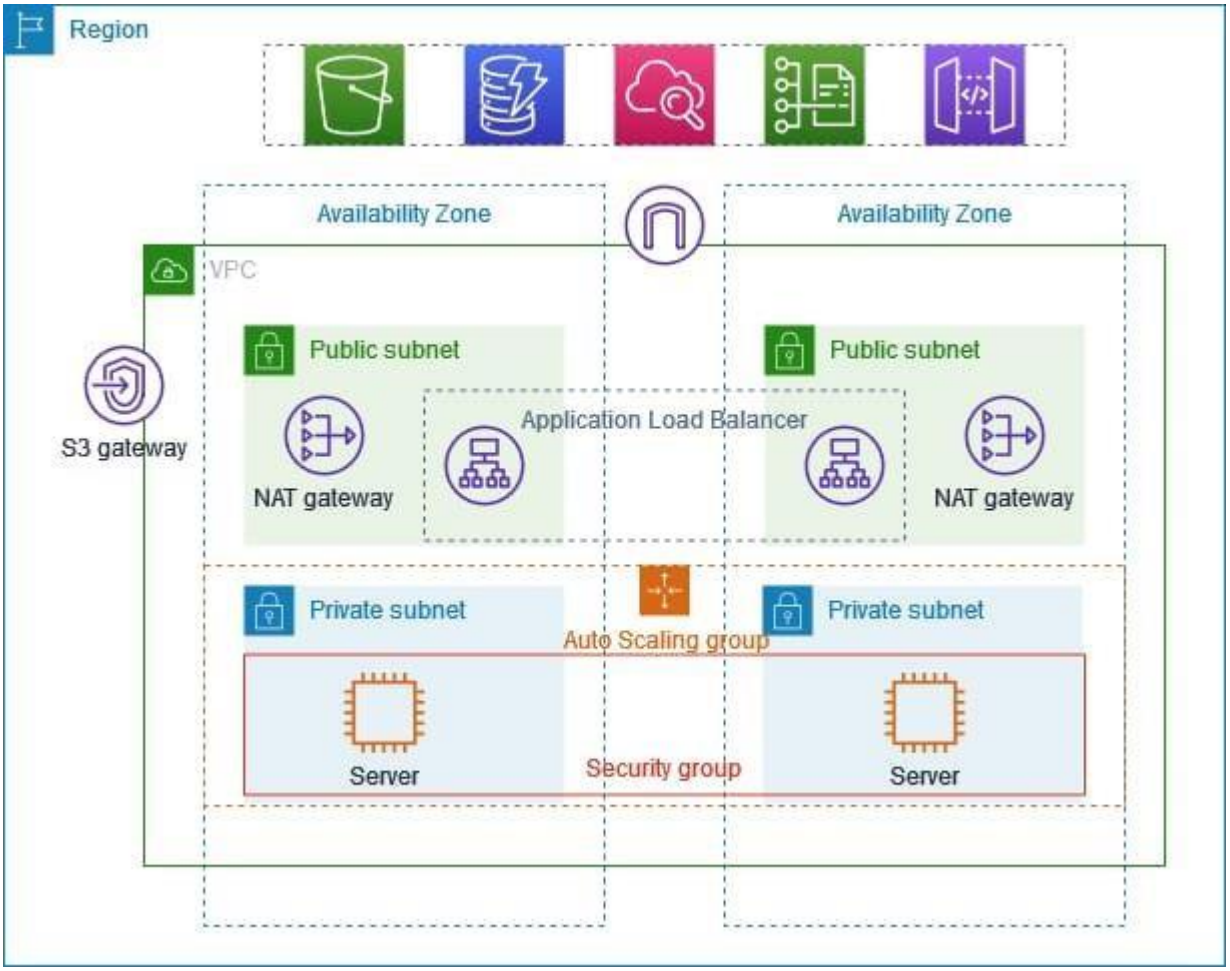
AWS VPC Project: Public-Private Subnet with Auto Scaling, Load Balancer, and Bastion Host



Overview

In this project, we set up a Virtual Private Cloud (VPC) with public and private subnets across two Availability Zones (AZs). The infrastructure includes an Auto Scaling group, an Application Load Balancer, and NAT Gateways to manage server traffic, enhance security, and ensure high availability in a production environment.

Architecture Components

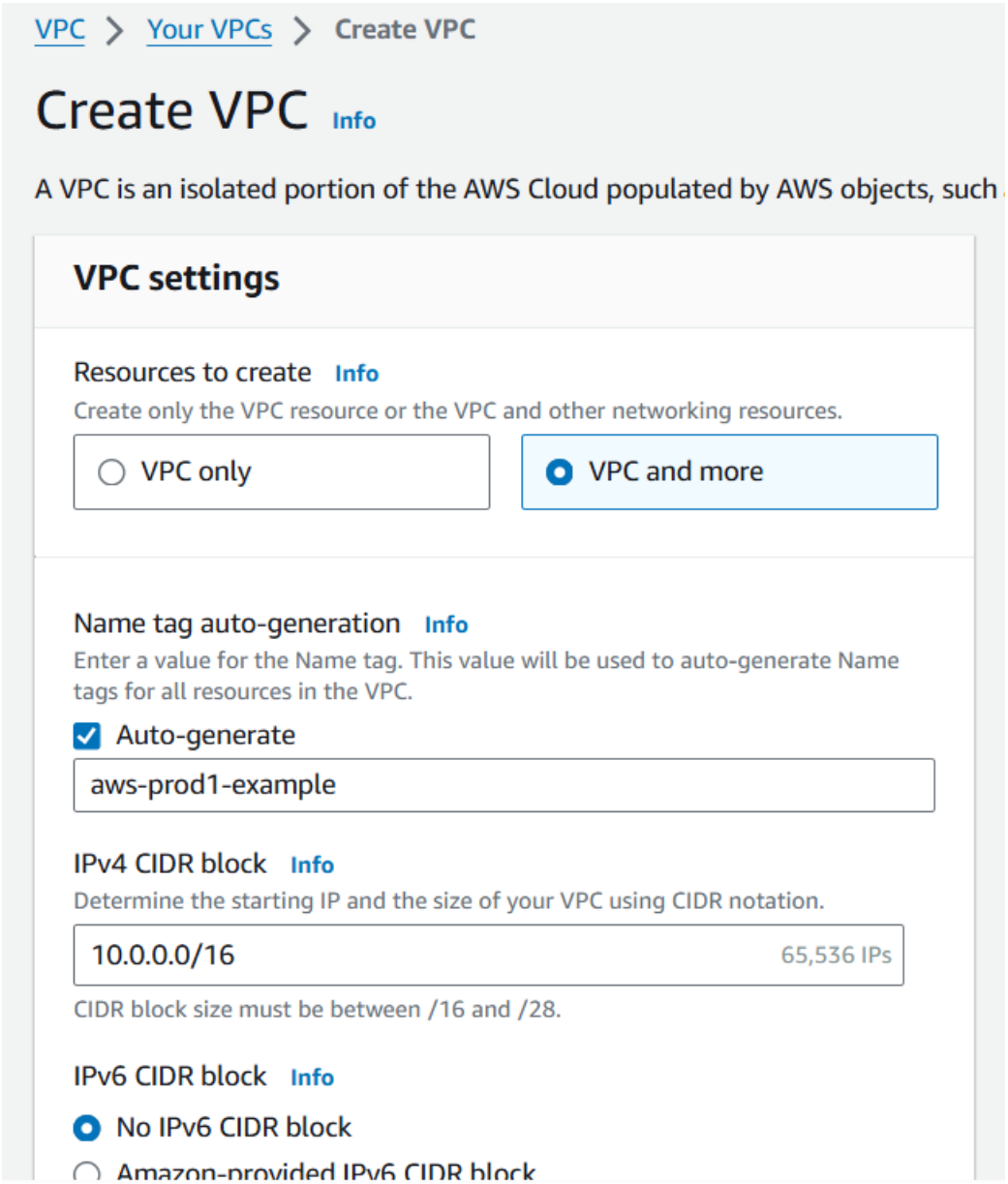
- **VPC with Public and Private Subnets:** The VPC spans two AZs, each containing both public and private subnets.
- **NAT Gateway:** Deployed in each AZ to allow instances in private subnets to access the internet securely.
- **Application Load Balancer (ALB):** Distributes incoming traffic across the instances in the private subnets.
- **Auto Scaling Group (ASG):** Automatically adjusts the number of instances in the private subnets based on demand.
- **Bastion Host:** Used to securely connect to instances in the private subnets.

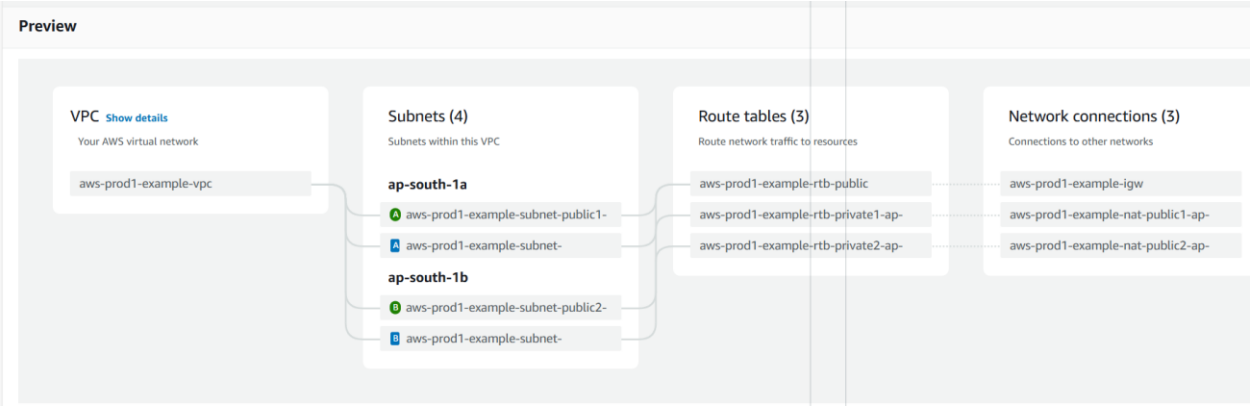
Step-by-Step Implementation

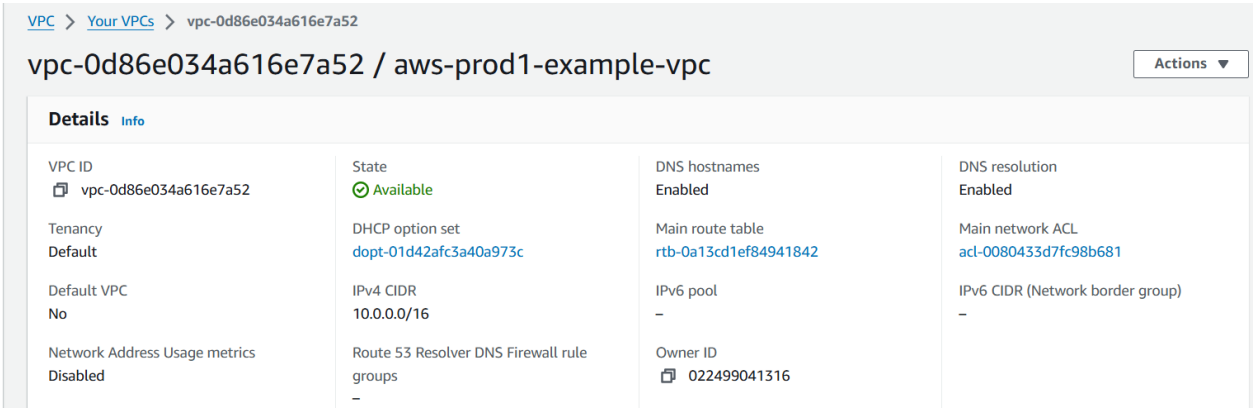
1. Create the VPC and Subnets

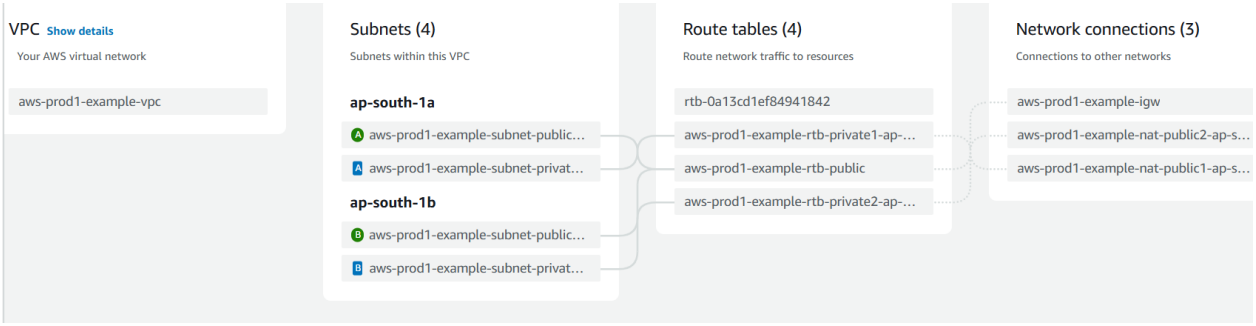
- **Go to the VPC dashboard** in AWS.

- **Create a VPC** with a CIDR block suitable for your environment (e.g., 10.0.0.0/16).
- **Create Public and Private Subnets** in two Availability Zones (e.g., 10.0.1.0/24, 10.0.2.0/24 for public, 10.0.3.0/24, 10.0.4.0/24 for private).









2. Configure Route Tables

- Attach an **Internet Gateway (IGW)** to the VPC.
- Create a **Route Table** for the public subnets and associate it with the subnets. Add a route to the IGW (0.0.0.0/0 -> IGW).
- For the private subnets, associate them with the default route table that has no direct internet access.

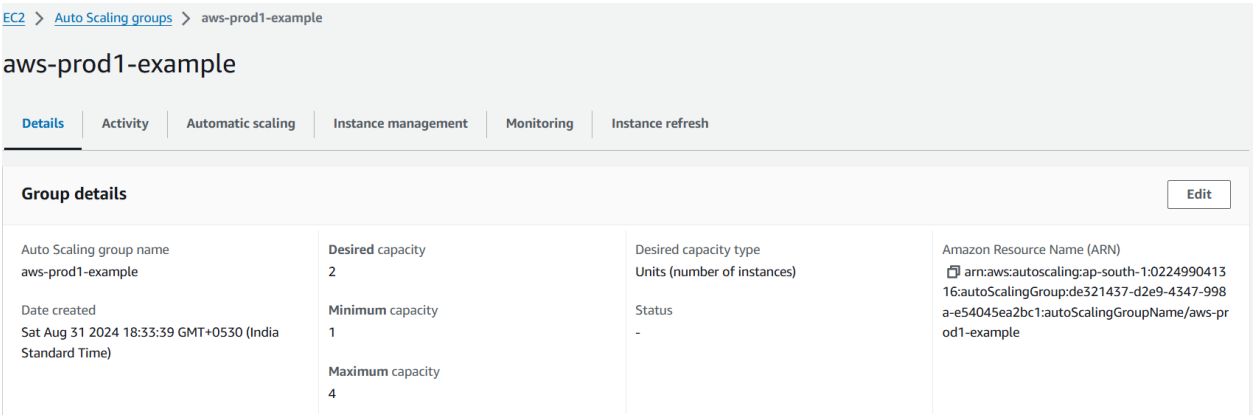
3. Create NAT Gateways

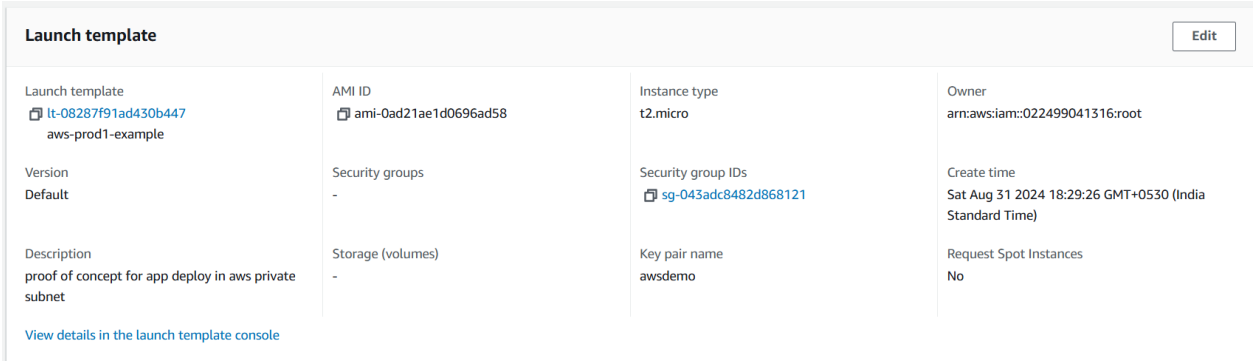
- **Go to the NAT Gateways** section in the VPC dashboard.
- Create a **NAT Gateway** in each public subnet, and associate each with an Elastic IP.
- Update the route table for the private subnets to direct internet-bound traffic to the NAT Gateway.

4. Launch Instances in Private Subnets via Auto Scaling

- **Go to EC2 -> Auto Scaling Groups.**
- **Create a Launch Template** with the following settings:
    - Select a recent AMI (Amazon Machine Image).
    - Configure the instance type (e.g., t2.micro).
    - **Create a new security group** with the following inbound rules:
        - SSH (Port 22) - Anywhere
        - Custom TCP (Port 8000) - Anywhere (for the Python app)
- **Create the Auto Scaling Group (ASG):**
    - Select the launch template created.
    - Choose the VPC and select the private subnets.
    - Set the desired capacity to 2, minimum to 1, and maximum to 4.

- Two instances will be launched without public IPs.



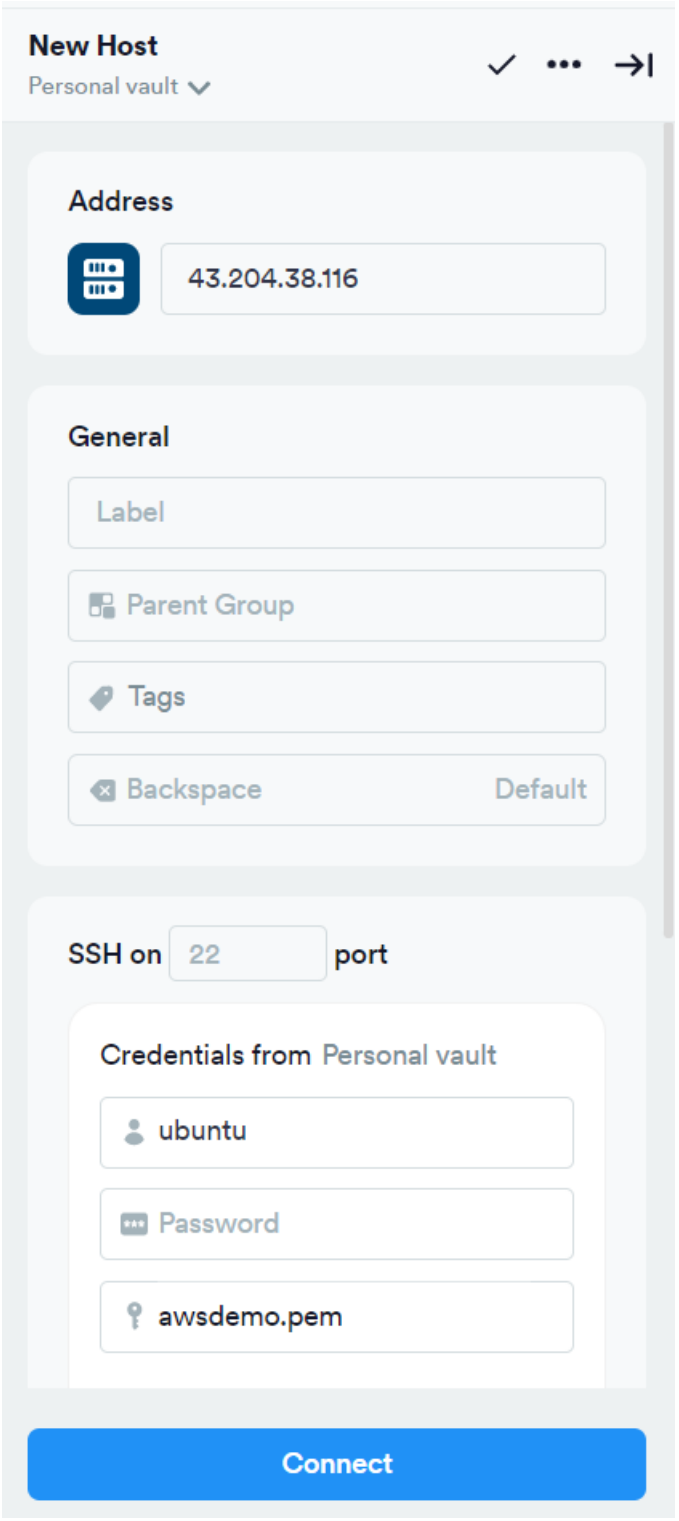


5. Set Up a Bastion Host

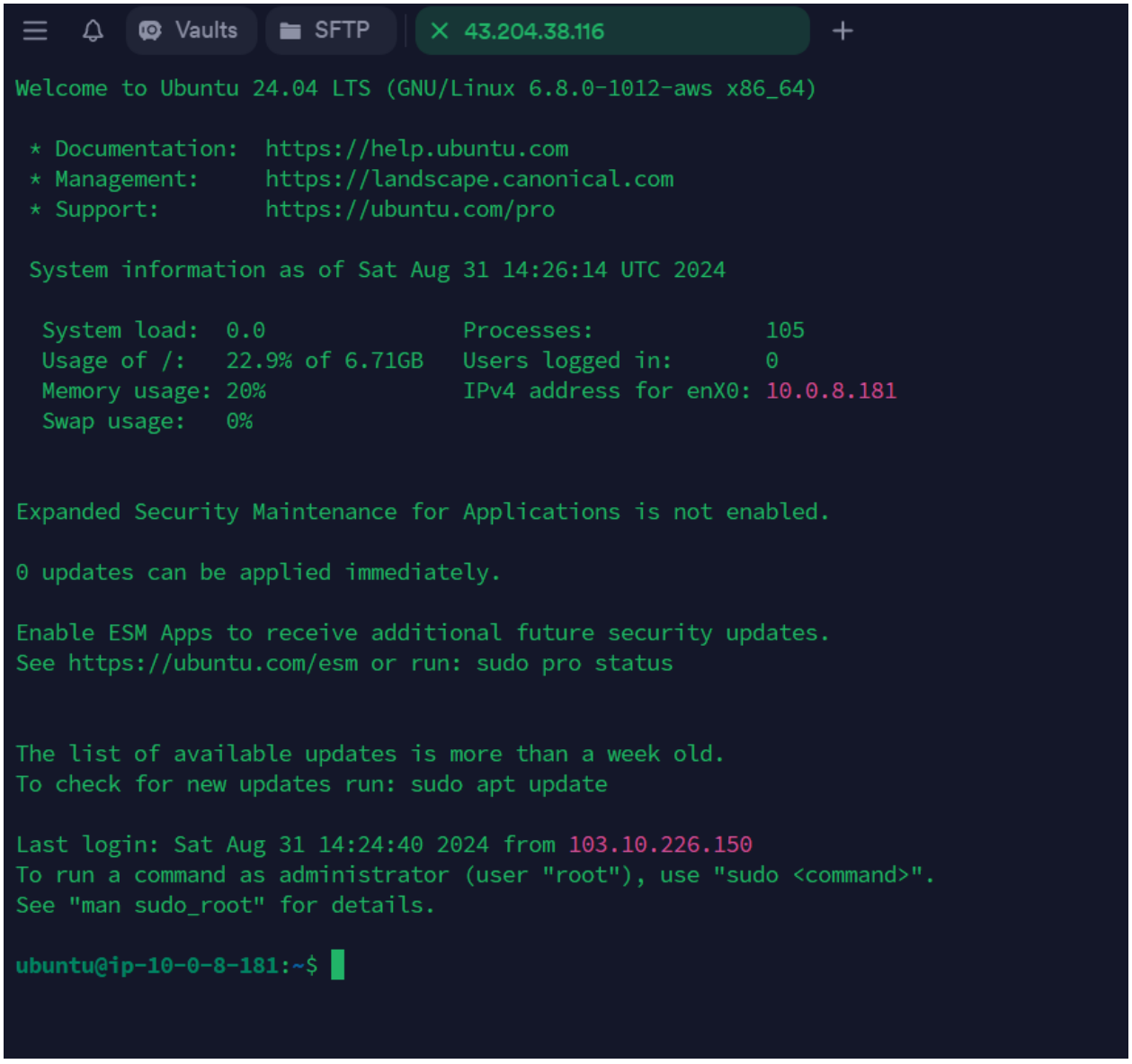
- **Launch a new EC2 instance** in the public subnet (AZ 1a).
- Enable the public IP and select the VPC created.
- Use the default AMI or your preferred Linux distribution.
- Once launched, **copy the key pair** from your local machine to the Bastion Host

Terminus is a popular terminal emulator and SSH client that allows users to securely connect to remote servers from their local machine. It provides a graphical interface for users to access command-line environments on remote servers over a network using SSH (Secure Shell) protocol.

- **Install Terminus**: Download and install Terminus from its official website.
- **Open Terminus**: Launch the Terminus application.
- **Create a New SSH Connection**: Click "New SSH Connection" and enter the host, port, username, and authentication details like key.
- **Connect to the Server**: Click "Connect" and, if prompted, accept the server's SSH



key fingerprint.


Vaults
SFTP
43.204.38.116
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Sat Aug 31 14:26:14 UTC 2024
System load: 0.0
Usage of /: 22.9% of 6.71GB
Memory usage: 20%
Swap usage: 0%
Processes: 105
Users logged in: 0
IPv4 address for enX0: 10.0.8.181
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.
To check for new updates run: sudo apt update
Last login: Sat Aug 31 14:24:40 2024 from 103.10.226.150
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-10-0-8-181:~$

```
ubuntu@ip-10-0-8-181:~$ pwd
/home/ubuntu
ubuntu@ip-10-0-8-181:~$ ls
awsdemo.pem
ubuntu@ip-10-0-8-181:~$ sudo cp awsdemo.pem/home/ubuntu/.ssh
cp: missing destination file operand after 'awsdemo.pem/home/ubuntu/.ssh'
Try 'cp --help' for more information.
ubuntu@ip-10-0-8-181:~$ chmod 400 awsdemo.pem
ubuntu@ip-10-0-8-181:~$ ssh -i awsdemo.pem ubuntu@43.204.38.116
The authenticity of host '43.204.38.116 (43.204.38.116)' can't be established.
ED25519 key fingerprint is SHA256:TpEd1fA8Q6ffEr+LKBRVlFXiQZ2MRytBLJPi+hfb7Ss.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '43.204.38.116' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Sat Aug 31 14:37:01 UTC 2024

  System load:  0.0               Processes:             110
  Usage of /:   22.9% of 6.71GB   Users logged in:       1
  Memory usage: 21%               IPv4 address for enX0: 10.0.8.181
  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.
To check for new updates run: sudo apt update

Last login: Sat Aug 31 14:26:15 2024 from 103.10.226.150
ubuntu@ip-10-0-8-181:~$
```

Now we have to login in one of the instance to install python application using

command

ssh -i keyname.pem ubuntu@private _ip_address_of_one_instance

inthis case

we used

ssh -I awsdemo.pem ubuntu@10.0.139.23

```
ubuntu@ip-10-0-8-181:~$ ls
awsdemo.pem
ubuntu@ip-10-0-8-181:~$ ssh -i awsdemo.pem ubuntu@10.0
.139.23
The authenticity of host '10.0.139.23 (10.0.139.23)' c
an't be established.
ED25519 key fingerprint is SHA256:A0nAZxA4frhPJY0SflUA
7MYC5tZXFtgdv3K5mY7g6Bg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[
fingerprint])? yes
Warning: Permanently added '10.0.139.23' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

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

 System information as of Sat Aug 31 14:43:20 UTC 2024

  System load:  0.0               Processes:             103
  Usage of /:   22.7% of 6.71GB   Users logged in:       0
  Memory usage: 19%               IPv4 address for enX0: 10.0.139.23
  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.
To check for new updates run: sudo apt update
```

Now we successfully login in one instance in private subnet

6.Deploy a Simple Web Application

- **Create a basic HTML file** on one of the private instances:

vim index.html

```
ubuntu@ip-10-0-139-23:~$ vim index.html
ubuntu@ip-10-0-139-23:~$ cat index.html
<!DOCTYPE html>
<html>
<body>
<h1>My First AWS PROJECT to demonstrate apps in privat
e subnet</h1>

</body>
</html>


ubuntu@ip-10-0-139-23:~$
```

- **Run the application** using Python's built-in HTTP server:

python3 -m http.server 8000

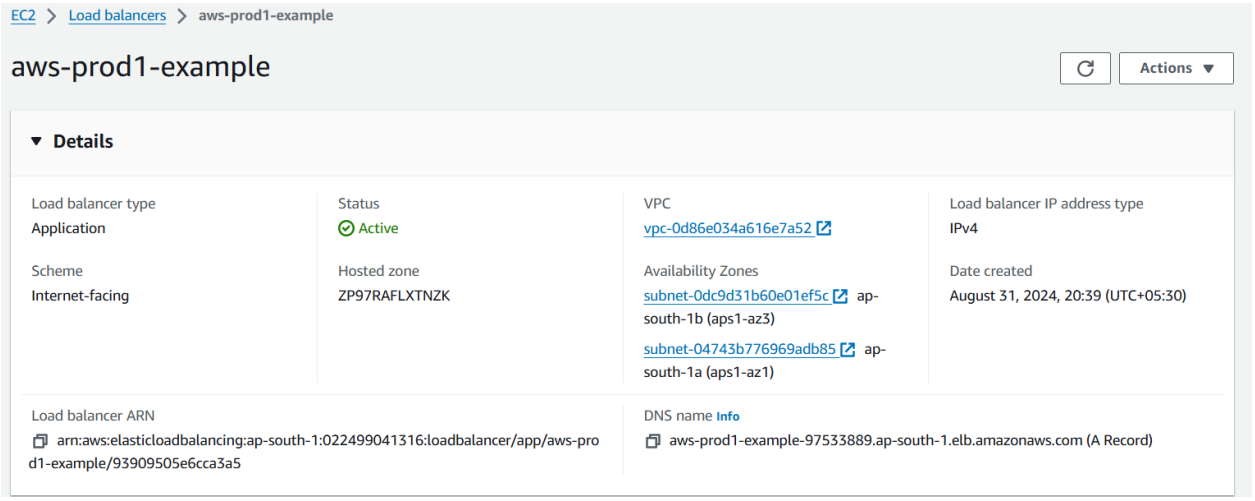
```
ubuntu@ip-10-0-139-23:~$ vim index.html
ubuntu@ip-10-0-139-23:~$ cat index.html
<!DOCTYPE html>
<html>
<body>
<h1>My First AWS PROJECT to demonstrate apps in privat
e subnet</h1>

</body>
</html>


ubuntu@ip-10-0-139-23:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000
/) ...
```

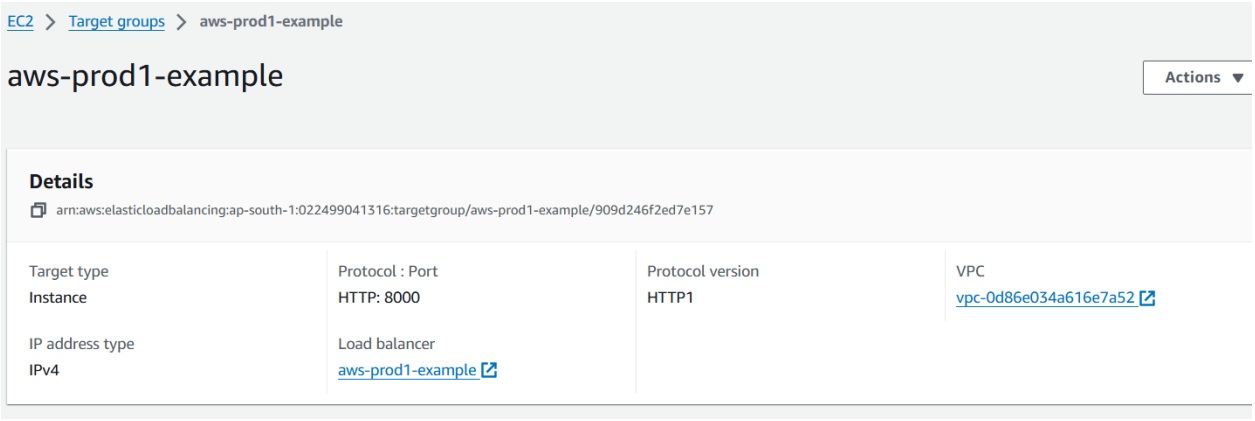
6. Create an Application Load Balancer

- **Go to the EC2 -> Load Balancers** section and create an Application Load Balancer.
- Select the VPC and choose the public subnets.
- Assign the security group created earlier to the Load Balancer.
- **Create a Target Group**:
    - Choose HTTP as the protocol and set the port to 8000.

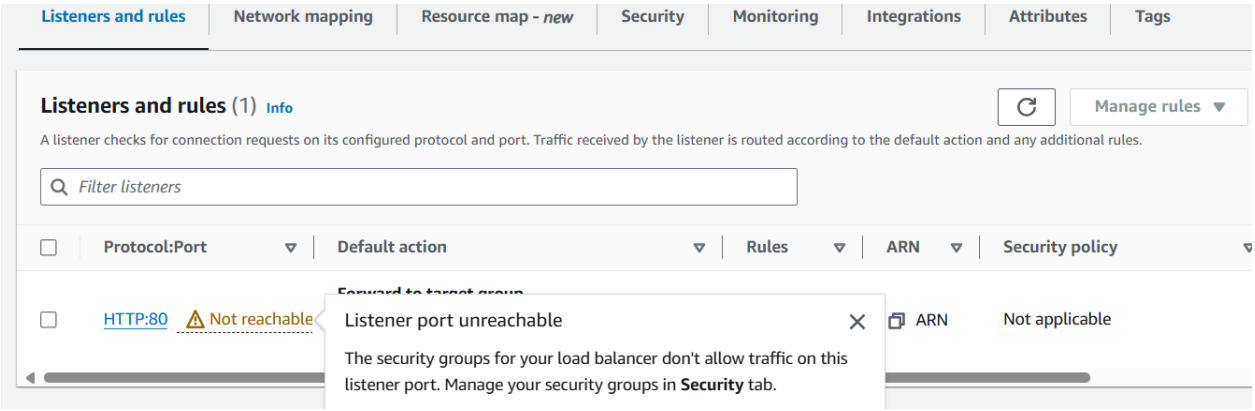
- Register the private instances in the target group.

- **Associate the Target Group** with the Load Balancer and finish the setup.

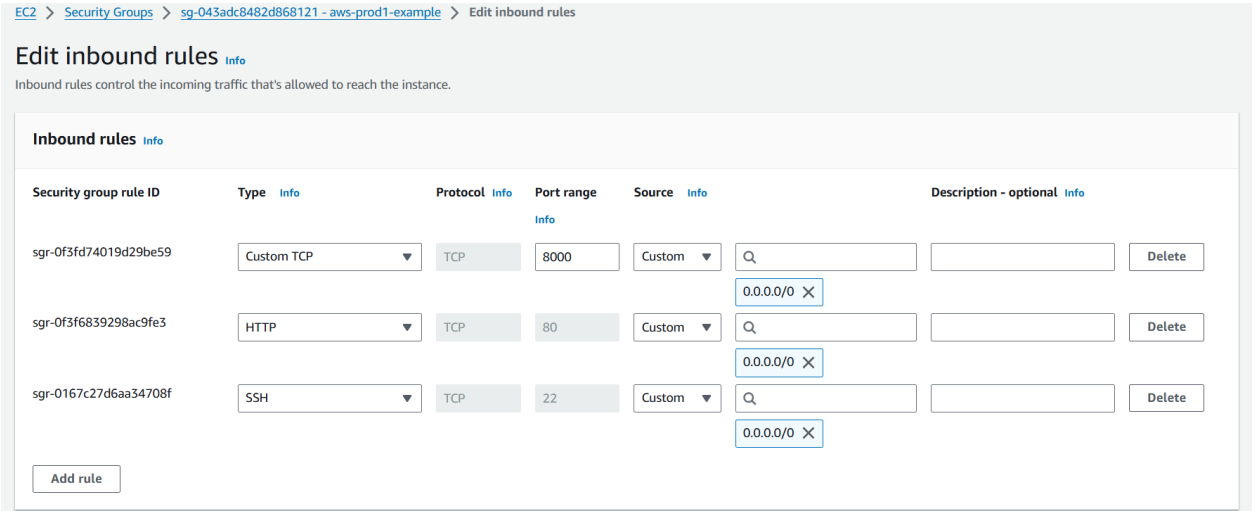




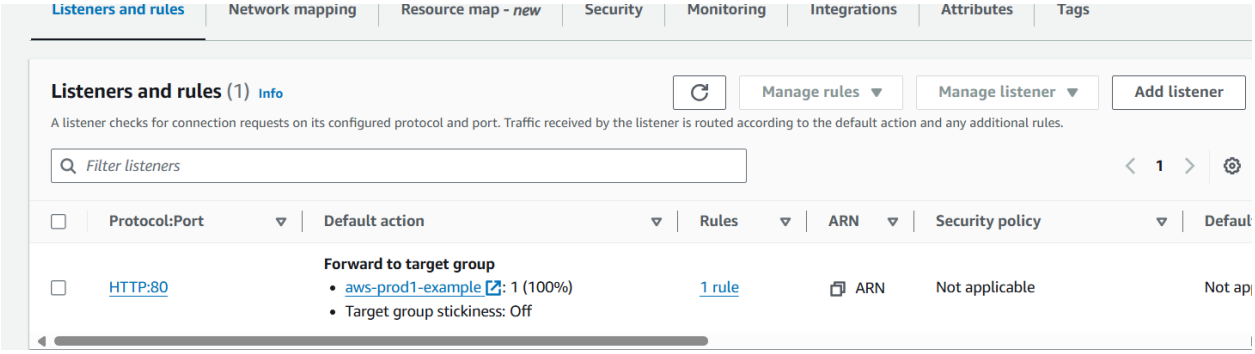
But in load balancer there is error



- **Update the Load Balancer's Security Group** to allow inbound traffic on HTTP (Port 80).



Now error must be resolve.



7. Test the Setup

- Obtain the **DNS name** of the Load Balancer from the AWS console.
- **Open the DNS name in a web browser.**
- The application should be accessible, and traffic will be directed to the healthiest instance in the private subnets.

Same process we have do for another instance in private subnet.