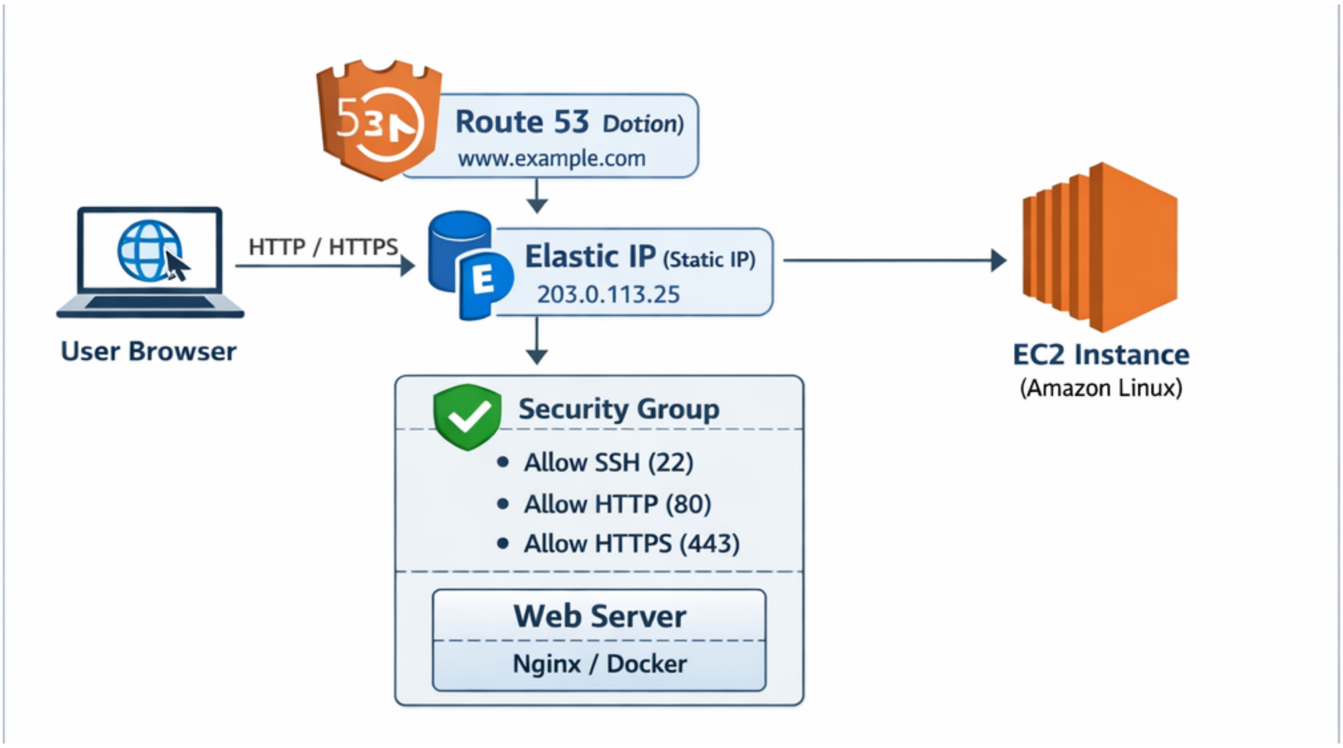
Task 4 — Application Access (Public IP / Elastic IP / Route 53)

Goal

Make the application accessible from a browser using:

- EC2 Public IP (temporary)
- Elastic IP (static / recommended)
- Route 53 domain (optional)

Architecture



Prerequisites

- App running on EC2 (Nginx/Docker)
- Security Group allows HTTP (80)

Step-by-Step (Amazon Linux)

1) Launch EC2 Instance

- AMI: Amazon Linux
- Type: t2.micro
- Inbound rules:
  - SSH 22 (My IP)
  - HTTP 80 (Anywhere)
  - HTTPS 443 (optional)

2) Connect using SSH

```
ssh -i your-key.pem ec2-user@EC2_PUBLIC_IP
```

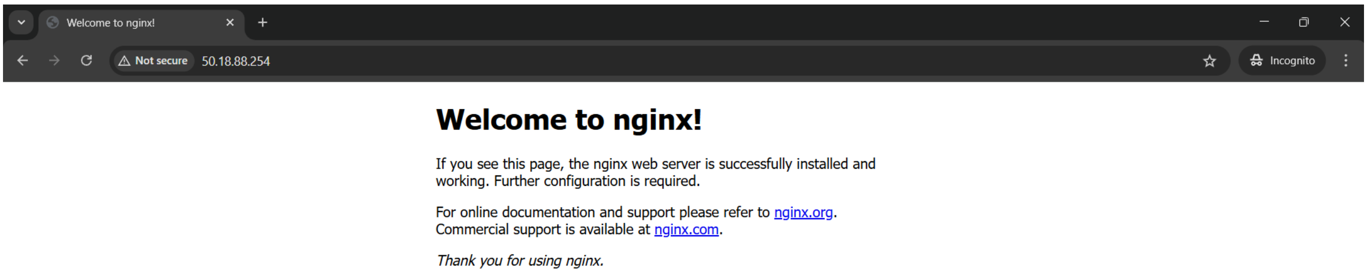
3) Install and Run Nginx (example)

```
sudo yum update -y
sudo yum install nginx -y
sudo systemctl start nginx
sudo systemctl enable nginx
systemctl status nginx
```

4) Access via EC2 Public IP

Open:

```
http://EC2_PUBLIC_IP
```



Welcome to nginx!

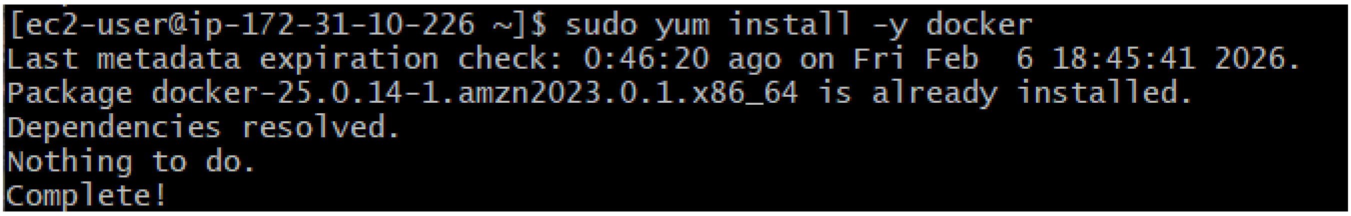
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

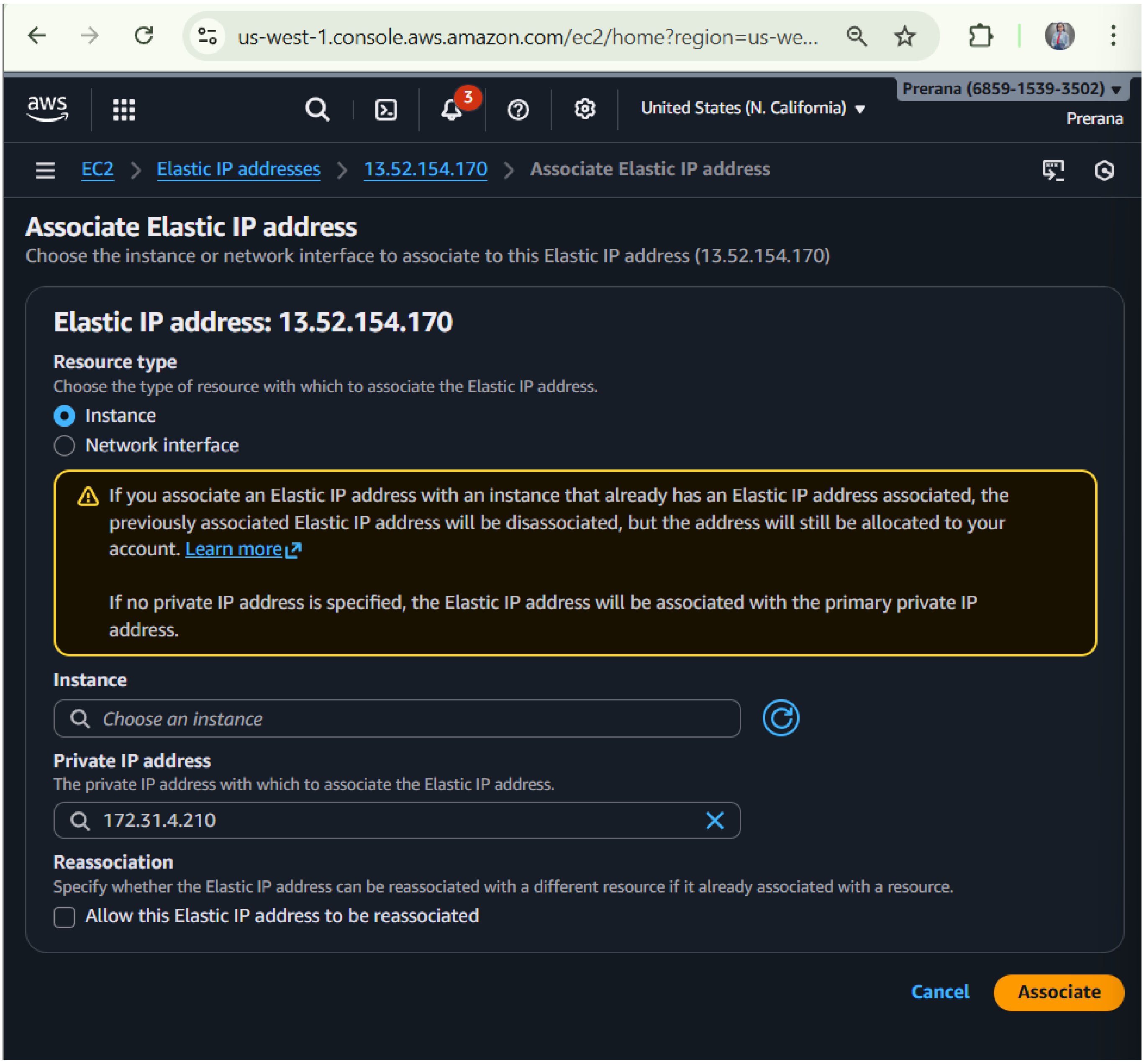
Thank you for using nginx.

5) Allocate & Associate Elastic IP

EC2 → Elastic IPs → Allocate → Associate to instance



```
[ec2-user@ip-172-31-10-226 ~]$ sudo yum install -y docker
Last metadata expiration check: 0:46:20 ago on Fri Feb  6 18:45:41 2026.
Package docker-25.0.14-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```


us-west-1.console.aws.amazon.com/ec2/home?region=us-we...
aws
3
United States (N. California) ▾
Prerana (6859-1539-3502) ▾
Prerana
EC2 > Elastic IP addresses > 13.52.154.170 > Associate Elastic IP address

Associate Elastic IP address

Choose the instance or network interface to associate to this Elastic IP address (13.52.154.170)

Elastic IP address: 13.52.154.170

Resource type

Choose the type of resource with which to associate the Elastic IP address.

● Instance

○ Network interface

If you associate an Elastic IP address with an instance that already has an Elastic IP address associated, the previously associated Elastic IP address will be disassociated, but the address will still be allocated to your account. Learn more

If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.

Instance

Choose an instance

Private IP address

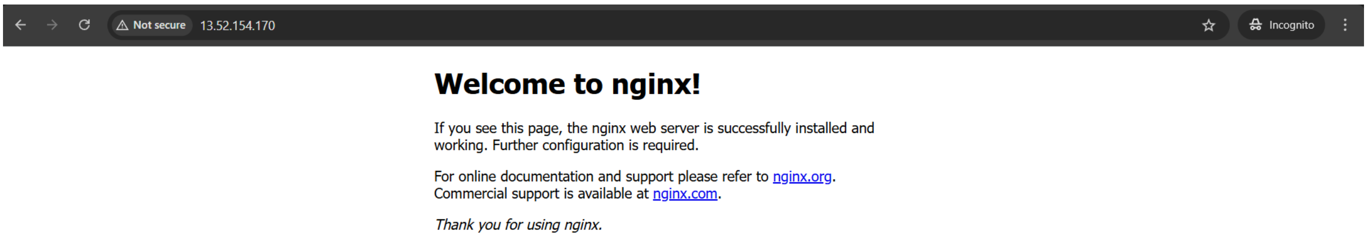
The private IP address with which to associate the Elastic IP address.

172.31.4.210

Reassociation

Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.

□ Allow this Elastic IP address to be reassociated



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

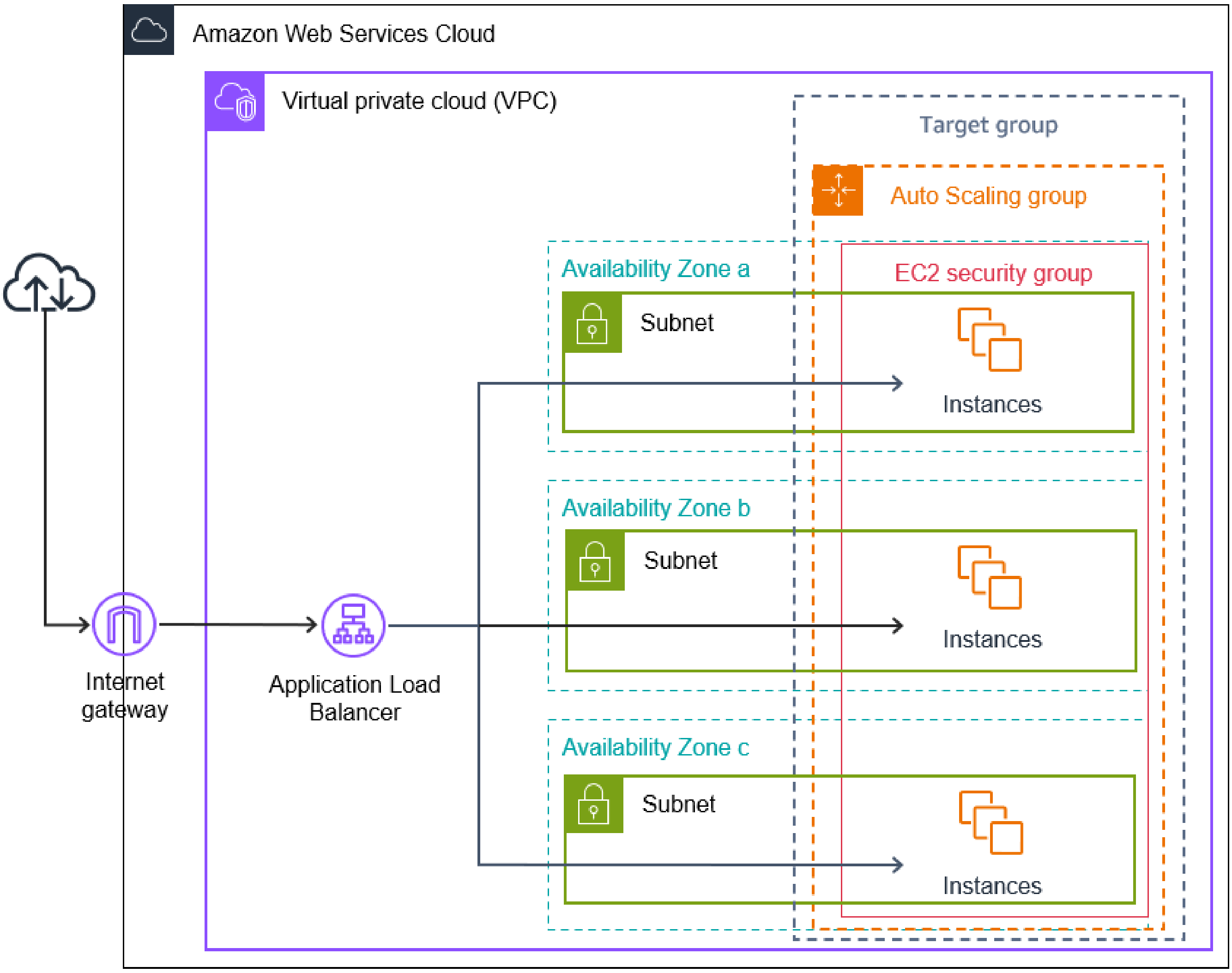
Thank you for using nginx.

Task 5 — Load Balancer & Auto Scaling (ALB + ASG)

Objective

The objective of this task is to configure an Application Load Balancer (ALB) and attach it to an Auto Scaling Group (ASG) so that application traffic is distributed automatically and EC2 instances scale based on CPU utilization.

Architecture



Configure

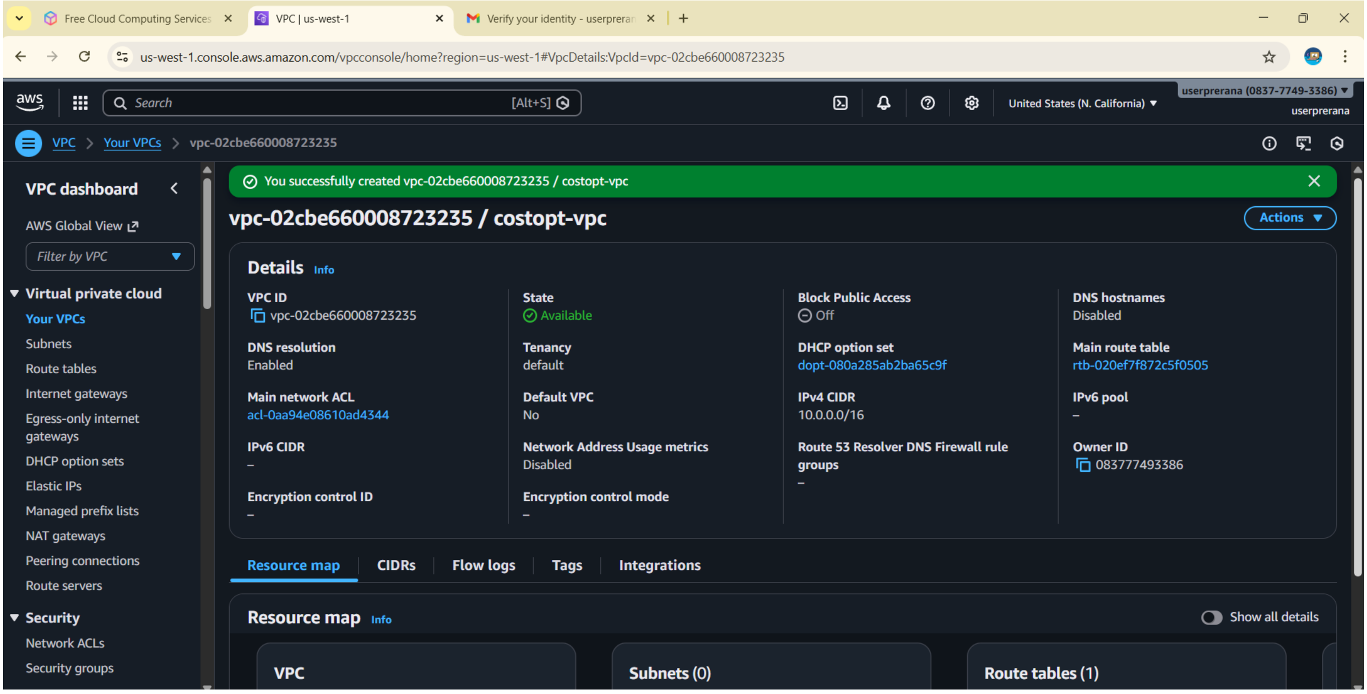
- Application Load Balancer (ALB)
- Auto Scaling Group (ASG)
- Scale based on CPU utilization

Region: us-west-1 (N. California)

Step-by-Step Implementation (with Screenshots)

Step 1: VPC Setup

- VPC: `costopt-vpc`
- CIDR: `10.0.0.0/16`



us-west-1.console.aws.amazon.com/vpcconsole/home?region=us-west-1#VpcDetails:VpcId=vpc-02cbe660008723235

VPC > Your VPCs > vpc-02cbe660008723235

You successfully created vpc-02cbe660008723235 / costopt-vpc

vpc-02cbe660008723235 / costopt-vpc

Details Info

VPC ID vpc-02cbe660008723235	State Available	Block Public Access Off	DNS hostnames Disabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-080a285ab2ba65c9f	Main route table rtb-020ef7f872c5f0505
Main network ACL acl-0aa94e08610ad4344	Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool –
IPv6 CIDR –	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 083777493386
Encryption control ID –	Encryption control mode –		

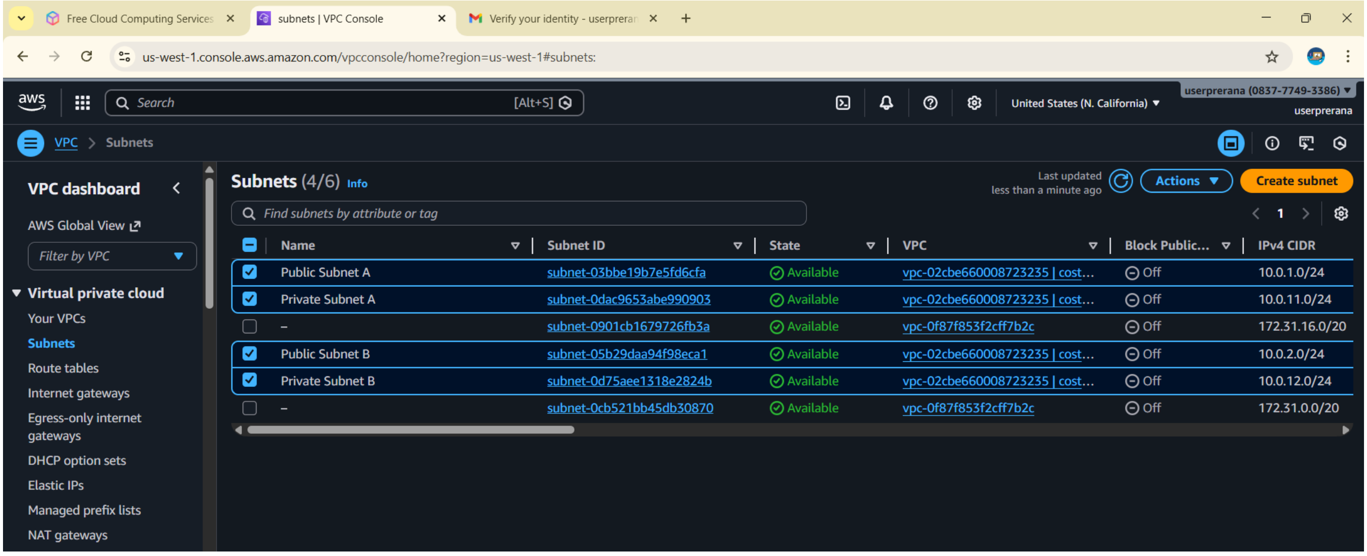
Resource map | CIDRs | Flow logs | Tags | Integrations

Resource map Info — Show all details

VPC | Subnets (0) | Route tables (1)

Step 2: Subnets

- Public A: 10.0.1.0/24
- Public B: 10.0.2.0/24
- Private A: 10.0.11.0/24
- Private B: 10.0.12.0/24



us-west-1.console.aws.amazon.com/vpcconsole/home?region=us-west-1#subnets:

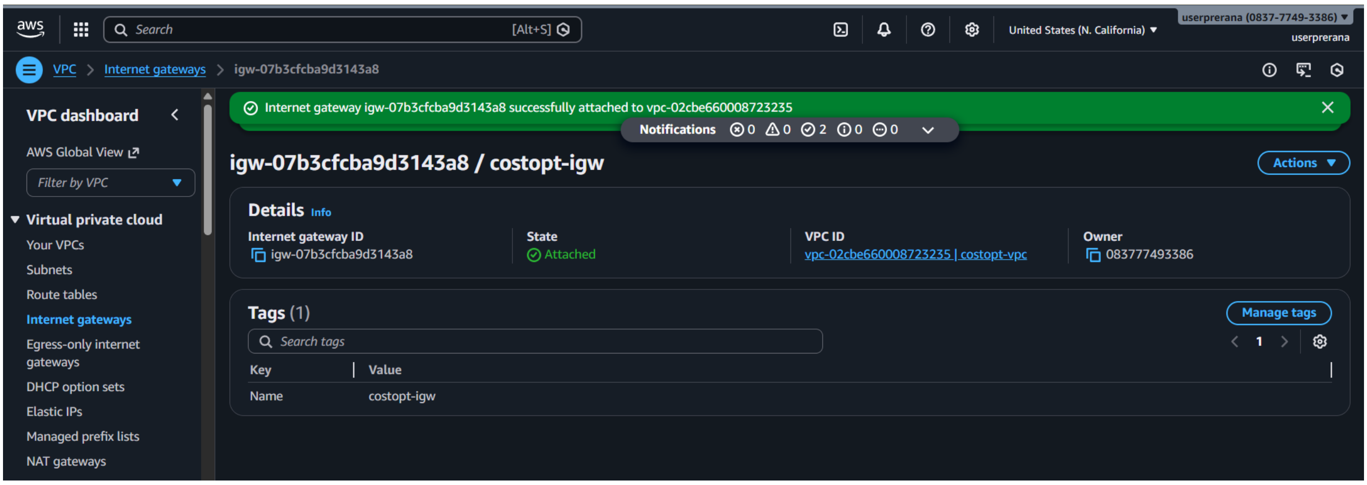
VPC > Subnets

Subnets (4/6) Info

Last updated less than a minute ago

	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
☑	Public Subnet A	subnet-03bbe19b7e5fd6cfa	Available	vpc-02cbe660008723235 | cost...	Off	10.0.1.0/24
☑	Private Subnet A	subnet-0dac9653abe990903	Available	vpc-02cbe660008723235 | cost...	Off	10.0.11.0/24
□	–	subnet-0901cb1679726fb3a	Available	vpc-0f87f853f2cff7b2c	Off	172.31.16.0/20
☑	Public Subnet B	subnet-05b29daa94f98eca1	Available	vpc-02cbe660008723235 | cost...	Off	10.0.2.0/24
☑	Private Subnet B	subnet-0d75aee1318e2824b	Available	vpc-02cbe660008723235 | cost...	Off	10.0.12.0/24
□	–	subnet-0cb521bb45db30870	Available	vpc-0f87f853f2cff7b2c	Off	172.31.0.0/20

Step 3: Internet Gateway



Internet gateway igw-07b3cfcba9d3143a8 successfully attached to vpc-02cbe660008723235

igw-07b3cfcba9d3143a8 / costopt-igw

Details Info

Internet gateway ID	State	VPC ID	Owner
igw-07b3cfcba9d3143a8	Attached	vpc-02cbe660008723235 | costopt-vpc	083777493386

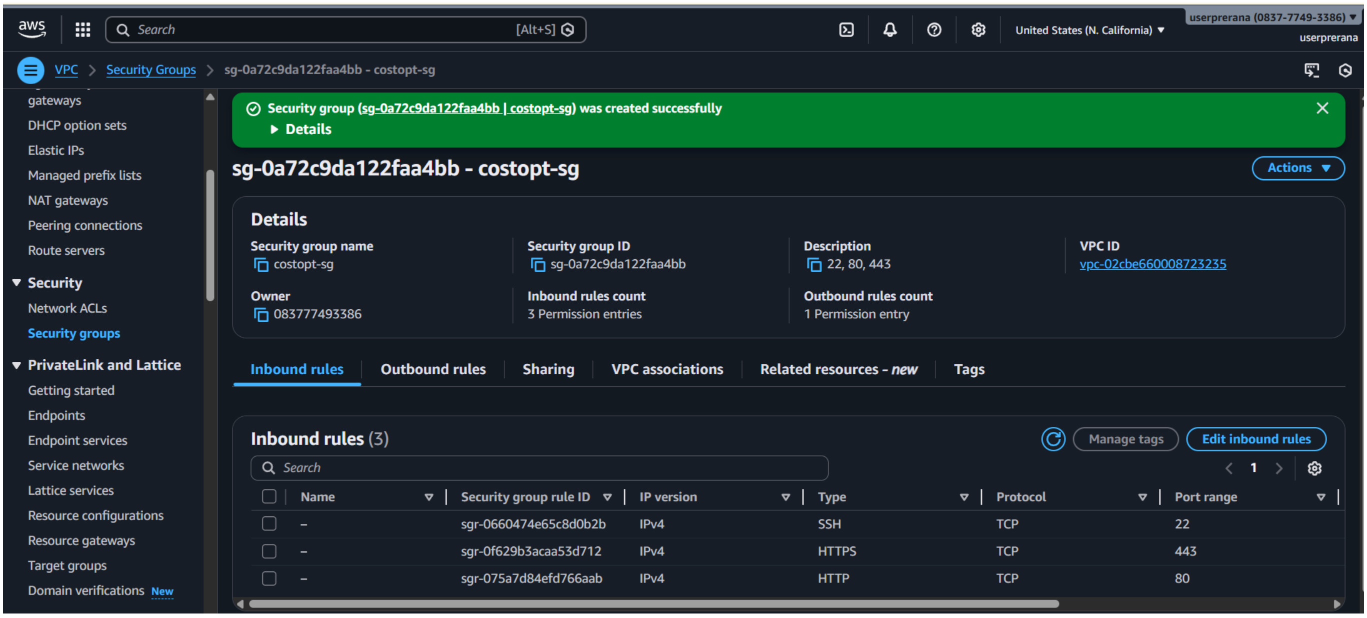
Tags (1)

Key	Value
Name	costopt-igw

Step 4: Security Group

Inbound:

- 22, 80, 443



Security group (sg-0a72c9da122faa4bb | costopt-sg) was created successfully
▶ Details

sg-0a72c9da122faa4bb - costopt-sg

Details

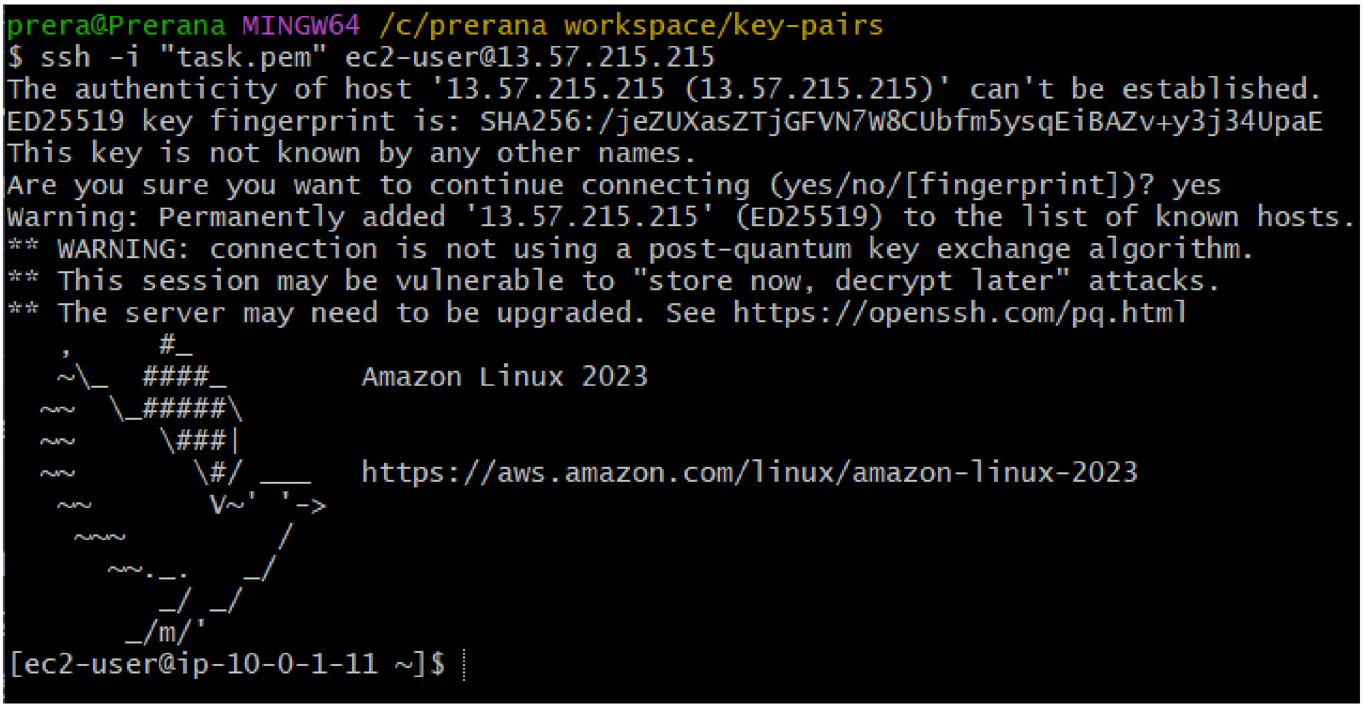
Security group name	Security group ID	Description	VPC ID
costopt-sg	sg-0a72c9da122faa4bb	22, 80, 443	vpc-02cbe660008723235

Owner	Inbound rules count	Outbound rules count
083777493386	3 Permission entries	1 Permission entry

Inbound rules (3)

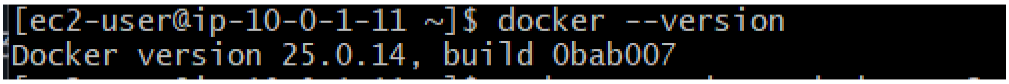
□	Name	Security group rule ID	IP version	Type	Protocol	Port range
□	–	sgr-0660474e65c8d0b2b	IPv4	SSH	TCP	22
□	–	sgr-0f629b3acaa53d712	IPv4	HTTPS	TCP	443
□	–	sgr-075a7d84efd766aab	IPv4	HTTP	TCP	80

Step 5: EC2 Access



```
prera@Prerana MINGW64 /c/prerana workspace/key-pairs
$ ssh -i "task.pem" ec2-user@13.57.215.215
The authenticity of host '13.57.215.215 (13.57.215.215)' can't be established.
ED25519 key fingerprint is: SHA256:/jeZUXasZTjGFVN7W8CUbfm5ysqEiBAZv+y3j34UpaE
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.57.215.215' (ED25519) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-11 ~]$
```

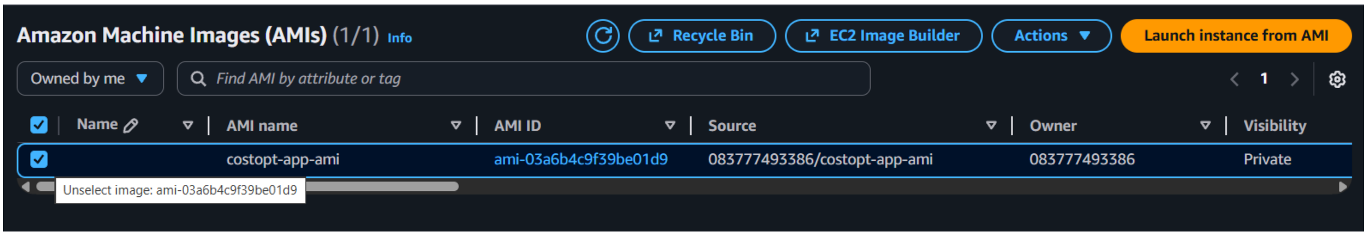
Step 6: Docker Installation



```
[ec2-user@ip-10-0-1-11 ~]$ docker --version
Docker version 25.0.14, build 0bab007
```

Step 7: Create AMI

- AMI Name: costopt-app-ami
- AMI ID: ami-03a6b4c9f39be01d9

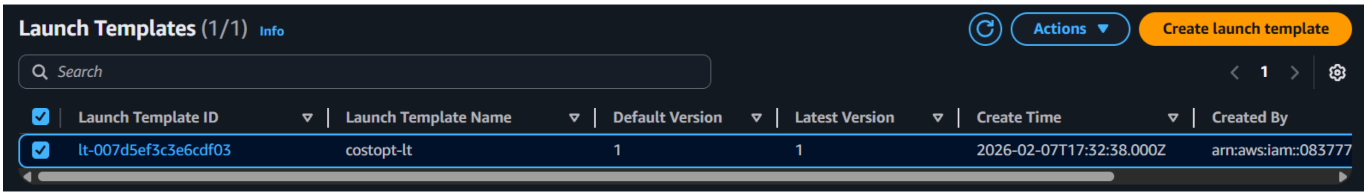


Amazon Machine Images (AMIs) (1/1)

☑	Name	AMI name	AMI ID	Source	Owner	Visibility
☑		costopt-app-ami	ami-03a6b4c9f39be01d9	083777493386/costopt-app-ami	083777493386	Private

Unselect image: ami-03a6b4c9f39be01d9

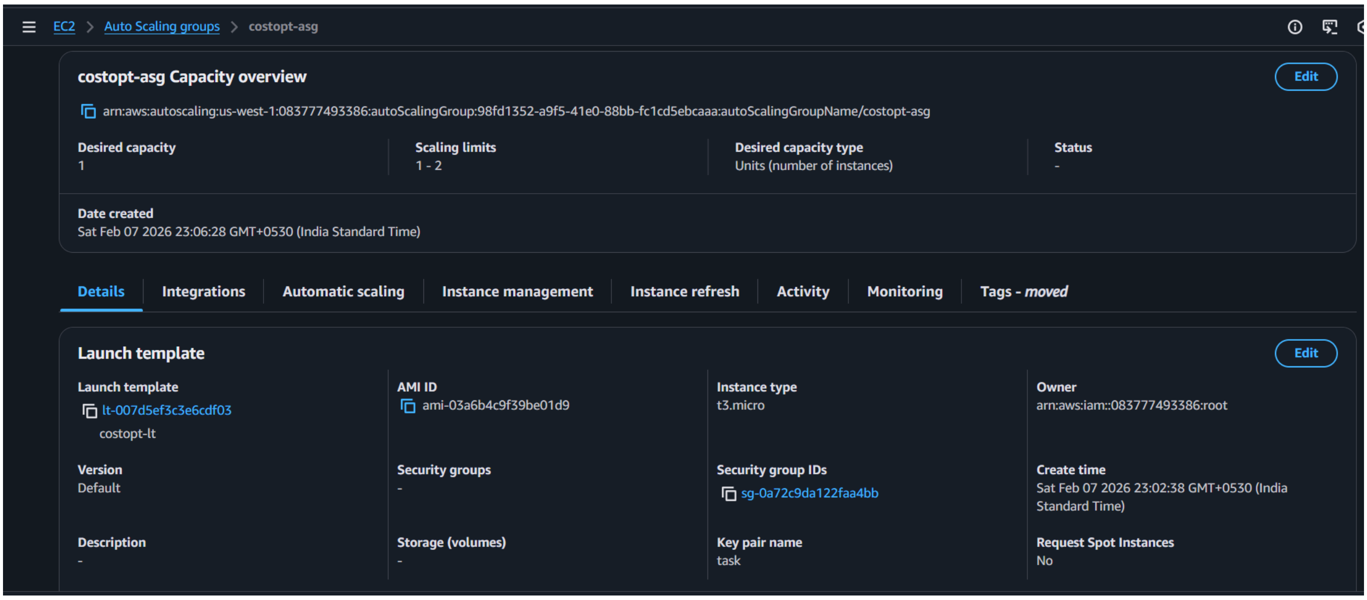
Step 8: Launch Template



Launch Templates (1/1)

☑	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
☑	lt-007d5ef3c3e6cdf03	costopt-lt	1	1	2026-02-07T17:32:38.000Z	arn:aws:iam::083777

Step 9: Auto Scaling Group



costopt-asg Capacity overview

arn:aws:autoscaling:us-west-1:083777493386:autoScalingGroup:98fd1352-a9f5-41e0-88bb-fc1cd5ebcaaa:autoScalingGroupName/costopt-asg

Desired capacity	Scaling limits	Desired capacity type	Status
1	1 - 2	Units (number of instances)	-

Date created
Sat Feb 07 2026 23:06:28 GMT+0530 (India Standard Time)

Launch template

Launch template lt-007d5ef3c3e6cdf03 costopt-lt	**AMI ID** ami-03a6b4c9f39be01d9	**Instance type** t3.micro	**Owner** arn:aws:iam::083777493386:root
Version Default	**Security groups** -	**Security group IDs** sg-0a72c9da122faa4bb	**Create time** Sat Feb 07 2026 23:02:38 GMT+0530 (India Standard Time)
Description -	**Storage (volumes)** -	**Key pair name** task	**Request Spot Instances** No

Step 10: CPU-Based Scaling Policy

- Target tracking: Average CPU utilization
- Target: 50%

Task 6 — Cost Optimization (Free-tier + Minimal + Auto Scaling)

Objective

Create a cost-optimized AWS setup by:

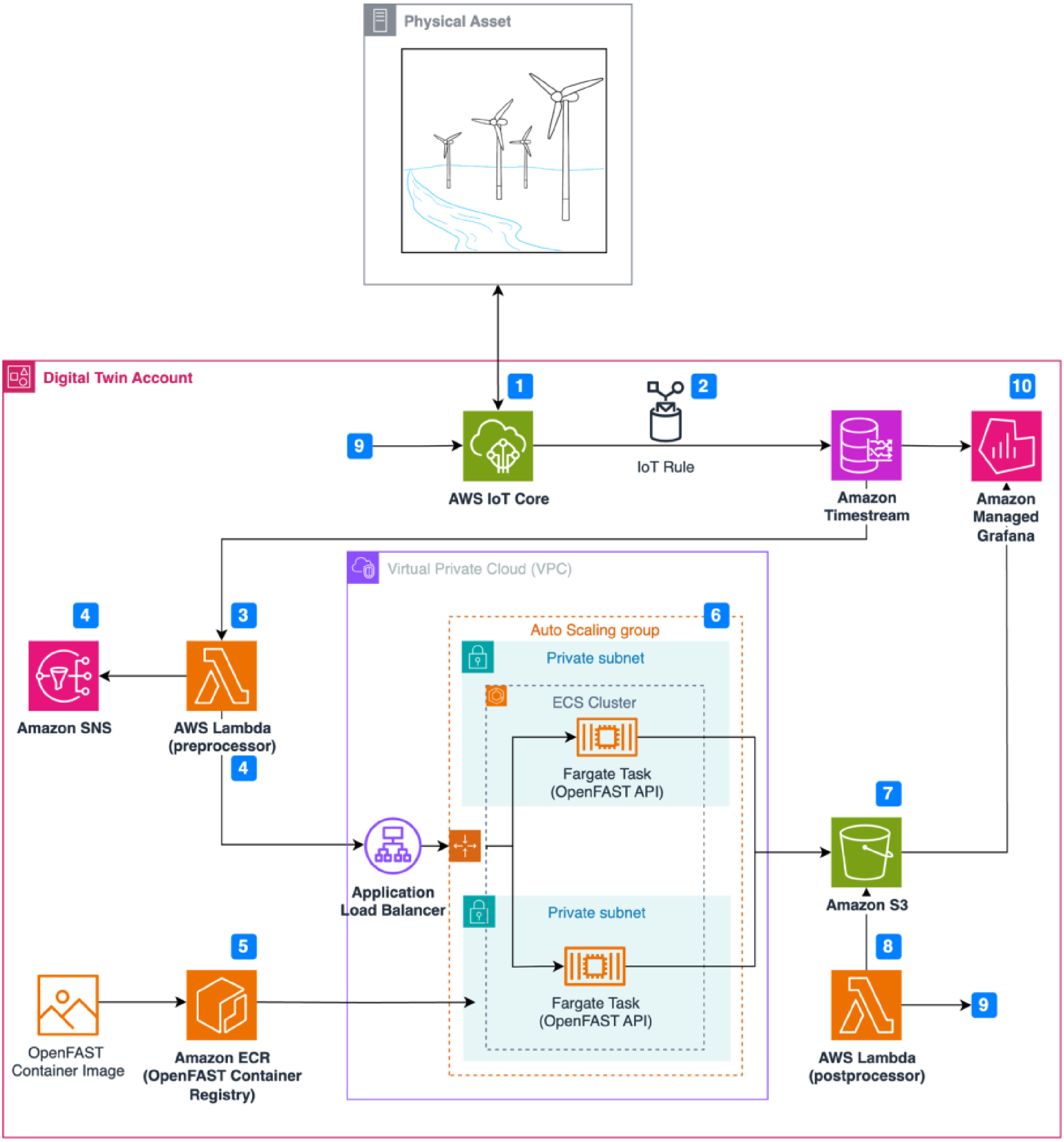
1. Using free-tier / low-cost EC2.
2. Keeping resources minimal (small instance, minimal storage, limited ports).
3. Auto Scaling based on CPU usage (avoid extra instances).

Scaling logic:

- CPU > 50% → add instance
- CPU < 50% → remove extra instance

Region: us-west-1 (N. California)

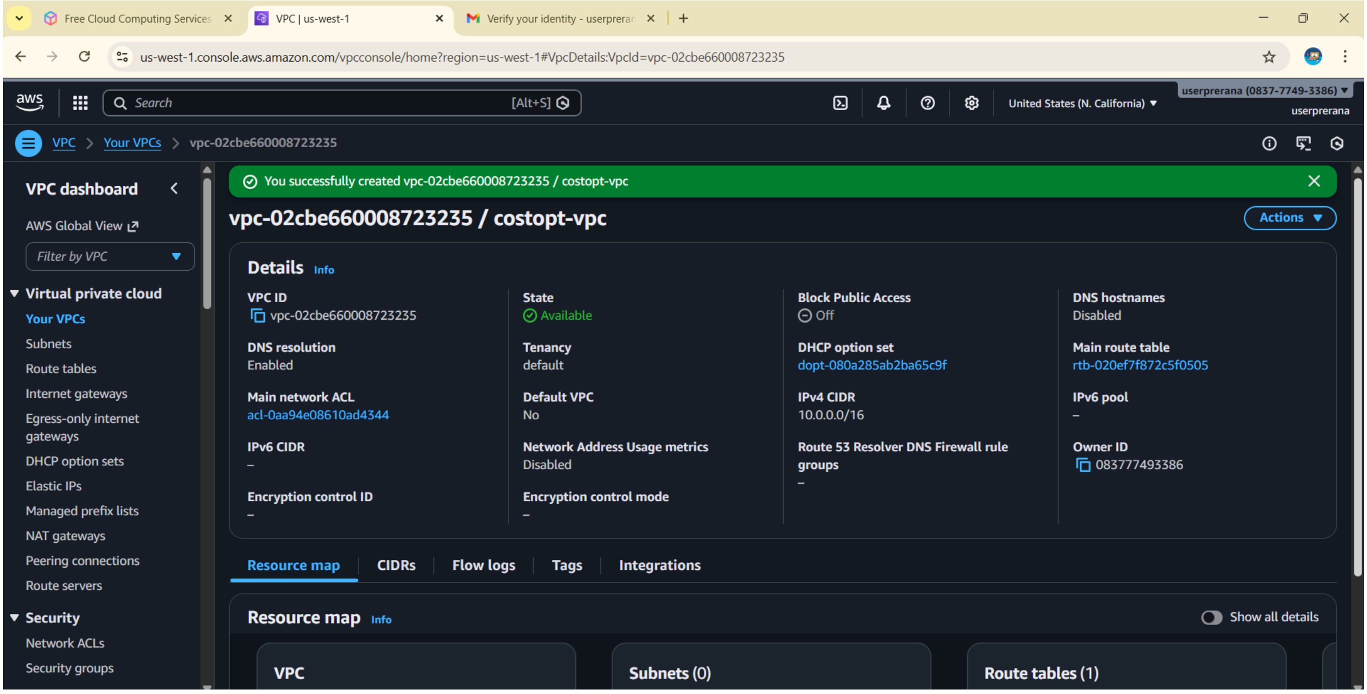
Architecture



Step-by-Step Implementation (with Screenshots)

Step 1: Create VPC

- VPC: `costopt-vpc`
- CIDR: `10.0.0.0/16`



You successfully created vpc-02cbe660008723235 / costopt-vpc

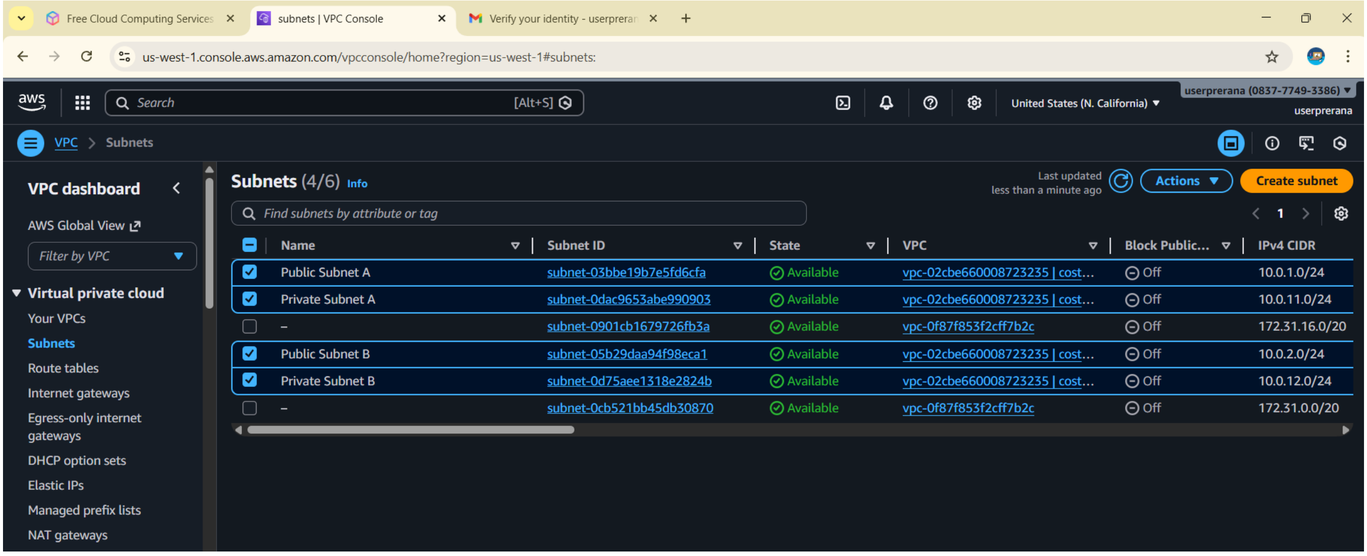
vpc-02cbe660008723235 / costopt-vpc

Details Info

VPC ID vpc-02cbe660008723235	State Available	Block Public Access Off	DNS hostnames Disabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-080a285ab2ba65c9f	Main route table rtb-020ef7f872c5f0505
Main network ACL acl-0aa94e08610ad4344	Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool –
IPv6 CIDR –	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 083777493386
Encryption control ID –	Encryption control mode –		

Resource map Info

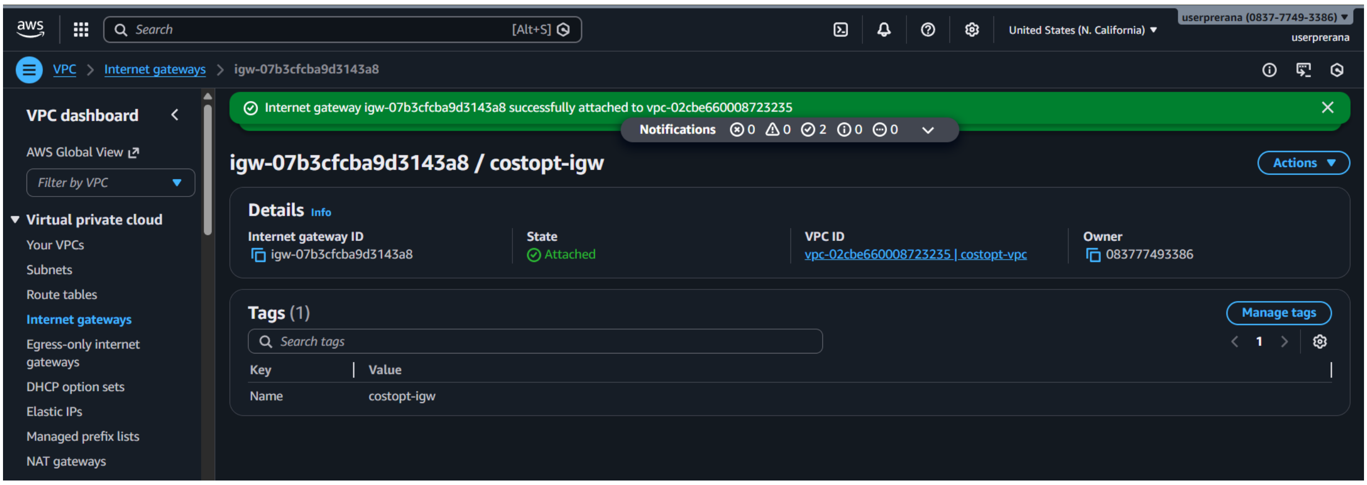
Step 2: Create Subnets (No NAT Gateway to save cost)


Free Cloud Computing Services
subnets | VPC Console
Verify your identity - userpreran
us-west-1.console.aws.amazon.com/vpcconsole/home?region=us-west-1#subnets:
aws
Search
[Alt+S]
United States (N. California) ▼
userprerana (0837-7749-3386) ▼
userprerana
VPC > Subnets
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Last updated less than a minute ago
Actions ▼
Create subnet
Find subnets by attribute or tag

Subnets (4/6) Info

	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
☑	Public Subnet A	subnet-03bbe19b7e5fd6cfa	Available	vpc-02cbe660008723235 | cost...	Off	10.0.1.0/24
☑	Private Subnet A	subnet-0dac9653abe990903	Available	vpc-02cbe660008723235 | cost...	Off	10.0.11.0/24
□	–	subnet-0901cb1679726fb3a	Available	vpc-0f87f853f2cff7b2c	Off	172.31.16.0/20
☑	Public Subnet B	subnet-05b29daa94f98eca1	Available	vpc-02cbe660008723235 | cost...	Off	10.0.2.0/24
☑	Private Subnet B	subnet-0d75aee1318e2824b	Available	vpc-02cbe660008723235 | cost...	Off	10.0.12.0/24
□	–	subnet-0cb521bb45db30870	Available	vpc-0f87f853f2cff7b2c	Off	172.31.0.0/20

Step 3: Attach Internet Gateway



Internet gateway igw-07b3cfcba9d3143a8 successfully attached to vpc-02cbe660008723235

igw-07b3cfcba9d3143a8 / costopt-igw

Details Info

Internet gateway ID igw-07b3cfcba9d3143a8	State Attached	VPC ID vpc-02cbe660008723235 | costopt-vpc	Owner 083777493386

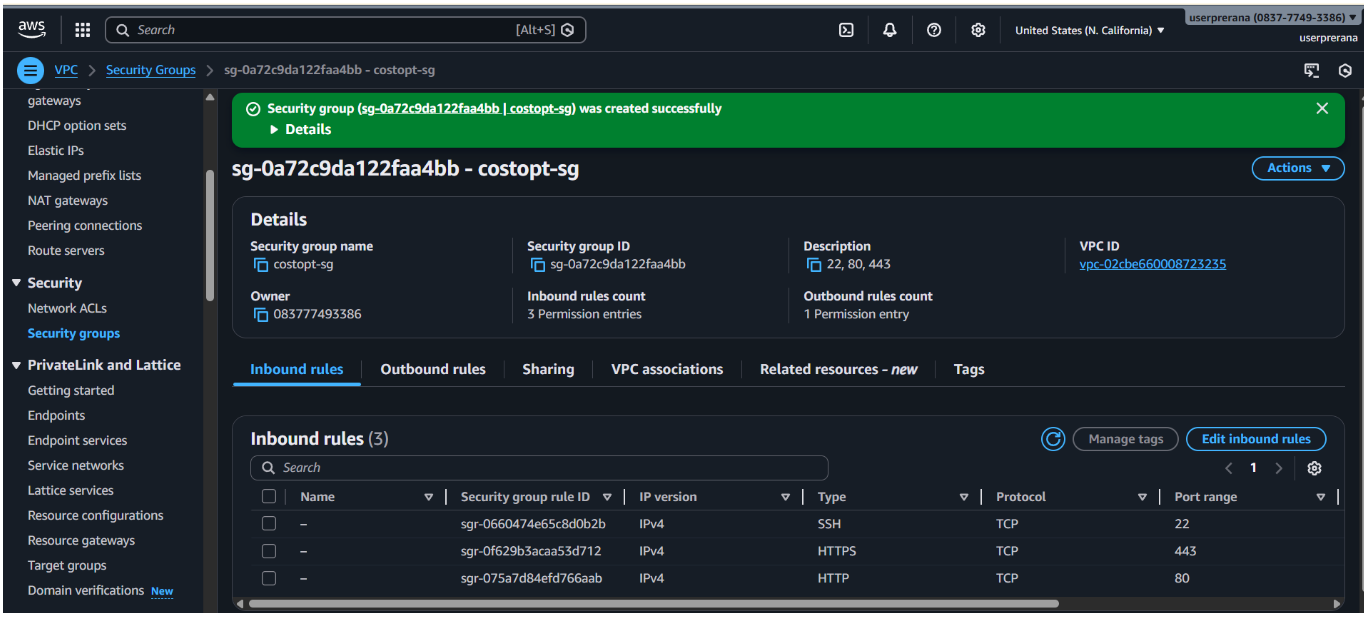
Tags (1)

Key	Value
Name	costopt-igw

Step 4: Security Group (minimal ports)

Inbound:

- 22, 80, 443 (optional)



Security group (sg-0a72c9da122faa4bb | costopt-sg) was created successfully
▶ Details

sg-0a72c9da122faa4bb - costopt-sg

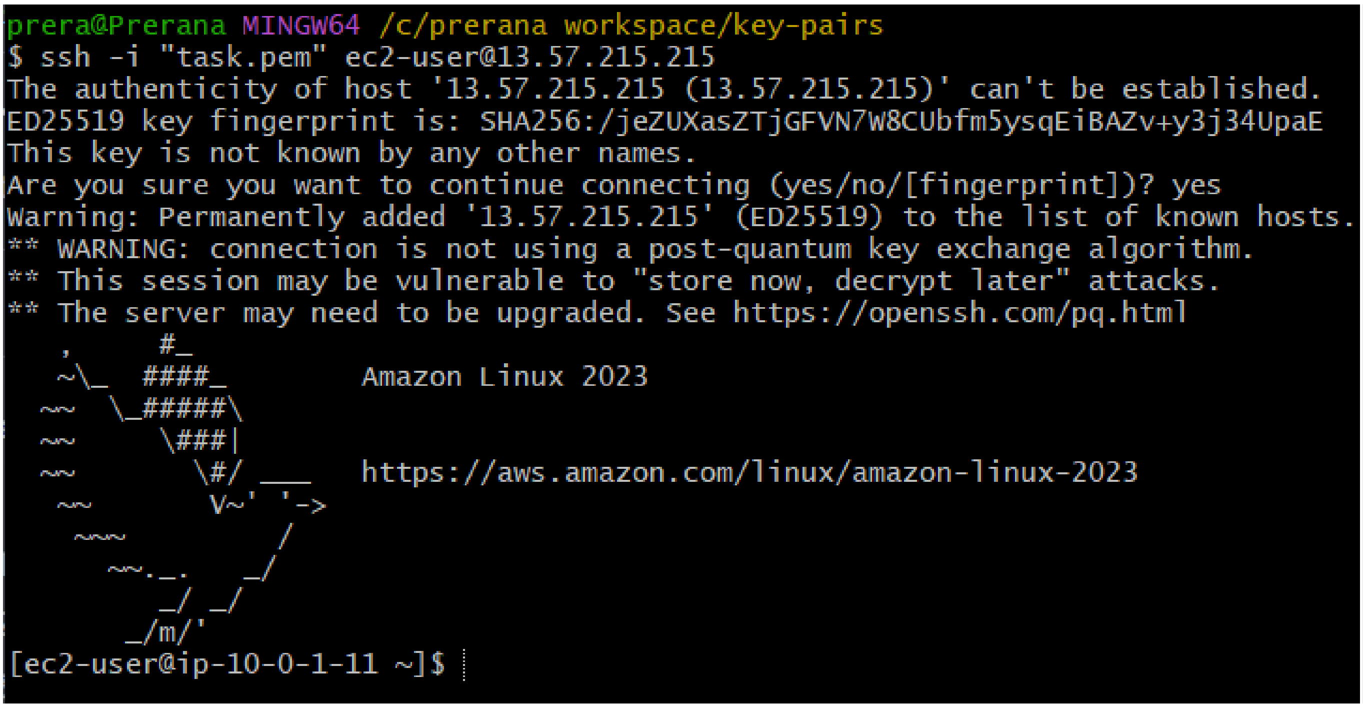
Details

Security group name: costopt-sg
Security group ID: sg-0a72c9da122faa4bb
Description: 22, 80, 443
VPC ID: vpc-02cbe660008723235
Owner: 083777493386
Inbound rules count: 3 Permission entries
Outbound rules count: 1 Permission entry

Inbound rules (3)

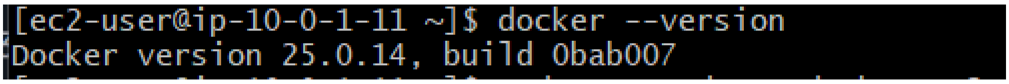
□	Name	Security group rule ID	IP version	Type	Protocol	Port range
□	–	sgr-0660474e65c8d0b2b	IPv4	SSH	TCP	22
□	–	sgr-0f629b3acaa53d712	IPv4	HTTPS	TCP	443
□	–	sgr-075a7d84efd766aab	IPv4	HTTP	TCP	80

Step 5: Launch EC2 and Verify Access



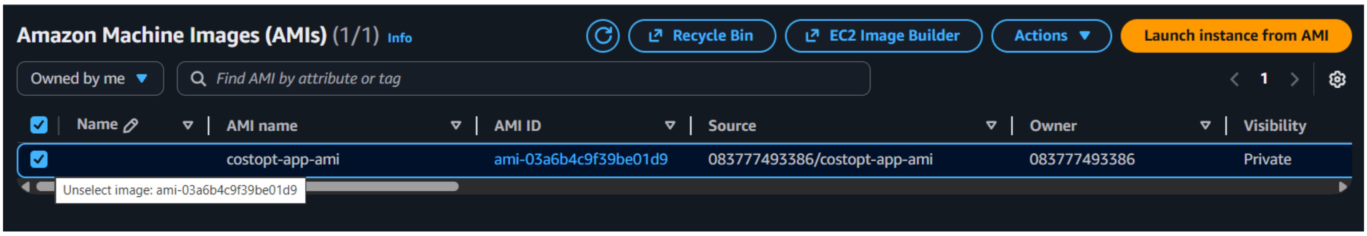
```
prera@Prerana MINGW64 /c/prerana workspace/key-pairs
$ ssh -i "task.pem" ec2-user@13.57.215.215
The authenticity of host '13.57.215.215 (13.57.215.215)' can't be established.
ED25519 key fingerprint is: SHA256:/jeZUXasZTjGFVN7W8CUbfm5ysqEiBAZv+y3j34UpaE
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.57.215.215' (ED25519) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-11 ~]$
```

Step 6: Install Docker



```
[ec2-user@ip-10-0-1-11 ~]$ docker --version
Docker version 25.0.14, build 0bab007
```

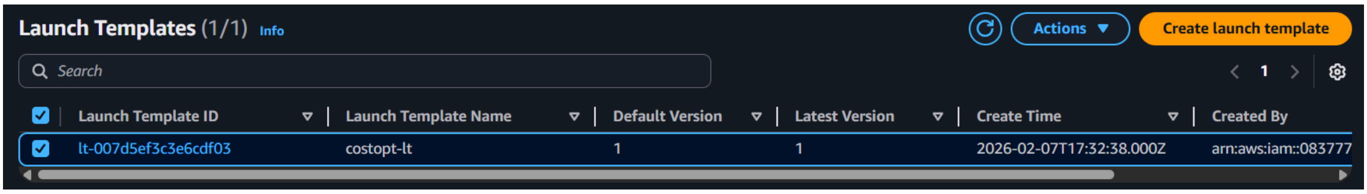
Step 7: Create AMI



Amazon Machine Images (AMIs) (1/1)

☑	Name	AMI name	AMI ID	Source	Owner	Visibility
☑		costopt-app-ami	ami-03a6b4c9f39be01d9	083777493386/costopt-app-ami	083777493386	Private

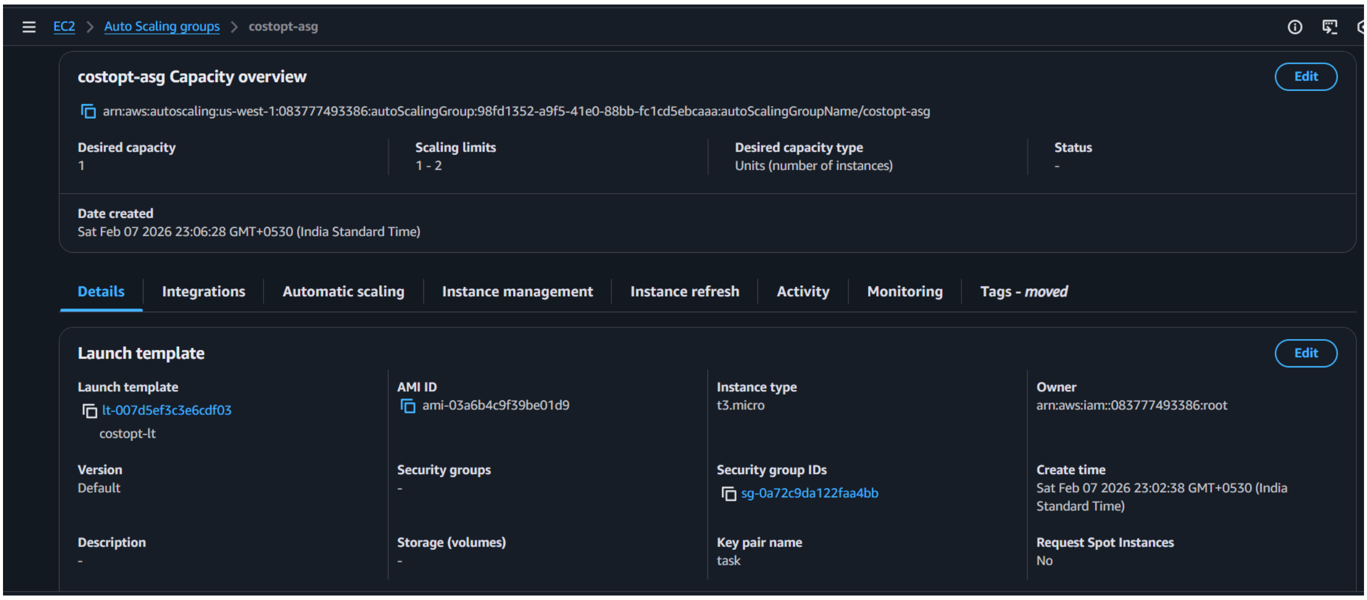
Step 8: Create Launch Template



Launch Templates (1/1)

☑	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
☑	lt-007d5ef3c3e6cdf03	costopt-lt	1	1	2026-02-07T17:32:38.000Z	arn:aws:iam::083777

Step 9: Create Auto Scaling Group



costopt-asg Capacity overview

arn:aws:autoscaling:us-west-1:083777493386:autoScalingGroup:98fd1352-a9f5-41e0-88bb-fc1cd5ebcaaa:autoScalingGroupName/costopt-asg

Desired capacity	Scaling limits	Desired capacity type	Status
1	1 - 2	Units (number of instances)	-

Date created
Sat Feb 07 2026 23:06:28 GMT+0530 (India Standard Time)

Launch template

Launch template: lt-007d5ef3c3e6cdf03 costopt-lt

AMI ID: ami-03a6b4c9f39be01d9

Instance type: t3.micro

Owner: arn:aws:iam::083777493386:root

Version: Default

Security groups: -

Security group IDs: sg-0a72c9da122faa4bb

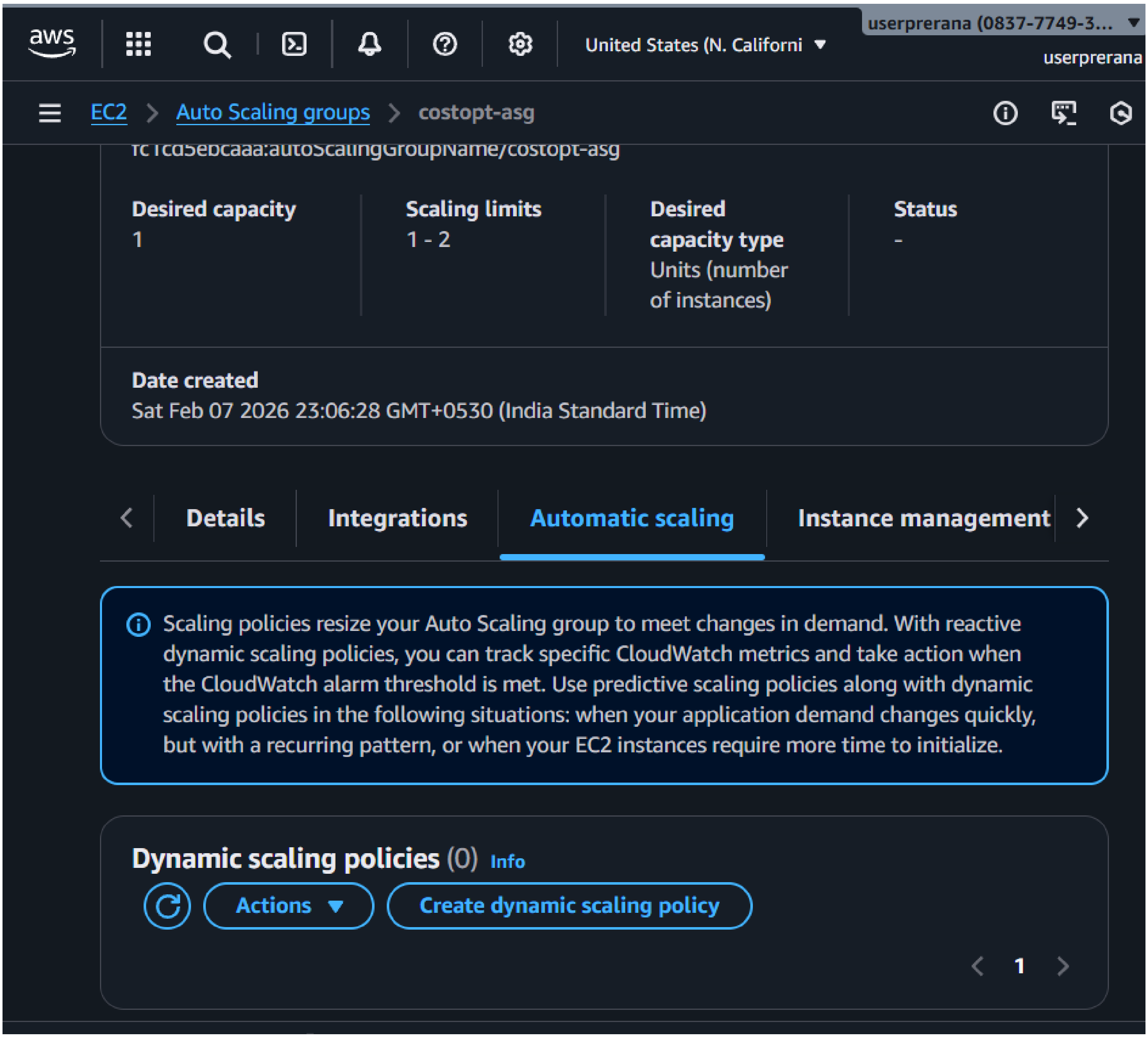
Create time: Sat Feb 07 2026 23:02:38 GMT+0530 (India Standard Time)

Description: -

Storage (volumes): -

Key pair name: task

Request Spot Instances: No



aws
United States (N. Californi
userprerana (0837-7749-3...
userprerana
EC2 > Auto Scaling groups > costopt-asg

fc1cd5ebcaaa:autoScalingGroupName/costopt-asg

Desired capacity
1

Scaling limits
1 - 2

Desired capacity type
Units (number of instances)

Status
-

Date created
Sat Feb 07 2026 23:06:28 GMT+0530 (India Standard Time)

Details
Integrations
Automatic scaling
Instance management

Scaling policies resize your Auto Scaling group to meet changes in demand. With reactive dynamic scaling policies, you can track specific CloudWatch metrics and take action when the CloudWatch alarm threshold is met. Use predictive scaling policies along with dynamic scaling policies in the following situations: when your application demand changes quickly, but with a recurring pattern, or when your EC2 instances require more time to initialize.

Dynamic scaling policies (0) Info

Step 10: CPU-Based Scaling Policy Screenshot (Required)

Add screenshot from: Auto Scaling Group → Automatic scaling → Dynamic scaling policies

---

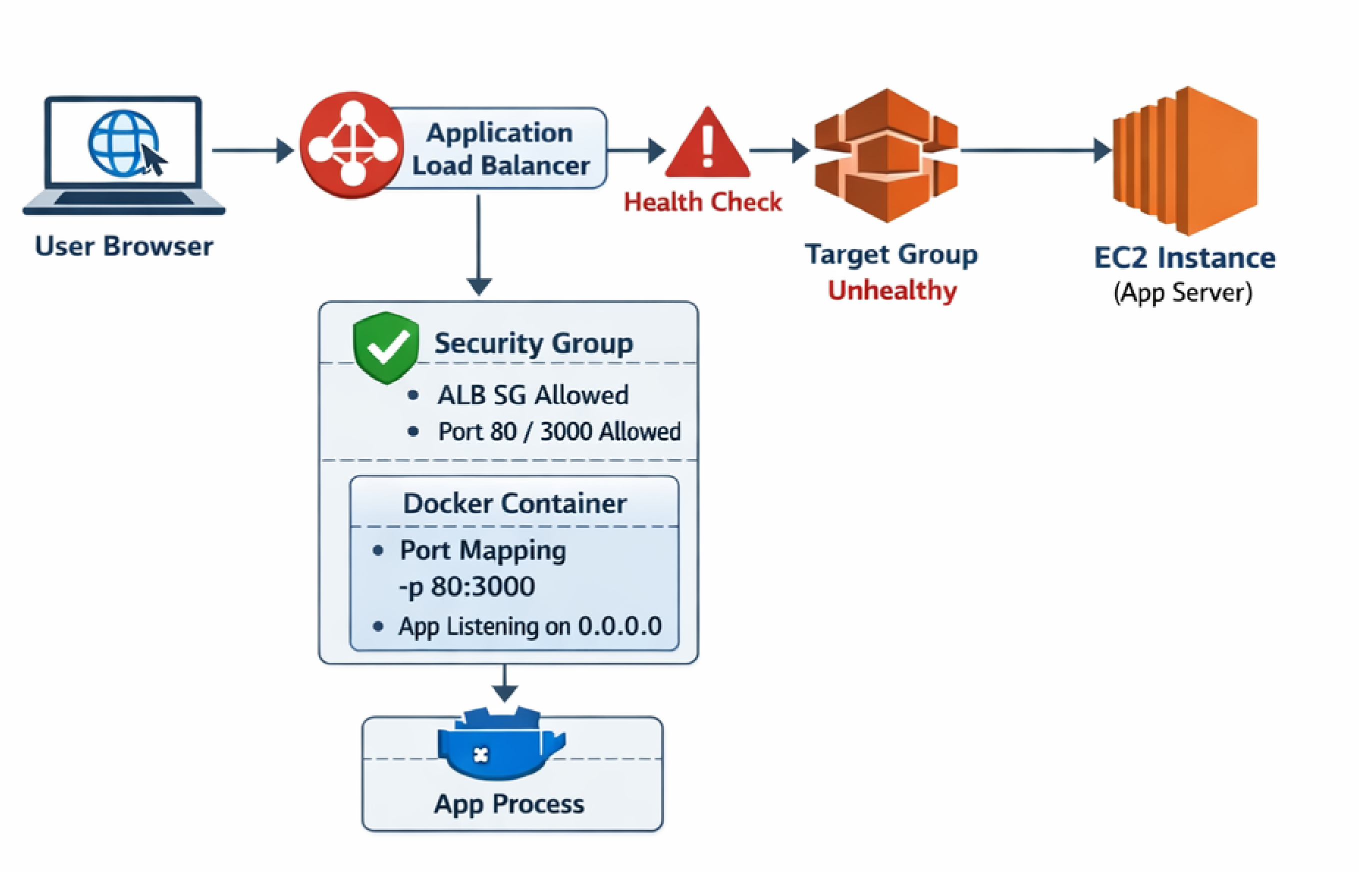
Final Notes

- Keep **all screenshots** inside `images/`
- Keep this file as **root** `README.md` for submission.

Task 7: Troubleshooting Guide

This guide explains common problems when deploying containerized applications and how to fix them in simple steps.

Architecture



1. App Not Accessible

Problem:
You open the application URL in the browser, but the page does not load.

What to check:

- **DNS / IP Address:**
  Make sure you are using the correct Load Balancer DNS name or EC2 Public IP.
- **Security Group Rules:**
  Check that inbound rules allow:
  - Port 80 (HTTP) or
  - Port 443 (HTTPS)
    from `0.0.0.0/0` or your IP address.
- **Internet Gateway:**
  If the instance is in a public subnet, ensure the VPC has an Internet Gateway attached.

---

2. Container Running but Port Not Reachable

Problem:
The Docker container status shows **running**, but the application is not accessible.

What to check:

- **Port Mapping:**
  Confirm correct port mapping. Example: -p 80:3000 This means port `80` on the host forwards to port `3000` inside the container.

- **Listening Address:**
  The application must listen on `0.0.0.0`, not `127.0.0.1`.
  If it listens on localhost, external traffic cannot reach it.
- **Container Logs:**
  Check logs to see if the app failed to start: docker logs <container_id>

---

3. ALB Health Check Failures

Problem:
The Application Load Balancer shows targets as **Unhealthy**.

What to check:

- **Health Check Path:**
  Ensure the ALB health check path matches your application endpoint.
  Example: `/` or `/health`
- **Success Response Code:**
  The app must return `200 OK`.
  Codes like `302` or `401` will cause health check failure.
- **Security Group Rules:**
  The EC2 instance security group must allow inbound traffic **from the ALB security group** on the application port.
- **Application Startup Time:**
  If the app takes time to start, increase:
- Health check interval
- Healthy threshold
  in the Target Group settings.

Conclusion

Most issues occur due to:

- Incorrect security group rules
- Wrong port mapping
- Application not listening on the correct address
- ALB health check misconfiguration

Fixing these usually resolves application access problems.