



Placement Empowerment Program
Cloud Computing and DevOps Centre

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

Name: CHELSIAH M

Department: CSE



Introduction

In cloud environments, securing access to private instances is crucial. A **Bastion Host** (or Jump Box) is a special-purpose instance that acts as a secure gateway to access EC2 instances in a private subnet. Instead of exposing private instances directly to the internet, users connect to the Bastion Host first and then access the private instances from there.

This setup **enhances security** by limiting direct SSH access to private instances and applying strict security controls.

Overview

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

What We Will Do?

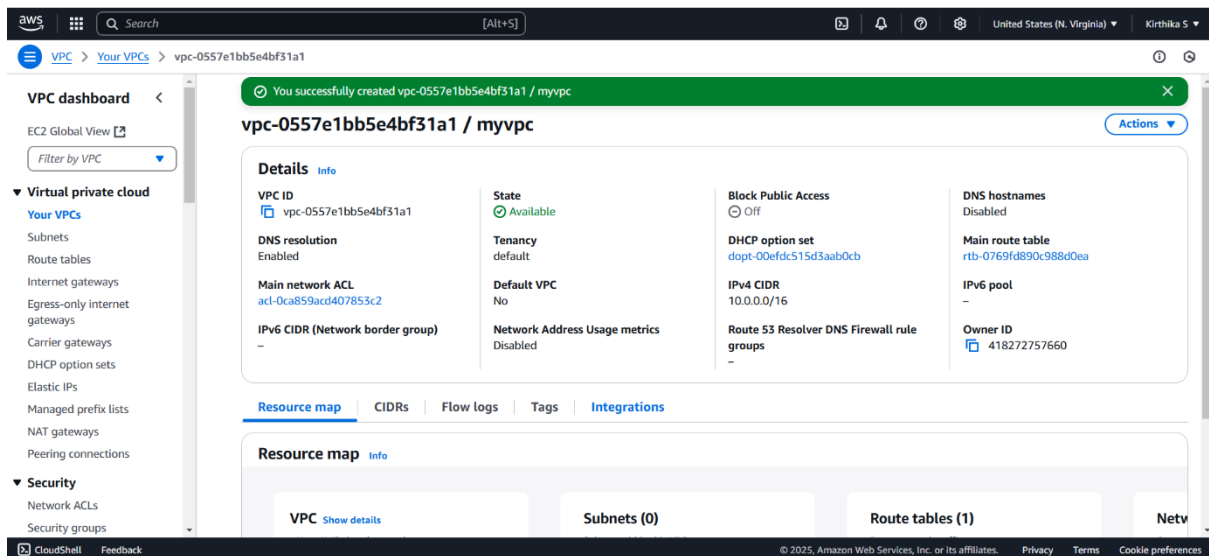
1. **Create a VPC** with a **Public and Private Subnet**.
2. **Set Up a Bastion Host** in the Public Subnet.
3. **Launch a Private EC2 Instance** in the Private Subnet.
4. **Configure Secure SSH Access** via the Bastion Host.
5. **Enhance Security** by restricting SSH access and considering AWS Systems Manager as an alternative.

Step 1:

Create a VPC with Public and Private Subnets

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.

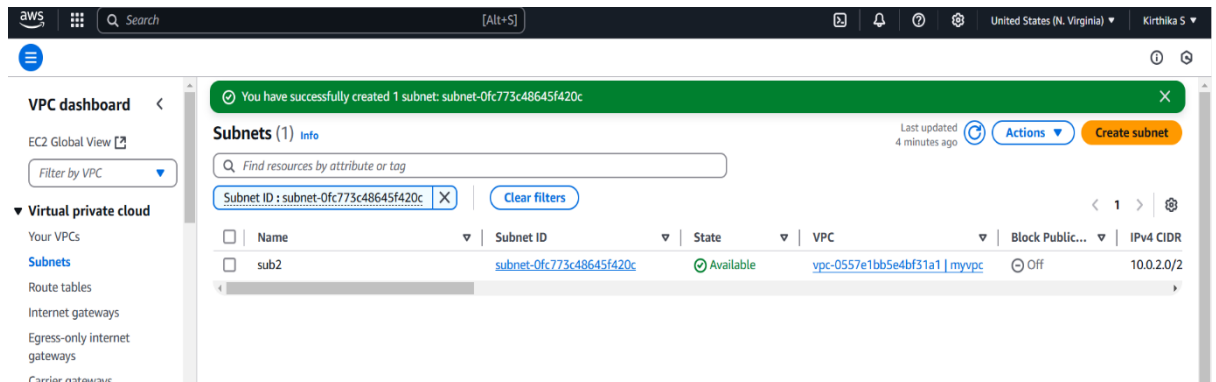


Create a Public Subnet

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

Create a Private Subnet

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

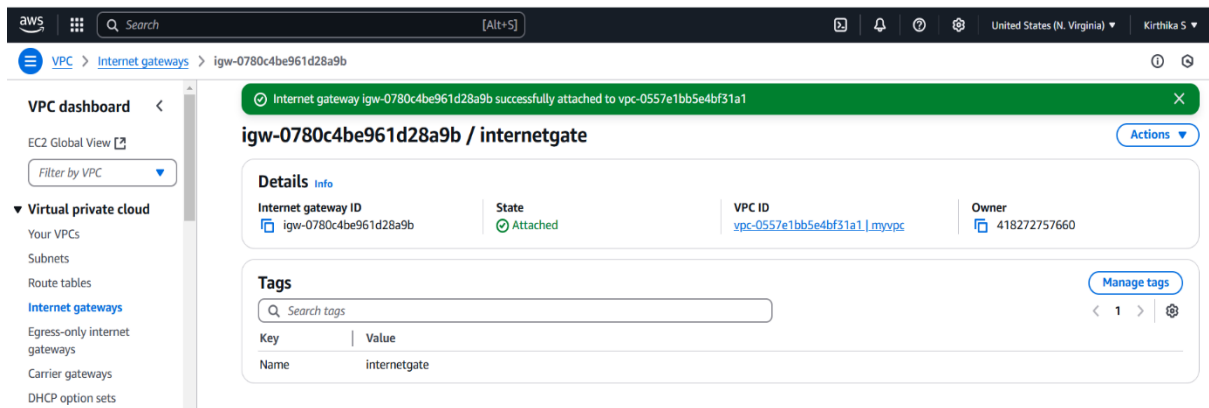


Step 2:

Configure Public Subnet for Internet Access Create an Internet Gateway (IGW)

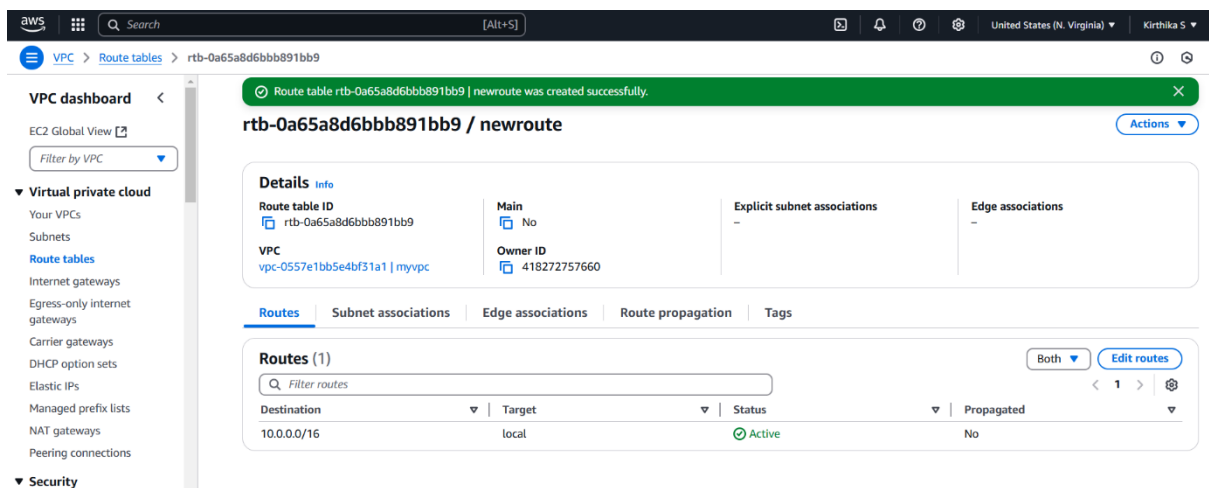
- Go to **Internet Gateways** → Click **Create Internet Gateway**.

- Name it **MyIGW**, attach it to **MyVPC**.



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**.

Key pair name - *required*
codesimple [Create new key pair](#)

Network settings [Info](#)

VPC - *required* [Info](#)
vpc-0557e1bb5e4bf31a1 (myvpc) [Create new VPC](#)

Subnet [Info](#)
subnet-0fc773c48645f420c [Create new subnet](#)
VPC: vpc-0557e1bb5e4bf31a1 Owner: 418272757660 Availability Zone: us-east-1a
Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.2.0/24

Auto-assign public IP [Info](#)
Disable

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
☒ Create security group ☐ Select existing security group

Security group name - *required*
launch-wizard-11

Summary

Number of instances [Info](#)
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...[read more](#)
ami-085ad6ae776d8f09c

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

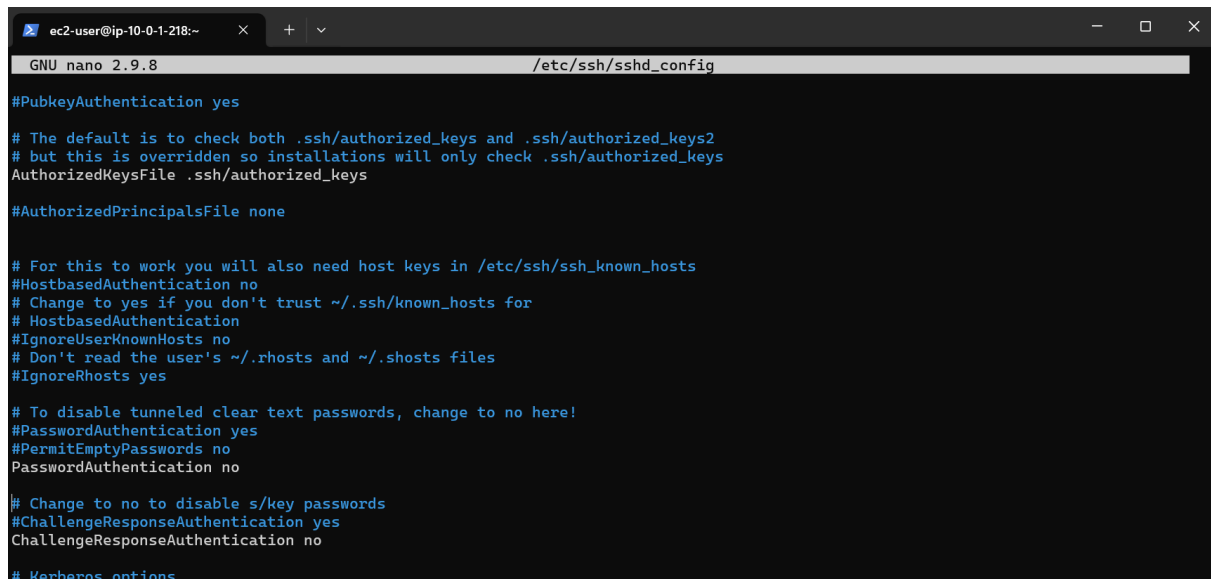
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which... [X](#)

[Cancel](#) [Launch instance](#) [Preview code](#)

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

```
ec2-user@ip-10-0-1-218:~  
PS C:\Users\oviya\Downloads> scp -i sam.pem sam.pem ec2-user@54.210.90.216::~/  
sam.pem                                     100% 1678      6.6KB/s   00:00  
PS C:\Users\oviya\Downloads> ^C  
PS C:\Users\oviya\Downloads> ssh -i sam.pem ec2-user@54.210.90.216  
Last login: Wed Feb  5 09:20:14 2025 from 182.74.154.218  
  
_#_  
|###_| Amazon Linux 2  
|\#####\  
||####| AL2 End of Life is 2026-06-30.  
|/#/  
V~! ~->  
A newer version of Amazon Linux is available!  
  
Amazon Linux 2023, GA and supported until 2028-03-15.  
https://aws.amazon.com/linux/amazon-linux-2023/  
  
[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:Y6FPLTZ5IAKtMwmbL3YqLSXOPKRyeylHTZPbylOrLY.  
ECDSA key fingerprint is MD5:d4:a6:0d:fa:99:92:df:21:ca:36:f0f: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  
|\#####\  
||####| AL2 End of Life is 2026-06-30.  
|/#/  
V~! ~->  
A newer version of Amazon Linux is available!  
  
Amazon Linux 2023, GA and supported until 2028-03-15.  
https://aws.amazon.com/linux/amazon-linux-2023/  
  
[ec2-user@ip-10-0-1-218 ~]$_$ |
```

```
sudo systemctl restart ssh
```



```
ec2-user@ip-10-0-1-218:~  
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.

Outcome

By completing this POC of setting up a Bastion Host in AWS, you will:

1. Deploy a bastion host in a public subnet and a private instance in a private subnet for secure access.
2. Enable SSH access to the private instance through the bastion host, ensuring the private instance remains isolated from the internet.
3. Configure security groups to restrict network traffic and enforce access control based on best practices.
4. Verify connectivity and communication between the bastion host and private instance within the VPC.