Task 1 (5%): In your opinion is it a good practice to create multiple users in AWS? Justify your answer and give examples.

Yes, creating multiple users in AWS is a good practice. This satisfies the principle of least privilege, with multiple users, each user can be assigned the minimum permissions required for their specific role. This reduces the risk of accidental or malicious actions by limiting access only to necessary resources. A developer working on a specific application doesn't need access to all the databases in AWS, only to the ones relevant to their work.

Task 2 (5%): What could be a use case for an IAM *role*?

Imagine if you want to let someone upload files to your S3 bucket without giving them full access to your AWS account. You can create an **IAM role** just for this purpose.

Task 3 (5%): What is the difference between an IAM *role* and an IAM *policy*?

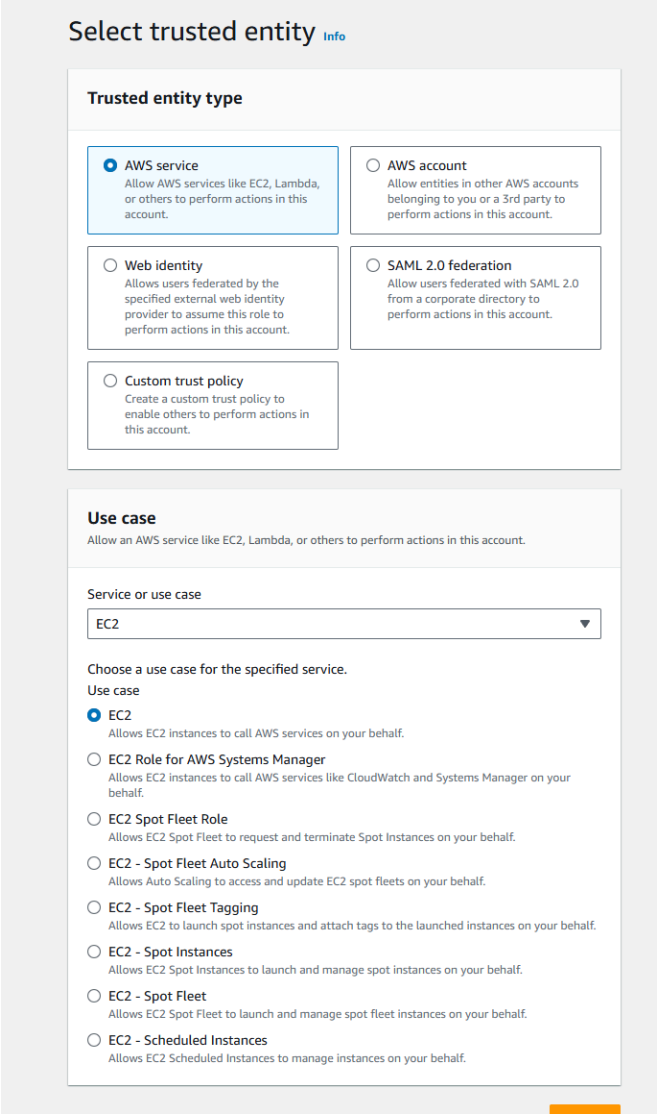
IAM Role: It is used to **grant temporary permissions** to entities (such as users, applications, or AWS services) that need access to AWS resources but don't have long-term credentials.
IAM Policy: A document that defines permissions. Policies specify which actions are allowed or denied on specific resources.

Task 4 (10%): What is needed in order for the EC2 instance to be able to access the newly created DynamoDB table? Please consider following the best practices.

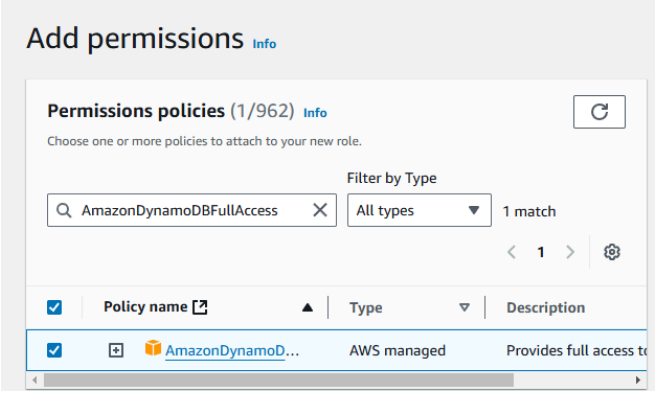
Wes should use an **IAM role** with the necessary permissions. The role should be created with a policy that grants access to the specific DynamoDB table. Once the role is created, attach it to the EC2 instance.

Task 5 (5%): Report the steps you took in order to EC2 instance access the DynamoDB table.

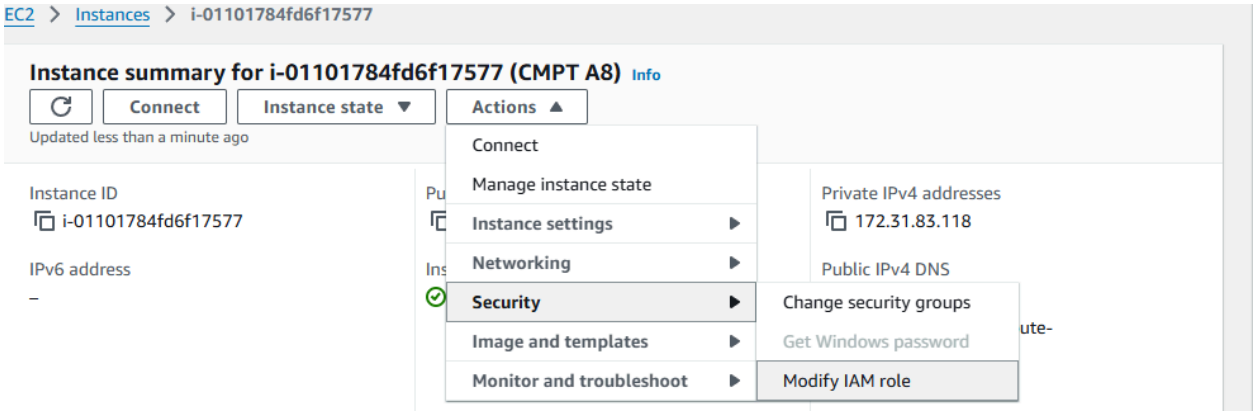
Go to IAM console > Click **Roles > Create Role** > Choose **AWS Service** and select **EC2**.

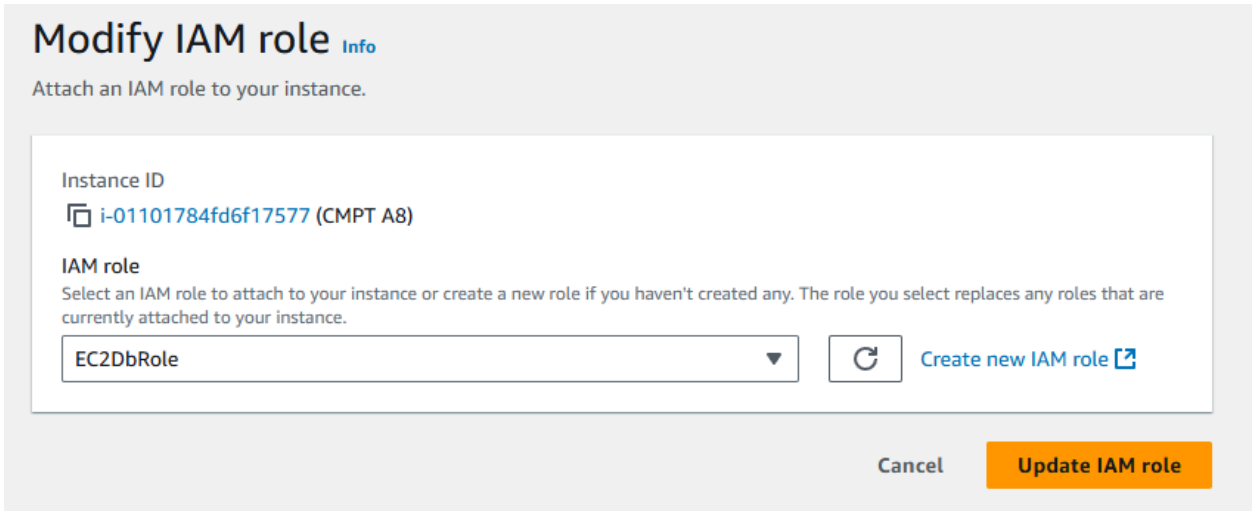


Attach the **AmazonDynamoDBFullAccess** policy



Go to the **EC2 Console >** Select the instance> Click **Actions > Security > Modify IAM Role**





Use the saved key pair to SSH into the instance:

```
S C:\WINDOWS\system32> ssh -i "C:\Users\danny\Downloads\A8 2.pem" ubuntu@54.167.66.72
he authenticity of host '54.167.66.72 (54.167.66.72)' can't be established.
D25519 key fingerprint is SHA256:hxOHJJadAzp5NLoW2qvzyBdG847/dq+A6LJz2cjGIhs.
his key is not known by any other names.
re you sure you want to continue connecting (yes/no/[fingerprint])? yes
arning: Permanently added '54.167.66.72' (ED25519) to the list of known hosts.
elcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/pro

System information as of Sat Nov  9 04:08:53 UTC 2024

 System load:  0.07              Processes:             103
 Usage of /:   23.0% of 6.71GB   Users logged in:       0
 Memory usage: 20%               IPv4 address for enX0: 172.31.83.118
 Swap usage:   0%

xpanded Security Maintenance for Applications is not enabled.

 updates can be applied immediately.

nable ESM Apps to receive additional future security updates.
```

Install AWS CLI

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

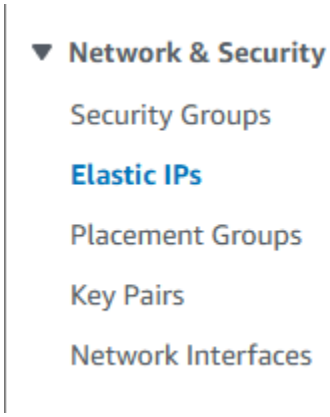
interact with the usersTable

```
ubuntu@ip-172-31-83-118:~$ aws dynamodb list-tables
{
    "TableNames": [
        "usersTable"
    ]
}
```

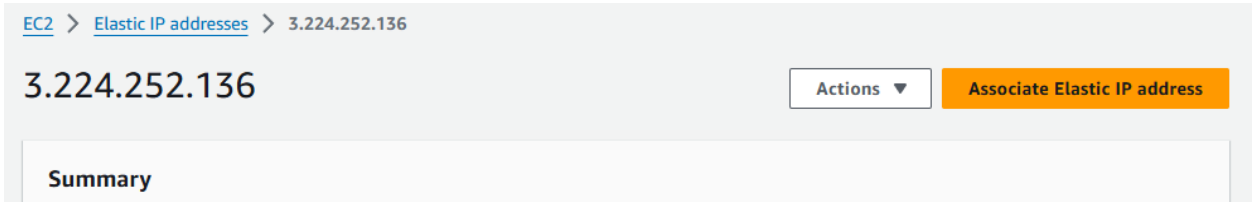
Task 6 (5%): If you go to EC2 > Instances and click on the Instance you have created, then you will notice that there is a plethora of information about your newly created machine. There is an IPv4 Public IP created for your EC2 instance. If you right click and Stop the machine and then Start it again, you will realize that the IP assigned to that machine is changed. Why is that? What would you do in order to give your machine an IP Address that persists through reboots.

AWS assigns a **dynamic public IP** by default. This IP is released when the instance is stopped and a new one is assigned upon restart.

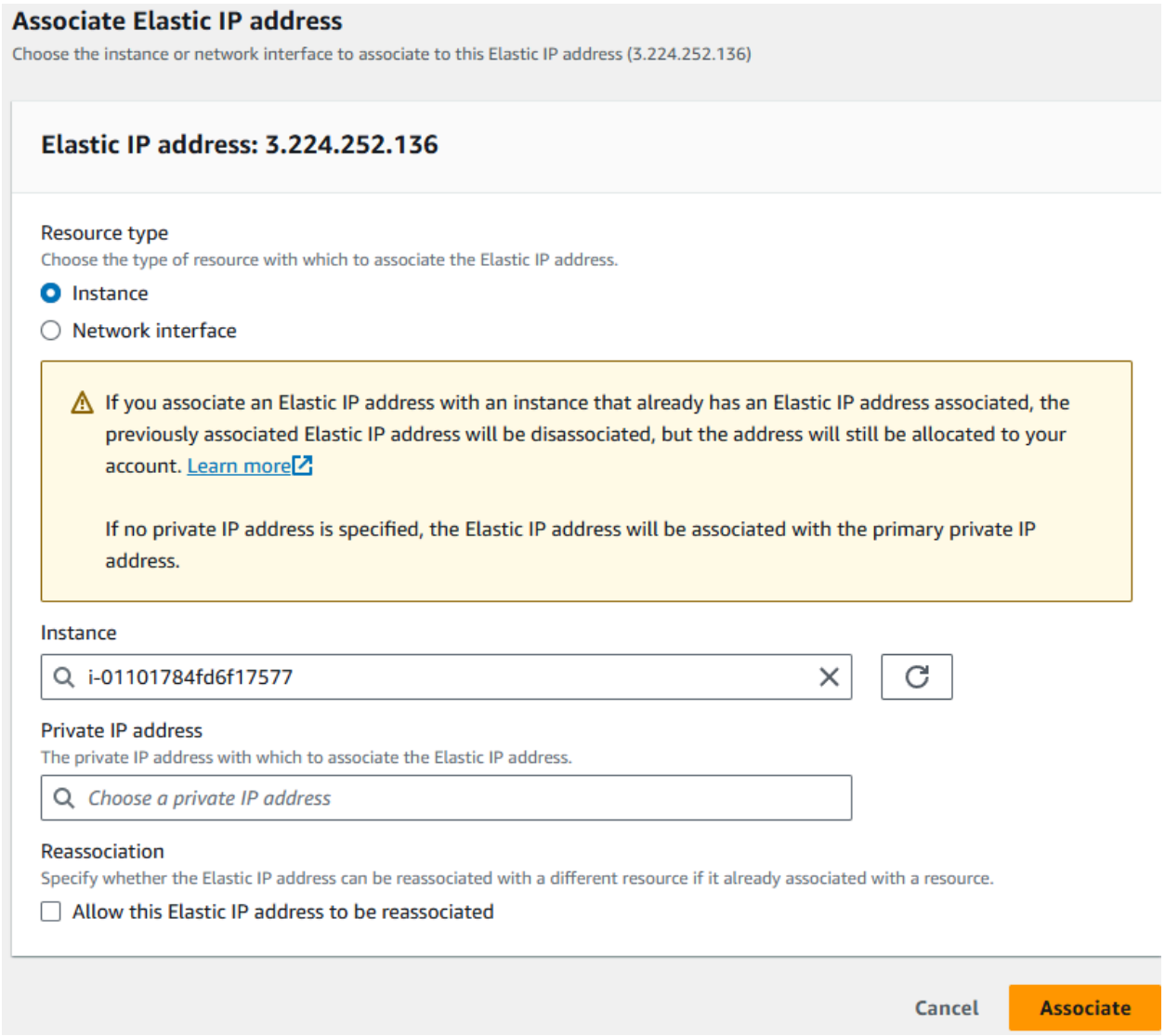
To ensure your EC2 instance retains the same public IP across reboots, we need to associate an **Elastic IP** with the instance.



After allocation, select the newly created Elastic IP. Click **Associate Elastic IP address**.



Choose the instance

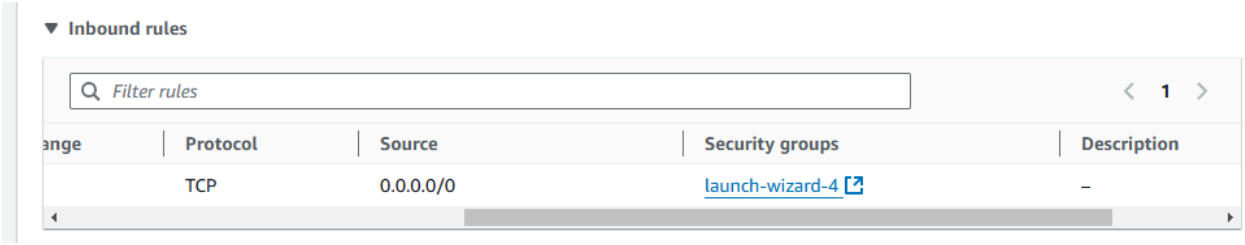


Now, even if we stop and start the instance, the assigned Elastic IP will persist.

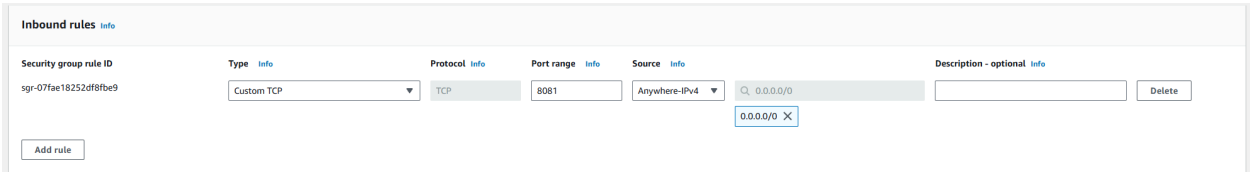
Task 7 (5%): If you try to setup a Web server listening to the port 8081 inside your instance you will soon realize that it is not accessible from the outside world. What is the AWS component responsible for allowing traffic to be sent to port 8081? What steps would you take in order to make it accessible from the outside world?

The AWS component responsible for controlling inbound and outbound traffic to the EC2 instance is the **Security Group**. Security Groups act as virtual firewalls, allowing or blocking traffic based on defined rules.

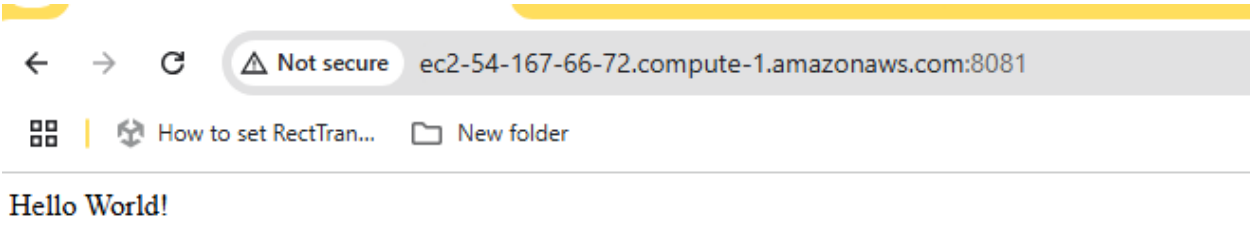
Go to the **EC2 Console**, select the instance.
In the **Description** tab, find the **Security Groups** section and click on the linked Security Group name.



Click on Inbound Rules and then Edit Inbound Rules.
Add a new rule



Once the rule is added, the web server running on port 8081 is accessible



Task 8 (5%): What is the range of the IPs in the VPC you just created?

Range of IPs:

- The first IP in this range is 10.0.0.0.
- The last IP in this range is 10.0.255.255.

Task 9 (5%): What is the difference between a VPC and a Virtual Private Network (VPN)?

A **VPC (Virtual Private Cloud)** allows us to create a virtual network within AWS where we can launch and manage resources like EC2 instances in a secure and isolated environment. It gives us control over networking configurations such as IP address ranges, subnets, and routing. In contrast, a **VPN (Virtual Private Network)** enables us to establish a secure, encrypted connection between our on-premises network or remote devices and another network.

Task 10 (5%): What are the IP ranges of the two subnets you created?

Public Subnet: CIDR Block: 10.0.1.0/24 IP Range: 10.0.1.0 to 10.0.1.255
Private Subnet: CIDR Block: 10.0.2.0/24 IP Range: 10.0.2.0 to 10.0.2.255

Task 11 (10%): Why would someone create a public and a private subnet. What are the uses of each of them? Provide an example.

Creating a public and a private subnet provides a secure and efficient architecture for deploying applications in the cloud. A public subnet is used for resources that need direct internet access, such as web servers, which are made accessible to users via an Internet Gateway. In contrast, a private subnet is designed for resources that should remain isolated from the internet, such as databases or backend services. For example, a web server in the public subnet can serve user requests while securely accessing a database in the private subnet.

Task 12 (5%): If we launch two instances, one in the public subnet and one in the private subnet, the one in the private subnet will not have internet access. How is it possible to connect to the instance in the private subnet through SSH?

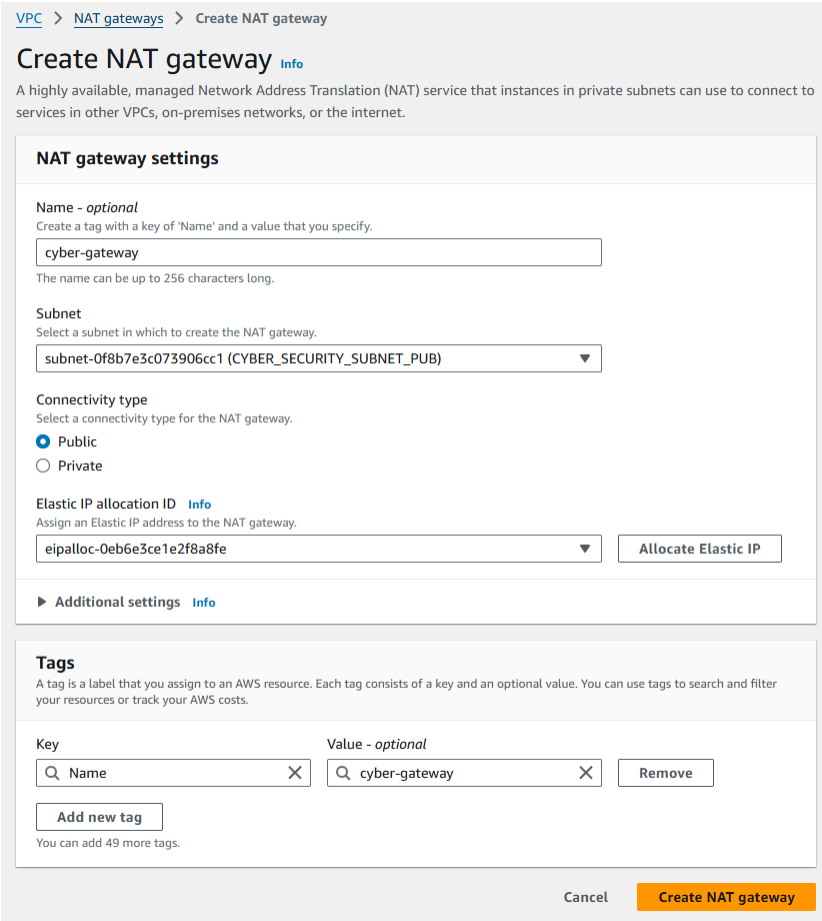
To connect to an instance in the **private subnet** through SSH, we can use a method called **jump server**, which involves first connecting to an instance in the **public subnet** and then using that instance to access the private one.

Task 13 (10%): We can give internet access to the private subnet by creating a NAT Gateway. What is the difference between the NAT Gateway and the Internet Gateway?

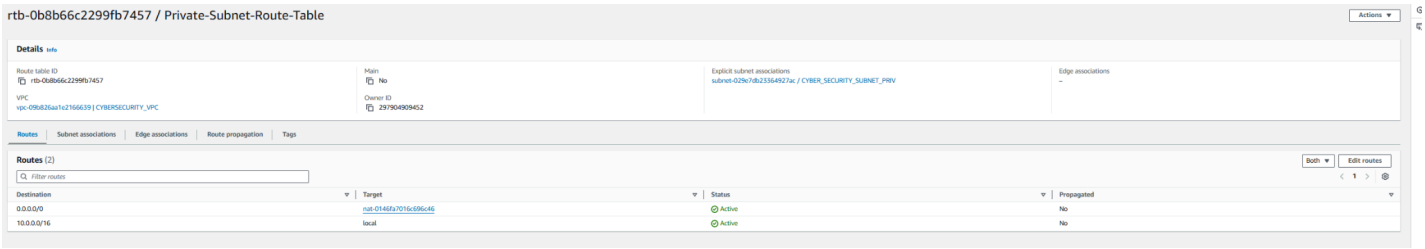
A NAT Gateway and an Internet Gateway serve different purposes in AWS networking. An Internet Gateway enables instances in a public subnet to have direct, two-way communication with the internet, allowing inbound and outbound traffic. It’s commonly used for resources like web servers that need to be accessible from the internet. In contrast, a NAT Gateway allows instances in a private subnet to initiate outbound connections to the internet (e.g., for software updates or accessing external services) while preventing any inbound traffic from the internet. This ensures that private subnet resources, such as databases or backend servers, remain isolated from direct internet access, enhancing security.

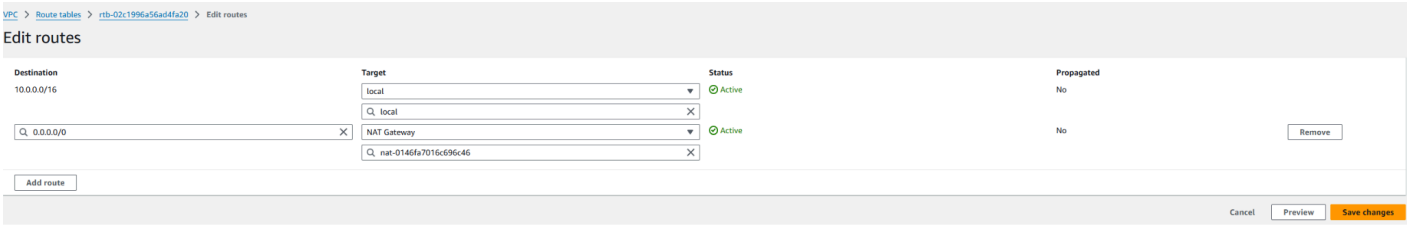
Task 14 (5%): What are the steps needed to be taken in order to create a NAT Gateway into the public subnet to provide the private subnet with internet access? You can try it by launching two instances and experimenting with the NAT Gateway.

Create a NAT Gateway
Go to the **VPC Console** > Select **NAT Gateways** > Click **Create NAT Gateway** > **Subnet**: Choose **public subnet** > Select the Elastic IP allocated (Note the Elastic IP was created in the previous task.)



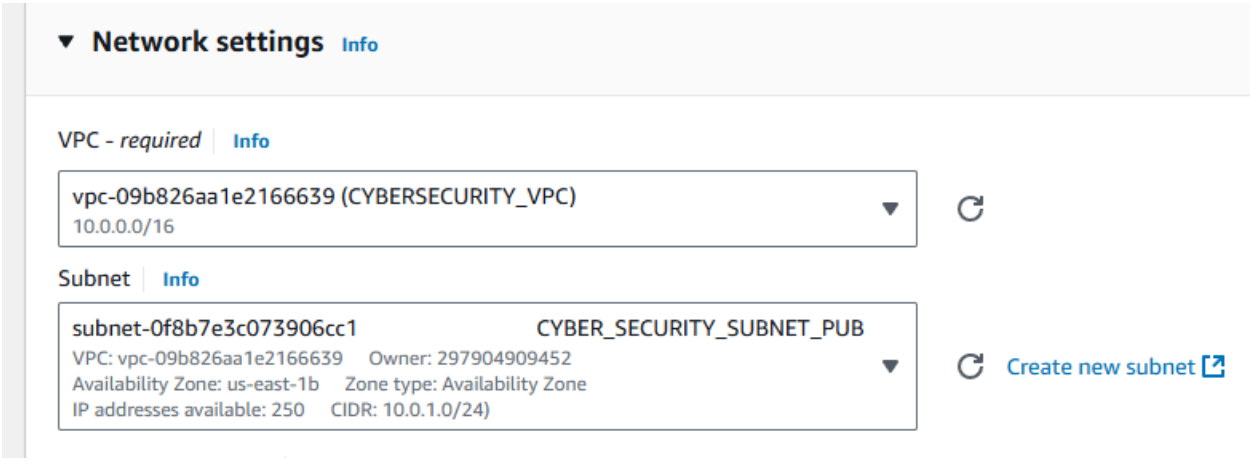
Create a Separate Route Table for the Private Subnet:
Click **Create Route Table** > VPC: Select the same VPC > Add a route for **0.0.0.0/0** pointing to the **NAT Gateway**:



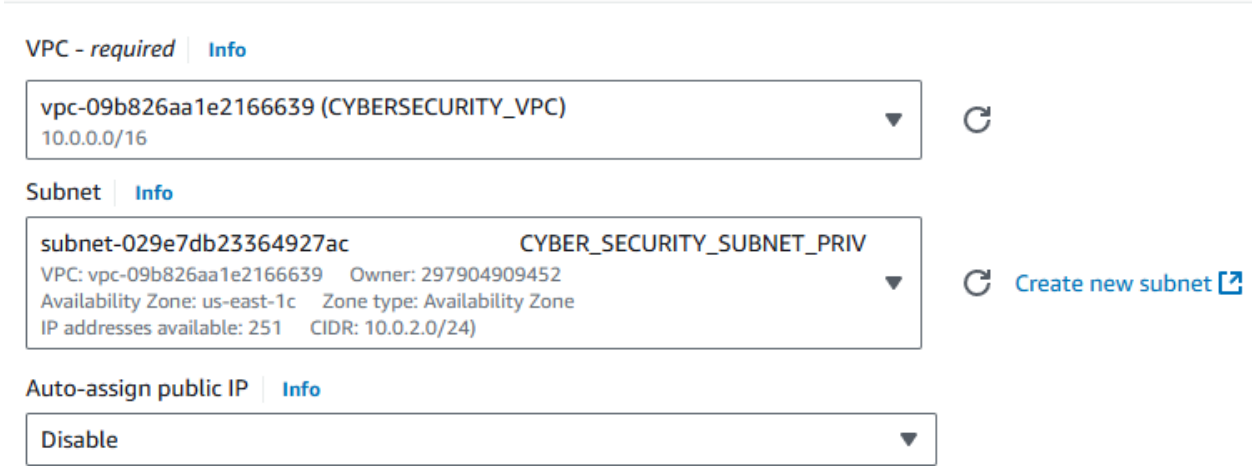


Launch Two Instances:

Public Subnet Instance: Launch an EC2 instance in the public subnet with a public IP.



Private Subnet Instance: Launch another EC2 instance in the private subnet without a public IP.

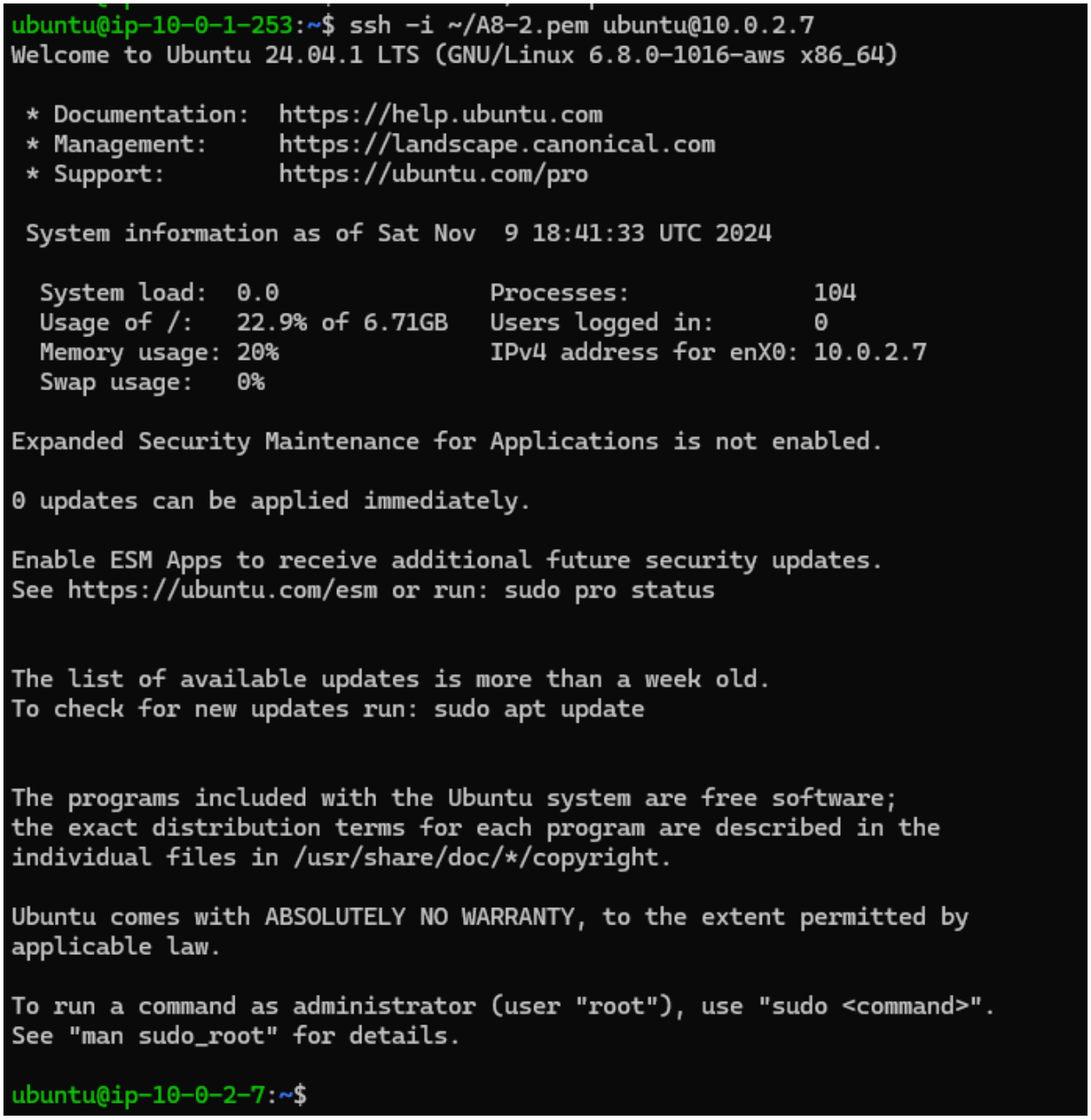


Connect to the Public Instance, then transfer the key to the public Instance

```
PS C:\Users\danny\downloads> scp -i "A8-2.pem" "A8-2.pem" ubuntu@98.81.36.191:/home/ubuntu
A8-2.pem                                                                    100% 1678    22.1KB/s   00:00
```

From the public instance, SSH into the private instance using its private IP.

By associating this NACL with the public subnet, we control traffic at the subnet level. The rules will be applied to all instances in the public subnet. Instances in the public subnet will still have internet access through the Internet Gateway, but now only traffic allowed by the NACL rules



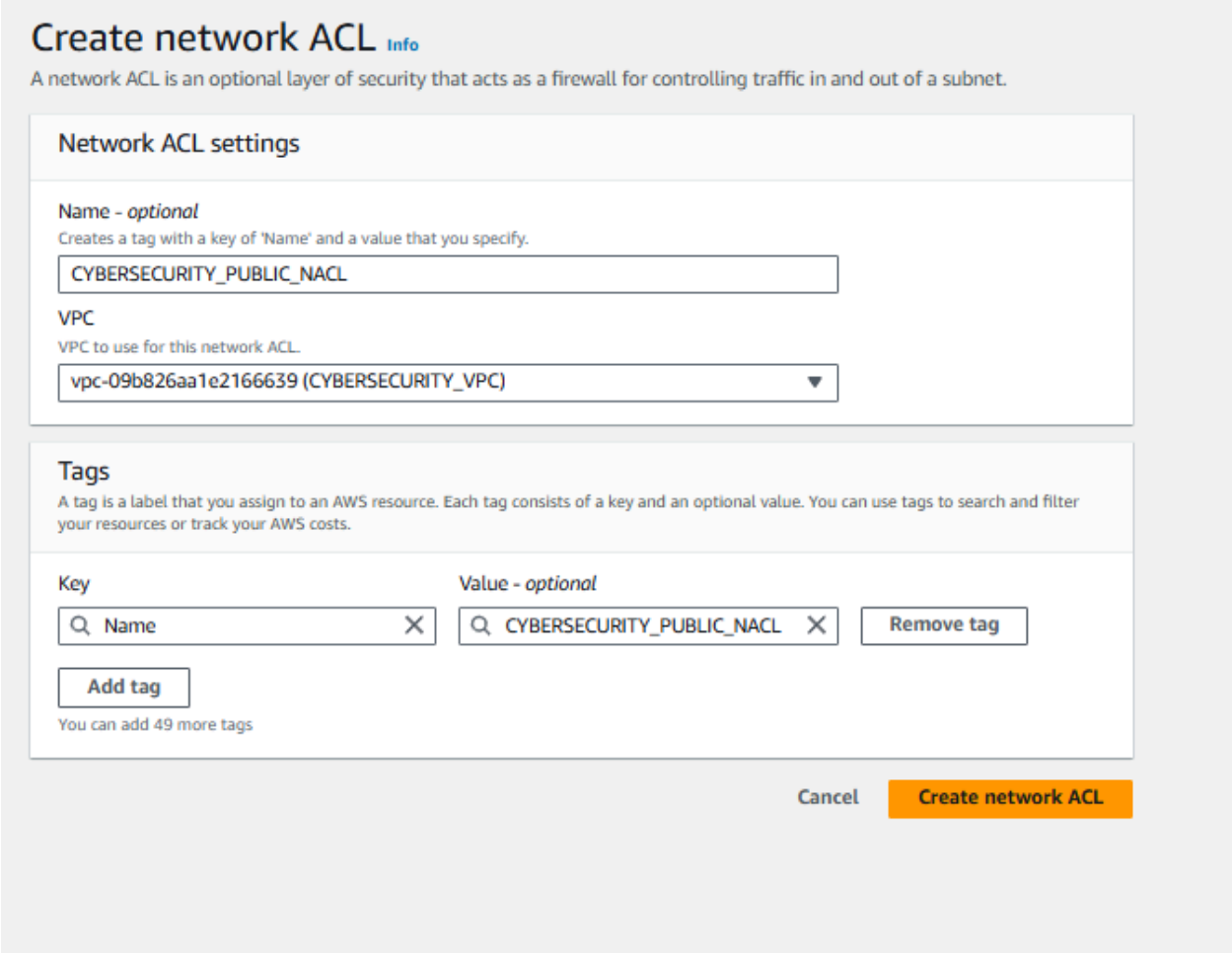
Task 15 (10%): In VPC under Security there is another module called Network ACL. What is the difference between Network ACL and Security Groups?

Security Groups and **Network ACLs (NACLs)** both control traffic in AWS, but they serve different purposes and operate at different levels. Security Groups are **stateful**, meaning they automatically allow return traffic for inbound or outbound connections, and they operate at the **instance level**, controlling access to individual instances. They only support **allow rules**. In contrast, Network ACLs are **stateless**, meaning both inbound and outbound traffic must be

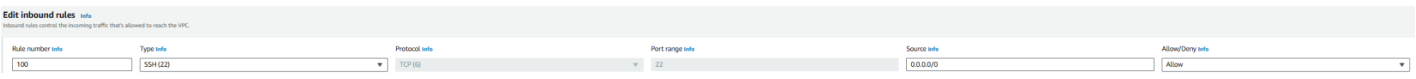
explicitly allowed, and they operate at the **subnet level**, affecting all instances within a subnet. NACLs support both **allow and deny rules**, providing more granular control.

Task 16 (5%): Report the steps required to create a Network ACL. How would you integrate it into the public subnet you previously created?

Create a **Network ACL (NACL)**

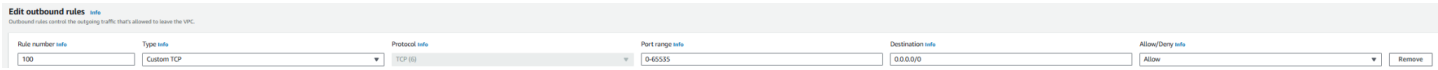


Add Inbound Rules:
Rule Number: 100 (lower numbers are evaluated first).
Type: SSH.
Protocol: TCP.
Port Range: 22.
Source: 0.0.0.0/0 (or your specific IP range).
Allow/Deny: Allow



Add Outbound Rules:
Rule Number: 100.

Type: All Traffic.
Protocol: All.
Port Range: 0-65535.
Destination: 0.0.0.0/0.
Allow/Deny: Allow



Associate the NACL with the Public Subnet