

PROJECT 2 — EC2 + NGINX + Docker + Domain Mapping + SSL

EC2 Instance with Domain Mapping and NGINX

a. Infrastructure Requirements:

i. EC2 Instance:

1. Deploy 2 **EC2 Instance** in a **Private Subnet**.
2. Attach an **Elastic IP** to the instance to ensure it has a static public IP address for domain mapping.

ii. Domain Mapping:

1. Associate with a domain name.
"ec2-docker1.<domain-name>" and
"ec2-instance1.<domain-name>"
"ec2-docker2.<domain-name>" and
"ec2-instance2.<domain-name>"

iii. Application Load Balancer (ALB) & Domain Mapping

1. Set up an **Application Load Balancer (ALB)** in **Public Subnets** to handle incoming HTTP/HTTPS traffic.
2. Set up SSL certificate. The website should not open HTTP.
3. HTTP traffic should be redirected to HTTPS.

4. Configure the ALB to associate with a domain name.
"ec2-alb-docker.<domain-name>"
"ec2-alb-instance.<domain-name>"

iv. Docker

1. Install **Docker**
2. Run a Docker container that responds with "**Namaste from Container**" on an **internal port** (e.g., 8080)

v. NGINX Configuration:

1. Install **NGINX**
2. Domain-based Content Serving:
 - a. "ec2-instance.<domain-name>" - Configure NGINX to serve the text "**Hello from Instance**"
 - b. "ec2-docker.<domain-name>" - Set up NGINX to forward requests to a Docker container running on the same instance, which serves the text "**Namaste from Container**".

vi. SSL/TLS with Let's Encrypt:

1. Set up **Let's Encrypt** to obtain the subdomain's **SSL certificate**.
2. Configure **NGINX** to use this certificate for **HTTPS** access.
Ensure that all **HTTP traffic is redirected to HTTPS** for the subdomain.

We will be using Same vpc and same subnets and same alb for this assignment so skipping vpc subnet steps

Step 1) Now we will create Security group for EC2 and ALB_EC2

ModifyInboundSecurityGroupRules

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-078bef40da7aa4b4a

Search [Alt+S]

Asia Pacific (Mumbai) Account ID: 3905-0378-1838 Shubham Mishra

Elastic Container Service

EC2 > Security Groups > sg-078bef40da7aa4b4a - ec2-sg > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

| Security group rule ID | Type | Protocol | Port range | Source | Description - optional | |
|------------------------|-------------|----------|------------|--------|------------------------|--------|
| - | All traffic | All | All | Custom | sg-058caa5d444ee378a | Delete |
| - | SSH | TCP | 22 | Custom | pl-0fa83cebf909345ca | Delete |

Add rule

Cancel Preview changes Save rules

SG for EC2

ModifyInboundSecurityGroupRules

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-058caa5d444ee378a

Search [Alt+S]

Asia Pacific (Mumbai) Account ID: 3905-0378-1838 Shubham Mishra

Elastic Container Service

EC2 > Security Groups > sg-058caa5d444ee378a - SG_ALB_FOR_EC2 > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

| Security group rule ID | Type | Protocol | Port range | Source | Description - optional | |
|------------------------|-------|----------|------------|----------|------------------------|--------|
| - | HTTP | TCP | 80 | Anywh... | 0.0.0.0/0 | Delete |
| - | HTTP | TCP | 80 | Anywh... | ::0/0 | Delete |
| - | HTTPS | TCP | 443 | Anywh... | 0.0.0.0/0 | Delete |
| - | HTTPS | TCP | 443 | Anywh... | ::0/0 | Delete |

Add rule

Rules with source of 0.0.0.0/0 or ::0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

SG for ALB

Step 2) We will launch two ec2 instances in private subnet and install nginx and docker on both the servers.

Launch instance | EC2 | ap-south-1

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: SERVER_1

Application and OS Images (Amazon Machine Image)

Search for your full catalog including 1000s of application and OS images.

Recent: Quick Start

Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM, SSD Volume Type)

Description: Ubuntu Server 24.04 LTS (HVM, SSD Volume Type). Support available from Canonical (http://www.ubuntu.com/cloud/services).

Architecture: 64-bit (x86) | AMI ID: ami-02b209d5e6b5954ef | Publish Date: 2025-10-22 | Username: ubuntu

Instance type

Instance type

Summary

Number of instances: 1

Software image (AMI): Canonical, Ubuntu, 24.04, amd64, read more

Virtual server type (instance type): t2.micro

Firewall (security group): SG_EC2

Storage (volumes): 1 volume(s) - 8 GB

Launch Instance | Preview code

Launch instance | EC2 | ap-south-1

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

Description

Ubuntu Server 24.04 LTS (HVM, SSD Volume Type). Support available from Canonical (http://www.ubuntu.com/cloud/services).

Canonical, Ubuntu, 24.04, amd64, read more

Architecture: 64-bit (x86) | AMI ID: ami-02b209d5e6b5954ef | Publish Date: 2025-10-22 | Username: ubuntu

Instance type

Instance type: t2.micro

Family: T2 | 1 vCPU | 1 GB Memory | Current generation: true | On-Demand Windows base pricing: \$0.17 USD per hour | On-Demand Linux base pricing: \$0.028 USD per hour | On-Demand Linux base pricing: \$0.024 USD per hour | On-Demand Linux base pricing: \$0.043 USD per hour | On-Demand Linux base pricing: \$0.024 USD per hour

Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required: CLOUD9DNA

Network settings

VPC - required: vpc-0c0a0f10b0e71b4b0 (CUSTOM_VPC)

Subnet: subnet-0a0f43f955a4053

Auto-assign public IP: Disable

Firewall (security groups):

Summary

Number of instances: 1

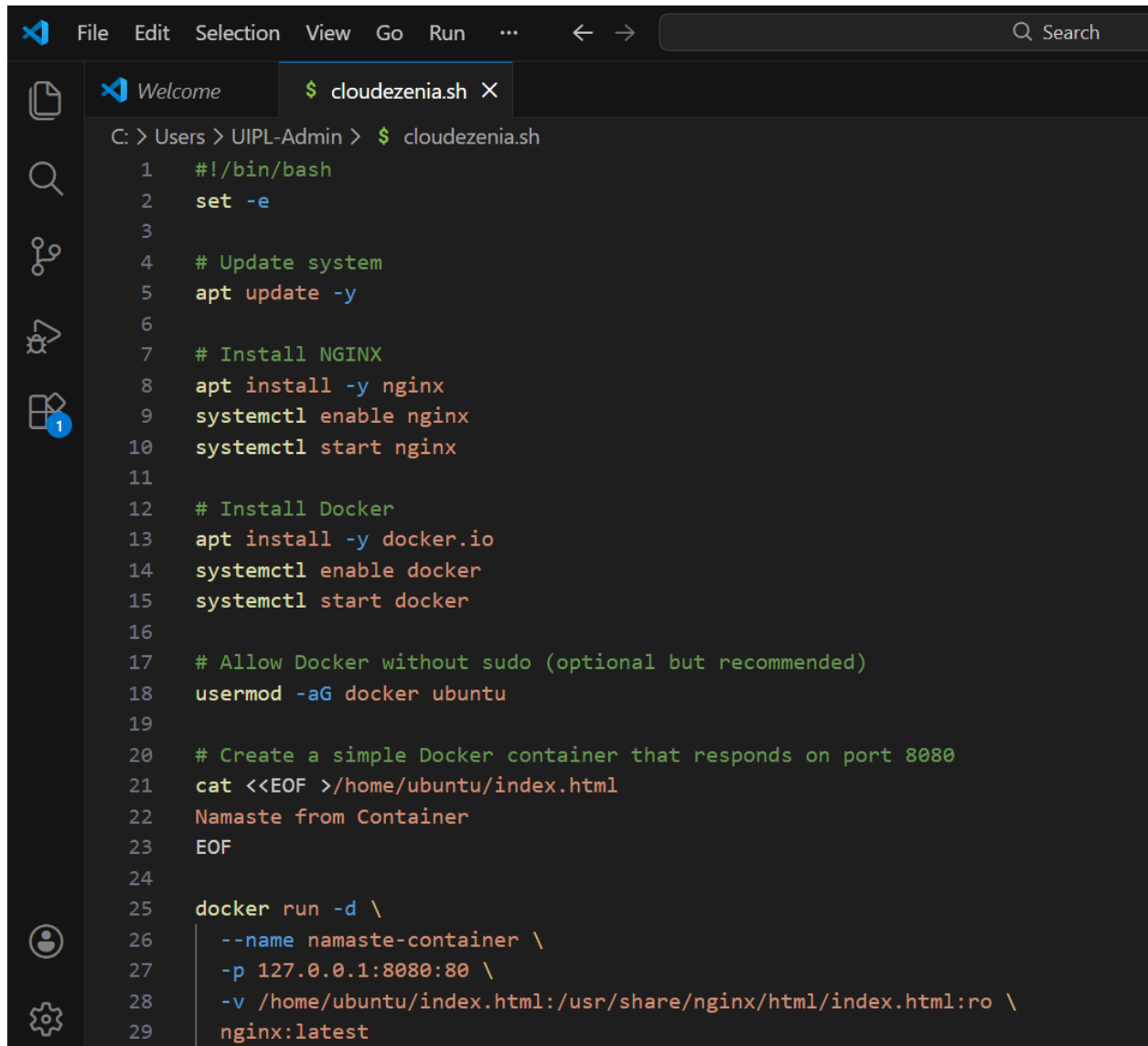
Software image (AMI): Canonical, Ubuntu, 24.04, amd64, read more

Virtual server type (instance type): t2.micro

Firewall (security group): SG_EC2

Storage (volumes): 1 volume(s) - 8 GB

Launch Instance | Preview code



```
C: > Users > UIPL-Admin > $ cloudezenia.sh
1  #!/bin/bash
2  set -e
3
4  # Update system
5  apt update -y
6
7  # Install NGINX
8  apt install -y nginx
9  systemctl enable nginx
10 systemctl start nginx
11
12 # Install Docker
13 apt install -y docker.io
14 systemctl enable docker
15 systemctl start docker
16
17 # Allow Docker without sudo (optional but recommended)
18 usermod -aG docker ubuntu
19
20 # Create a simple Docker container that responds on port 8080
21 cat <<EOF >/home/ubuntu/index.html
22 Namaste from Container
23 EOF
24
25 docker run -d \
26     --name namaste-container \
27     -p 127.0.0.1:8080:80 \
28     -v /home/ubuntu/index.html:/usr/share/nginx/html/index.html:ro \
29     nginx:latest
```

Paste this user data into servers to install during bootstrapping.

Launch an instance | EC2 | ap-south-1

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

Metadata accessible: Enabled

Metadata IPv6 endpoint: Select

Metadata version: V2 only (token required)

Metadata response hop limit: 2

Allow tags in metadata: Select

User data - optional:
☐ User data has already been base64 encoded

Summary

Number of instances: 1

Software image (AMI): Canonical, Ubuntu, 24.04, amd64, read more

Virtual server type (instance type): t2.micro

Firewall (security group): SG_EC2

Storage (volumes): 1 volume(s) - 8 GB

Launch instance

Instances | EC2 | ap-south-1

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:instanceState=running;tag:Name=SERVER_1,SERVER_2;v=3;\$....

Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running X Name = SERVER_1 X Name = SERVER_2 X

Clear filters

| | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Ava |
|--------------------------|----------|---------------------|----------------|---------------|-------------------|---------------|------|
| <input type="checkbox"/> | SERVER_2 | i-0ef8ffba0551991d1 | Running | t2.micro | 2/2 checks passed | View alarms + | ap-s |
| <input type="checkbox"/> | SERVER_1 | i-0d1c39b23ea04d9ff | Running | t2.micro | 2/2 checks passed | View alarms + | ap-s |

Step 3) Now we will launch bastion server to configure nginx on both servers in private subnets.

Launch an instance | EC2 | ap-south-1

NatGateways | VPC Console

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

EC2 > Instances > Launch an instance

Name and tags

Name

BASTION_SERVER

Add additional tags

▼ Application and OS images (Amazon Machine Image)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Q Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Debian

Browse more AMIs

Including AMIs from AWS Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), 550 Volume Type

ami-02b629d5e85954ef / ami-02b629d5e85954ef (AMI ID)

Virtualization: hvm - ENA enabled: true - Root device type: ebs

Description

Ubuntu Server 24.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture

AMI ID

Publish Date

Username

Verified provider

▼ Summary

Number of instances

1

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...read more

ami-02b629d5e85954ef

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch instance

Preview code

▼ Instance type

Instance type

Launch an instance | EC2 | ap-south-1

NatGateways | VPC Console

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

EC2 > Instances > Launch an instance

Key pair name - required

CLOUDZENIA

Create new key pair

▼ Network settings

VPC - required

vpc-0c3d91d56e71b4fb7 (CUSTOM_VPC)

172.16.0.0/16

Subnet - info

subnet-0a71c6d9f130171

ALB_PUBLIC_SUBNET_1

VPC: vpc-0c3d91d56e71b4fb7 Owner: 390503781838 Availability Zone: ap-south-1a (ap1-ae1) Zone type: Availability Zone - IP addresses available: 250 CIDR: 172.16.0.0/24

Auto-assign public IP

Enable

Firewall (security group)

Create new security group

Select existing security group

Security group name - required

bastion_sg

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _ (09, 001-14, 09)

Description - required

launch-wizard-2 created 2025-12-14T13:40:18.477Z

Inbound Security Group Rules

▼ Security group rule 1 (TCP: 22, 0.0.0.0/0)

Type

Protocol

Port range

Source

Source

Description - optional

▼ Summary

Number of instances

1

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...read more

ami-02b629d5e85954ef

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch instance

Preview code

Instances | EC2 | ap-south-1

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:instanceState=running;v=3;\$case=tags:true%5C,client:false;\$...

Account ID: 3905-0378-1838

Shubham Mishra

EC2 > Instances

Instances (3) Info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Clear filters

| | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Avail... |
|--------------------------|----------------|---------------------|----------------|---------------|-------------------|---------------|----------|
| <input type="checkbox"/> | SERVER_2 | i-0ef8ffba0551991d1 | Running | t2.micro | 2/2 checks passed | View alarms + | ap-s... |
| <input type="checkbox"/> | SERVER_1 | i-0d1c39b23ea04d9ff | Running | t2.micro | 2/2 checks passed | View alarms + | ap-s... |
| <input type="checkbox"/> | BASTION_SERVER | i-0bb1e8845cfc74e2c | Running | t2.micro | Initializing | View alarms + | ap-s... |

Step 4) Now we will ssh into bashion server and copy pem file to ssh into private servers to configure nginx

Instance details | EC2 | ap-south-1

EC2 Instance Connect | ap-south-1

NatGateways | VPC Console

ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-0bb1e8845...

Account ID: 3905-0378-1838

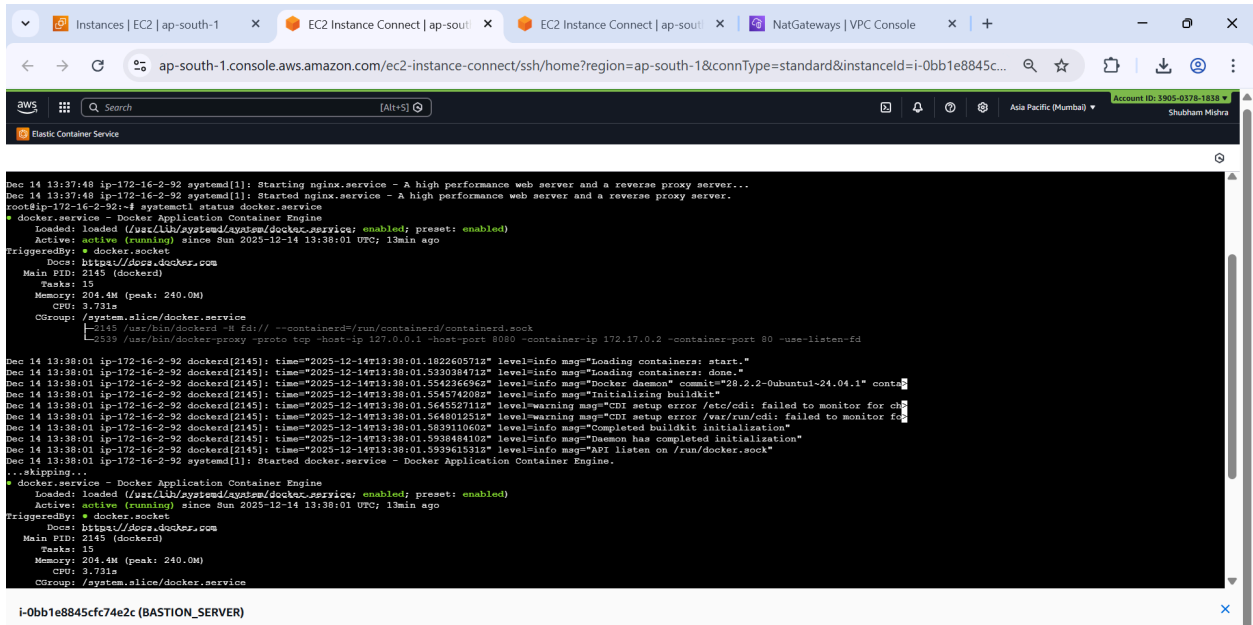
Shubham Mishra

```

AZDuLm9iYa3qe1jBQC1s4HbsWDhOIWX/V+T16TtfgAUWw2h9ZvwWogQ7ZsM+mUrd
Cffa6oenk2AabLuUeMH/kwMGoRMLdhnNYUwWgThyfcX9v2JBngEEfoATSo3WmByO
YN2RaKbE4Sg19tE8z8zwca9D7EZON+VnhQ5ja+b4Xd7p0p6+H9fpNUHv1P11TQaC
luRePLS/H95z8BHyx9Nmap20AEcgk1ka1d4ex/v41RPqQDKbkl4xNfxb494GKz
NLPte0jBAoGBA091FueS81Fgh3yvwIP1UziJpxBOU+eVuedAaHEBP1iGR4KkZecq
a+XJMSqfCeuQLICGqufabGYxj1kaNHZc8pMNOXvP+K0FH8IX66PKigSVWhJYqjUD
jpJpc/jy2QtzBn8I/axqMyImogweGyE9rthOYe2bdkKr+Uj5a0kaCOXaxAoGBAN+E
OV7MR2pcUfJbs+U0S1wXRgy6hHORJv6hMD1UIM7wUshY+rVJDLkWCfS15Aq2SxyZ
HtydghX8OTWwVowwvaOefsm3lFPhFoCm1Ctw/0x2dL3vY1hJF+L+egkl1Bfdi2R
VXx67xfTKGJSKUEBIK1C11gnuluf5mo1/5tphU1AoGAJ0z6OI804MfqYD42CbPR
qLt4cMXbIwHbE0oD2h31T1wkC8gWlpvUitG/v5aCY2i01ZjGH/AM0a4awxyZnkFG
FXwbTacytPzdXN0i36GVjPEkueDyeGeQFVbz3A+3J00KSPqgEdVNL7MV8VVT8N4W
YOM1bGYFXUCJK3RLEN46HTECgYBwvYBgGonUWolurUnF5CILVPj2TbTDCIvhCAaR
6OmusTLeeivpGF04oqVBdtTqjCbLBV01ZANynrwb6QkqXMsJQD9CycvJCxfTe2q
RJTvefY8g18hTtRXQfymf71x0Ycbu7WNKr+r6iUc6teZo29iFa8w0jDyVqdRfx2
E2VfQKBgQCTEglnYNh0aNchP1K261LwVA5hEZqCAWSrhqvtb4P1Aor1r1LKW80R
enehj8YWMVgH4JmSD1WZdobtLZYpPxPdVRRQieZIOwXlu7VapKGVVaVsep/xqVO
uN8bJfTzAbMOH8z5NDuzdB/8+EmipTtuuuTFDxNuMpXI3vgpDNZaWQ==
-----END RSA PRIVATE KEY-----
  
```

27,29 Bot

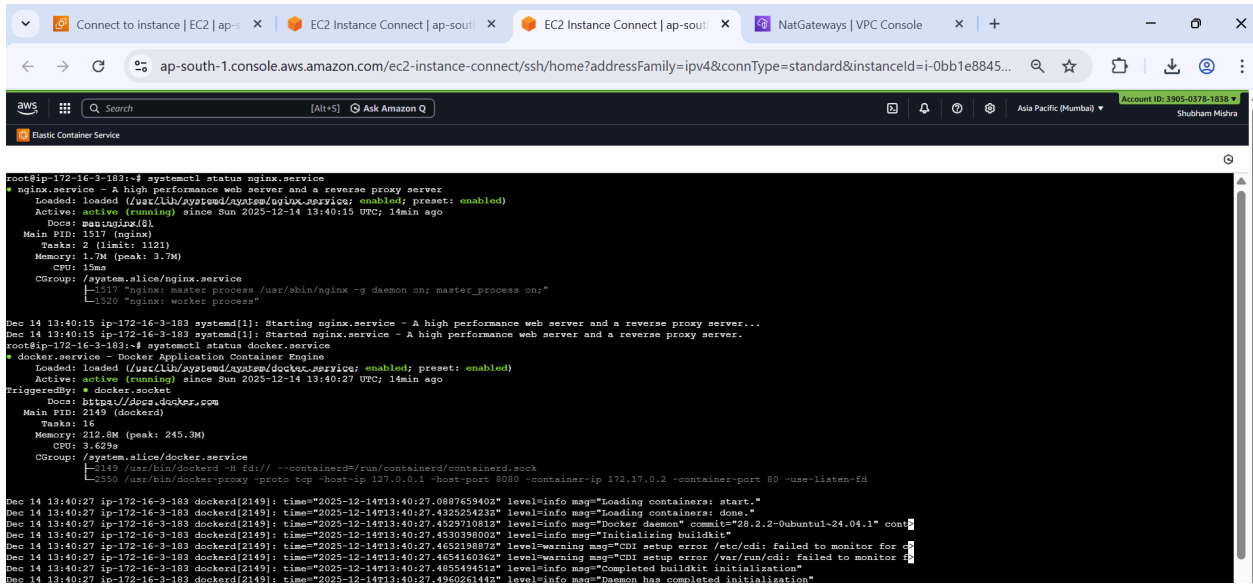
i-0bb1e8845cfc74e2c (BASTION_SERVER)



```
Dec 14 13:37:48 ip-172-16-2-92 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Dec 14 13:37:48 ip-172-16-2-92 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
root@ip-172-16-2-92:~# systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-12-14 13:38:01 UTC; 13min ago
     TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 2145 (dockerd)
     Tasks: 15
    Memory: 204.4M (peak: 240.0M)
       CPU: 3.731s
    CGroup: /system.slice/docker.service
            └─2145 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
               └─2539 /usr/bin/docker-proxy -proto tcp --host-ip 127.0.0.1 --host-port 8080 --container-ip 172.17.0.2 --container-port 80 --use-listen-fd

Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.182260571Z" level=info msg="Loading containers: start."
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.53038471Z" level=info msg="Loading containers: done."
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.55426636Z" level=info msg="Docker daemon" commit="28.2.2-0ubuntu1~24.04.1" cont
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.554574208Z" level=info msg="Initializing buildkit"
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.564552711Z" level=warning msg="CDI setup error /etc/cdi: failed to monitor for ch
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.564801251Z" level=warning msg="CDI setup error /var/run/cdi: failed to monitor fo
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.583911060Z" level=info msg="Completed buildkit initialization"
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.593848410Z" level=info msg="Daemon has completed initialization"
Dec 14 13:38:01 ip-172-16-2-92 dockerd[2145]: time="2025-12-14T13:38:01.593961531Z" level=info msg="API listen on /run/docker.sock"
Dec 14 13:38:01 ip-172-16-2-92 systemd[1]: Started docker.service - Docker Application Container Engine.
...skipping...
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-12-14 13:38:01 UTC; 13min ago
     TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 2145 (dockerd)
     Tasks: 15
    Memory: 204.4M (peak: 240.0M)
       CPU: 3.731s
    CGroup: /system.slice/docker.service
            └─2145 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
               └─2539 /usr/bin/docker-proxy -proto tcp --host-ip 127.0.0.1 --host-port 8080 --container-ip 172.17.0.2 --container-port 80 --use-listen-fd
```

Successfull ssh into server 1 and we can see nginx and docker installed successfully.



```
root@ip-172-16-3-183:~# systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-12-14 13:40:15 UTC; 14min ago
     Docs: man:nginx(8)
   Main PID: 1517 (nginx)
     Tasks: 2 (limit: 1121)
    Memory: 1.7M (peak: 3.7M)
       CPU: 15ms
    CGroup: /system.slice/nginx.service
            └─1517 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
               └─1520 "nginx: worker process"

Dec 14 13:40:15 ip-172-16-3-183 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Dec 14 13:40:15 ip-172-16-3-183 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
root@ip-172-16-3-183:~# systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-12-14 13:40:27 UTC; 14min ago
     TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 2149 (dockerd)
     Tasks: 16
    Memory: 212.8M (peak: 245.3M)
       CPU: 3.423s
    CGroup: /system.slice/docker.service
            └─2149 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
               └─2550 /usr/bin/docker-proxy -proto tcp --host-ip 127.0.0.1 --host-port 8080 --container-ip 172.17.0.2 --container-port 80 --use-listen-fd

Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.088765940Z" level=info msg="Loading containers: start."
Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.432525423Z" level=info msg="Loading containers: done."
Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.432971081Z" level=info msg="Docker daemon" commit="28.2.2-0ubuntu1~24.04.1" cont
Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.453039800Z" level=info msg="Initializing buildkit"
Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.465219887Z" level=warning msg="CDI setup error /etc/cdi: failed to monitor for c
Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.465416036Z" level=warning msg="CDI setup error /var/run/cdi: failed to monitor
Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.485549451Z" level=info msg="Completed buildkit initialization"
Dec 14 13:40:27 ip-172-16-3-183 dockerd[2149]: time="2025-12-14T13:40:27.496026144Z" level=info msg="Daemon has completed initialization"
```

Successfull ssh into server 2 and we can see nginx and docker installed successfully.

Step 5) Now we will remove default file at location etc/nginx/sites-available and create two files on both private ec2 servers

a) ec2-instance.conf

nginx

```
server {  
    listen 80;  
    server_name ec2-instance1.shubham-mishra.online;  
  
    location / {  
        return 200 "Hello from Instance\n";  
    }  
}
```

b) ec2-docker.conf

```
server {  
    listen 80;  
    server_name ec2-docker1.shubham-mishra.online;  
  
    location / {  
        proxy_pass http://127.0.0.1:8080;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
    }  
}
```

```
root@ip-172-16-2-92:/etc/nginx/sites-available# ls
ec2-docker.conf  ec2-instance.conf
root@ip-172-16-2-92:/etc/nginx/sites-available#
```

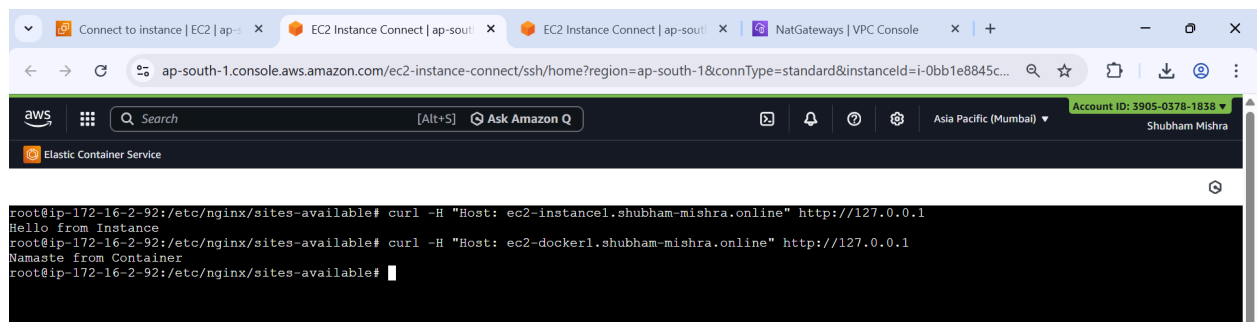
c) Then run

```
sudo nginx -t
sudo systemctl reload nginx
```

d) Then

```
curl -H "Host: ec2-instance1.shubham-mishra.online" http://127.0.0.1
curl -H "Host: ec2-docker1.shubham-mishra.online" http://127.0.0.1
```

e) Got output

A screenshot of the AWS Management Console interface. The browser tabs at the top include 'Connect to instance | EC2 | ap-south-1', 'EC2 Instance Connect | ap-south-1', and 'NatGateways | VPC Console'. The address bar shows the URL 'ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=ap-south-1&connType=standard&instanceId=i-0bb1e8845c...'. The console header displays the AWS logo, a search bar, and the account information 'Account ID: 3905-0378-1838' and 'Shubham Mishra'. The main content area shows a terminal window with the following output:

```
root@ip-172-16-2-92:/etc/nginx/sites-available# curl -H "Host: ec2-instance1.shubham-mishra.online" http://127.0.0.1
Hello from Instance
root@ip-172-16-2-92:/etc/nginx/sites-available# curl -H "Host: ec2-docker1.shubham-mishra.online" http://127.0.0.1
Namaste from Container
root@ip-172-16-2-92:/etc/nginx/sites-available#
```

Step 6) Now we will associate elastic ip as per assignment requirement to both private servers

The image displays two screenshots of the AWS Management Console, specifically the Elastic IP addresses page in the ap-south-1 region. Both screenshots show a green success message at the top: "Elastic IP address associated successfully. Elastic IP address [IP] has been associated with instance [Instance ID]".

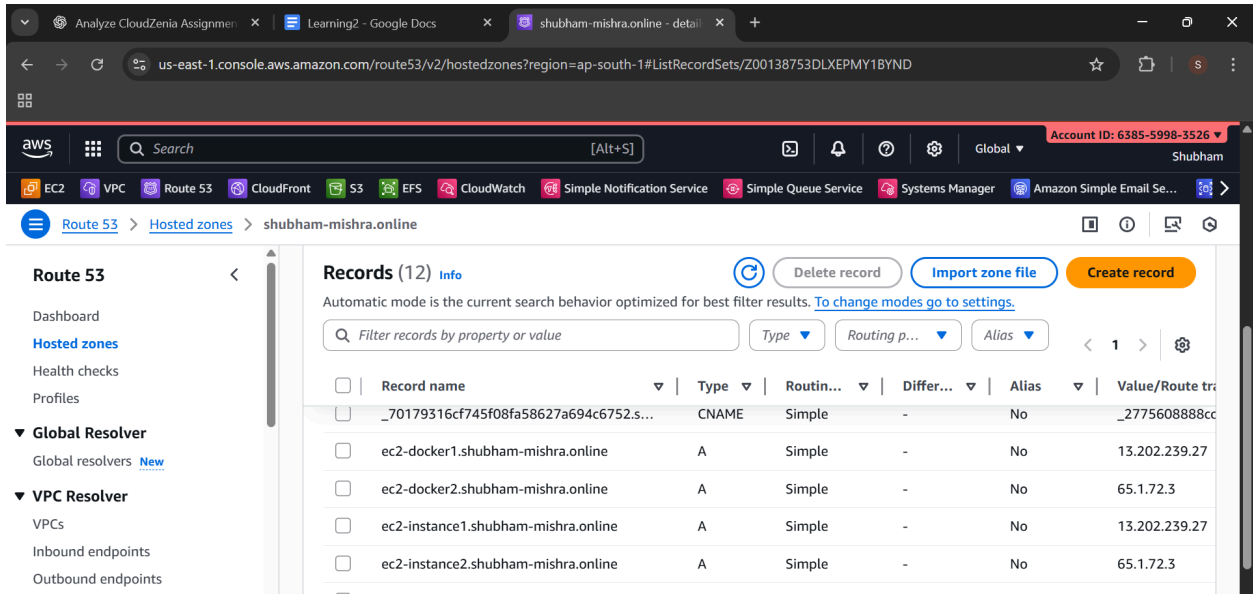
Top Screenshot: Shows the Elastic IP address 13.202.239.27 associated with instance i-0d1c39b23ea04d9ff. The summary table is as follows:

| Summary | | | |
|--|---|---|--|
| Allocated IPv4 address 13.202.239.27 | Type Public IP | Allocation ID eipalloc-0a4fc362fd3afde6d | Reverse DNS record - |
| Association ID eipassoc-06b53d51991d5daf0 | Scope VPC | Associated instance ID i-0d1c39b23ea04d9ff | Private IP address 172.16.2.92 |
| Network interface ID eni-0a81eba06e4c4bd97 | Network interface owner account ID 390503781838 | Public DNS ec2-13-202-239-27.ap-south-1.compute.amazonaws.com | NAT Gateway ID - |
| Address pool Amazon | Network border group ap-south-1 | Service managed - | |

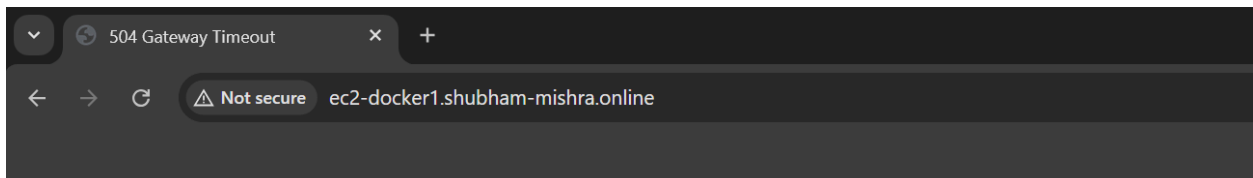
Bottom Screenshot: Shows the Elastic IP address 65.1.72.3 associated with instance i-0ef8ffba0551991d1. The summary table is as follows:

| Summary | | | |
|--|---|---|---|
| Allocated IPv4 address 65.1.72.3 | Type Public IP | Allocation ID eipalloc-0b8779428461ec645 | Reverse DNS record - |
| Association ID eipassoc-0f902f203f3b9988e | Scope VPC | Associated instance ID i-0ef8ffba0551991d1 | Private IP address 172.16.3.183 |
| Network interface ID eni-01ec24f3941ce0c5e | Network interface owner account ID 390503781838 | Public DNS ec2-65-1-72-3.ap-south-1.compute.amazonaws.com | NAT Gateway ID - |
| Address pool Amazon | Network border group ap-south-1 | Service managed - | |

Step 7) Now we will create records in route53 for the servers for domain mapping to elastic ip



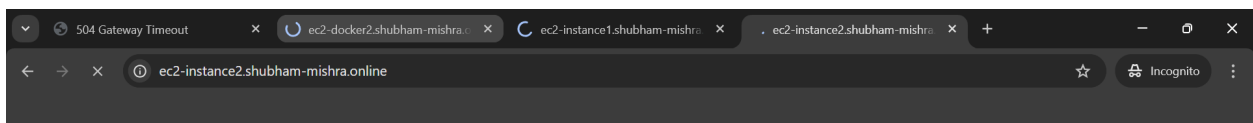
Step 8) Now we will test the record



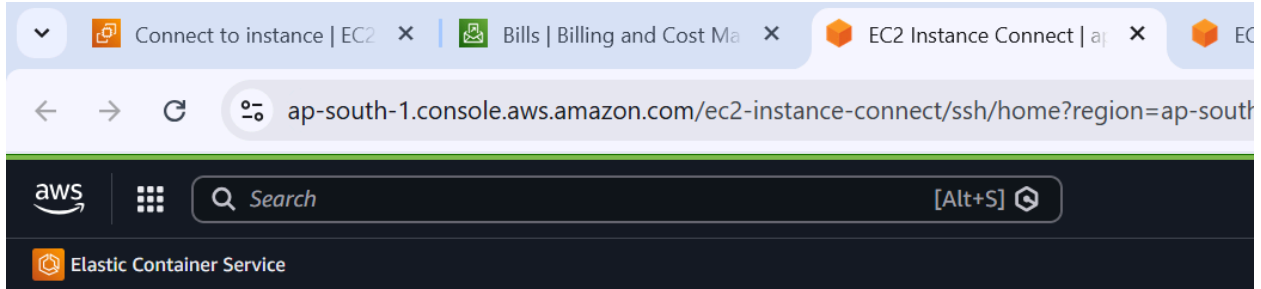
Gateway Timeout

Server error - server 13.202.239.27 is unreachable at this moment.

Please retry the request or contact your administrator.



Timeout error as elastic ip needs Internet gateway not NAT gateway so fix is either use Internet gateway or move instances in public subnet
Also modify subnet setting to allow auto assign public ip address
Then working now

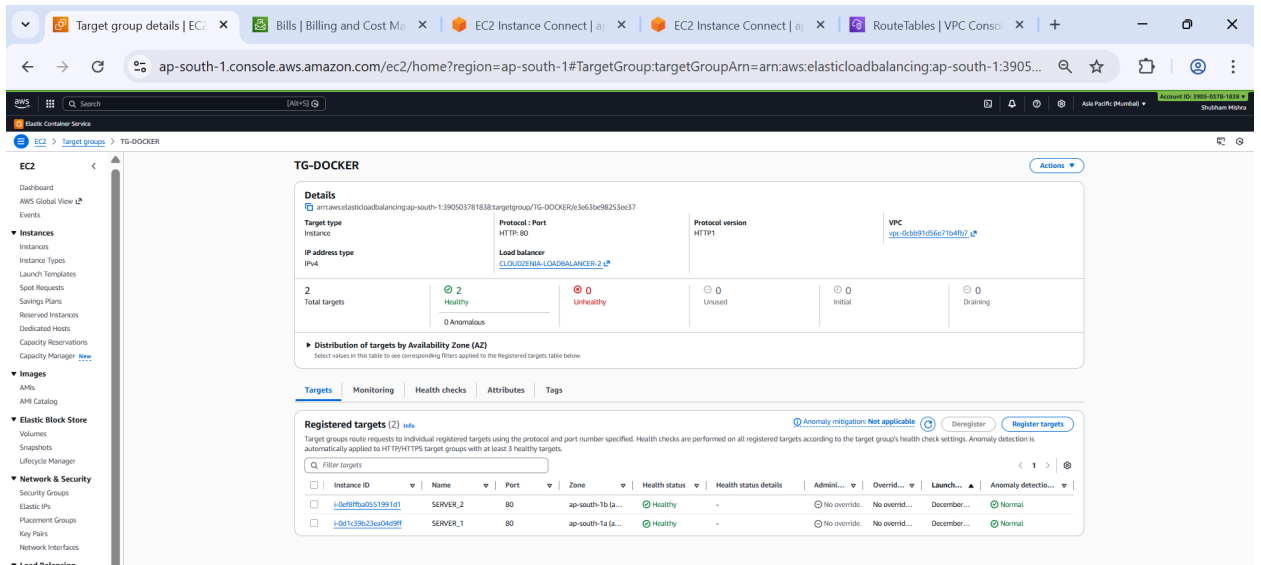


```

root@ip-172-16-0-228:~# curl http://ec2-instance1.shubham-mishra.online/
curl http://ec2-instance2.shubham-mishra.online/
curl http://ec2-docker1.shubham-mishra.online/
curl http://ec2-docker2.shubham-mishra.online/
Hello from Instance
Hello from Instance
Namaste from Container
Namaste from Container
root@ip-172-16-0-228:~#

```

Step 9) Now we will create TG_DOCKER and TG_INSTANCE FOR new ALB CLOUDZENIA-LOADBALANCER-2



The screenshot displays the AWS Management Console for the 'Target group details | EC2' page. The breadcrumb navigation shows 'EC2 > Target groups > TG-INSTANCE'. The main content area is titled 'TG-INSTANCE' and includes a 'Details' section with the following information:

- Target type:** Instance
- Protocol - Port:** HTTP-80
- Protocol version:** HTTP1
- VPC:** [vpc-d8b8f4f5a771b4b7](#)
- Load balancer:** [CLB03Z7NIA-LOADBALANCER-2](#)
- IP address type:** IPv4

Below the details, there are statistics: 2 Total targets, 2 Healthy, 0 Unhealthy, 0 Anomalous, 0 Unused, 0 Initial, and 0 Draining. A section for 'Distribution of targets by Availability Zone (AZ)' is also present.

The 'Targets' tab is active, showing a table of registered targets:

| Instance ID | Name | Port | Zone | Health status | Health status details | Administrative override | Override details |
|------------------------------------|----------|------|--------------------|---------------|-----------------------|-------------------------|-------------------------------|
| i-0d4f89ba551991d1 | SERVER_2 | 80 | ap-south-1a (a...) | Healthy | - | No override | No override is currently a... |
| i-0d4f89ba551991d1 | SERVER_1 | 80 | ap-south-1a (a...) | Healthy | - | No override | No override is currently a... |

Step 10) Now we will create new ALB

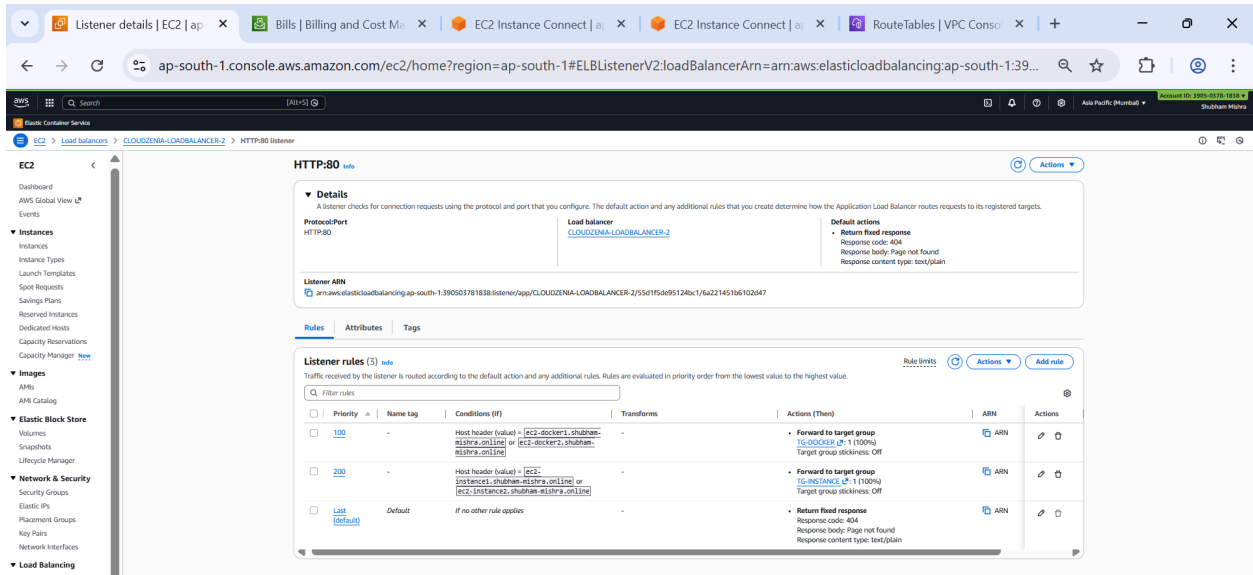
The screenshot displays the AWS Management Console for the 'Load balancer details | EC2' page. The breadcrumb navigation shows 'EC2 > Load balancers > CLOUDZENIA-LOADBALANCER-2'. The main content area is titled 'CLOUDZENIA-LOADBALANCER-2' and includes a 'Details' section with the following information:

- Load balancer type:** Application
- Status:** Active
- VPC:** [vpc-d8b8f4f5a771b4b7](#)
- Load balancer IP address type:** IPv4
- Scheme:** Internet-facing
- Hosted zone:** ZP978AM/LX7N2X
- Availability Zones:** [subnet-066547a285d59a2a](#) (ap-south-1a (ap1-a23)), [subnet-0a87f5a89f130711](#) (ap-south-1a (ap1-a21))
- Date created:** December 14, 2023, 21:16 (UTC-05:30)
- Load balancer ARN:** [arn:aws:elasticloadbalancing:ap-south-1:390503781838:loadbalancer/app/CLOUDZENIA-LOADBALANCER-2/5d41f5d695124bc1](#)
- DNS name:** [CLOUDZENIA-LOADBALANCER-2-149909915.ap-south-1.elb.amazonaws.com \(A Record\)](#)

The 'Listeners and rules' tab is active, showing a table of listeners:

| Protocol/Port | Default action | Rules | ARN | Security policy | Default SSL/TLS certificate | mTLS | Trust store |
|---------------|---|-------------------------|-----|-----------------|-----------------------------|----------------|---------------|
| HTTP/80 | Return fixed response Response code: 404 Response body: Page not found Response content type: text/plain | 3 rules | ARN | Not applicable | Not applicable | Not applicable | Not applic... |

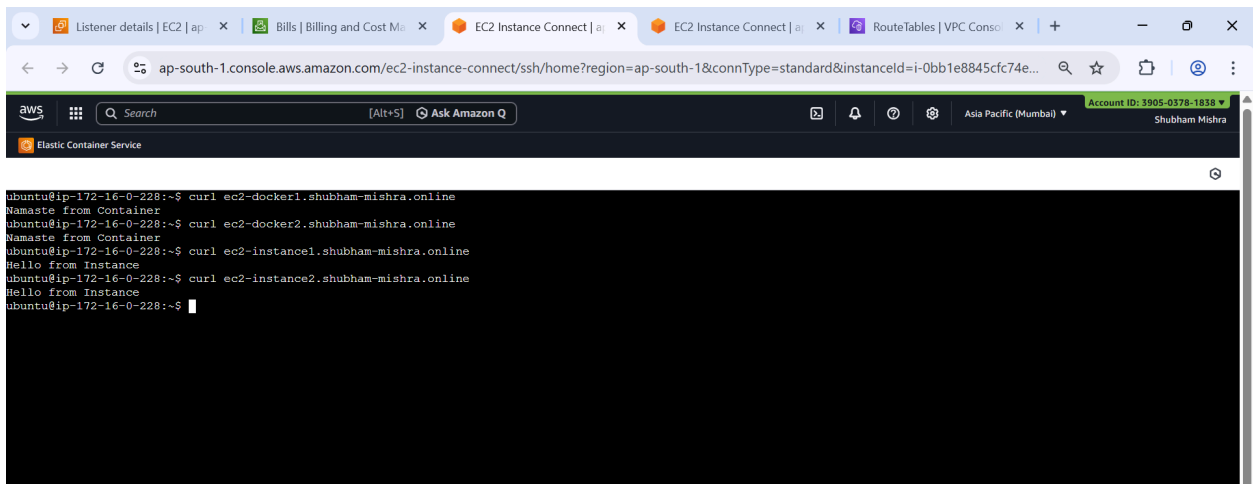
Step 11) Now we will create rules for both TG based on header



Step 12) Now we will create new records in Route 53

| | | | | | | | | |
|--------------------------|---|-------|--------|---|----|--|-----|---|
| <input type="checkbox"/> | microservice.shubham-mishra.online | CNAME | Simple | - | No | CLOUDZENIA-ALB-451697647.ap-south-1.elb.amazonaws.com | 300 | - |
| <input type="checkbox"/> | ec2-docker1.shubham-mishra.online.shubham-mishra.online | CNAME | Simple | - | No | CLOUDZENIA-LOADBALANCER-2-149909915.ap-south-1.elb.amazonaws.com | 300 | - |
| <input type="checkbox"/> | ec2-docker2.shubham-mishra.online.shubham-mishra.online | CNAME | Simple | - | No | CLOUDZENIA-LOADBALANCER-2-149909915.ap-south-1.elb.amazonaws.com | 300 | - |
| <input type="checkbox"/> | ec2-instance1.shubham-mishra.online.shubham-mishra.online | CNAME | Simple | - | No | CLOUDZENIA-LOADBALANCER-2-149909915.ap-south-1.elb.amazonaws.com | 300 | - |
| <input type="checkbox"/> | ec2-instance2.shubham-mishra.online.shubham-mishra.online | CNAME | Simple | - | No | CLOUDZENIA-LOADBALANCER-2-149909915.ap-south-1.elb.amazonaws.com | 300 | - |
| <input type="checkbox"/> | wordpress.shubham-mishra.online | CNAME | Simple | - | No | CLOUDZENIA-ALB-451697647.ap-south-1.elb.amazonaws.com | 300 | - |

Step 13) Now we will verify new records are working or not



This confirms its working correctly

Step 14) Now we will add SSL certificate for this new ALB

ap-south-1.console.aws.amazon.com/acm/home?region=ap-south-1#/certificates/bba5ef05-fd1a-4f85-b459-0bc01e4ad69c

AWS Certificate Manager (ACM)

List certificates
Request certificate
Import certificate
AWS Private CA

bba5ef05-fd1a-4f85-b459-0bc01e4ad69c [Delete](#)

Certificate status

Identifier: bba5ef05-fd1a-4f85-b459-0bc01e4ad69c
ARN: arn:aws:acm:south-1:390503781838:certificate/bba5ef05-fd1a-4f85-b459-0bc01e4ad69c
Type: Amazon Issued
Status: **Issued**

Domains (1) [Create records in Route 53](#) [Export to CSV](#)

| Domain | Status | Renewal status | Type | CNAME name | CNAME value |
|------------------------|---------|----------------|-------|--|--|
| *shubham-mishra.online | Success | - | CNAME | _70179316cf745f08fa58627a694c6752.shubham-mishra.online. | _2775608888ccf4dba5002886dd39bb77.validations.aws. |

Details

| | | | |
|-------------|---|---|---------------------|
| In use | Serial number | Requested at | Renewal eligibility |
| No | 05:bff5ca9b7c6:49:27:e1:58:f197a7:37:2eb9 | December 14, 2025, 21:30:59 (UTC+05:30) | Ineligible |
| Domain name | Public key info | Issued at | Export option |

Step 15) New SSL added in route53

Listener: HTTPS:443

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol: HTTPS **Port:** 443

Default action: [View](#)

The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Forwarding action: [View](#)

Forwarding action: [Forward to target group](#)

Health check: [View](#)

Forward to target group: [View](#)

Target group: [View](#)

Target group: [TG-DOCKERS](#) (100%) [Remove](#)

Target group: [TG-INSTANCE](#) (100%) [Remove](#)

Secure listener settings: [View](#)

Security policy: [View](#)

Security policy: [Default](#)

Default SSL/TLS server certificate: [View](#)

Default SSL/TLS server certificate: [Amazon Elastic Load Balancing - Default](#)

Client certificate handling: [View](#)

Client certificate handling: [Default](#)

Step 17) Now we will add https rule

HTTPS:443

Details

Protocol/Port: HTTPS:443

Default actions: [Forward to target group](#)

Listener ARN: [arn:aws:elasticloadbalancing:ap-south-1:990503781838:listener/app/CLOUDEZNA-LOADBALANCER-2/55d1f5de9124bc1/63099a2e4063b7be](#)

Rules **Certificates** **Security** **Attributes** **Tags**

Listener rules (3) [View](#) [Add rule](#)

Traffic routed by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

| Priority | Name tag | Conditions (If) | Transforms | Actions (Then) | ARN | Actions |
|----------------|----------|--|------------|---|----------------------|---|
| 100 | | Host header (value) = [ec2-dockers-shubham-mishra.onLine, shubham-mishra.onLine] or [ec2-dockers-shubham-mishra.onLine, shubham-mishra.onLine] | | Forward to target group TG-DOCKERS (100%) Target group stickiness: Off | View | Edit Delete |
| 200 | | Host header (value) = [ec2-instance1-shubham-mishra.onLine] or [ec2-instance2-shubham-mishra.onLine, shubham-mishra.onLine] | | Forward to target group TG-INSTANCE (100%) Target group stickiness: Off | View | Edit Delete |
| Last (Default) | Default | If no other rule applies | | Forward to target group TG-DOCKERS (100%) TG-INSTANCE (100%) Target group stickiness: Off | View | Edit Delete |

Step 18) Now we will redirect all traffic on http to https

Listener details | EC2 | ap-south-1 | Bills | Billing and Cost Manager | EC2 Instance Connect | Certificate details | bba5...

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ELBListenerV2:loadBalancerArn=arn:aws:elasticloadbalancing:ap-south-1:39...

EC2 > Load balancers > CLOUDSENA-LOADBALANCER-2 > HTTP-80 Listener

HTTP-80

Details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol/Port
HTTP-80

Load balancer
[CLOUDSENA-LOADBALANCER-2](#)

Default actions

- Redirect to HTTPS://#host:443/#path?#query
Status code: HTTP_301

Listener ARN
[arn:aws:elasticloadbalancing:ap-south-1:390505781838:listener/app/CLOUDSENA-LOADBALANCER-2/5d1f5de95124bc1f6a221451b6162d47](#)

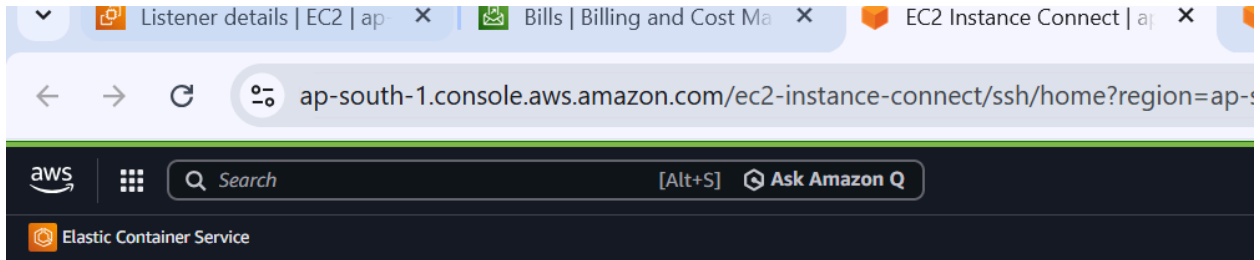
Rules | **Attributes** | **Tags**

Listener rules (3)

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

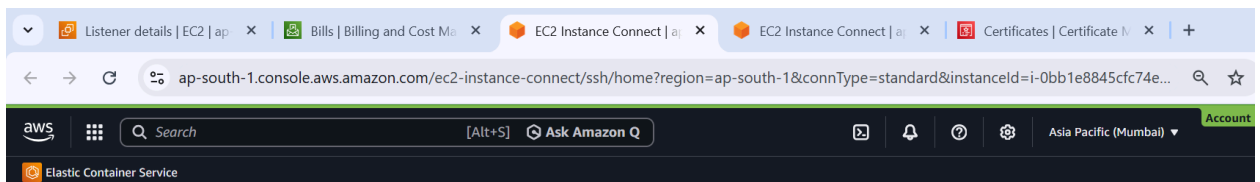
| Priority | Name tag | Conditions (If) | Transforms | Actions (Then) | ARN | Actions |
|----------------|----------|--|------------|--|---------------------|---------|
| 100 | | Host header (value) = [ec2-socket1.shubham-elphra.onlin... or ec2-socket2.shubham-elphra.onlin...] | | <ul style="list-style-type: none">Forward to target group TG-DOCKER-LB (1/100%) Target group stickiness: Off | ARN | |
| 200 | | Host header (value) = [ec2-instance1.shubham-elphra.onlin... or ec2-instance2.shubham-elphra.onlin...] | | <ul style="list-style-type: none">Forward to target group TG-INSTANCE-LB (1/100%) Target group stickiness: Off | ARN | |
| Last (default) | Default | If no other rule applies | | <ul style="list-style-type: none">Redirect to HTTPS://#host:443/#path?#query Status code: HTTP_301 | ARN | |

Step 19) This confirms http redirected to https



```
ubuntu@ip-172-16-0-228:~$ curl ec2-docker1.shubham-mishra.online.shubham-mishra.online
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
ubuntu@ip-172-16-0-228:~$ curl ec2-docker2.shubham-mishra.online.shubham-mishra.online
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
ubuntu@ip-172-16-0-228:~$ curl ec2-instance2.shubham-mishra.online.shubham-mishra.online
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
ubuntu@ip-172-16-0-228:~$ curl ec2-instance1.shubham-mishra.online.shubham-mishra.online
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
ubuntu@ip-172-16-0-228:~$
```

Step 20) Confirmation setup is successful



```
ubuntu@ip-172-16-0-228:~$ curl ec2-instance2.shubham-mishra.online.shubham-mishra.online
Namaste from Container
ubuntu@ip-172-16-0-228:~$ curl ec2-instance2.shubham-mishra.online.shubham-mishra.online
Hello from Instance
ubuntu@ip-172-16-0-228:~$ curl ec2-instance1.shubham-mishra.online.shubham-mishra.online
Hello from Instance
ubuntu@ip-172-16-0-228:~$ curl ec2-docker1.shubham-mishra.online.shubham-mishra.online
Namaste from Container
ubuntu@ip-172-16-0-228:~$ curl ec2-docker2.shubham-mishra.online.shubham-mishra.online
Namaste from Container
```

Multi-Domain Traffic Routing Using AWS ALB, EC2, Docker, and Route 53

Overview

Designed and implemented a highly available, domain-based routing architecture on AWS that cleanly separates EC2-hosted services and Docker-based microservices using Application Load Balancers and Route 53 DNS.

Key Outcomes Achieved

1. Domain-Based Traffic Segregation

- Successfully routed multiple custom domains to different backend workloads.
- EC2 instance domains returned **“Hello from Instance”**.
- Docker container domains returned **“Namaste from Container”**.
- Ensured deterministic routing using ALB listener rules and host headers.

2. Load Balancer Integration with Route 53

- Integrated Application Load Balancers with Route 53 using **A (Alias) records**.
- Eliminated direct Elastic IP exposure, enabling AWS-managed, scalable traffic handling.
- Achieved seamless DNS resolution across multiple subdomains.

3. Secure HTTPS Enablement

- Implemented TLS termination at the ALB using **AWS Certificate Manager (ACM)**.
- Enabled HTTP-to-HTTPS redirection (301) to enforce secure access.
- Used wildcard certificates to support multiple subdomains with a single certificate.

4. Backend Isolation and Clean Architecture

- Separated EC2 and Docker workloads using **dedicated target groups** and **separate ALBs** for simplicity.
- Prevented accidental traffic leakage by explicitly defining routing rules and server blocks.
- Ensured Docker services were isolated on internal ports and accessed only via NGINX proxying.

5. High Availability and Observability Readiness

- Configured health checks for all target groups to ensure fault-tolerant routing.
- Designed the setup to support future enhancements such as Auto Scaling, WAF, and CloudFront.
- Validated end-to-end flow using curl, DNS lookups, and TLS handshake verification.

Technologies Used

- **AWS Services:** EC2, Application Load Balancer (ALB), Route 53, AWS Certificate Manager (ACM)
- **Containers:** Docker, NGINX
- **Networking:** HTTP/HTTPS, TLS, DNS, Host-based routing
- **OS:** Linux (Ubuntu)

Final Result

Delivered a **secure, scalable, and interview-ready AWS architecture** demonstrating real-world troubleshooting, DNS and load-balancer integration, HTTPS configuration, and multi-service routing—aligned with production best practices.
