how to check if subnet is private or public

**ChatGPT said:**

To check if a subnet is private or public in AWS, you need to verify if the subnet has a route to an Internet Gateway (IGW). Here’s how you can do it:

**Steps to Check if Subnet is Public or Private:**

1. **Log in to the AWS Console**:
   * Open the AWS Management Console and log in.
2. **Navigate to the VPC Dashboard**:
   * From the AWS Console, go to the **VPC Dashboard** by searching for "VPC" in the search bar.
3. **Check the Route Table of the Subnet**:
   * On the left panel, click on **Subnets** under **Virtual Private Cloud**.
   * Find and select the subnet you want to check.
   * Take note of the **Route Table** ID associated with the subnet (listed under the **Route Table** column).
4. **Inspect the Route Table**:
   * Click on **Route Tables** in the VPC Dashboard on the left panel.
   * Find the route table that corresponds to your subnet and click on it.
   * Check the **Routes** tab.
   * **If the route table has a route for 0.0.0.0/0 pointing to an Internet Gateway (IGW)**, the subnet is **public**.
   * **If the route table does not have a route to an IGW** but has a route to a **NAT Gateway** or **NAT Instance**, the subnet is **private**.

**Key Indicators:**

* **Public Subnet**:
  + Has a route to the **Internet Gateway (IGW)** for 0.0.0.0/0.
* **Private Subnet**:
  + Has no direct route to an Internet Gateway (IGW) for 0.0.0.0/0.
  + Often has a route to a **NAT Gateway** or **NAT Instance** to allow outbound traffic to the internet.

**Summary:**

* **Public Subnet** = Route to **Internet Gateway** (IGW).
* **Private Subnet** = No route to **Internet Gateway** (IGW), but may have a route to **NAT Gateway** or **NAT Instance**.

**Part 2**

how to check if subnet is private or public

###### ChatGPT said:

To check if a subnet is private or public in AWS, you need to verify if the subnet has a route to an Internet Gateway (IGW). Here’s how you can do it:

### Steps to Check if Subnet is Public or Private:

1. **Log in to the AWS Console**:
   * Open the AWS Management Console and log in.
2. **Navigate to the VPC Dashboard**:
   * From the AWS Console, go to the **VPC Dashboard** by searching for "VPC" in the search bar.
3. **Check the Route Table of the Subnet**:
   * On the left panel, click on **Subnets** under **Virtual Private Cloud**.
   * Find and select the subnet you want to check.
   * Take note of the **Route Table** ID associated with the subnet (listed under the **Route Table** column).
4. **Inspect the Route Table**:
   * Click on **Route Tables** in the VPC Dashboard on the left panel.
   * Find the route table that corresponds to your subnet and click on it.
   * Check the **Routes** tab.
   * **If the route table has a route for 0.0.0.0/0 pointing to an Internet Gateway (IGW)**, the subnet is **public**.
   * **If the route table does not have a route to an IGW** but has a route to a **NAT Gateway** or **NAT Instance**, the subnet is **private**.

### Key Indicators:

* **Public Subnet**:
  + Has a route to the **Internet Gateway (IGW)** for 0.0.0.0/0.
* **Private Subnet**:
  + Has no direct route to an Internet Gateway (IGW) for 0.0.0.0/0.
  + Often has a route to a **NAT Gateway** or **NAT Instance** to allow outbound traffic to the internet.

### Summary:

* **Public Subnet** = Route to **Internet Gateway** (IGW).
* **Private Subnet** = No route to **Internet Gateway** (IGW), but may have a route to **NAT Gateway** or **NAT Instance**.

Let me know if you need further assistance with any step!

4o

##### You said:

how to disable public ip address while creating ec2 isntace

###### ChatGPT said:

To disable the automatic assignment of a public IP address while creating an EC2 instance in AWS, follow these steps:

### Steps to Disable Public IP Address during EC2 Instance Creation:

1. **Login to AWS Console**:
   * Open the **AWS Management Console** and log in.
2. **Navigate to the EC2 Dashboard**:
   * From the AWS Console, search for **EC2** in the search bar and go to the **EC2 Dashboard**.
3. **Launch a New EC2 Instance**:
   * Click on the **Launch Instance** button to create a new EC2 instance.
4. **Choose an Amazon Machine Image (AMI)**:
   * Select your desired AMI (e.g., Amazon Linux 2, Ubuntu, etc.) and click **Select**.
5. **Select Instance Type**:
   * Choose an instance type (e.g., t2.micro) and click **Next: Configure Instance Details**.
6. **Configure Instance Details**:
   * In the **Configure Instance Details** section, scroll down to find the **Network** settings.
   * Under **Network**, select your desired **VPC**.
   * Under **Subnet**, select a **Private Subnet** or any subnet where you do not want a public IP assigned.
7. **Disable Public IP Assignment**:
   * Below the **Subnet** field, you’ll see a setting called **Auto-assign Public IP**.
     + By default, **Auto-assign Public IP** is set to **Enable**.
     + **Change it to Disable** by selecting **Disable** from the drop-down menu.
8. **Complete the Remaining Steps**:
   * Finish the rest of the setup process (e.g., Add Storage, Configure Security Group, etc.) and click **Review and Launch**.
   * After reviewing your settings, click **Launch** to create the EC2 instance.

### Summary:

* When **Auto-assign Public IP** is set to **Disable**, the instance will not be assigned a public IP address during its creation, making it accessible only within the VPC unless other network configurations (e.g., NAT Gateway) are used.

Let me know if you need further clarification or additional steps!

10.0.0.128/25 – 32-25=7 = 2 ki power 7 it has ip addres assocaiate with subnet

**Part 3**

I understand now! Here's how the VPC architecture breaks down:

1. **VPC (Virtual Private Cloud)**: The outermost container that holds your network resources. It is a logically isolated network within AWS where you define your network settings.
   * **Inside the VPC**, you have:
     + **Subnets**: These are subdivisions of the VPC network. Subnets can be either:
       - **Public Subnet**: Can directly access the internet via an Internet Gateway (IGW).
       - **Private Subnet**: Does not have direct access to the internet. Uses a NAT Gateway or NAT Instance for outbound traffic to the internet.
   * **Inside the Subnet**, you deploy your resources, such as:
     + **EC2 Instances**: Your application is deployed here (for example, in the private subnet if you want it to be isolated from direct internet access).
   * **Route Tables**:
     + Each subnet is associated with a **Route Table**, which determines the traffic routing between subnets, to/from the internet, and between different resources inside your VPC.
     + The **Private Subnet Route Table** will route internal traffic and may route external traffic through a **NAT Gateway** for internet access.
     + The **Public Subnet Route Table** will route traffic directly to the **Internet Gateway (IGW)** for direct internet access.

**Visualizing this structure:**

* **VPC (outermost box)**:
  + **Public Subnet**:
    - Contains resources like web servers, which need internet access.
  + **Private Subnet**:
    - Contains application servers, databases, or backend systems that do not need direct internet access.
  + **Route Tables**: Associated with the subnets for traffic routing.

Let me know if you'd like me to generate a new diagram with this structure!

4o

Top of Form

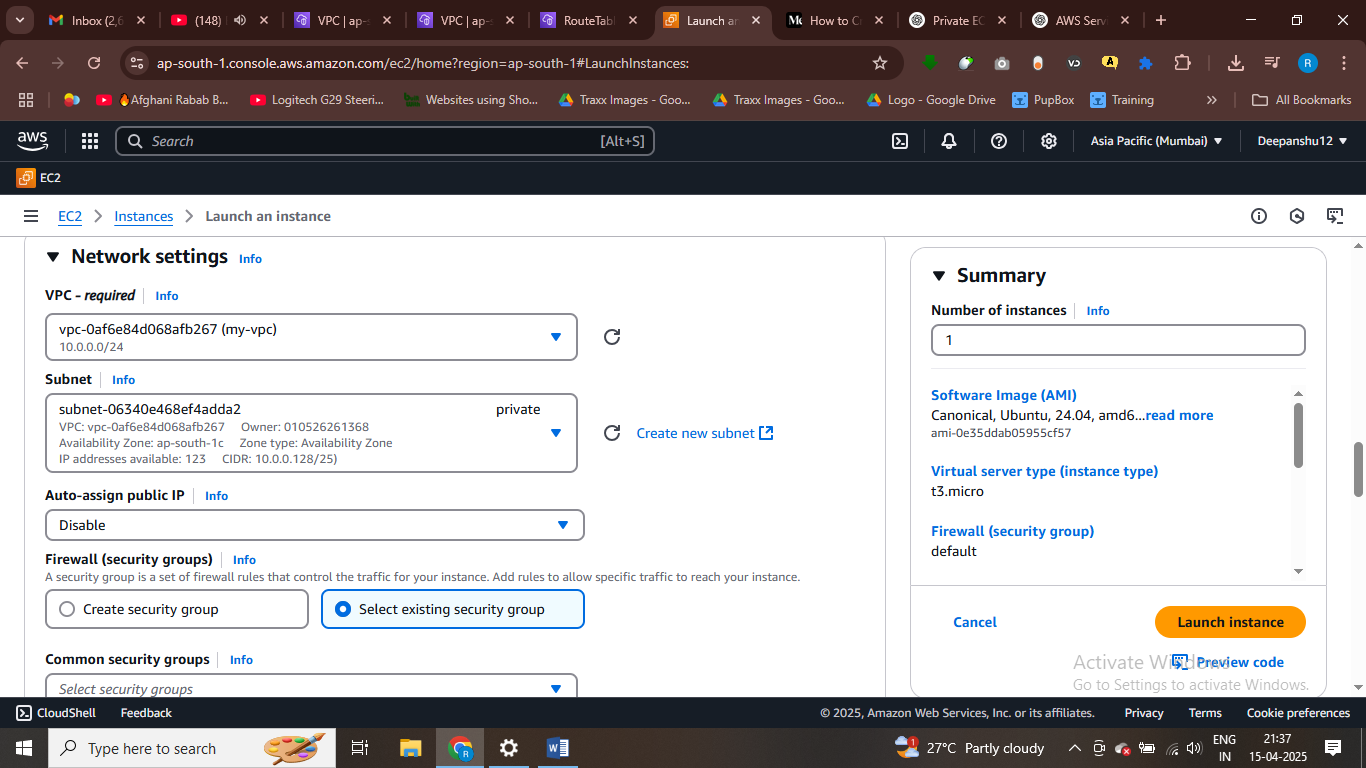


Search

Deep research

Bottom of Form

ChatGPT can make mistakes. Check important info.



**Part 4**

how to ssh into private instance tell me please

**ChatGPT said:**

To SSH into a private EC2 instance (which does not have a public IP), you'll need to follow these steps:

**1. Ensure You Have a Bastion Host (Jump Box)**

* **Bastion Host**: A Bastion Host is an EC2 instance located in a **public subnet** that allows you to access resources in the **private subnet** via SSH.
* This host will act as a "jump box" to securely SSH into private instances.

**2. Set Up the Bastion Host**

* **Launch an EC2 instance** in a **public subnet**.
* Ensure this instance has a **public IP** and **SSH access** (via security group settings) enabled.
* **Security Group**: The security group of the Bastion host should allow incoming SSH traffic (port 22) from your IP address.

**3. Configure Security Groups for the Private Instance**

* For your **private EC2 instance**, make sure the security group allows inbound SSH traffic **only from the Bastion Host**.
* To do this, go to the security group of the private instance, and under **Inbound Rules**, set the source of port 22 (SSH) to the **private IP** of the Bastion host (or the security group ID of the Bastion host).

**4. SSH Into the Bastion Host**

* From your local machine, SSH into the Bastion host first using its public IP:

bash

CopyEdit

ssh -i /path/to/your/bastion-key.pem ec2-user@bastion-public-ip

**5. SSH Into the Private Instance From the Bastion Host**

* Once you are inside the Bastion host, you can SSH into the private instance using its **private IP**:

bash

CopyEdit

ssh -i /path/to/your/private-instance-key.pem ec2-user@private-instance-private-ip

**6. Optional: Use SSH Agent Forwarding**

* If you don’t want to store your private keys on the Bastion host, you can use **SSH agent forwarding**.
* This allows you to forward your SSH key from your local machine to the Bastion host and directly access the private instance.
* Use the -A flag when connecting to the Bastion host:

bash

CopyEdit

ssh -A -i /path/to/your/bastion-key.pem ec2-user@bastion-public-ip

**7. Verify Connectivity**

* After following the steps above, you should be able to access the private EC2 instance via the Bastion host.

**Diagram:**

* **Public Subnet**: Bastion Host → Public IP
* **Private Subnet**: Private Instance → Private IP (accessible via Bastion Host)

Let me know if you need more details or run into any issues during the process!

**Part 6**

steps for ssh into private instance using bostion host - please only steps with cmds that we used

###### ChatGPT said:

Here are the steps to SSH into a private instance using a Bastion Host with the necessary commands:

### Steps to SSH into Private Instance Using Bastion Host:

#### 1. **SSH into the Bastion Host (Public Instance)**

* **Command**:

bash

CopyEdit

ssh -i /path/to/your/bastion-key.pem ec2-user@bastion-public-ip

* Replace /path/to/your/bastion-key.pem with the path to your **Bastion Host private key** and bastion-public-ip with the **public IP** of your Bastion Host.

#### 2. **SSH from the Bastion Host to the Private Instance**

* Once logged into the Bastion Host, SSH into the **Private EC2 Instance** using its **Private IP**:

bash

CopyEdit

ssh -i /path/to/your/private-instance-key.pem ec2-user@private-instance-private-ip

* Replace /path/to/your/private-instance-key.pem with the **private key** for the **private instance** and private-instance-private-ip with the **private IP** of your private EC2 instance.

#### 3. **Optional: Using SSH Agent Forwarding**

* If you don’t want to store your private key on the Bastion Host, use **SSH Agent Forwarding**.
* **SSH into Bastion Host with Agent Forwarding**:

bash

CopyEdit

ssh -A -i /path/to/your/bastion-key.pem ec2-user@bastion-public-ip

* Then, from the Bastion Host, SSH into the private instance:

bash

CopyEdit

ssh ec2-user@private-instance-private-ip

### Summary of Commands:

1. **SSH into Bastion Host**:

bash

CopyEdit

ssh -i /path/to/your/bastion-key.pem ec2-user@bastion-public-ip

1. **SSH from Bastion Host to Private Instance**:

bash

CopyEdit

ssh -i /path/to/your/private-instance-key.pem ec2-user@private-instance-private-ip

OR with **SSH Agent Forwarding**:

bash

CopyEdit

ssh -A -i /path/to/your/bastion-key.pem ec2-user@bastion-public-ip

ssh ec2-user@private-instance-private-ip

# How to Create a VPC on AWS and SSH into Private EC2 Instances Using MobaXterm

[[Taimoor Haider Ishtiaq](https://medium.com/@taimoorhaiderx?source=post_page---byline--7b962d29d2d0---------------------------------------)](https://medium.com/@taimoorhaiderx?source=post_page---byline--7b962d29d2d0---------------------------------------)

[Taimoor Haider Ishtiaq](https://medium.com/@taimoorhaiderx?source=post_page---byline--7b962d29d2d0---------------------------------------)

·

Follow

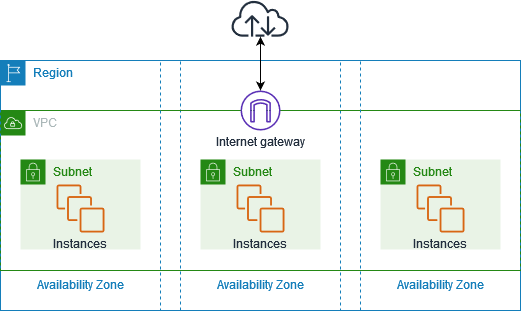
6 min read

·

Mar 19, 2025

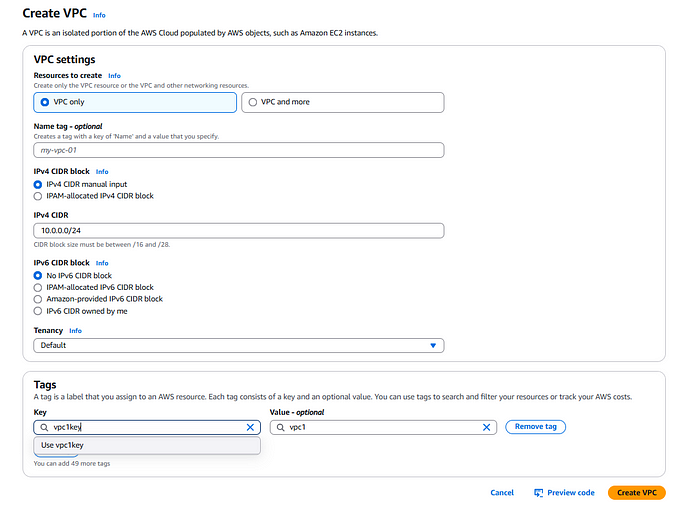
1

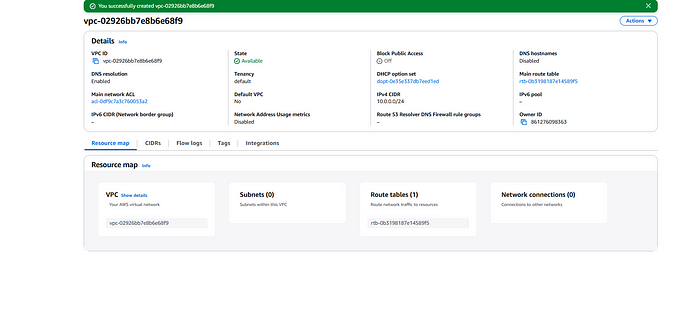
First, we will create a **VPC**, then set up **public and private subnets**. Next, we will assign **routing tables** (with an **Internet Gateway**) to the **public subnet**. Finally, we will create **EC2 instances** in both subnets and access them using **MobaXterm**.



# 1. Creating a VPC

We will assign a **CIDR block** (e.g., **10.0.0.0/24**) to define the IP range for our **VPC**.





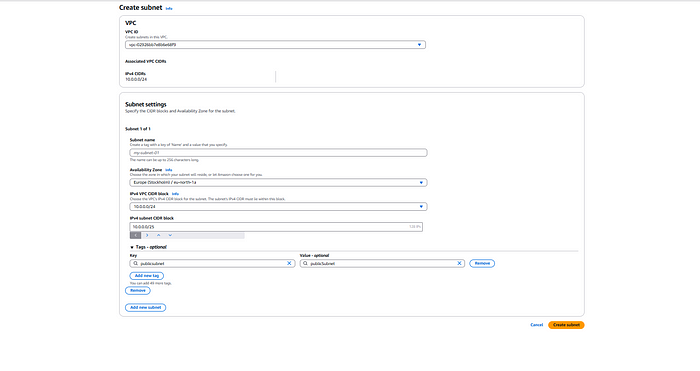
Now, we will perform **subnetting** on **10.0.0.0/24** (our **Parent Network**).

# **Subnetting**

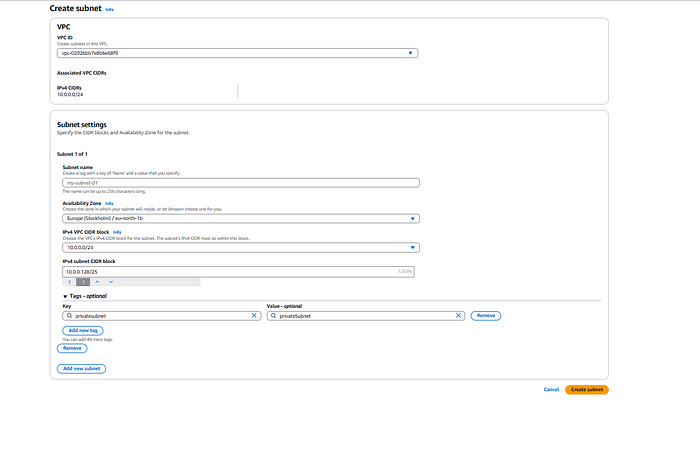
*Subnetting the****10.0.0.0/24****network involves dividing it into two smaller subnets by borrowing 1 additional bit from the host portion. The original****/24 subnet****has****256 IP addresses (10.0.0.0–10.0.0.255)****. By increasing the subnet mask to****/25****, we create****two equal subnets****, each with****128 IP addresses****. The first subnet is****10.0.0.0/25****(usable: 10.0.0.1–10.0.0.126, broadcast: 10.0.0.127), and the second is****10.0.0.128/25****(usable: 10.0.0.129–10.0.0.254, broadcast: 10.0.0.255). This process efficiently organizes IP allocation while maintaining logical separation.*

# 1.1 Creating Subnet 1 (Public Subnet)

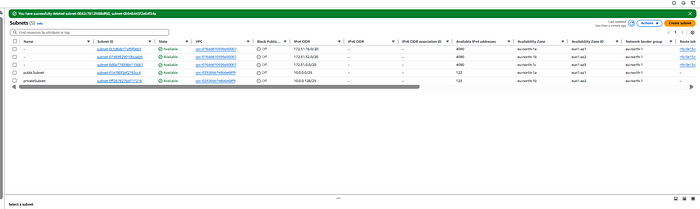
Create subnets in differenct availibility zones.



# 1.2 Creating Subnet 2 (Private Subnet)



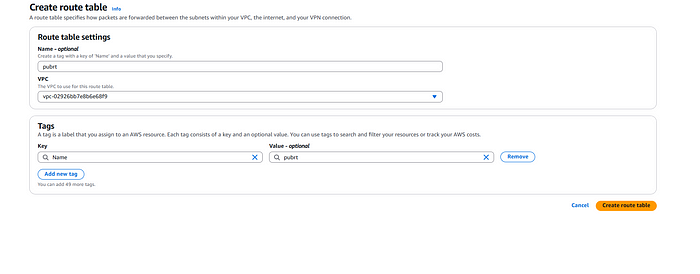
Our subnets are created.



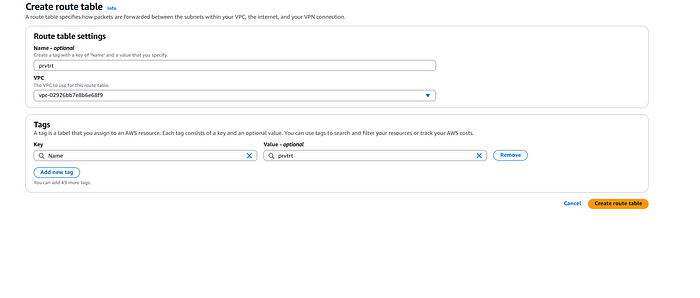
# 2. Creating Routing Tables

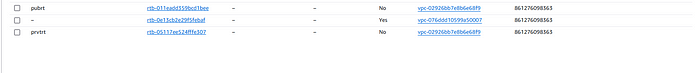
Routing tables define how network traffic is directed within the VPC. We will create and configure two routing tables:

* **Public Route Table:** This will be associated with the public subnet and will have a route pointing to the Internet Gateway, allowing instances in the public subnet to communicate with the internet.

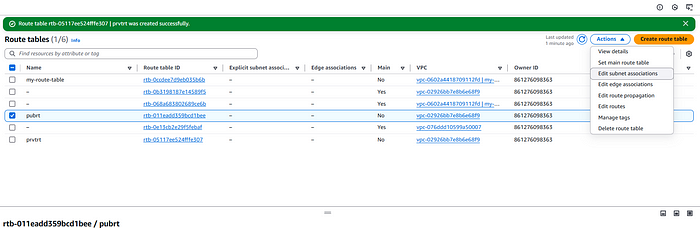


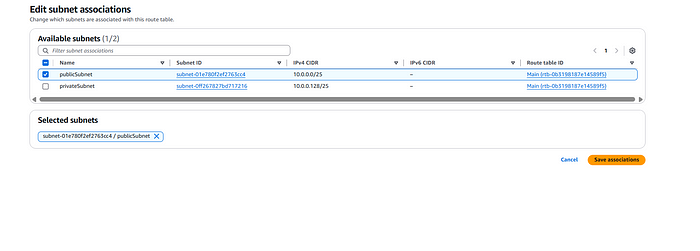
* **Private Route Table:** This will be associated with the private subnet, ensuring private instances do not have direct internet access.

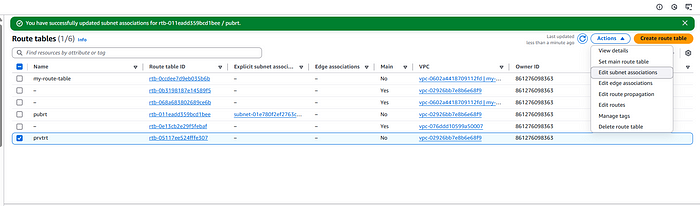


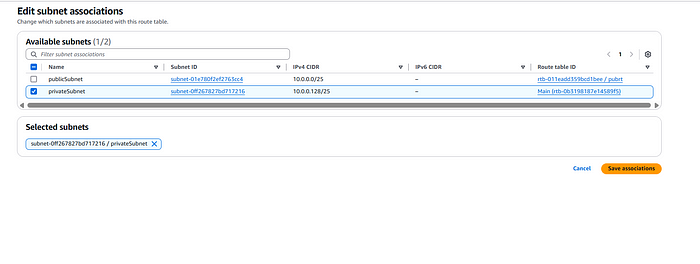


Associating the routing tables to the public and private subnets.







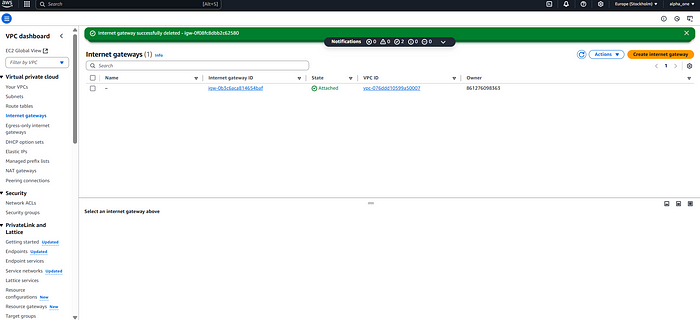


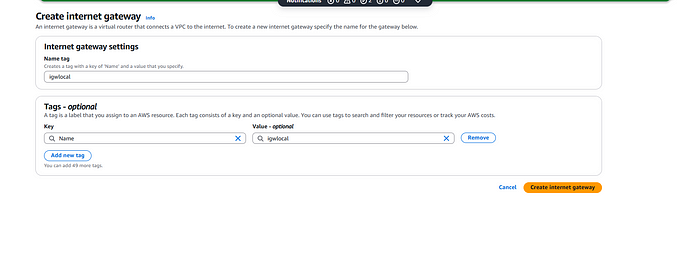
# 3. Creating an Internet Gateway

An **Internet Gateway (IGW)** allows instances in the public subnet to access the internet while enabling inbound traffic from external sources. In this step, we will:

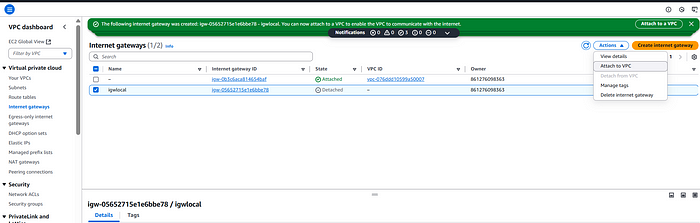
* Create an **Internet Gateway** in AWS.
* Attach the **Internet Gateway** to the VPC.
* Update the **Public Route Table** to direct internet-bound traffic (0.0.0.0/0) through the Internet Gateway.

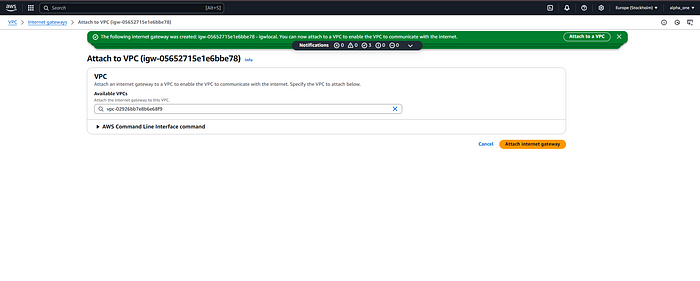
This ensures that instances in the **public subnet** can access the internet, enabling software updates, external API calls, and SSH connections.

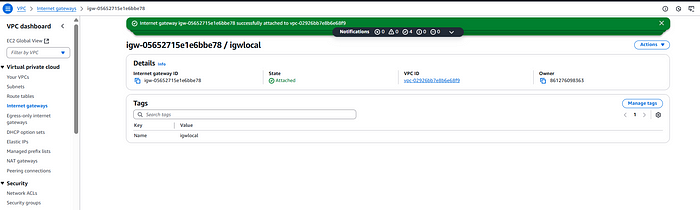




We have created the internet gateway (i.e igwlocal). e will attach this to our VPC.





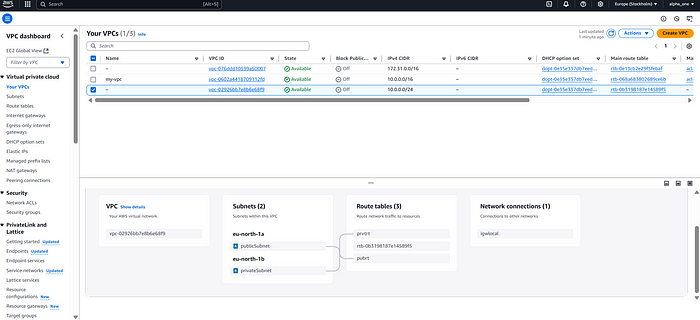


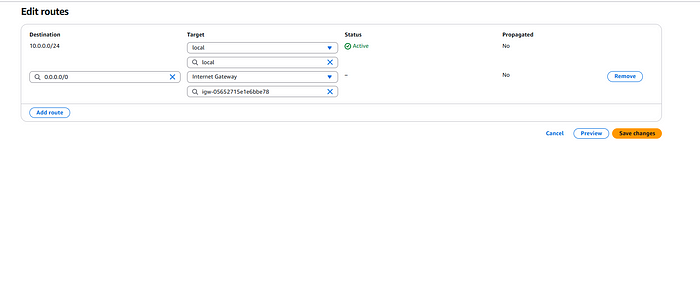
# 4. Attaching the Internet Gateway to the Public Subnet

While the **Internet Gateway (IGW)** is attached to the **VPC**, it does not automatically provide internet access to instances. We need to ensure that the public subnet can route traffic through it.

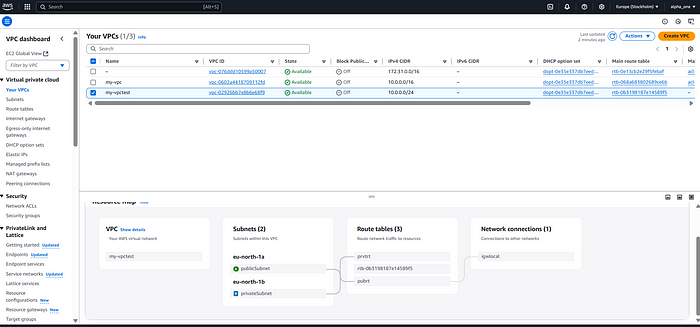
In this step, we will:

* Modify the **Public Route Table** to include a route for 0.0.0.0/0, directing traffic to the **Internet Gateway**.
* Associate the **Public Route Table** with the **Public Subnet**.





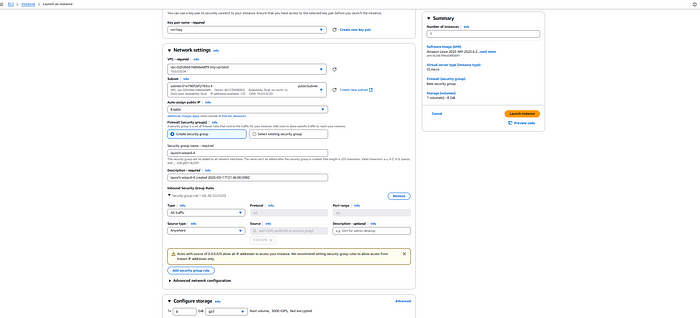
We can see in the Resource map of the VPC that the internet gateway is attached to the public subnet. This step ensures that instances in the **public subnet** can communicate with the internet while keeping the **private subnet** isolated.



# 5. Launching EC2 Instances in Public and Private Subnets

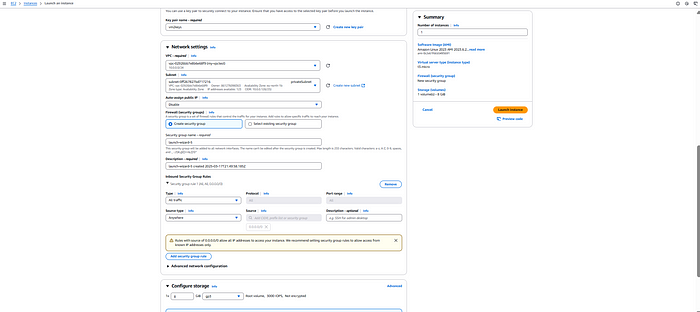
Now, we will create two **EC2 instances**:

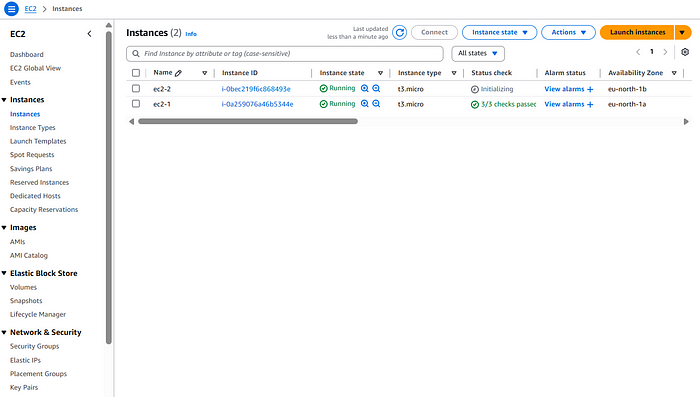
1. **Public Instance:** This instance will be launched in the **public subnet** and will have a **public IP address** assigned, allowing direct SSH access over the internet.



1. **Private Instance:** This instance will be launched in the **private subnet**, and we must **disable the public IP** to ensure it remains isolated from direct internet access.

The **public instance** will act as a **bastion host**, which we will later use to access the **private instance** securely.





# 6. Accessing the Public Instance Using MobaXterm

Now, we will use **MobaXterm** to **SSH into the public EC2 instance** in the **public subnet**.

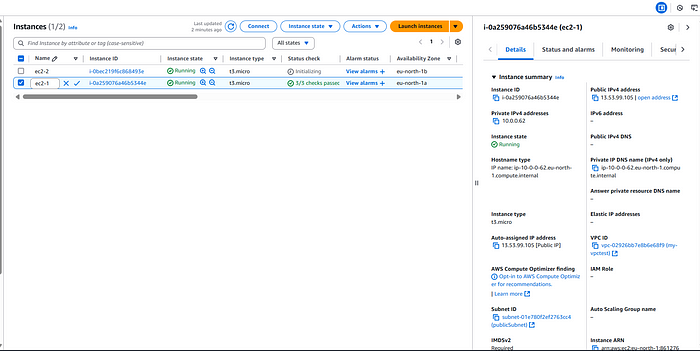
## What is MobaXterm?

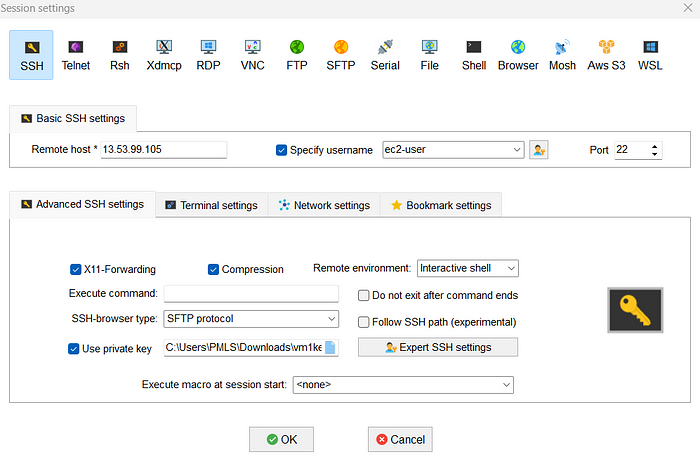
MobaXterm is a powerful **remote access tool** that provides an enhanced terminal for developers and system administrators. It supports **SSH, SFTP, RDP, and many other remote protocols**, making it ideal for managing cloud-based servers like AWS EC2 instances.

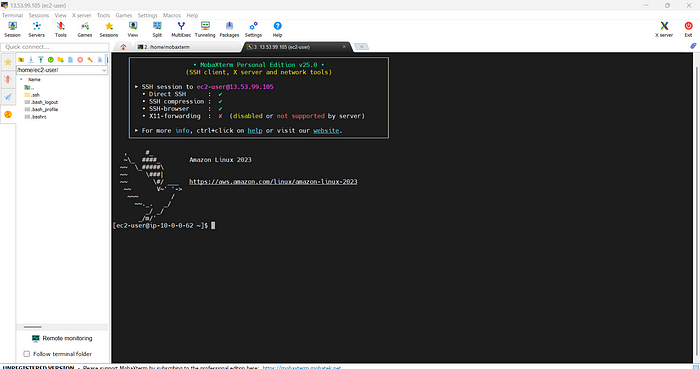
## Steps to SSH into the Public Instance using MobaXterm

1. **Open MobaXterm** on your local machine.
2. Click on **“Session”** → Select **“SSH”**.
3. In the **“Remote Host”** field, enter the **public IP address** of your **public EC2 instance**.
4. Under **“Advanced SSH settings”**, browse and select your **.pem key file**.
5. Click **“OK”** to start the SSH session.

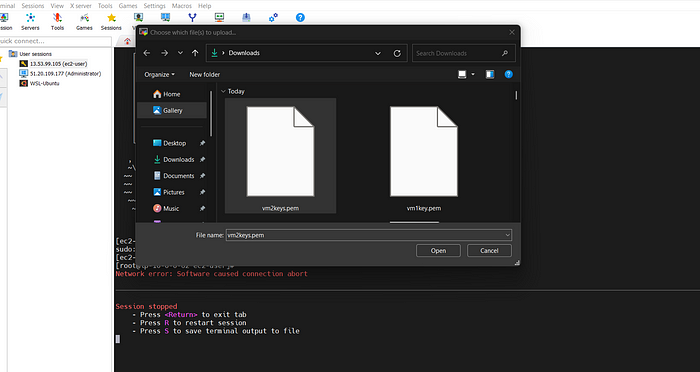
Once connected, we can **manage the public EC2 instance** directly from MobaXterm. Later, we will use this public instance to securely connect to the **private instance** inside the VPC.



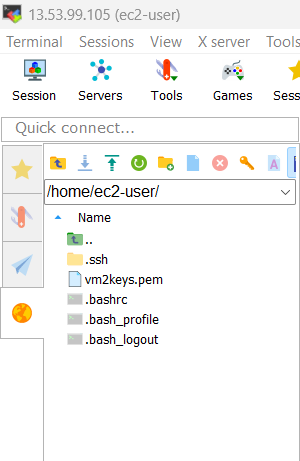




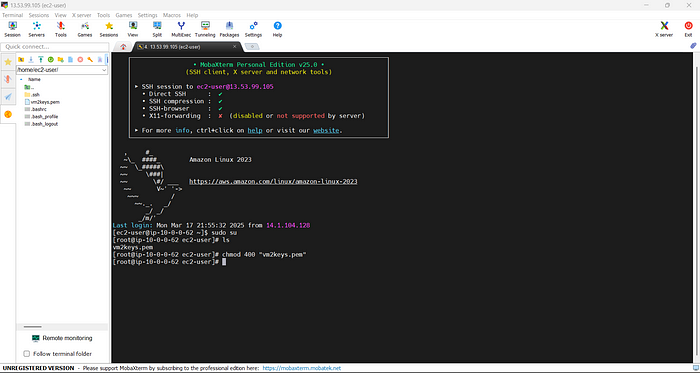
Now to jump to the instance in our private subnet we have to use the key pair that we created for private instance we have to follow these steps:-



Upload the private virtual machine .pem file here.

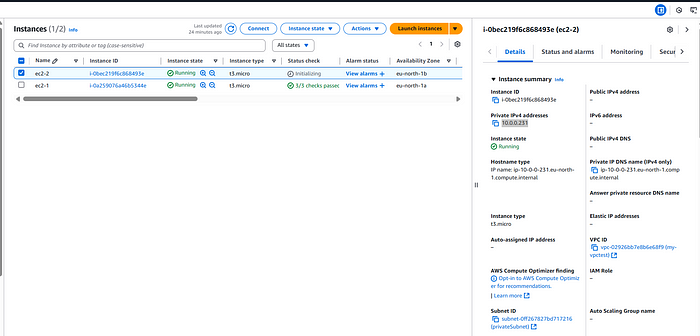


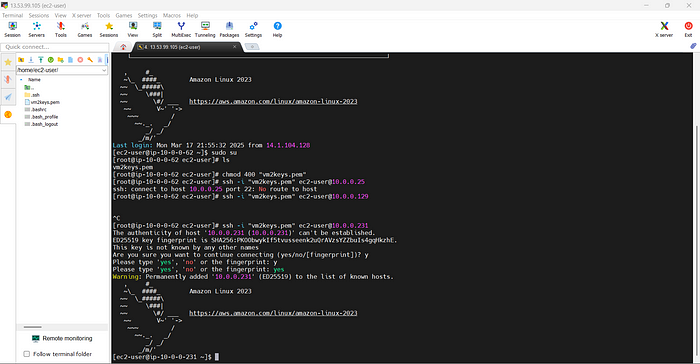
Giving permission using **chmod**command:-



This is important because the private virtual machine does not have the public ip address so.You must **set the correct permissions** using chmod 400, or SSH will reject the key.

Now we use our private network private address to access it(ssh).





Now, you are inside the Private EC2 Instance!

Finally now we can access private virtual machine inside the private subnet.