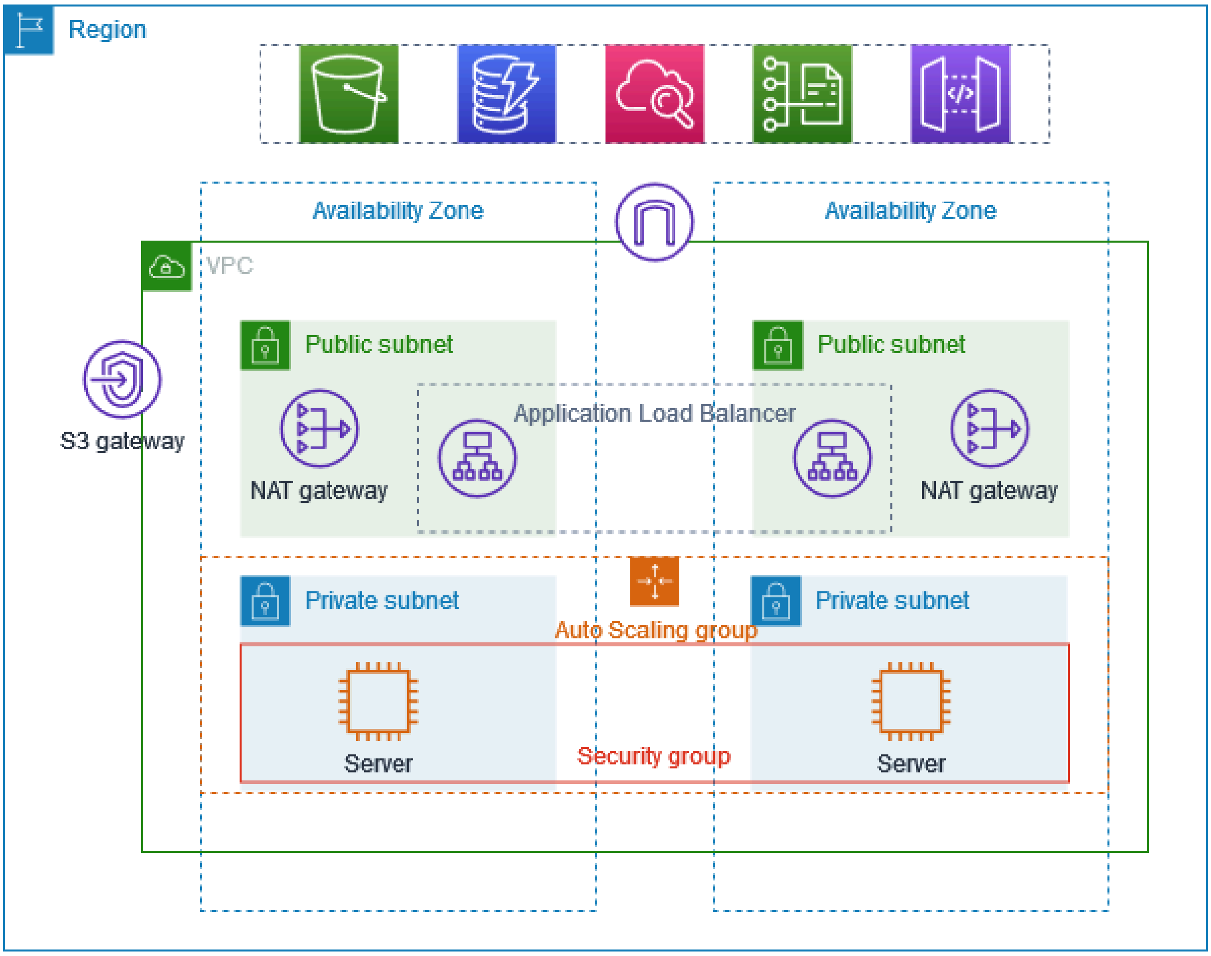
AWS VPC and Private Subnet Application Deployment

Project Overview

This project demonstrates deploying an application in a **private subnet** using AWS services, ensuring security, scalability, and high availability. The architecture includes:

- **Bastion Host** for secure access to private instances.
- **Auto Scaling Group (ASG)** with EC2 instances in private subnets.
- **Application Load Balancer (ALB)** for distributing traffic.
- **NAT Gateway** for outbound internet access from private subnets.
- **Security Groups & Route Tables** for network segmentation.



AWS Services Used

- **Virtual Private Cloud (VPC)** – Custom network for secure deployment.
- **Subnets** – Public and private subnets for network separation.
- **Internet Gateway (IGW)** – Allows public subnet resources to access the internet.
- **NAT Gateway** – Enables private instances to securely access the internet.
- **EC2 Instances** – Compute resources for the application.
- **Bastion Host** – Acts as a secure intermediary to access private instances.
- **Security Groups** – Firewall rules to control access.
- **Application Load Balancer (ALB)** – Distributes traffic across instances.
- **Auto Scaling Group (ASG)** – Ensures high availability and scalability.

Implementation Steps

1. Create a VPC

1. Navigate to the **AWS VPC Dashboard**.
2. Click **Create VPC**, provide a **name** (e.g., `aws-prod`).
3. Choose an **IPv4 CIDR Block** (e.g., `10.0.0.0/16`).
4. Click **Create VPC**.

2. Create Public and Private Subnets

1. In the **VPC Dashboard**, go to **Subnets**, click **Create Subnet**.
2. Choose the **created VPC** (`aws-prod`).
3. **Create two public subnets** in different Availability Zones (e.g., `10.0.1.0/24`, `10.0.3.0/24`).
4. **Create two private subnets** in different Availability Zones (e.g., `10.0.2.0/24`, `10.0.4.0/24`).
5. Ensure subnets are **properly tagged** for identification.

3. Attach an Internet Gateway (IGW) to the VPC

1. In the **VPC Dashboard**, go to **Internet Gateways**.
2. Click **Create Internet Gateway**, provide a **name** (e.g., `aws-prod-igw`).
3. Click **Attach to VPC**, select `aws-prod`, and confirm.

Update Route Tables:

1. **Create a new Route Table** for the **Public Subnets**.
2. Add a route: `0.0.0.0/0 → Internet Gateway`.
3. Associate this route table with the **public subnets**.

4. Create and Attach a NAT Gateway

1. Go to **NAT Gateways**.
2. Click **Create NAT Gateway**.
3. Assign it to one of the **public subnets** (e.g., `10.0.1.0/24`).
4. **Allocate an Elastic IP** and attach it to the NAT Gateway.

Modify the Private Subnet Route Tables:

1. Add a route: `0.0.0.0/0 → NAT Gateway`.
2. Associate this route table with the **private subnets**.

---

5. Create Security Groups

Bastion Host Security Group

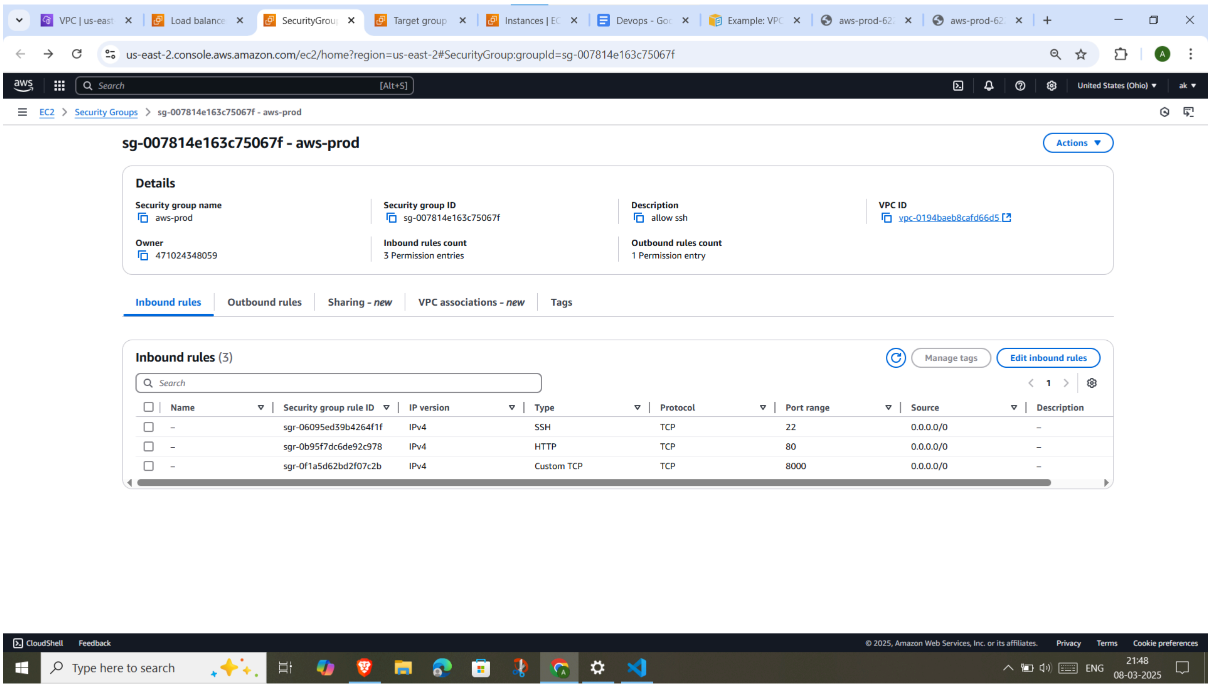
- **Inbound**: Allow SSH (`port 22`) **only from your IP**.

Private Instance Security Group

- **Inbound**: Allow SSH (`port 22`) **only from Bastion Host**.

Load Balancer Security Group

- **Inbound**: Allow HTTP (`port 80`) **from the internet**.
- **Inbound**: Allow application traffic (`port 8000`).



us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#SecurityGroup:groupId=sg-007814e163c75067f

sg-007814e163c75067f - aws-prod

Details

Security group name	Security group ID	Description	VPC ID
aws-prod	sg-007814e163c75067f	allow ssh	vpc-0194baeb8cafd66d5
Owner	**Inbound rules count**	**Outbound rules count**	
471024348059	3 Permission entries	1 Permission entry	

Inbound rules (3)

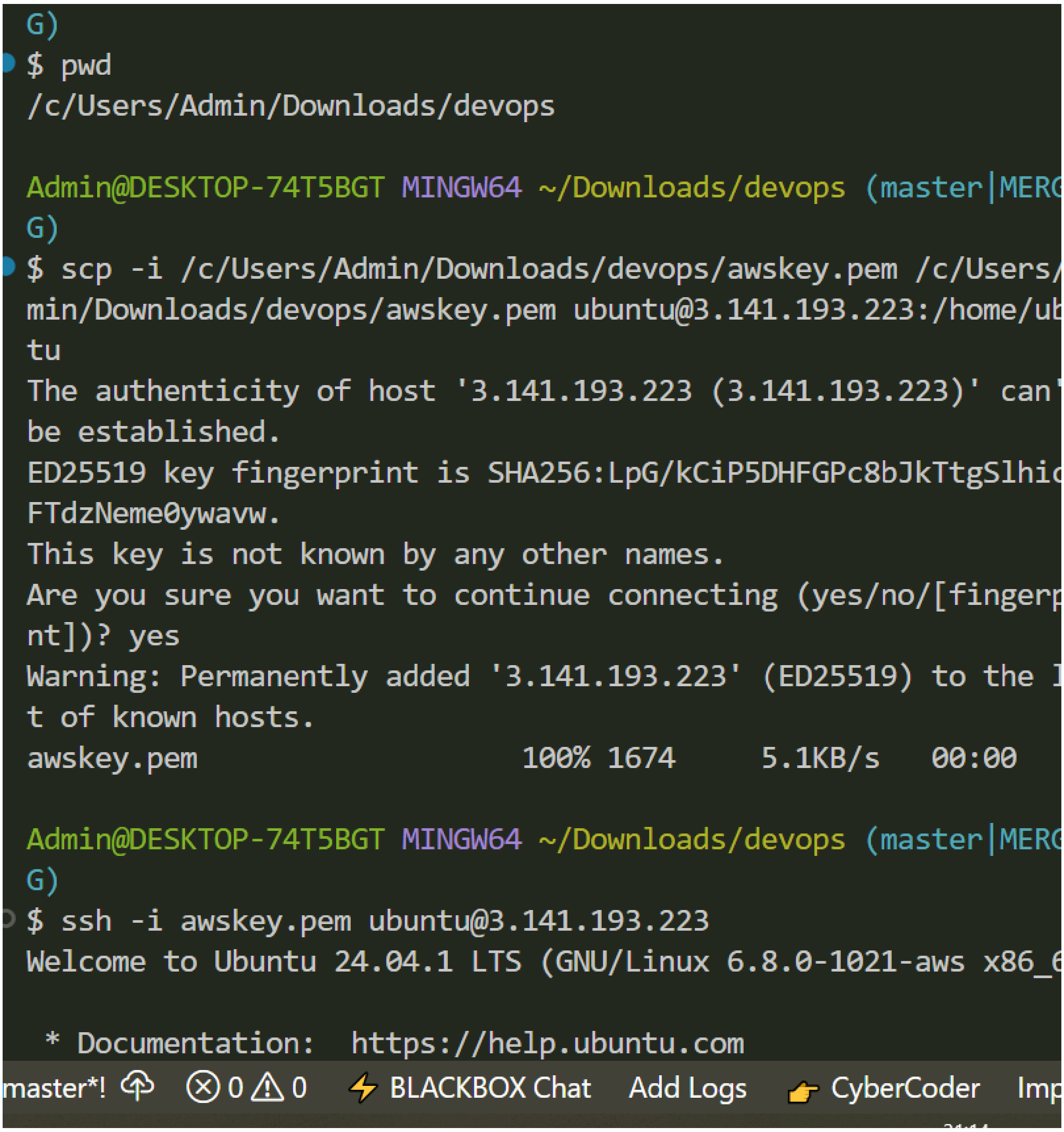
	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
□	–	sgr-06095ed39b4264f1f	IPv4	SSH	TCP	22	0.0.0.0/0	–
□	–	sgr-0b95f7dc6de92c978	IPv4	HTTP	TCP	80	0.0.0.0/0	–
□	–	sgr-0f1a5d62bd2f07c2b	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	–

6. Deploy a Bastion Host in Public Subnet

1. Launch an **EC2 Instance** in the **public subnet**.
2. Choose **Ubuntu** as the AMI.
3. Select **T2.micro** instance type.
4. Attach **Bastion Host Security Group**.
5. **Enable Auto-assign Public IP**.

SSH into the Bastion Host:

ssh -i my-key.pem ubuntu@<bastion-public-ip>



```
G)
$ pwd
/c/Users/Admin/Downloads/devops

Admin@DESKTOP-74T5BGT MINGW64 ~/Downloads/devops (master|MERG
G)
$ scp -i /c/Users/Admin/Downloads/devops/awskey.pem /c/Users/
min/Downloads/devops/awskey.pem ubuntu@3.141.193.223:/home/ub
tu
The authenticity of host '3.141.193.223 (3.141.193.223)' can'
be established.
ED25519 key fingerprint is SHA256:LpG/kCiP5DHFGPc8bJkTtgSlhic
FTdzNeme0ywavw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerp
nt])? yes
Warning: Permanently added '3.141.193.223' (ED25519) to the l
t of known hosts.
awskey.pem                    100% 1674     5.1KB/s   00:00

Admin@DESKTOP-74T5BGT MINGW64 ~/Downloads/devops (master|MERG
G)
$ ssh -i awskey.pem ubuntu@3.141.193.223
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1021-aws x86_6

 * Documentation:  https://help.ubuntu.com
```

7. Deploy Application Instances in Private Subnet

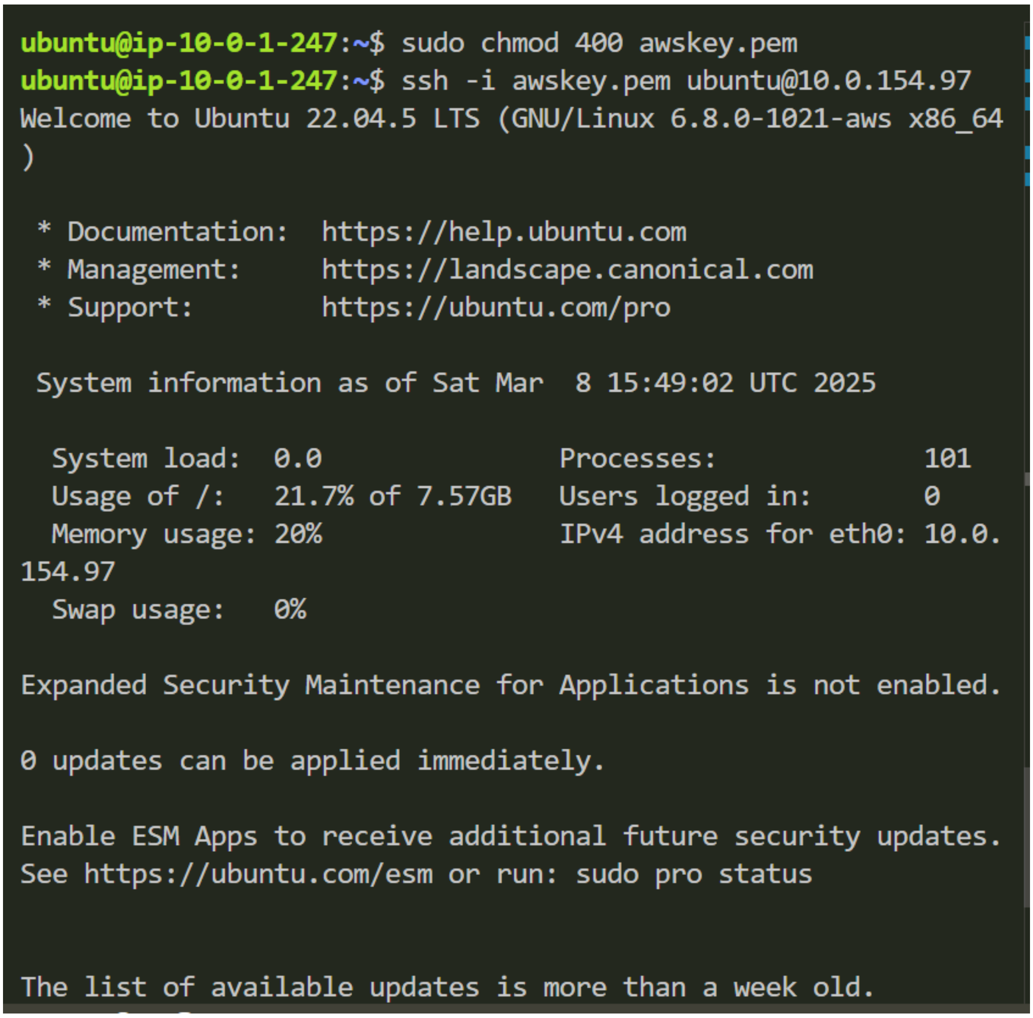
1. Launch **two EC2 instances** in the **private subnet**.
2. Attach **Private Instance Security Group**.

Copy SSH Key to Bastion Host:

scp -i my-key.pem my-key.pem ubuntu@<bastion-public-ip>:~

SSH from Bastion Host to Private Instance:

ssh -i my-key.pem ubuntu@<private-instance-ip>



```
ubuntu@ip-10-0-1-247:~$ sudo chmod 400 awskey.pem
ubuntu@ip-10-0-1-247:~$ ssh -i awskey.pem ubuntu@10.0.154.97
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1021-aws x86_64
)

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

 System information as of Sat Mar  8 15:49:02 UTC 2025

  System load:  0.0               Processes:             101
  Usage of /:   21.7% of 7.57GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for eth0: 10.0.
154.97
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
```

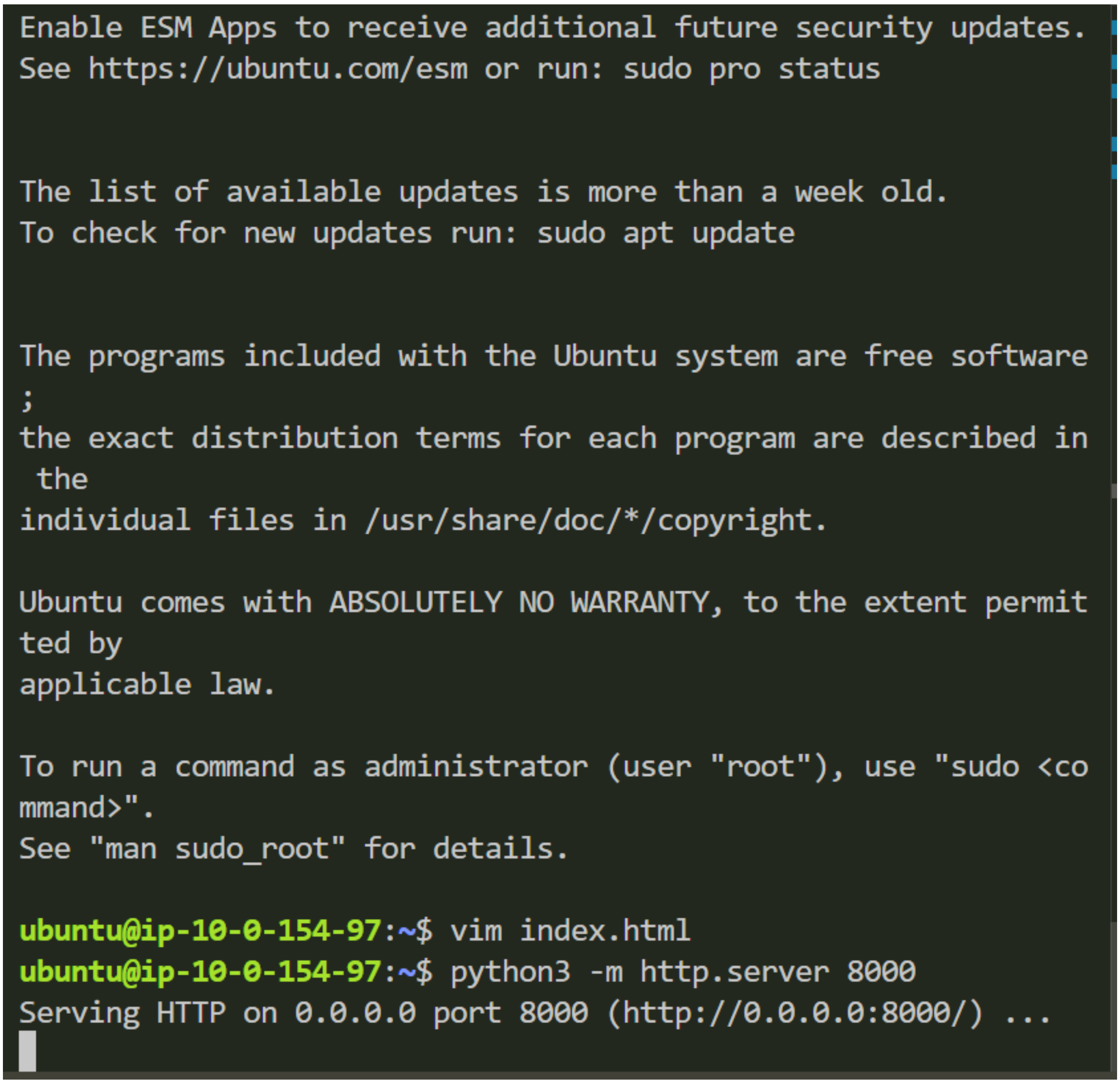
8. Install a Simple Python Web Application

On the **private EC2 instance**, create an `index.html` file:

```
<html>
<head><title>My first AWS Private to demonstrate app in private subnet</title></head>
<body>
  <h1>My first project using bastion-host</h1>
</body>
</html>
```

Start a **Python HTTP Server**:

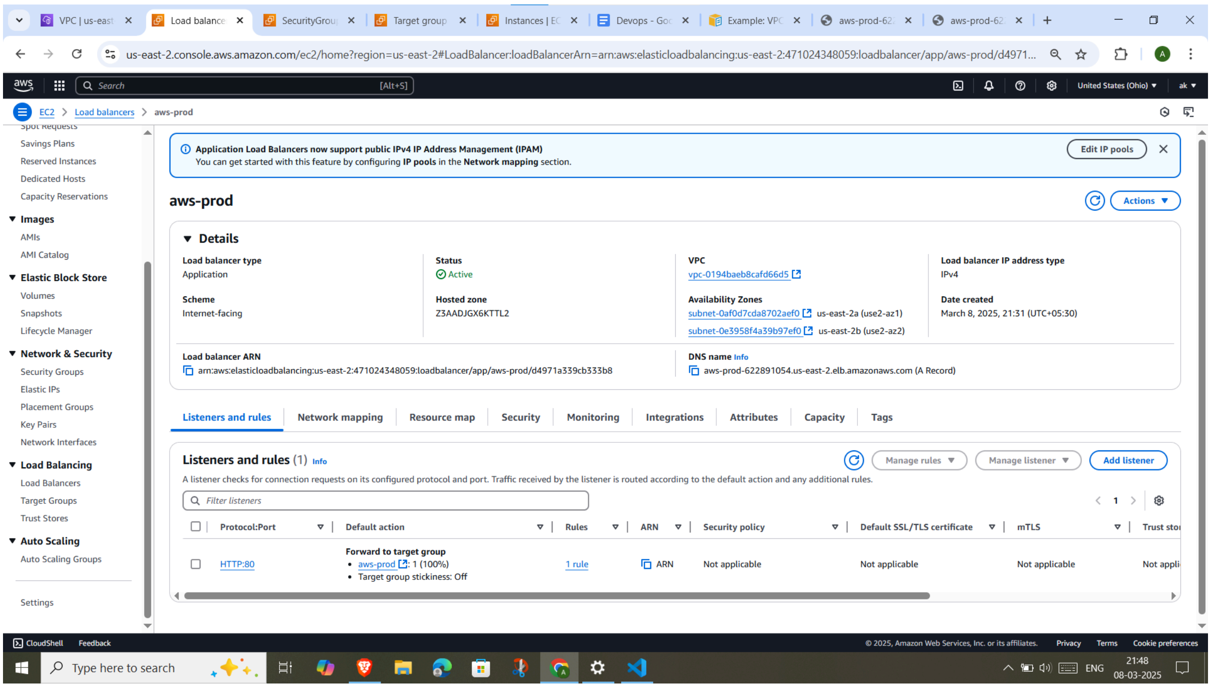
```
python3 -m http.server 8000
```



```
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software
;
the exact distribution terms for each program are described in
 the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permit
ted by
applicable law.

To run a command as administrator (user "root"), use "sudo <co
mmand>".
See "man sudo_root" for details.

ubuntu@ip-10-0-154-97:~$ vim index.html
ubuntu@ip-10-0-154-97:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

9. Create an Application Load Balancer (ALB)

1. Go to **EC2 > Load Balancers**.
2. Choose **Application Load Balancer**.
3. Set it as **Internet-facing**.
4. Assign it to the **Public Subnet**.
5. Attach **Load Balancer Security Group**.
6. Configure a **Target Group** to include **private EC2 instances**.
7. Configure **Health Checks** on **Port 80**.



Application Load Balancers now support public IPv4 IP Address Management (IPAM)
You can get started with this feature by configuring **IP pools** in the **Network mapping** section.

aws-prod

Details

Load balancer type
Application

Status
Active

VPC
vpc-0194baeb8cafd66d5

Load balancer IP address type
IPv4

Scheme
Internet-facing

Hosted zone
Z3AADJGX6KTTL2

Availability Zones
subnet-0af0d7cda8702aef0 us-east-2a (use2-az1)
subnet-0e3958f4a39b97ef0 us-east-2b (use2-az2)

Date created
March 8, 2025, 21:31 (UTC+05:30)

Load balancer ARN
arn:aws:elasticloadbalancing:us-east-2:471024348059:loadbalancer/app/aws-prod/d4971a339cb333b8

DNS name Info
aws-prod-622891054.us-east-2.elb.amazonaws.com (A Record)

Listeners and rules | Network mapping | Resource map | Security | Monitoring | Integrations | Attributes | Capacity | Tags

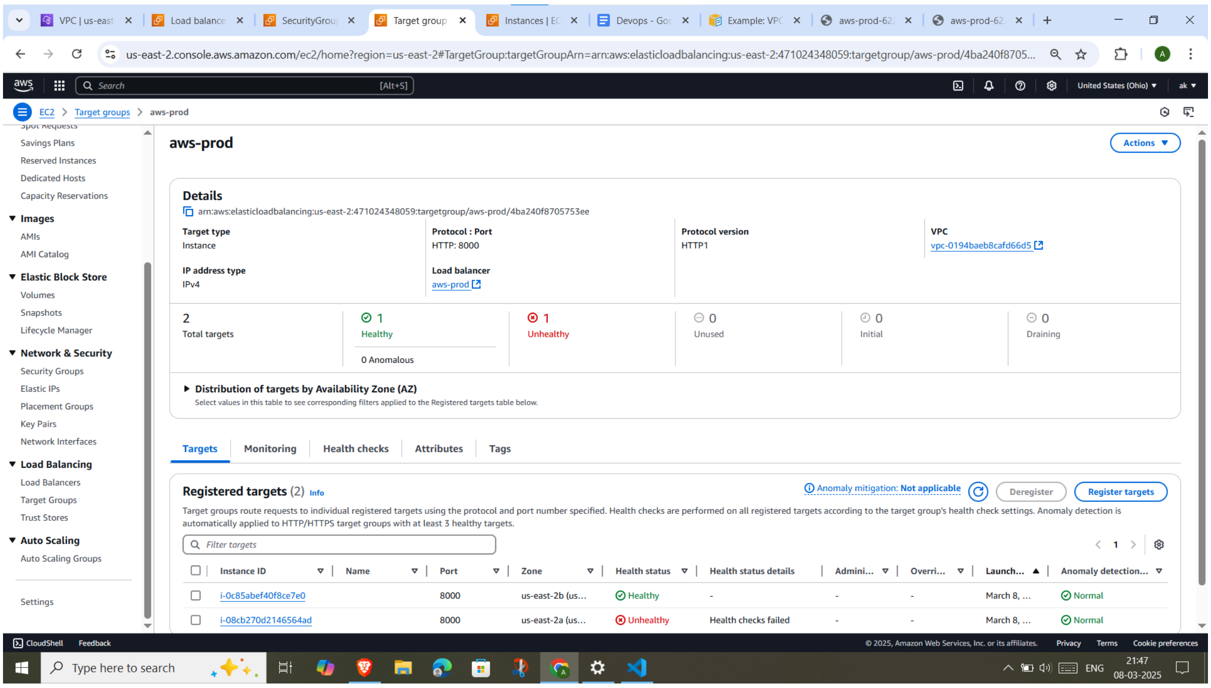
Listeners and rules (1) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

	Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust stor
□	HTTP:80	**Forward to target group** • aws-prod: 1 (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable	Not appli

10. Configure Auto Scaling Group (ASG)

1. Go to **EC2 > Auto Scaling Groups**.
2. Create a **Launch Template** with private instance settings.
3. Set **desired, minimum, and maximum** instances.
4. Attach the **Target Group** from Load Balancer.



aws-prod

Details

arn:aws:elasticloadbalancing:us-east-2:471024348059:targetgroup/aws-prod/4ba240f8705753ee

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 8000	HTTP1	vpc-0194baeb8cafd66d5
IP address type	**Load balancer**		
IPv4	aws-prod		

2	1	1	0	0	0
Total targets	Healthy 0 Anomalous	Unhealthy	Unused	Initial	Draining

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Registered targets (2) Info

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Instance ID	Name	Port	Zone	Health status	Health status details	Admini...	Overri...	Launch...	Anomaly detection...
i-0c85abef40f8ce7e0		8000	us-east-2b (us...	Healthy	-	-	-	March 8, ...	Normal
i-08cb270d2146564ad		8000	us-east-2a (us...	Unhealthy	Health checks failed	-	-	March 8, ...	Normal

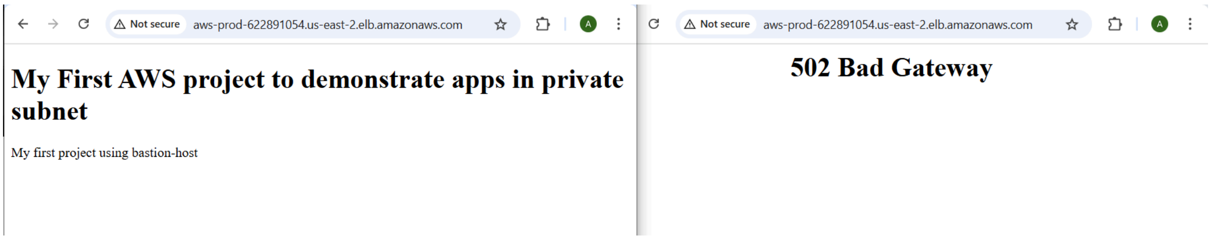
11. Test Load Balancer & Auto Scaling

1. **Get Load Balancer DNS Name** from EC2 Console.
2. Access the application in a browser:

http://<load-balancer-dns>

Verify Load Balancer

- **One instance responds** with `index.html` output.
- **Other shows "Bad Gateway"** (because it's missing `index.html`).
- **Target Group** shows **one healthy, one unhealthy** instance.


Not secure
aws-prod-622891054.us-east-2.elb.amazonaws.com
Not secure
aws-prod-622891054.us-east-2.elb.amazonaws.com

My First AWS project to demonstrate apps in private subnet

My first project using bastion-host

502 Bad Gateway

Key Takeaways

✅ **Bastion Host** allows secure SSH access to private instances.
✅ **Load Balancer** distributes traffic and enhances availability.
✅ **Auto Scaling** ensures high availability and scalability.
✅ **Security Groups** control access effectively.
✅ **NAT Gateway** provides secure outbound connectivity for private instances.