Placement Empowerment Program

Cloud Computing and DevOps Centre

Secure Access with a Bastion HostSet up a bastion host in a public subnet to securely access instances in a private subnet.

Name:Shalini D Department: IT

St. JOSEPH'S COLLEGE OF ENGINEERING | St. JOSEPH'S INSTITUTE OF TECHNOLOGY
AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction

A bastion host, deployed in a public subnet of your VPC, serves as a secure gateway to access instances in a private subnet. By launching an EC2 instance in the public subnet and assigning it a public IP or Elastic IP, you can create a security group to allow SSH access only from trusted IP addresses. Once set up, use the bastion host to connect securely to instances in the private subnet, ensuring controlled and protected access.

Overview

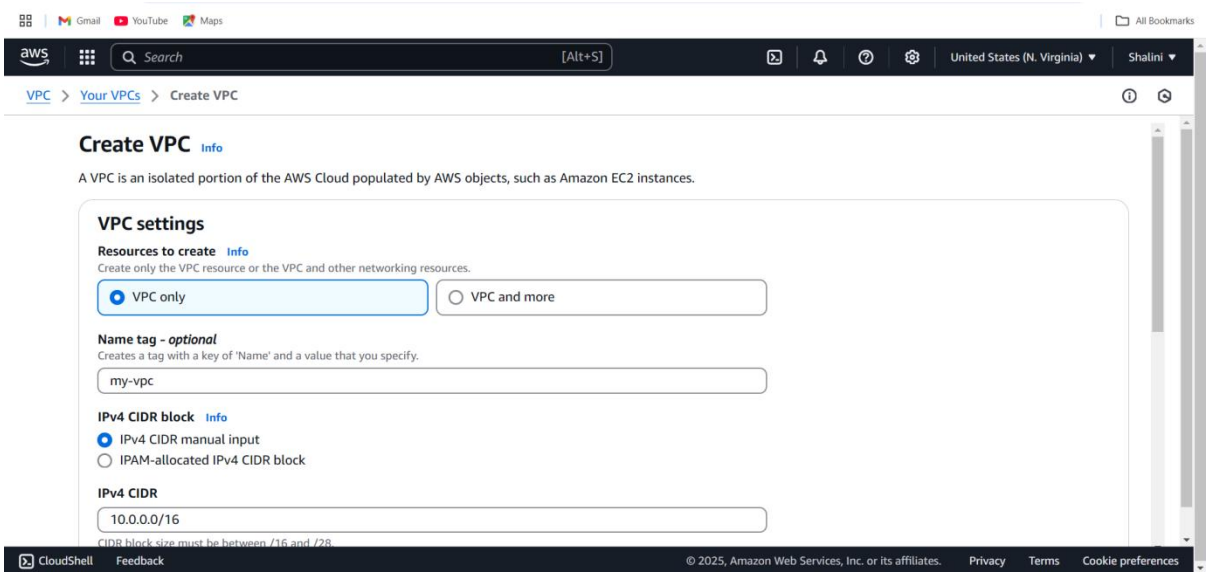
We will set up a **Bastion Host** in a **public subnet** that provides controlled SSH access to instances inside a **private subnet**.

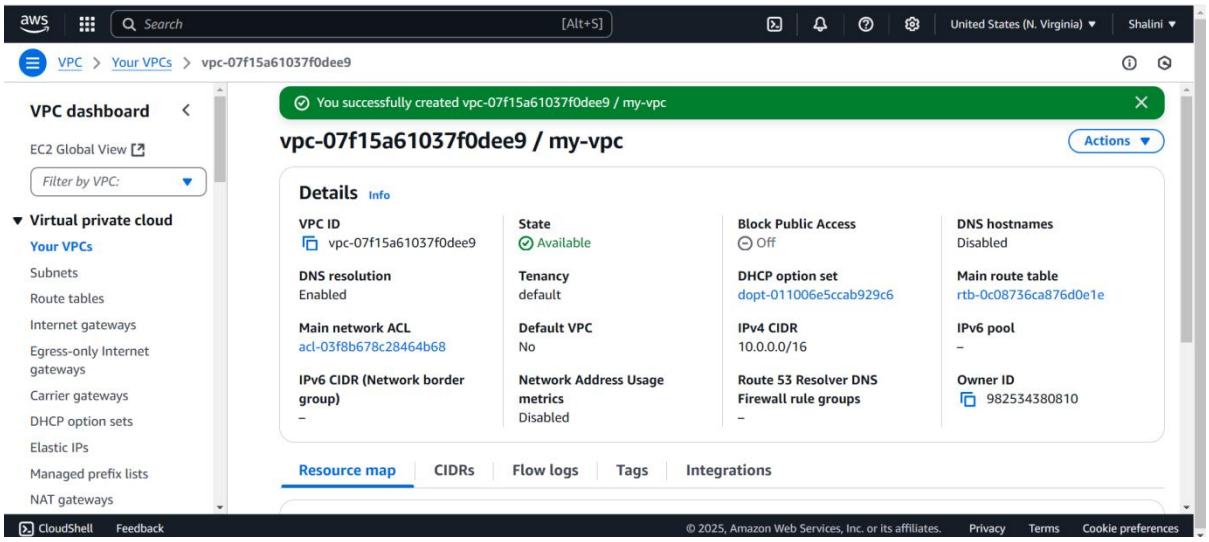
Step 1:

Create a VPC with Public and Private Subnets

1.1 Create a VPC

- Go to AWS Console → VPC Dashboard.
- Click Create VPC and name it MyVPC.
- Set IPv4 CIDR Block: 10.0.0.0/16.
- Click Create VPC.
-



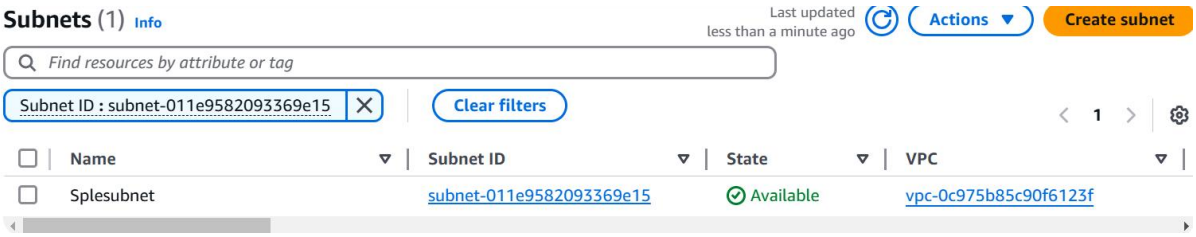


1.2 Create a Public Subnet

- Go to **Subnets → Create Subnet**.
- Select **MyVPC** and set CIDR block 10.0.1.0/24.
- Enable **Auto-Assign Public IP**.

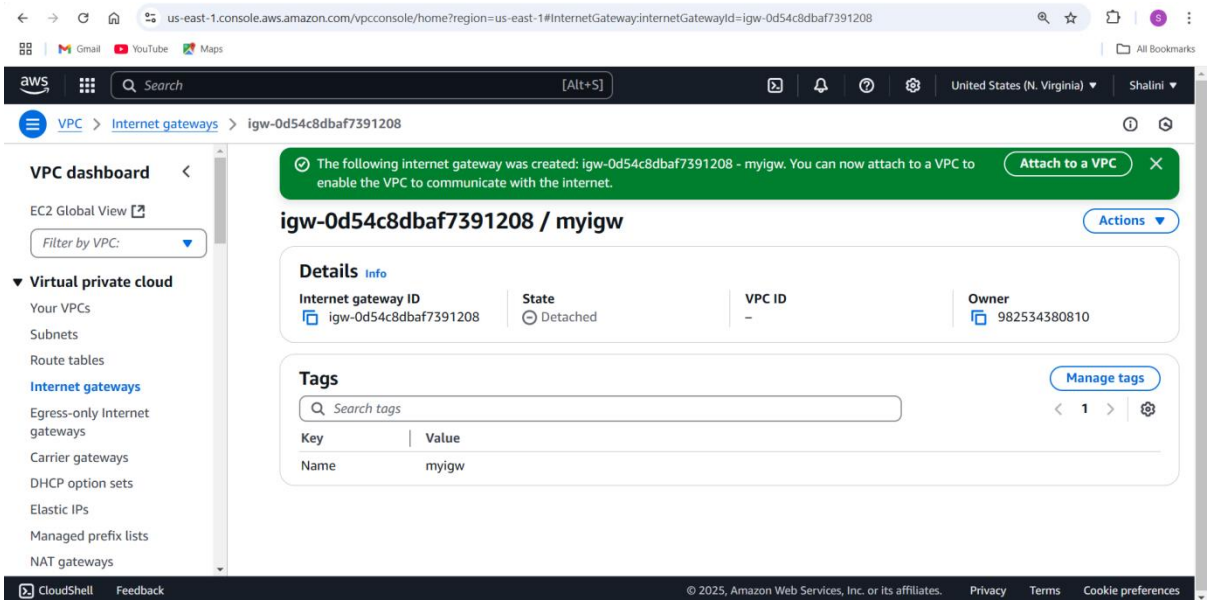
1.3 Create a Private Subnet

- Repeat the same process, but use CIDR block 10.0.2.0/24.
- **Do not enable** Auto-Assign Public IP.



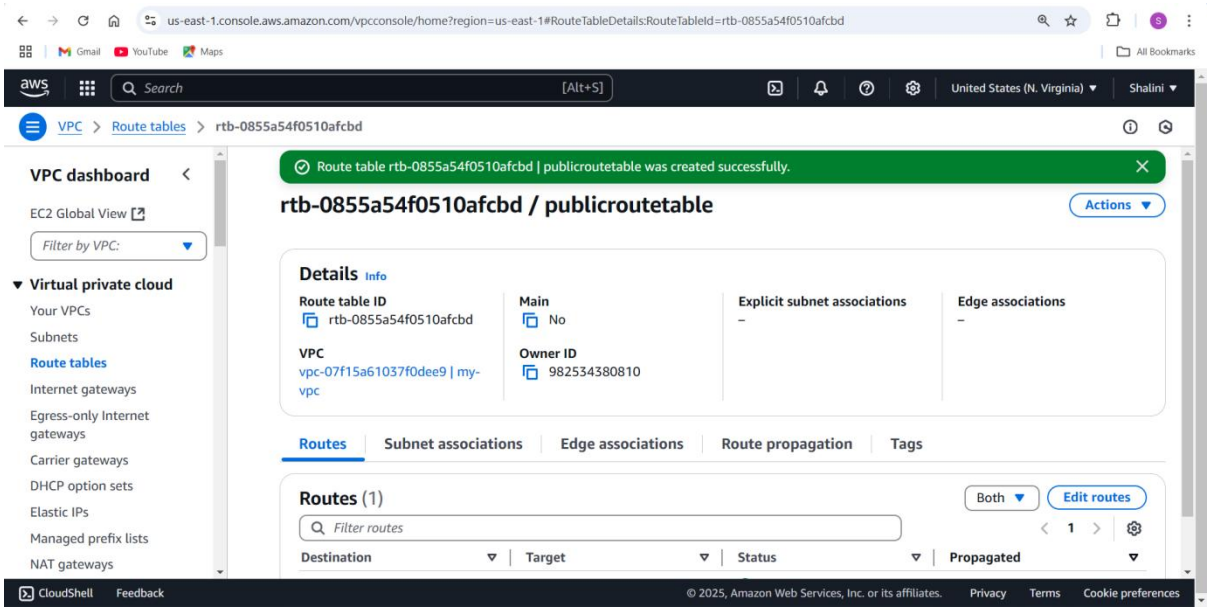
Step 2:
Configure Public Subnet for Internet Access
2.1 Create an Internet Gateway (IGW)

- Go to **Internet Gateways** → Click **Create Internet Gateway**.
- Name it **MyIGW**, attach it to **MyVPC**.



2.2 Update Public Route Table

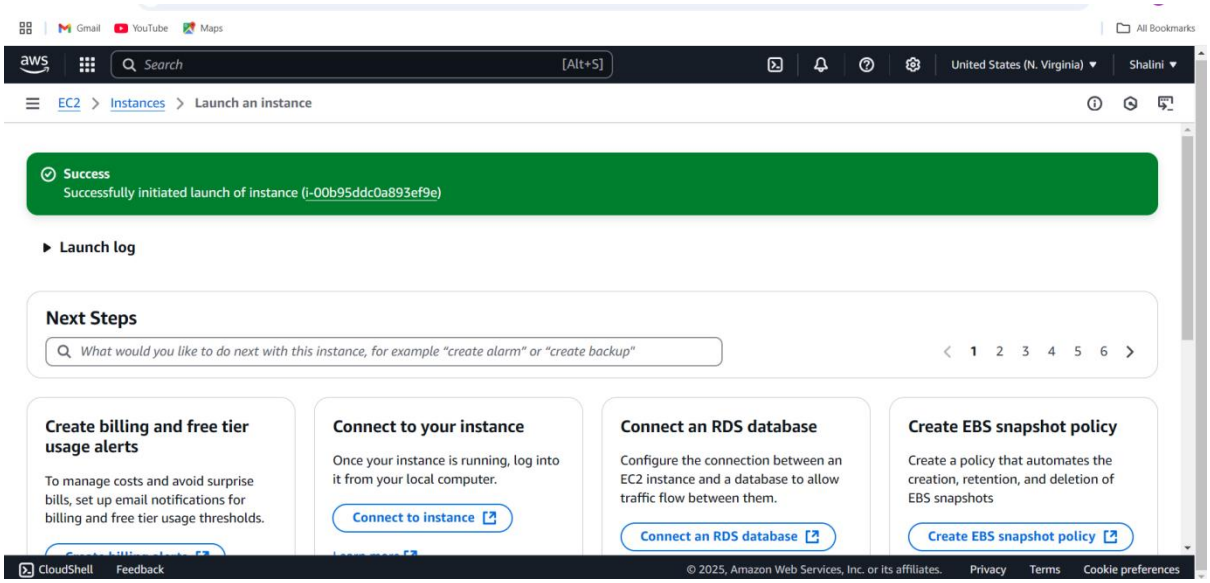
- Go to **Route Tables** → **Create Route Table** → Name it **PublicRouteTable**.
- Associate it with **PublicSubnet**.
- Add a route:
  - **Destination:** 0.0.0.0/0
  - **Target: Internet Gateway (MyIGW)**



Step 3:

Launch a Bastion Host (Public Subnet)

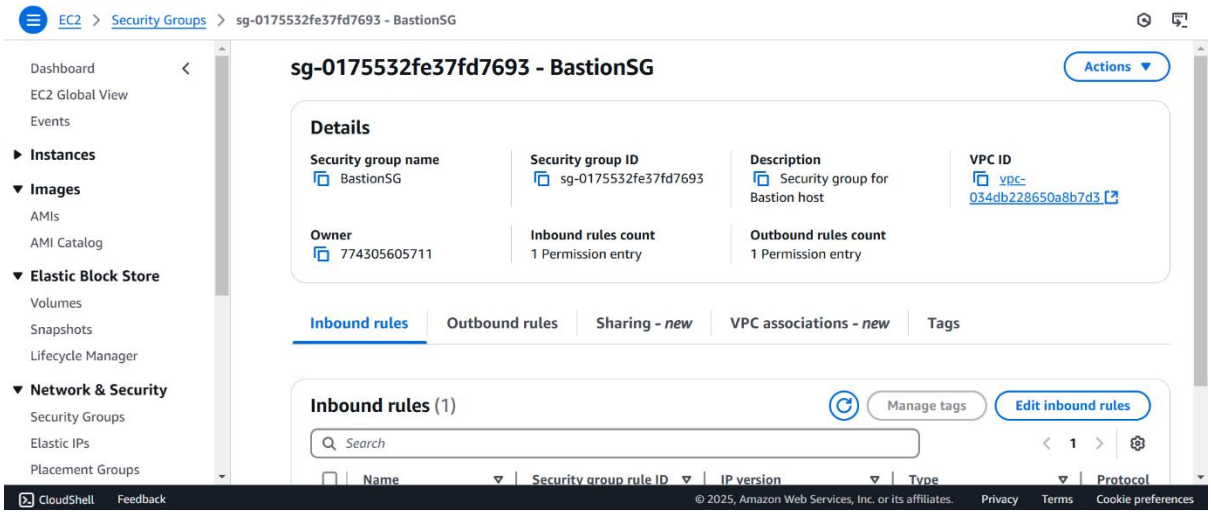
1. Go to **EC2 Dashboard → Launch Instance**.
2. Select **Amazon Linux 2** (or **Ubuntu**).
3. Choose **t2.micro (Free Tier Eligible)**.
4. Place it in **PublicSubnet** with **Auto-Assign Public IP enabled**.
5. Create a **Security Group (BastionSG)**:
    - Allow **SSH (Port 22) from Your IP** (xx.xx.xx.xx/32).
6. Create or use an **existing key pair** (e.g., bastion-key.pem).
7. Click **Launch**.



Step 4:

Launch a Private EC2 Instance

1. Go to **EC2 Dashboard → Launch Instance**.
2. Choose **Amazon Linux 2** (or **Ubuntu**).
3. Choose **t2.micro** and place it in **PrivateSubnet**.
4. **Disable Auto-Assign Public IP**.
5. Create a **Security Group (PrivateSG)**:
    - Allow **SSH (Port 22) only from Bastion Host's Security Group**.
6. Use the same **key pair** (bastion-key.pem).
7. Click **Launch**.



Step 5: Connect to the Private Instance Using the Bastion Host

5.1 Connect to the Bastion Host

ssh -i bastion-key.pem ec2-user@<bastion-public-ip>

(Replace <bastion-public-ip> with the actual Bastion Host public IP.)

```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\shalni> cd Downloads
PS C:\Users\shalni\Downloads> ssh -i key1.pem ec2-user@44.220.85.250
ssh: connect to host 44.220.85.250 port 22: Connection timed out
PS C:\Users\shalni\Downloads> ssh -i key1.pem ec2-user@10.0.2.62
ssh: connect to host 10.0.2.62 port 22: Connection timed out
PS C:\Users\shalni\Downloads>
```

5.2 SSH from Bastion to Private Instance

1. Copy the bastion-key.pem file to the Bastion Host:

scp -i bastion-key.pem bastion-key.pem ec2-user@<bastion-public-ip>:~/

2. Connect to the Bastion Host:

ssh -i bastion-key.pem ec2-user@<bastion-public-ip>

3. Change permissions for the key file:

chmod 400 bastion-key.pem

4. SSH into the Private Instance from the Bastion Host:

ssh -i bastion-key.pem ec2-user@<private-instance-ip>

(Replace <private-instance-ip> with the private IP of your instance.)

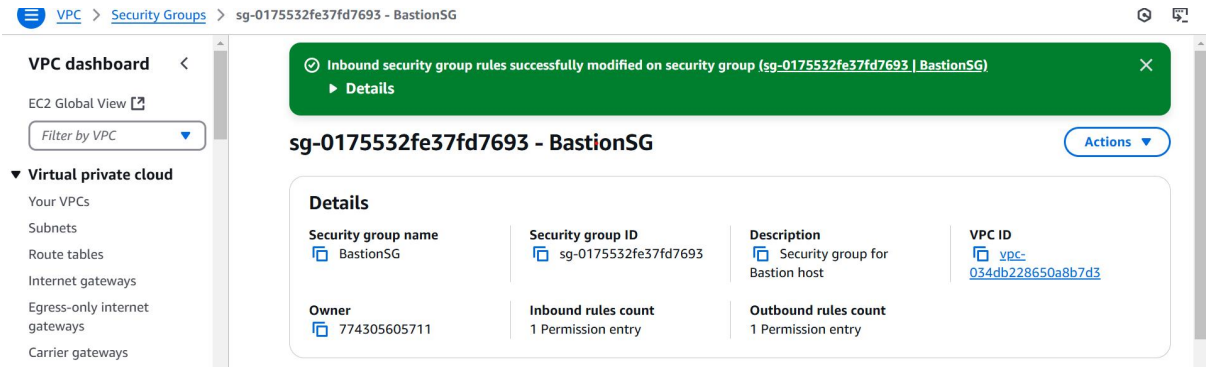
```
[ec2-user@ip-10-0-1-218 ~]$ chmod 400 sam.pem
[ec2-user@ip-10-0-1-218 ~]$ ssh -i sam.pem ec2-user@10.0.1.218
The authenticity of host '10.0.1.218 (10.0.1.218)' can't be established.
ECDSA key fingerprint is SHA256:Y6FPLIZ5IAKtMNwnbL3Yq1SXQPKRyey1HTZPbylOrLY.
ECDSA key fingerprint is MD5:d4:a6:0d:fa:99:92:df:21:ca:36:0f:39:5f:ed:ba:cd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.1.218' (ECDSA) to the list of known hosts.
Last login: Wed Feb  5 14:18:12 2025 from 223.178.84.112
       __|  __|_  )
       _|  (     /   Amazon Linux 2
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 1 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-1-218 ~]$
```

Step 6: Secure Your Bastion Host

6.1 Restrict SSH Access

- **Go to Security Group (BastionSG)** → Edit Inbound Rules.
- **Allow SSH only from your IP address (xx.xx.xx.xx/32)** instead of allowing all (0.0.0.0/0)



6.2 Disable Password Authentication

1. Edit SSH config:

sudo nano /etc/ssh/sshd_config

2. Find and update these lines:

PasswordAuthentication no
PermitRootLogin no

1. Restart SSH service:

sudo systemctl restart sshd

```
#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none


# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
```

Step 7:
Alternative - Use AWS Systems Manager (SSM) Instead of SSH

1. **Attach SSM Managed Policy to EC2 IAM Role** (AmazonSSMManagedInstanceCore).
2. **Enable SSM Agent** (Pre-installed on Amazon Linux & Ubuntu).
3. Use **AWS Systems Manager > Session Manager** to connect to instances without SSH.

Conclusion

This setup prevents direct internet exposure, enforces security group rules, and allows monitoring/logging of access.
For even better security, consider eliminating SSH and using AWS Systems Manager (SSM) Session Manager instead.