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

Introduction

Securing access to your cloud infrastructure is vital, and setting up a bastion host is a proven approach to achieve this. A bastion host acts

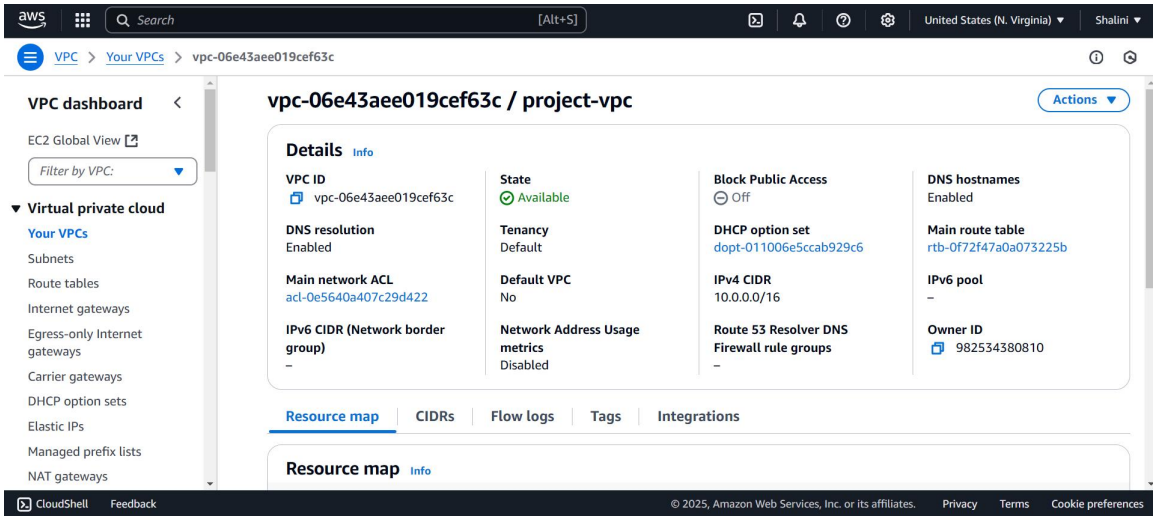
as a gateway that you can securely access instances in a private subnet without exposing them to the public internet.

Overview

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

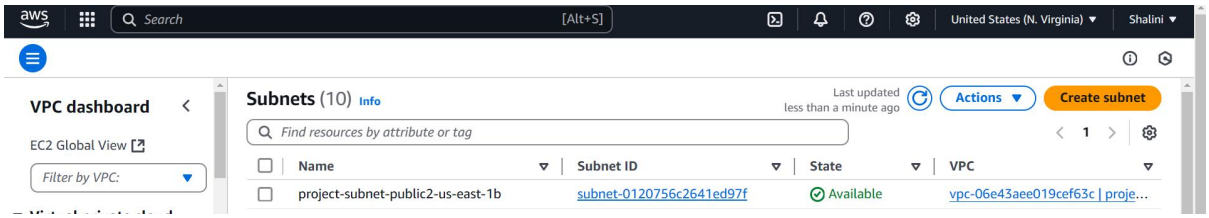
Step 1:

Create a VPC with Public and Private Subnets



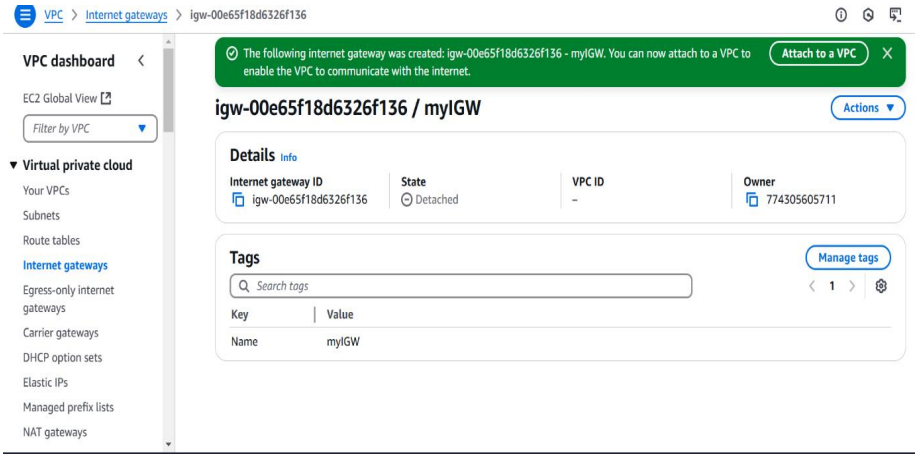
1.2 Create a Public Subnet

1.3 Create a Private Subnet

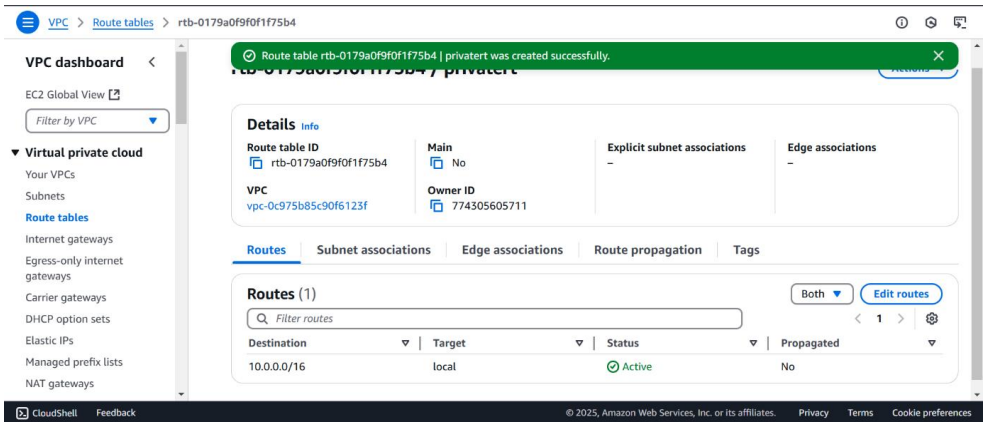


Step 2:

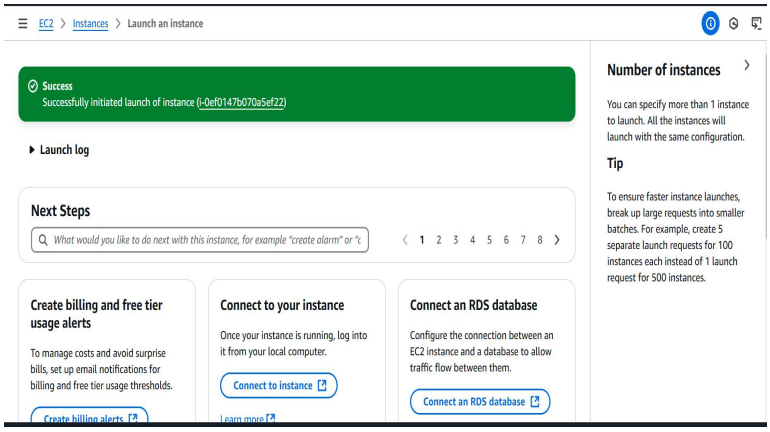
Configure Public Subnet for Internet Access



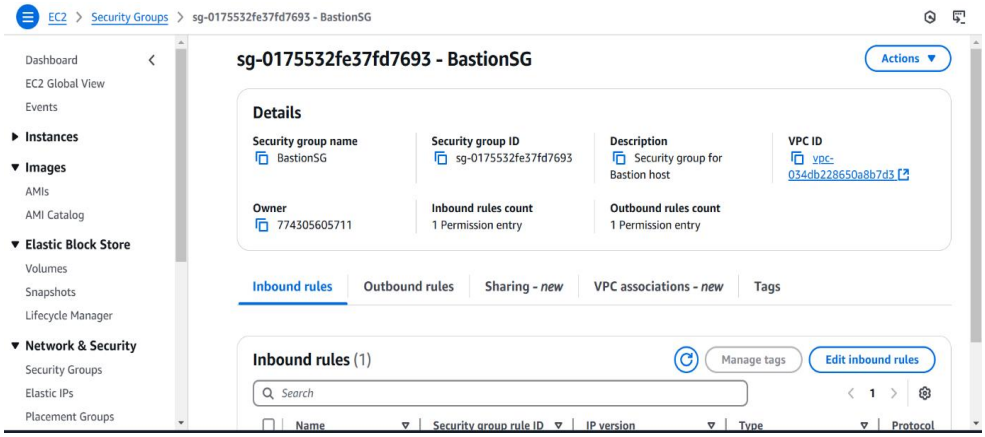
2. Update Public Route Table



Step 3:
Launch a Bastion Host (Public Subnet)



Step 4:
Launch a Private EC2 Instance



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

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.)

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>

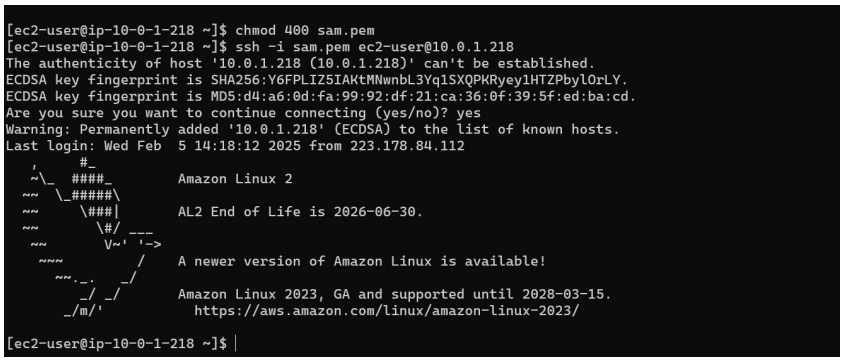
3. Change permissions for the key file:

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.)

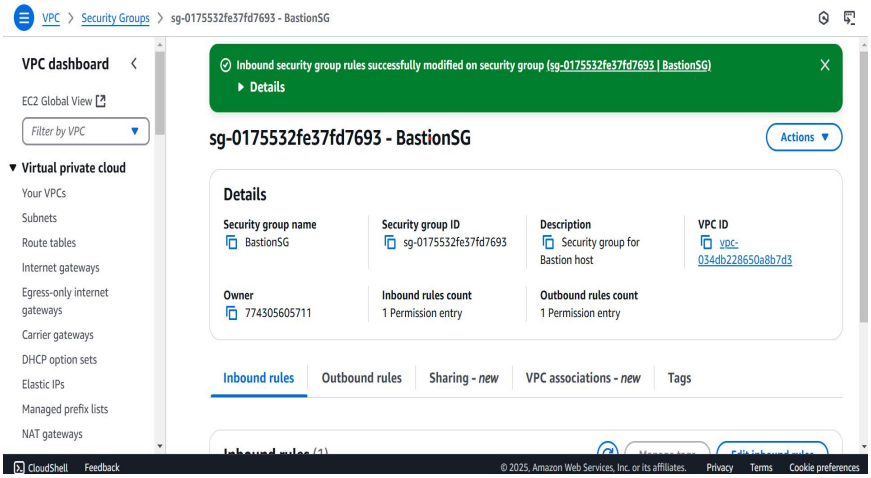


Step 6: Secure Your Bastion Host

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)



Disable Password Authentication

1. Edit SSH config:

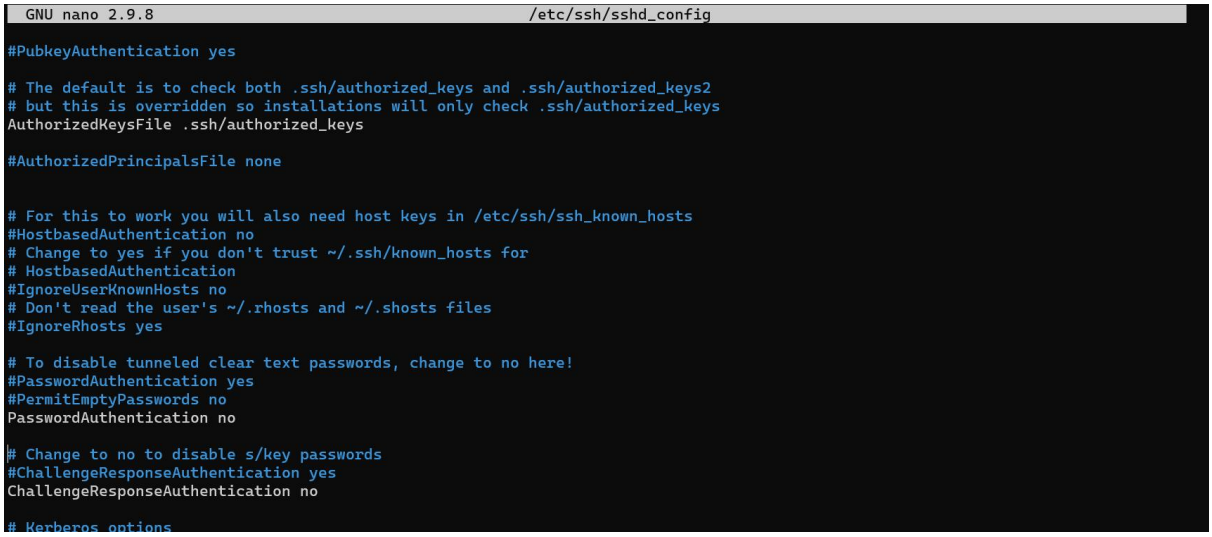
sudo nano /etc/ssh/sshd_config

2. Find and update these lines:

PasswordAuthentication no
PermitRootLogin no

1. Restart SSH service:s

sudo systemctl estart sshd



```
  GNU nano 2.9.8                    /etc/ssh/sshd_config

#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

Using a Bastion Host significantly enhances security by acting as a controlled access point to private instances. 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.