



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: Sylashri Rajendran
Department: IT



Introduction

A bastion host is a secure server that acts as a bridge between public and private networks. In cloud environments, a bastion host is used to securely access instances in private subnets, as direct internet access is restricted for security reasons. This Proof of Concept (POC) demonstrates how to set up a bastion host in AWS to access private instances while ensuring robust network security.

Overview

In this POC, we design and implement a secure architecture using AWS services. The project involves:

1. Creating a custom Virtual Private Cloud (VPC) with public and private subnets.
2. Launching an EC2 instance (bastion host) in the public subnet and a private instance in the private subnet.
3. Configuring security groups to control network traffic and enable secure access.
4. Using the bastion host as an intermediary to SSH into the private instance without exposing it directly to the internet.

The POC verifies secure access by testing connectivity, verifying the private instance's setup, and ensuring proper configurations.

Objectives

The primary objectives of this POC are:

1. Learn Network Segmentation:

Understand how to segregate public and private resources within a VPC.

2. Secure Private Resources:

Enable access to private instances without exposing them to the internet.

3. Practice Secure Access Techniques:

Use a bastion host to securely SSH into a private instance.

4. Apply Security Best Practices:

Use key-based authentication, restrict inbound traffic, and follow the principle of least privilege in security group configurations.

Importance

This POC is essential for anyone aiming to:

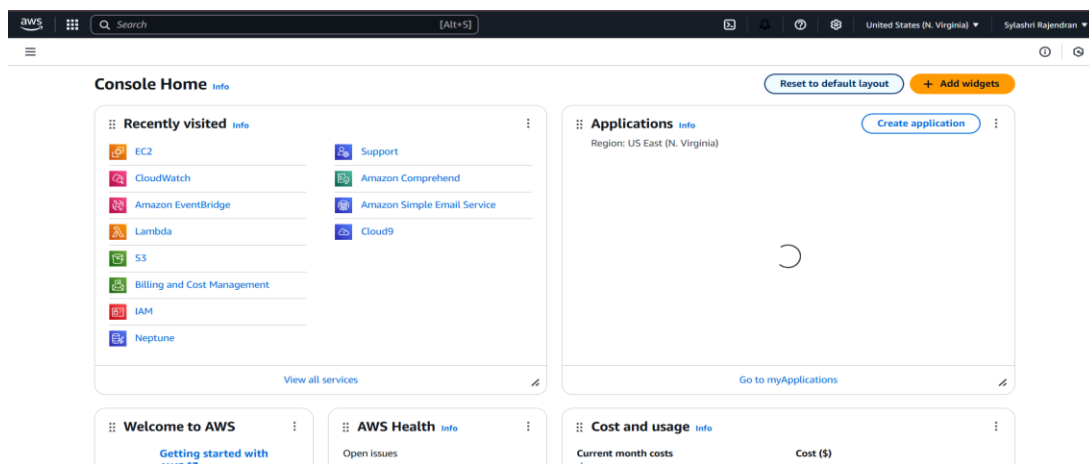
- 1. Enhance Security Skills:** Learn the fundamentals of securing cloud-based architectures by isolating sensitive resources.

2. **Prepare for Real-World Scenarios:** Bastion hosts are frequently used in enterprise environments where private resources need secure access.
3. **Develop Cloud Expertise:** Gain hands-on experience with AWS services like EC2, VPC, and security groups.
4. **Build Foundational Knowledge:** This knowledge is crucial for advanced cloud topics, such as setting up VPNs, NAT gateways, or using AWS Systems Manager for access.

Step-by-Step Overview

Step 1:

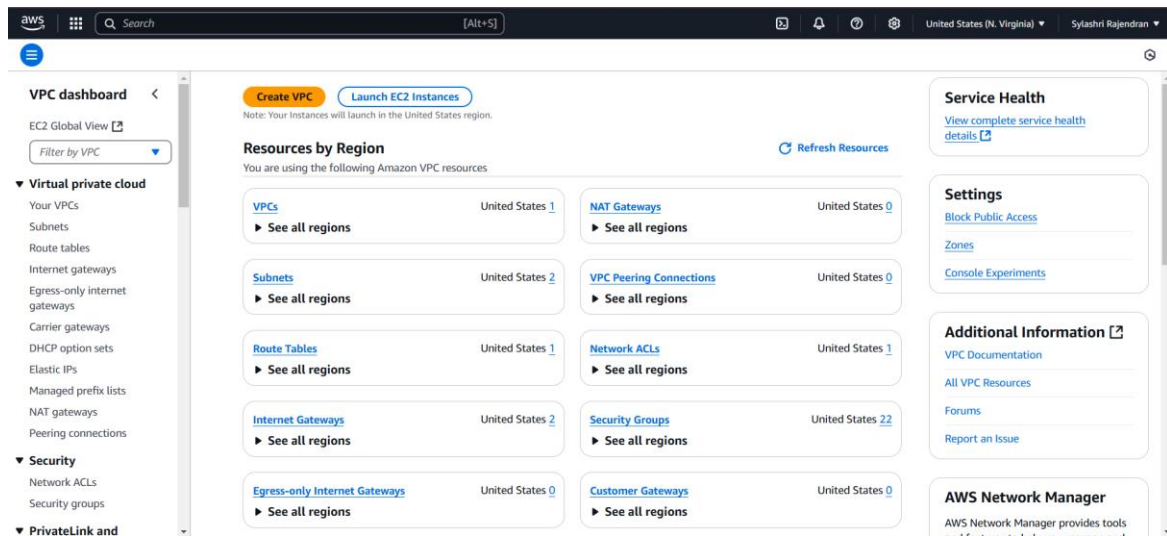
1. Go to [AWS Management Console](#).
2. Enter your username and password to log in.



Step 2:

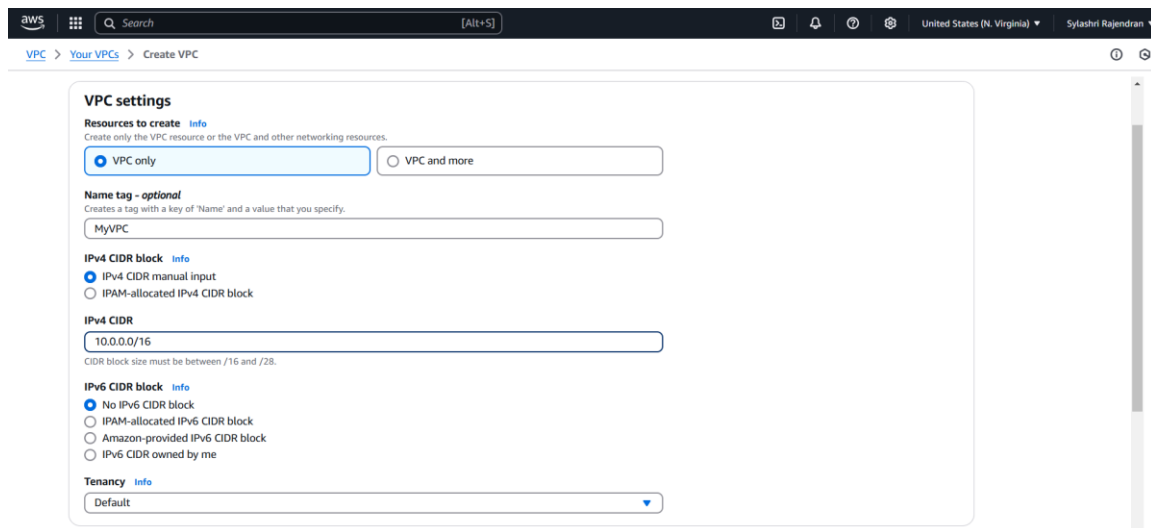
Search for **VPC** in the AWS search bar and click on it.

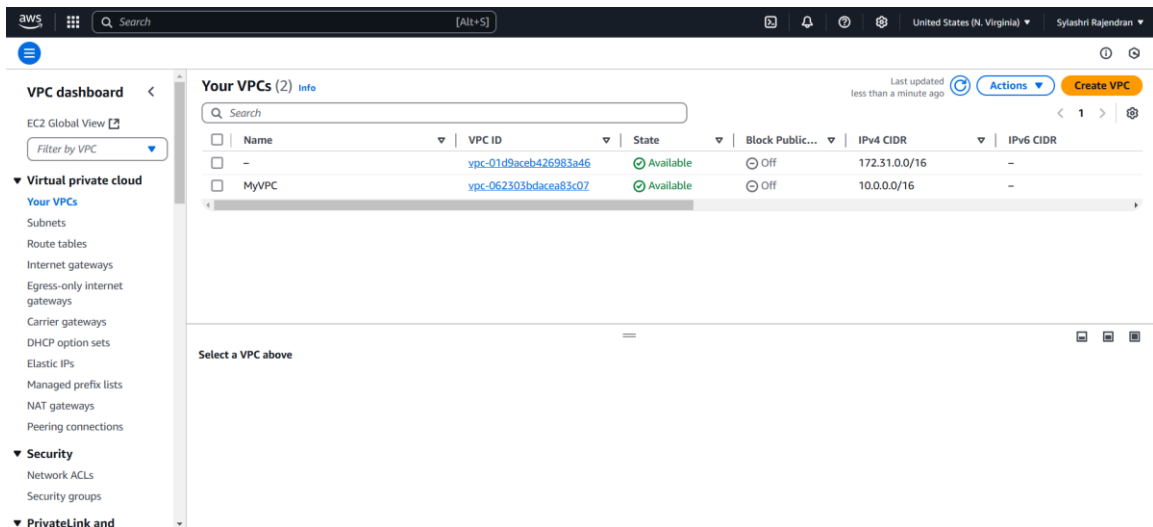
Click on **Create VPC**.



Step 3:

Create a new VPC by selecting **VPC only** and filling in the following details: set the **Name Tag** as *MyVPC* and the **IPv4 CIDR Block** as *10.0.0.0/16*. Leave all other settings as default, then click **Create VPC**. Once created, the new VPC will appear in the VPC list.

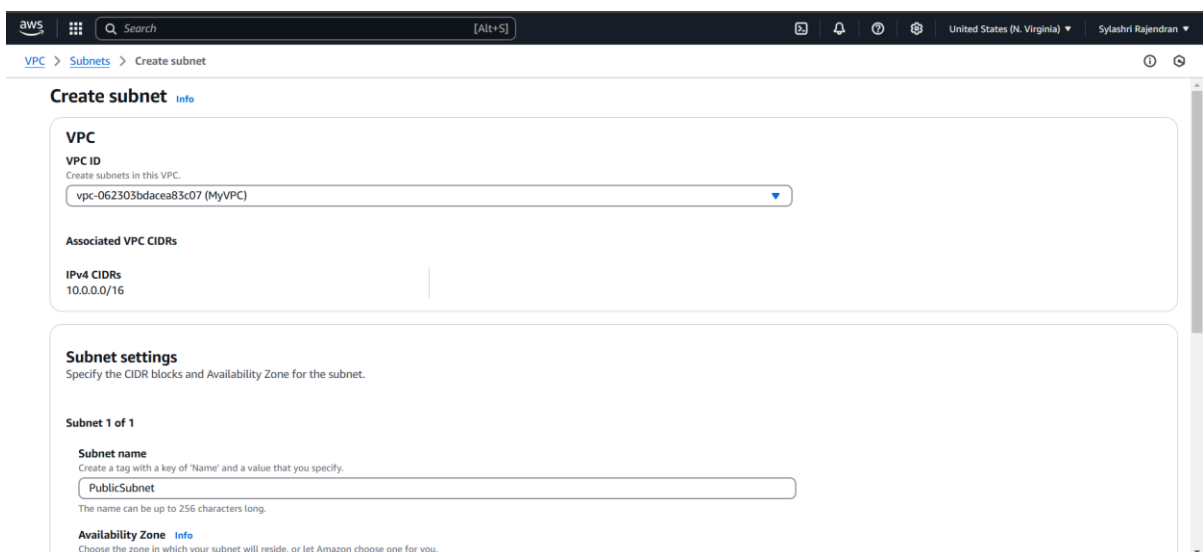




Step 4:

In the **VPC Dashboard**, go to **Subnets** and click **Create Subnet**.

Select the **VPC ID** of the VPC you created earlier (*MyVPC*). Enter the **Subnet Name** as *PublicSubnet*, choose an **Availability Zone** (e.g., *us-east-1a*), and set the **IPv4 CIDR Block** as *10.0.1.0/24*. Click **Create Subnet**.



Step 5:

Select your **PublicSubnet** from the list, click **Actions** → **Modify auto-assign IP settings**, check **Enable auto-assign public IPv4 address**, and click **Save**.

The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user information. Below the navigation bar, a breadcrumb trail indicates the path: VPC > Subnets > subnet-033090b42286470af > Edit subnet settings. The main content area is titled 'Edit subnet settings' with an 'Info' link. It contains three sections: 1. 'Subnet' section showing 'Subnet ID' as 'subnet-033090b42286470af' and 'Name' as 'PublicSubnet'. 2. 'Auto-assign IP settings' section with a description: 'Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.' It has two checkboxes: 'Enable auto-assign public IPv4 address' (checked) and 'Enable auto-assign customer-owned IPv4 address' (unchecked, with a note 'Option disabled because no customer owned pools found.'). 3. 'Resource-based name (RBN) settings' section with a description: 'Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.' It has two checkboxes: 'Enable resource name DNS A record on launch' (unchecked) and 'Enable resource name DNS AAAA record on launch' (unchecked). Below these is a 'Hostname type' section with two radio buttons: 'Resource name' (unchecked) and 'IP name' (checked).

Edit subnet settings [Info](#)

Subnet

Subnet ID [Info](#) subnet-033090b42286470af

Name [Info](#) PublicSubnet

Auto-assign IP settings [Info](#)

Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

☒ Enable auto-assign public IPv4 address [Info](#)

☐ Enable auto-assign customer-owned IPv4 address [Info](#)
Option disabled because no customer owned pools found.

Resource-based name (RBN) settings [Info](#)

Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

☐ Enable resource name DNS A record on launch [Info](#)

☐ Enable resource name DNS AAAA record on launch [Info](#)

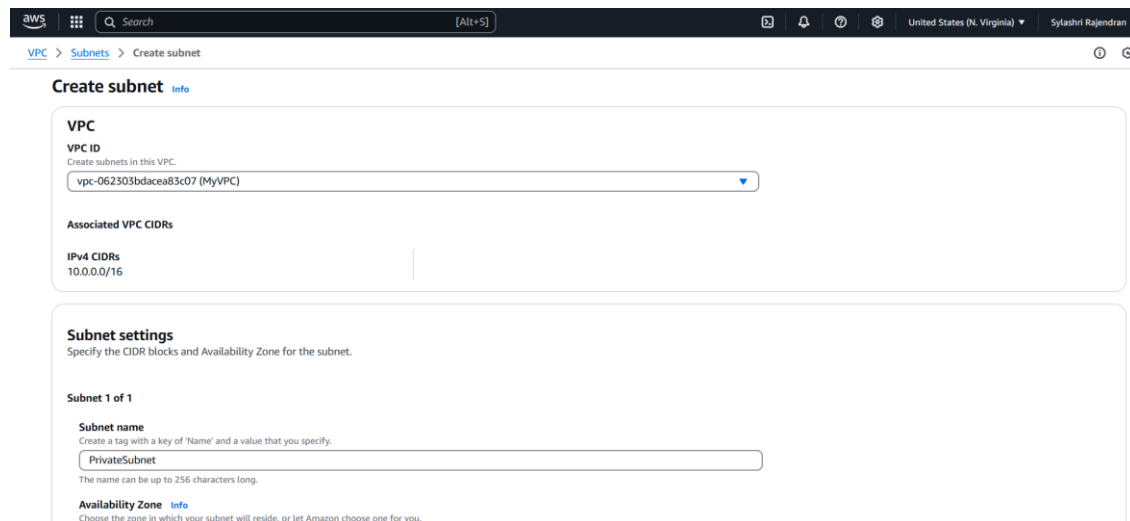
Hostname type [Info](#)

☐ Resource name

☒ IP name

Step 6:

Click **Create Subnet** again and fill in the details: select the same **VPC ID** (*MyVPC*), set **Subnet Name** to *PrivateSubnet*, use the same **Availability Zone** as the public subnet (e.g., *us-east-1a*), and set the **IPv4 CIDR Block** to *10.0.2.0/24*. Leave **auto-assign public IP** disabled and click **Create Subnet**.



The screenshot shows the AWS Management Console interface for the 'Create subnet' page. The breadcrumb navigation at the top indicates the path: VPC > Subnets > Create subnet. The page title is 'Create subnet' with an 'Info' link. The form is divided into two main sections: 'VPC' and 'Subnet settings'. In the 'VPC' section, the 'VPC ID' is set to 'vpc-062303bdacea83c07 (MyVPC)' from a dropdown menu. Below it, the 'Associated VPC CIDRs' section shows 'IPv4 CIDRs' as '10.0.0.0/16'. The 'Subnet settings' section has a subtitle 'Specify the CIDR blocks and Availability Zone for the subnet.' and indicates 'Subnet 1 of 1'. The 'Subnet name' field is set to 'PrivateSubnet' with a note that the name can be up to 256 characters long. The 'Availability Zone' field is currently empty, with a note to choose the zone or let Amazon choose one.

Step 7:

In the **VPC Dashboard**, go to **Internet Gateways** and click **Create Internet Gateway**. Name it *MyInternetGateway* and click **Create Internet Gateway**. Select your new gateway, click **Actions** → **Attach to VPC**, choose your VPC (*MyVPC*), and click **Attach Internet Gateway**.

The screenshot displays the AWS Management Console interface for the 'Internet gateways' section. The left-hand navigation pane is visible, with 'Virtual private cloud' expanded. The main area shows a table of Internet Gateways. The gateway 'MyInternetGateway' (ID: igw-01e0e69c946f48230) is selected. An 'Actions' dropdown menu is open, with 'Attach to VPC' highlighted. Below the table, the details for the selected gateway are shown, including its ID, state (Detached), and VPC ID.

Name	Internet gateway ID	State	VPC ID
-	igw-058eb02413d91fb73	Detached	-
-	igw-0e218aca30053bbd9	Attached	vpc-01d9aceb426983
MyInternetGateway	igw-01e0e69c946f48230	Detached	-

igw-01e0e69c946f48230 / MyInternetGateway

Details

Internet gateway ID igw-01e0e69c946f48230	State Detached	VPC ID -	Owner 842676011451
--	-------------------	-------------	-----------------------

The screenshot displays the AWS Management Console interface. At the top, the navigation bar shows 'AWS', a search bar, and the region 'United States (N. Virginia)'. The breadcrumb trail indicates the path: 'VPC' > 'Internet gateways' > 'igw-01e0e69c946f48230'. The left-hand navigation pane lists various VPC services, with 'Internet gateways' selected. The main content area features a green success banner at the top stating 'Internet gateway igw-01e0e69c946f48230 successfully attached to vpc-062303bdacea83c07'. Below this, the title 'igw-01e0e69c946f48230 / MyInternetGateway' is shown with an 'Actions' button. The 'Details' section includes a table with the following information:

Internet gateway ID	State	VPC ID	Owner
igw-01e0e69c946f48230	Attached	vpc-062303bdacea83c07 MyVPC	842676011451

Below the details, the 'Tags' section shows a search bar and a table with one tag:

Key	Value
Name	MyInternetGateway

Step 8:

In the **VPC Dashboard**, go to **Route Tables** and click **Create Route Table**. Name it *PublicRouteTable*, select your VPC (*MyVPC*), and click **Create Route Table**. Then, select *PublicRouteTable*, go to the **Routes** tab, click **Edit routes**, and add a route with **Destination** as *0.0.0.0/0* and **Target** as *MyInternetGateway*. Click **Save changes**.

The screenshot shows the 'Create route table' page in the AWS Management Console. The breadcrumb navigation is 'VPC > Route tables > Create route table'. The page title is 'Create route table' with an 'Info' link. A description states: 'A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.'

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.
PublicRouteTable

VPC
The VPC to use for this route table.
vpc-062303bdacea83c07 (MyVPC)

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key
Name

Value - optional
PublicRouteTable

Buttons: Remove, Add new tag, Cancel, Create route table.

The screenshot shows the 'Route table details' page for 'rtb-09826776f6c4cca0c / PublicRouteTable'. A green notification banner at the top says 'Updated routes for rtb-09826776f6c4cca0c / PublicRouteTable successfully'. The breadcrumb navigation is 'VPC > Route tables > rtb-09826776f6c4cca0c'. The left sidebar shows the 'VPC dashboard' with a 'Filter by VPC' dropdown and a list of VPC resources under 'Virtual private cloud', 'Security', and 'Privatel ink and'.

Details

Route table ID
rtb-09826776f6c4cca0c

VPC
vpc-062303bdacea83c07 | MyVPC

Main
No

Owner ID
842676011451

Explicit subnet associations
-

Edge associations
-

Routes (2)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-01e0e69c946f48230	Active	No
10.0.0.0/16	local	Active	No

Step 10:

Next, go to the **Subnet associations** tab of *PublicRouteTable*, click **Edit subnet associations**, check the box for *PublicSubnet*, and click **Save associations**.

The screenshot shows the AWS Management Console interface for editing subnet associations. The breadcrumb trail is: VPC > Route tables > rtb-09826776f6c4cca0c > Edit subnet associations. The page title is "Edit subnet associations" with a subtitle "Change which subnets are associated with this route table." Below this is a section titled "Available subnets (1/2)" with a search bar "Filter subnet associations". A table lists the available subnets:

	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/>	PublicSubnet	subnet-033090b42286470af	10.0.1.0/24	-	Main (rtb-08e26e6d8be86d576)
<input type="checkbox"/>	PrivateSubnet	subnet-00be8f05f0f37c70c	10.0.2.0/24	-	Main (rtb-08e26e6d8be86d576)

Below the table is a section titled "Selected subnets" which contains a single entry: "subnet-033090b42286470af / PublicSubnet" with a close button. At the bottom right are "Cancel" and "Save associations" buttons.

Step 10:

In the **EC2 Dashboard**, click **Launch Instance** and configure: set **Name** as *BastionHost*, select *Amazon Linux 2 AMI (HVM)* - Free Tier eligible, and choose

t2.micro as the **Instance Type**. For **Key Pair**, create or select one, downloading the .pem file if creating. Under **Network Settings**, select *MyVPC* for the **VPC**, *PublicSubnet* for the **Subnet**, and ensure **Auto-assign Public IP** is enabled. Create a **Security Group** to allow SSH (port 22) access, setting **Source** to *MyIP*. Use the default storage of 8 GiB, click **Launch Instance**, and wait for it to initialize.

The screenshot shows the AWS Management Console interface for launching an instance. The 'Network settings' section is expanded, showing the following configuration:

- VPC - required**: vpc-062303bdacea83c07 (MyVPC)
- Subnet**: subnet-033090b42286470af (PublicSubnet)
- Auto-assign public IP**: Enable
- Firewall (security groups)**: Create security group (selected)
- Security group name - required**: launch-wizard-13
- Description - required**: launch-wizard-13 created 2025-02-14T16:04:38.023Z

The 'Summary' section on the right shows the following configuration:

- Number of instances**: 1
- Software Image (AMI)**: Amazon Linux 2023.6.2...read more
- Virtual server type (instance type)**: t2.micro
- Firewall (security group)**: New security group
- Storage (volumes)**: 1 volume(s) - 8 GiB

The 'Launch instance' button is visible at the bottom right of the 'Summary' section.

Step 11: Paste the command copied in the SSH client and connect it by using your key pair.


```
#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 13:

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

Attach SSM Managed Policy to EC2 IAM Role
(AmazonSSMManagedInstanceCore).

Enable SSM Agent (Pre-installed on Amazon Linux & Ubuntu).

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.
5. Gain a practical understanding of secure cloud networking and foundational AWS services like EC2, VPC, and key-based authentication.