# Secure AWS VPC using Public and Private Subnets

In AWS, a **Virtual Private Cloud (VPC)** is a logically isolated section of the cloud where you can launch AWS resources in a virtual network you define. One of the most effective ways to **secure your infrastructure** within a VPC is by designing it with **public and private subnets**.

**What Are Public and Private Subnets?**

- **Public Subnet**: A subnet whose instances can communicate directly with the internet through an **Internet Gateway (IGW)**. Typically used for resources that need to be publicly accessible (e.g., Load Balancers, NAT Gateways, Bastion Hosts).
- **Private Subnet**: A subnet that does **not** have direct internet access. Instances in private subnets cannot initiate or receive traffic from the internet directly. These are ideal for **backend servers**, **databases**, **application servers**, etc.

**Key Components to Secure the VPC**

**1. Internet Gateway (IGW)**

- Attached to the VPC to allow traffic from public subnets to access the internet.
- Only the public subnets have route tables pointing to the IGW.

**2. NAT Gateway**

- Deployed in a public subnet, it allows instances in private subnets to access the internet **for outbound connections only** (e.g., software updates) while keeping them **inaccessible from the outside**.

**3. Route Tables**

- Public Subnet Route Table: Contains a route to the internet via the IGW.
- Private Subnet Route Table: Contains a route to the NAT Gateway for outbound internet access but **no route to the IGW**.

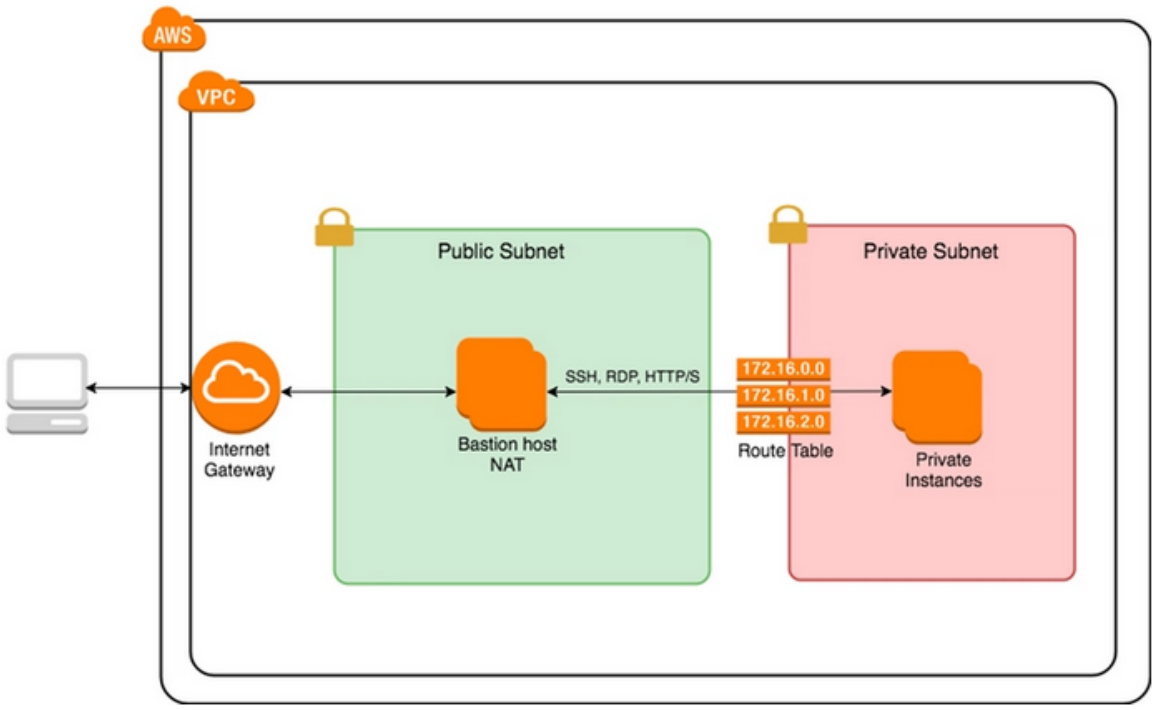
**4. Security Groups and Network ACLs**

- **Security Groups**: Acts as a virtual firewall for EC2 instances. Strictly define inbound and outbound rules.

- **NACLs** (Network Access Control Lists): Stateless filters for controlling traffic at the subnet level. Used for additional layers of security.

**5. Bastion Host (Jump Box)**

- A secure EC2 instance placed in the public subnet.
- Used to SSH/RDP into private instances using key-pairs, reducing exposure to the public internet.




***Prerequisite —***

- AWS Account with Admin Access.
- Knowledge about basic networking concepts (such as IP Addressing, CIDR notation, and routing), an understanding with basic cloud operations.
- Familiarity with navigating the AWS Management Console.
- **AWS LEVEL — BEGINNER — AWS 100**

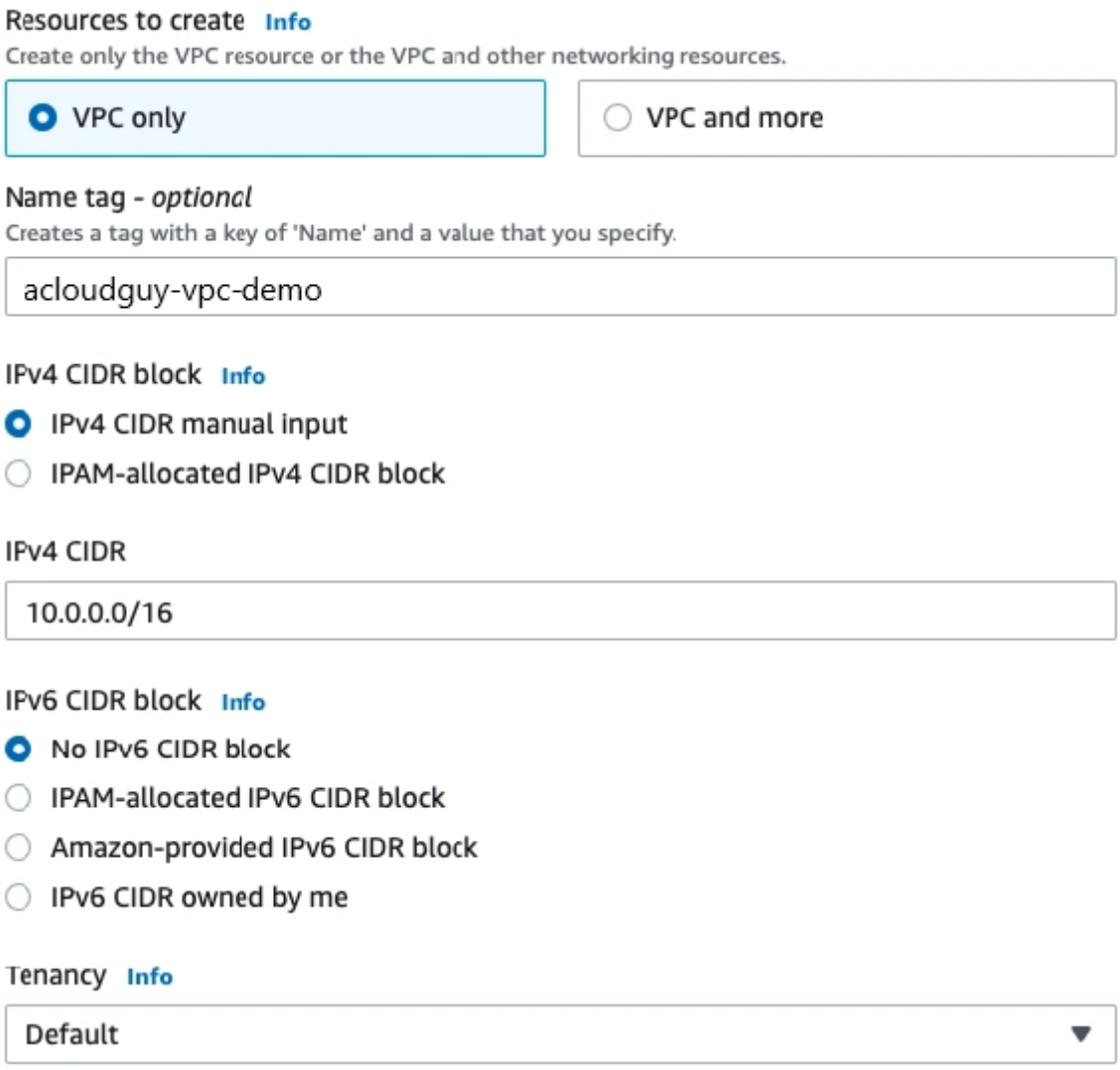
***AWS Services Usage —***

- AWS VPC, EC2, SSM, IGW, NGW, Route Table, SG, NACL and IAM

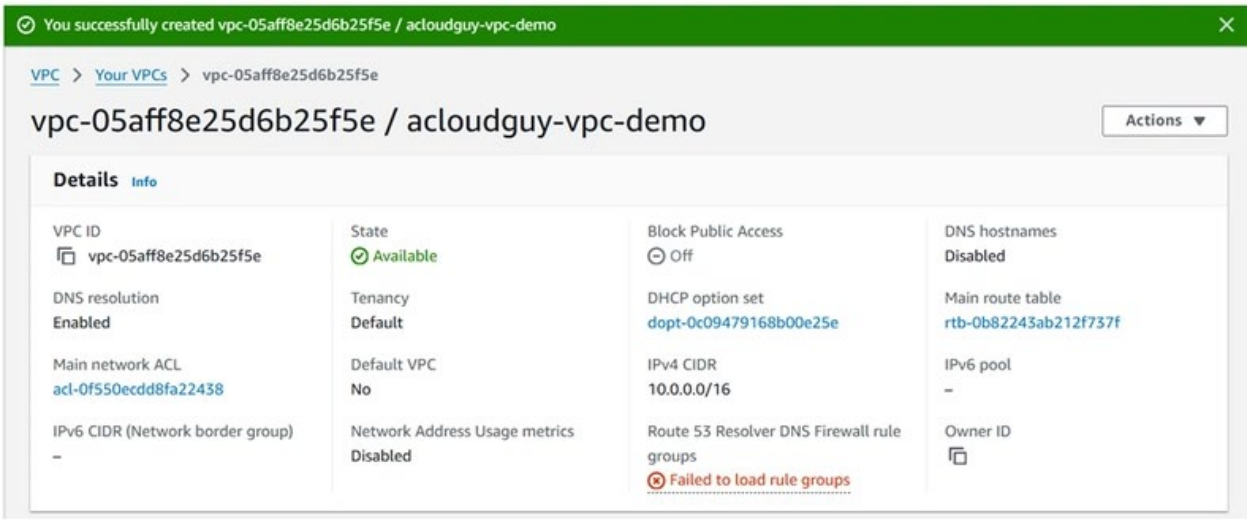
***STEP BY STEP GUIDE -***

**STEP 1 : Create VPC**

- Login to AWS Console with an IAM user having Admin privileges
- Select **us-west-2** region.
- In the AWS Management Console search bar, enter ***VPC***, and click the **VPC** result under **Services.**
- Click **Your VPCs** in the left navigation pane.
- Click **Create VPC** to begin creating a new VPC.
- Specify the following VPC details:
- **Resources to create**: Select **VPC only**
- **Name tag**: Enter *acloudguy-vpc-demo*
- **CIDR block**: Enter *10.0.0.0/16* (This is a CIDR block from the private (non-publicly routable) IP address ranges as specified in RFC 1918.)
- **Tenancy**: Select **Default** (Dedicated tenancy ensures your instances run on single-tenant hardware. For the purposes of this Lab, the default is fine though.)



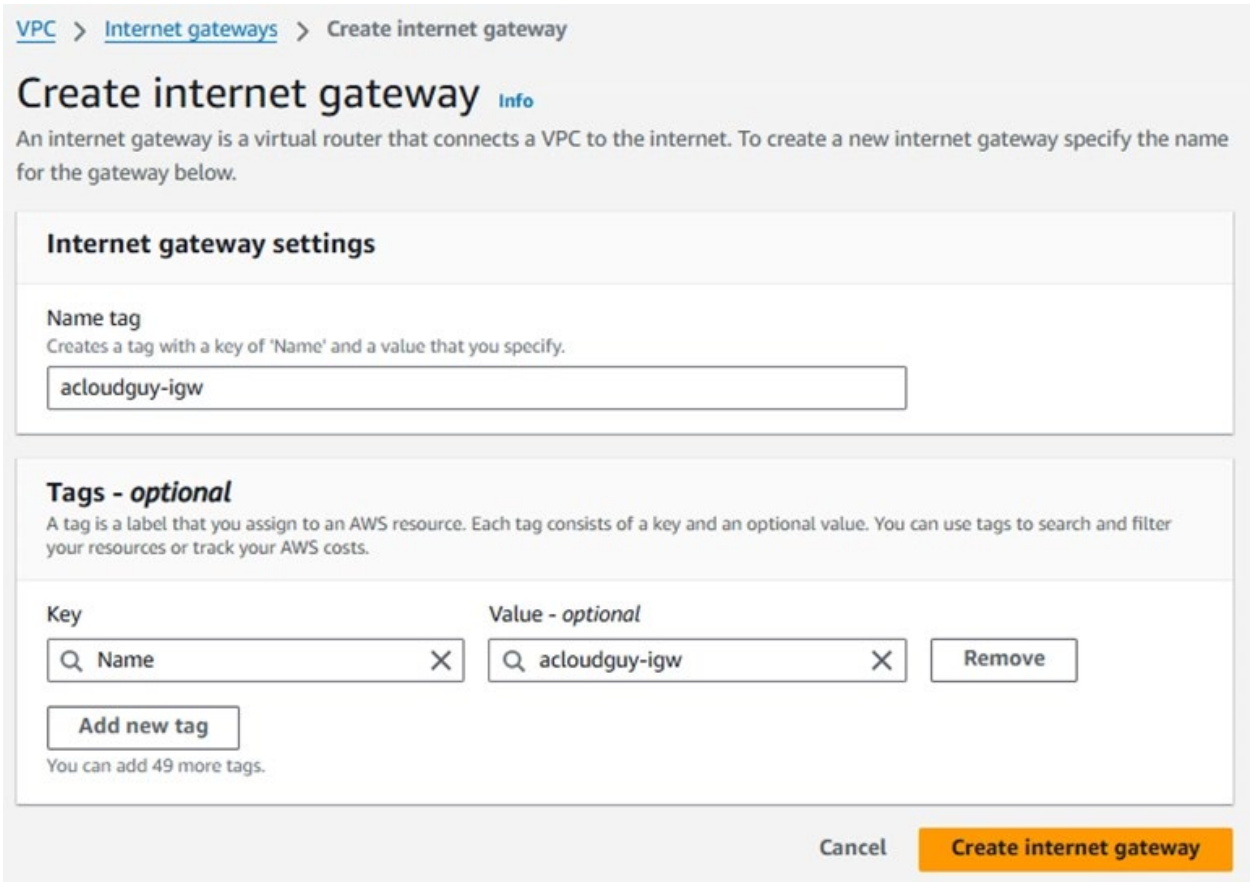

- Notice the **VPC and more.** option. Choosing this option launches a wizard that makes setting up and configuring a new VPC very simple.
- For learning the core concepts we are taking VPC only approach.
- Scroll to the bottom of the page and click **Create VPC**.



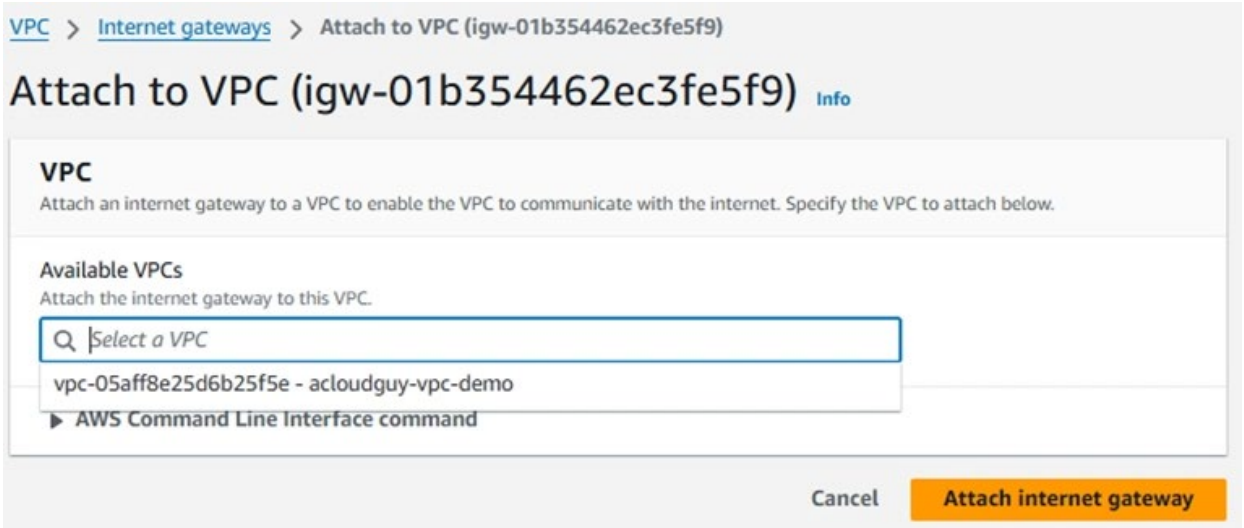

Amazon creates the requested VPC and the following linked services:
**DHCP options set**: Enables DNS for instances that need to communicate over the VPC's Internet gateway
**Main route table**: Table that contains a set of rules, called routes, that are used to determine where network traffic is directed
**Network ACL**: List of rules to determine whether traffic is allowed in or out of any subnet associated with the network ACL

**STEP 2 : Create IGW**

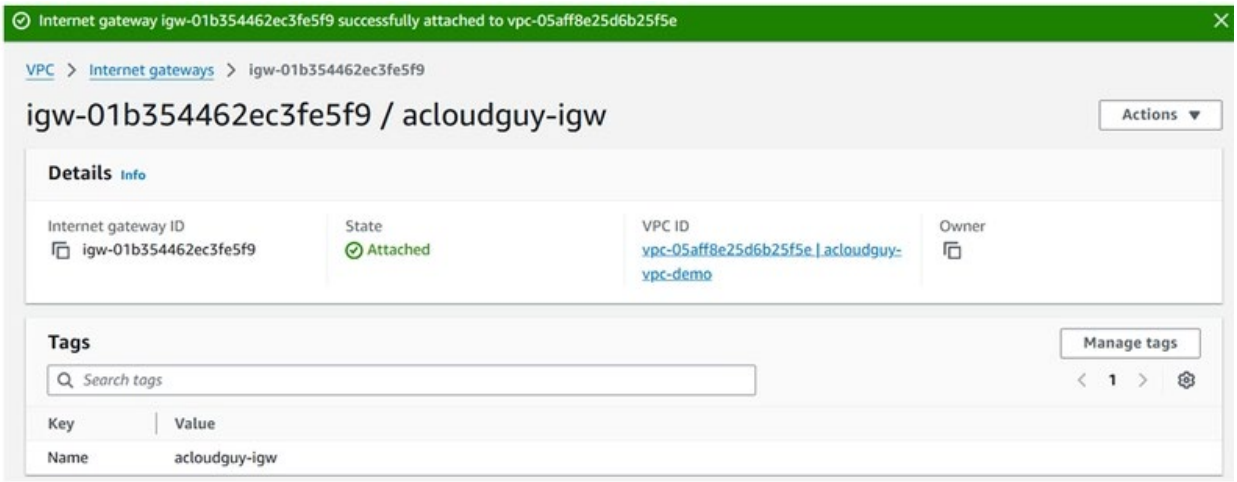
- An Internet Gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the Internet.
- It imposes no availability risks or bandwidth constraints on your network traffic.
- From the **VPC Dashboard**, click **Internet Gateways** in the left navigation pane.
- Click **Create internet gateway** to begin creating a new gateway with the following -
- **Name tag —** acloudguy-igw
- Click **Create Internet Gateway**




- The **State** of your Internet Gateway will be **detached** to start.
- Now you need to attach the new gateway to the VPC you created earlier.
- Click **Actions** then **Attach to VPC**
- Click **Attach internet gateway**

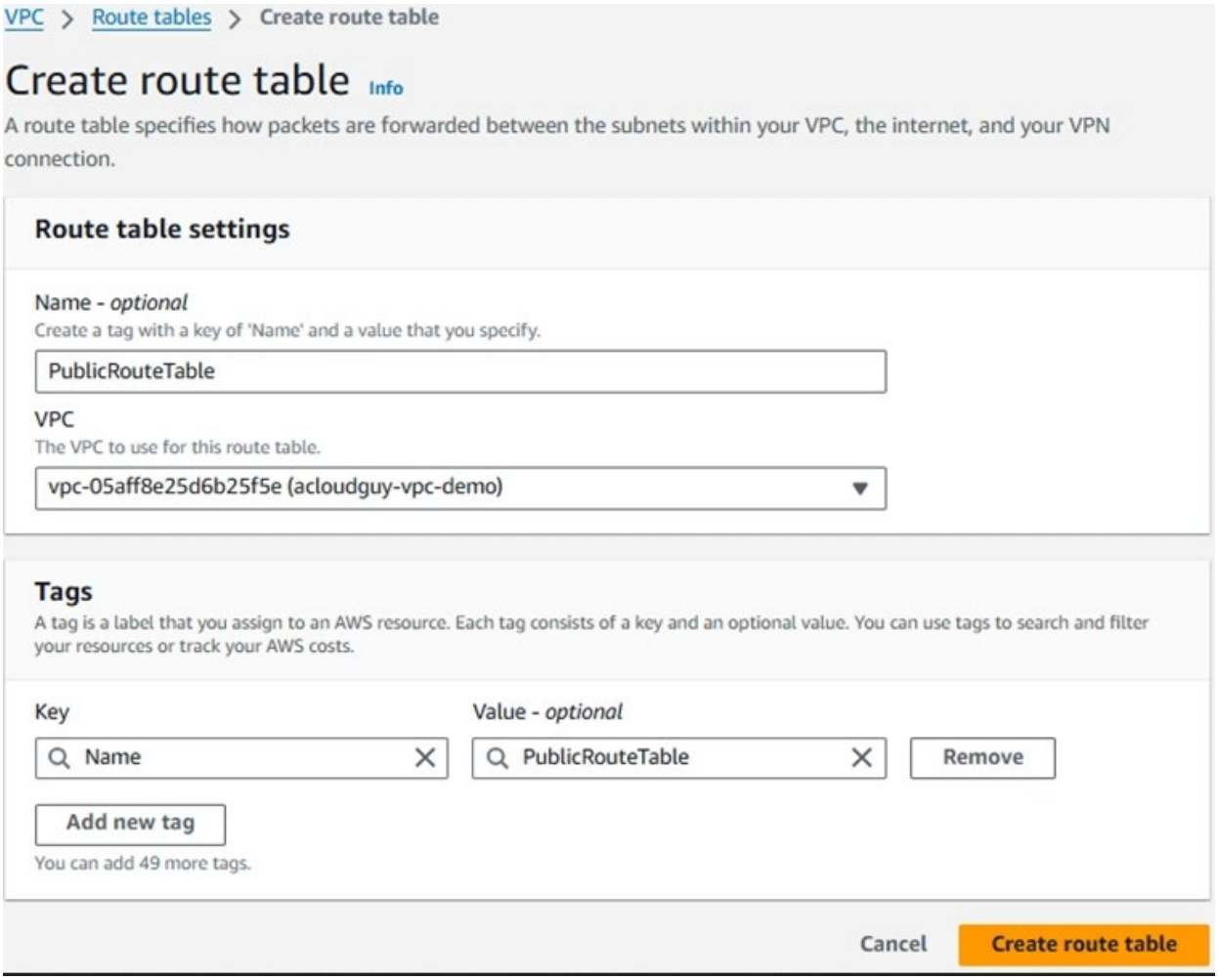


**IMPORTANT** An Internet Gateway can only be attached to one VPC.

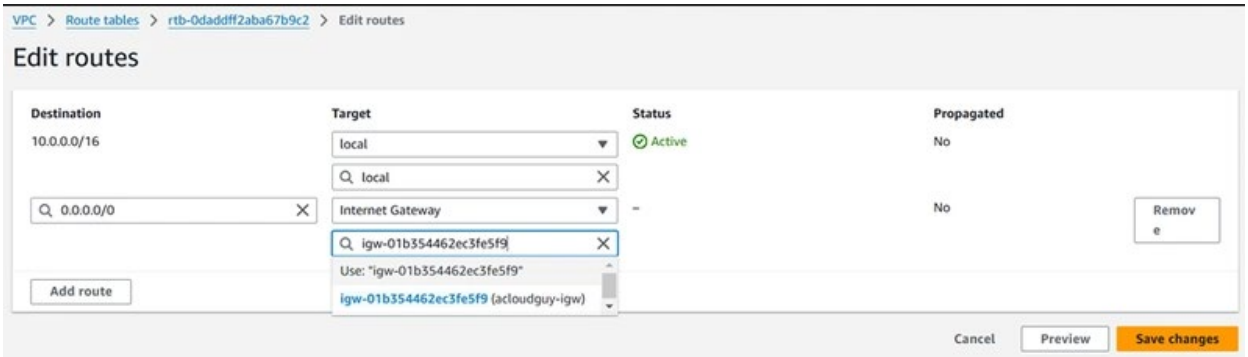



**STEP 3 : Create Public Subnet**

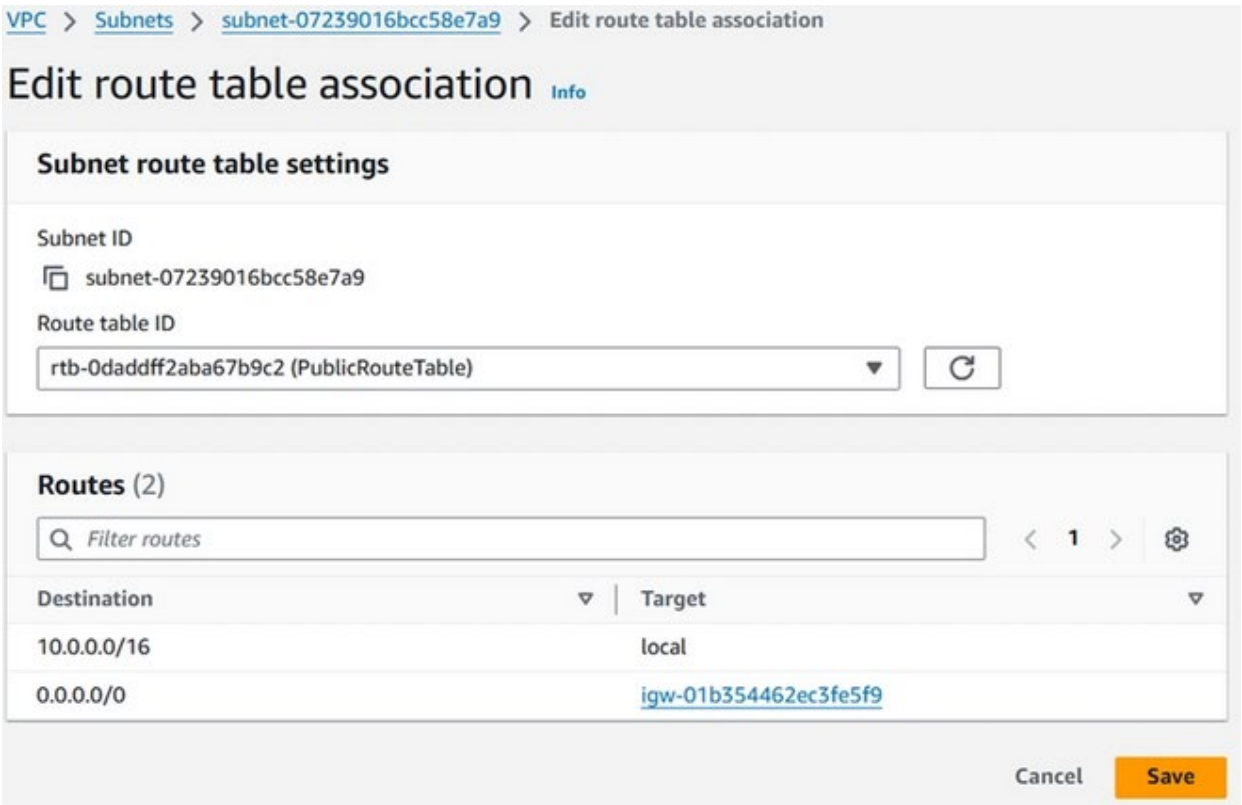
- In the **VPC Dashboard**, click **Subnets,** Click **Create subnet.**
- Configure the following **Public subnet** details:
- **VPC ID**: Select **acloudguy-vpc-demo**
- **Subnet name**: Enter *Public-A*
- **Availability Zone**: Select **us-west-2a** from the drop-down menu
- **IPv4 subnet CIDR block**: Enter *10.0.20.0/24*
- Click **Create subnet.**
- In the left navigation pane, click **Route Tables,** Click **Create route table.**
- Configure the following route table settings:
- **Name**: Enter *PublicRouteTable*
- **VPC**: Select the **acloudguy-vpc-demo** VPC from the drop-down menu.




- Scroll to the bottom of the page and click **Create route table.**
- On the route details page, switch to the **Routes** tab and click **Edit routes.**
- Click **Add route** and Configure the following route settings.
- **Destination**: Enter *0.0.0.0/0.*
- **Target**: Select **Internet Gateway**, then acloudguy-igw.
- Click **Save changes.**




- Select the **Public-A** subnet and click the **Route table** tab.
- Click the **Edit route table association** button
- Select **PublicRouteTable** from the **Route table ID** drop-down menu and confirm the following routes
- Click **Save**




- This **Public subnet** will require a route to the internet, so the associated route table has now been configured to use **PublicRouteTable** to determine traffic rules.

**STEP 3 : Create NAT Gateway**

- In the **VPC Dashboard**, click NAT Gateways.
- Click **Create NAT gateway.**
- **Name**: Enter *NAT-GW*
- **Subnet**: Select **Public-A**
- **Connectivity type**: Ensure **Public** is selected
- The **Public** connectivity type will allow this NAT Gateway the ability to access the public internet.
- Click **Allocate Elastic IP** next to the **Elastic IP allocation ID**
- Click **Create NAT gateway.**

**STEP 4 : Create Private Subnet**

- In the **VPC Dashboard**, click **Subnets, **Click **Create subnet.**
- Configure the following **Private subnet** details:
- **VPC ID**: Select **acloudguy-vpc-demo**
- **Subnet name**: Enter *Private-A*
- **Availability Zone**: Select **us-west-2a** from the drop-down menu
- **IPv4 subnet CIDR block**: Enter *10.0.10.0/24*
- Click **Create subnet.**
- In the left navigation pane, click **Route Tables,** Click **Create route table.**
- Configure the following route table settings:
- **Name**: Enter *PrivateRouteTable*
- **VPC**: Select the **acloudguy-vpc-demo** VPC from the drop-down menu.
- Click **Create route table.**
- In the **PrivateRouteTable** details page, in the **Routes** tab, click **Edit routes**
- Click **Add route** and configure the following route settings:
- **Destination**: Enter *0.0.0.0/0*
- **Target**: Select **IGW for temporary testing.**

- This is to make you understand why Instance is not able to reach Internet even if IGW is attached.
- Click **Save changes.**
- Click **Subnets** from the left navigation pane, then select the **Private-A** subnet.
- In the **Route Table** tab, and click **Edit route table association**
- Select **PrivateRouteTable** from the **Route table ID** drop-down menu.
- Click **Save.**

**STEP 5 : Create a Network ACL for a Private Subnet**

- A **Network Access Control List (NACL) **is an optional layer of security that acts as a firewall for controlling traffic in and out of a subnet.
- Click **Network ACLs **under **Security.**
- Click **Create Network ACL**
- Configure the following Network ACL settings:
- **Name**: Enter *Private-NACL*
- **VPC**: Select **acloudguy-vpc-demo **from the drop-down menu
- Click **Create network ACL**
- Select **Private-NACL** from the Network ACLs list and click the **Subnet associations** tab
- Click **Edit subnet associations**: Select the check box for the **Private-A** subnet to associate it with the network ACL.
- Click **Save changes**

**STEP 6 : Add rules to a Private Network ACL.**

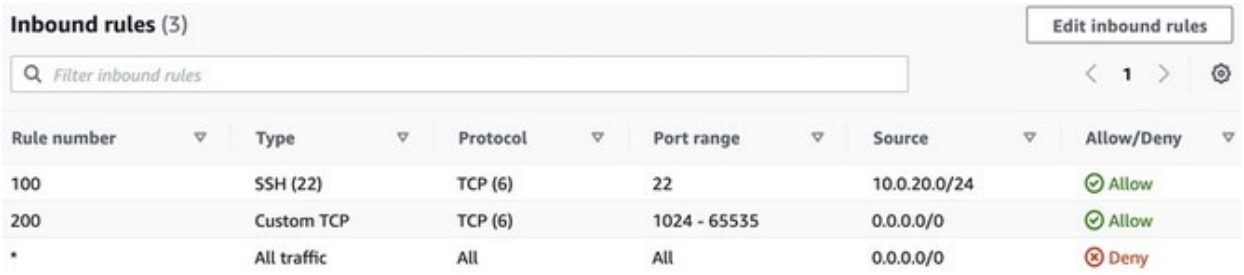
- Select **Private-NACL** from the list of Network ACLs
- Click the **Inbound rules** tab below the table and click **Edit inbound rules**
- Click **Add new rule** and configure the following:

**Rule number**: Enter *100*
**Type**: Select **SSH**
**Source**: Enter *10.0.20.0/24*
**Allow / Deny**: Select **Allow** from the drop-down menu

- For the second rule, click **Add new rule** and configure the following:

**Rule number**: Enter *200*
**Type**: Select **Custom TCP Rule**
**Port Range**: Enter *1024–65535*
**Source**: Enter *0.0.0.0/0*
**Allow / Deny**: Select **Allow** from the drop-down menu

- This will allow return traffic for the outbound rules you will add shortly (the range is specified as *1024–65535* because these are the available ports and not reserved). This enables resources inside the subnet to receive responses to their outbound traffic.
- Click **Save changes.**
- Ensure the **Private-NACL** is still selected then click the **Inbound rules** tab below the table to verify your inbound rules match the following.

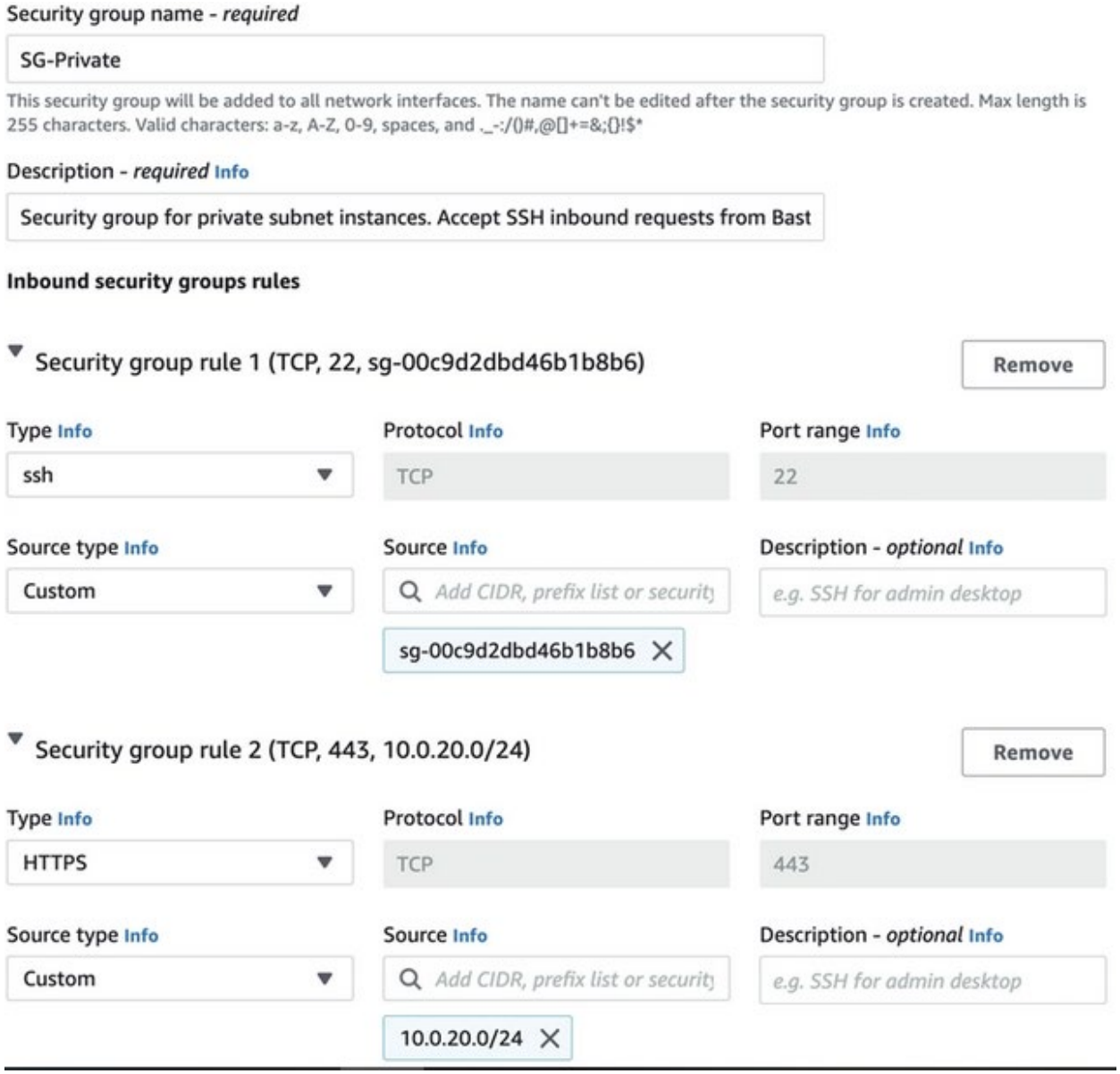



- **IMP : **When you add or remove rules from a network ACL, the changes are automatically applied to the subnets it is associated with. NACLs may take longer to propagate, as opposed to security groups, which take effect almost immediately.

**STEP 7 : Launching EC2 Instance on a Private Subnet.**

- In the AWS Management Console search bar, enter *EC2*, and click the **EC2** result under **Services.**
- Create a Key pair from EC2 left pane.
- Click **Launch instances.**
- In the **Name and tags** section, enter *private* under **Name.**
- In the **Instance Type** section, you should not change any options. Simply make sure the default **t2.micro **is selected.
- Select the Key Pair created earlier from drop down.
- In the **Network settings** section, click **Edit**, and configure the following instance details:
- **VPC**: Select the **acloudguy-vpc-demo** VPC
- **Subnet**: Select the **Private-A **subnet
- **Auto-assign Public IP**: Make sure this is **disabled**
- **Firewall**: Select **Create security group**
- **Security group name**: Enter *SG-Private*
- **Description**: Enter *Security group for private subnet instances. Accept SSH inbound requests from Bastion host only*.
- **Type**: SSH
- **Protocol**: TCP
- **Port**: 22
- **Source type**: Custom
- **Source**: SG-bastion
- *Tip: *If you don't recall the name of your bastion host's security group, leave the **Source** as* **Custom**, and start typing "*bastion*". It will find the security group for you. (Example: SG-bastion)
- Click **Add security group rule**
- **Type**: HTTPS
- **Protocol**: TCP
- **Port**: 443
- **Source type**: Custom

- **Source**: *10.0.20.0/24 (Public VPC CIDR)*
- *Note*: If you also needed Windows access, you would add another rule: Type RDP; Protocol TCP; Port 3389; Source *SG-bastion*




- Review the **Summary** section and click **Launch instance**

**STEP 8 : Test Internet access from EC2 Instance on a Private Subnet.**

- Connect to EC2 instance & hit sudo yum update

- Although the private instance security group is configured correctly, and you should have outbound access to the internet, it still timed out.
- The time out is caused by the private NACL denying inbound HTTP traffic.
- You will need Network Address Translation (NAT) to allow your private instance *outgoing *connectivity to the Internet.

**STEP 9 : Change Routes for Private Subnet from IGW to NGW.**

- In the **PrivateRouteTable** details page, in the **Routes** tab, click **Edit routes**
- Click **Add route** and configure the following route settings:
- **Destination**: Enter *0.0.0.0/0*
- **Target**: Remove IGW & ADD NGW created in STEP 3*.*
- This route will eventually send traffic originating from your private subnet and bound for the public internet, to a NAT device.
- Click **Save changes.**
- Click **Subnets** from the left navigation pane, then select the **Private-A** subnet.
- In the **Route Table** tab, and click **Edit route table association**
- Select **PrivateRouteTable** from the **Route table ID **drop-down menu.
- Click **Save.**

**STEP 10 : Final Test Internet access from EC2 Instance on a Private Subnet.**

- ***Important!***
- There are **two important configurations worth mentioning** again as to why this command should work in your lab environment:
- The private NACL has an Outbound Rule permitting HTTP (port 80) or HTTPS (port 443) access to anywhere on the internet (0.0.0.0/0)
- The security group for the NAT device allows HTTP/S access from any instance in the private subnet (that uses the private instance security group, which permits any destination as well)
- Connect to EC2 instance and Run **sudo yum update -y**
- SUCCESS !! It worked!


Running transaction test
Transaction test succeeded
Running transaction
Updating : yum-3.4.3-158.amzn2.0.2.noarch 1/6
Updating : yum-utils-1.1.31-46.amzn2.0.1.noarch 2/6
Updating : yum-plugin-priorities-1.1.31-46.amzn2.0.1.noarch 3/6
Cleanup : yum-plugin-priorities-1.1.31-45.amzn2.0.1.noarch 4/6
Cleanup : yum-utils-1.1.31-45.amzn2.0.1.noarch 5/6
Cleanup : yum-3.4.3-154.amzn2.0.1.noarch 6/6
Verifying : yum-3.4.3-158.amzn2.0.2.noarch 1/6
Verifying : yum-utils-1.1.31-46.amzn2.0.1.noarch 2/6
Verifying : yum-plugin-priorities-1.1.31-46.amzn2.0.1.noarch 3/6
Verifying : yum-plugin-priorities-1.1.31-45.amzn2.0.1.noarch 4/6
Verifying : yum-3.4.3-154.amzn2.0.1.noarch 5/6
Verifying : yum-utils-1.1.31-45.amzn2.0.1.noarch 6/6
Updated:
yum.noarch 0:3.4.3-158.amzn2.0.2 yum-plugin-priorities.noarch 0:1.1.31-46.amzn2.0.1
yum-utils.noarch 0:1.1.31-46.amzn2.0.1
Complete!