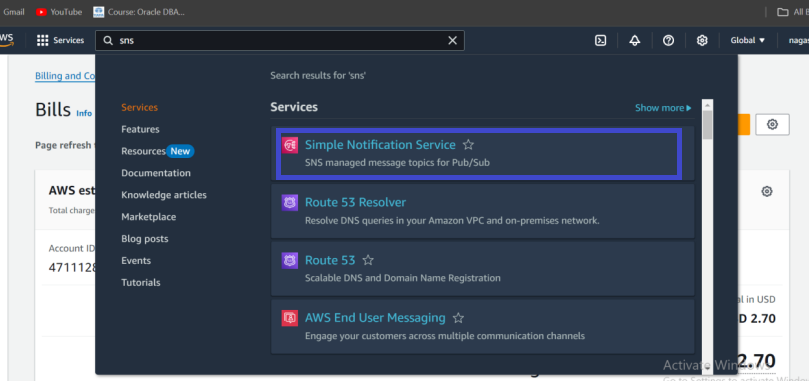
**SNS**

Amazon SNS (Simple Notification Service) is a fully managed messaging service provided by AWS that allows you to send messages or notifications to a variety of subscribers, including mobile devices, email, and other AWS services. Here are some key features and concepts related to Amazon SNS:

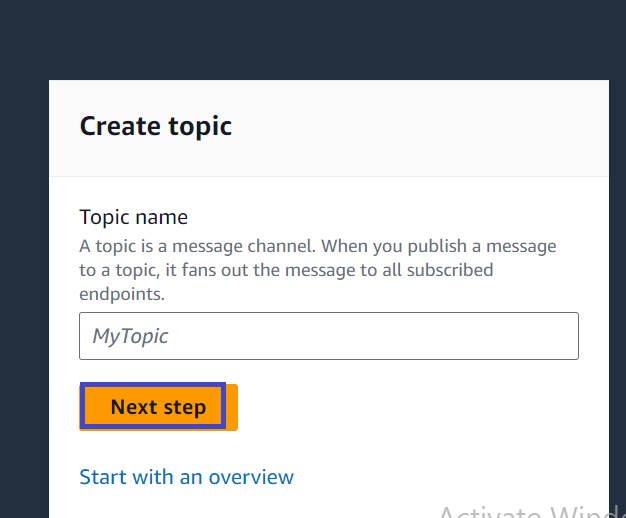
**Key Features**

1. **Pub/Sub Messaging**: SNS supports the publisher/subscriber (pub/sub) messaging model, allowing messages to be sent to multiple subscribers at once.
2. **Multiple Protocols**: You can send notifications via several protocols:
   * **Email/Email-JSON**: For email notifications.
   * **SMS**: For sending text messages to mobile phones.
   * **HTTP/HTTPS**: For sending messages to web servers.
   * **Amazon SQS**: For queuing messages.
   * **Application**: For sending messages to mobile apps (via push notifications).
3. **Topic Management**: You can create topics to which subscribers can subscribe. Publishers send messages to topics, and SNS takes care of delivering those messages to all subscribers.
4. **Message Filtering**: Allows subscribers to receive only specific messages based on filtering criteria.
5. **Delivery Status Logging**: You can monitor message delivery status and track issues through logs.
6. **Durable Storage**: SNS can store messages until they are successfully delivered to subscribers.

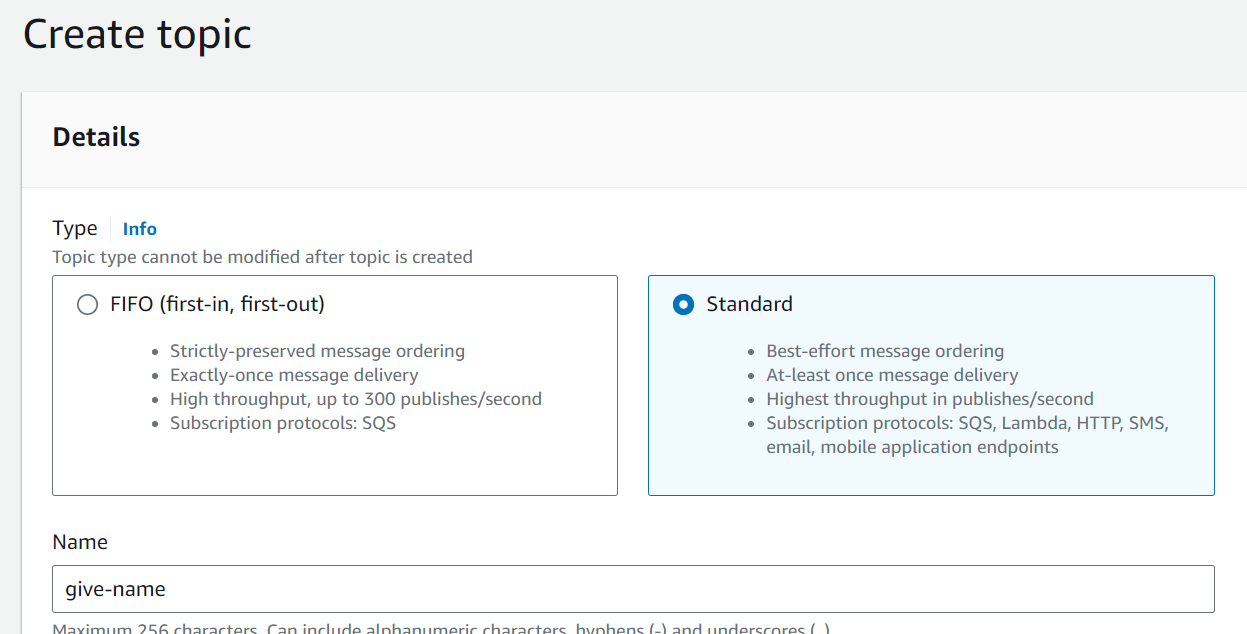
**Search and select SNS Service**

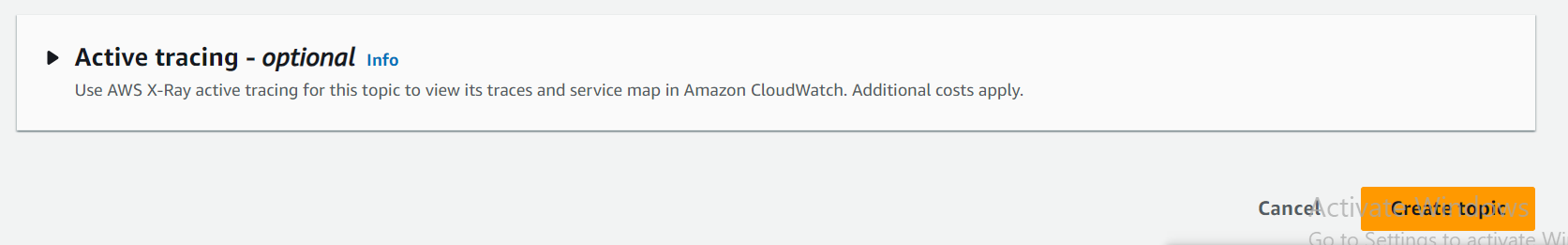


**Give the topic name and click on next step**

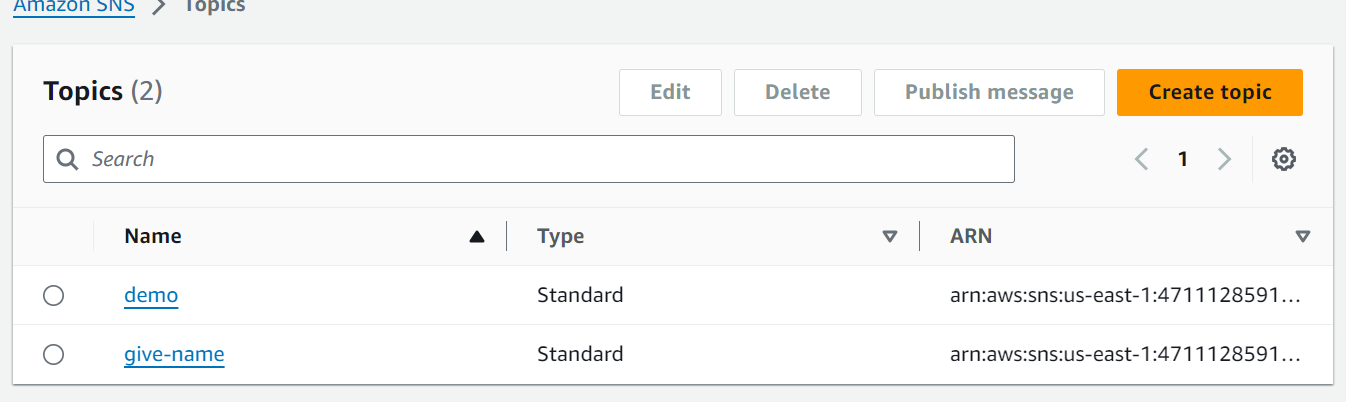


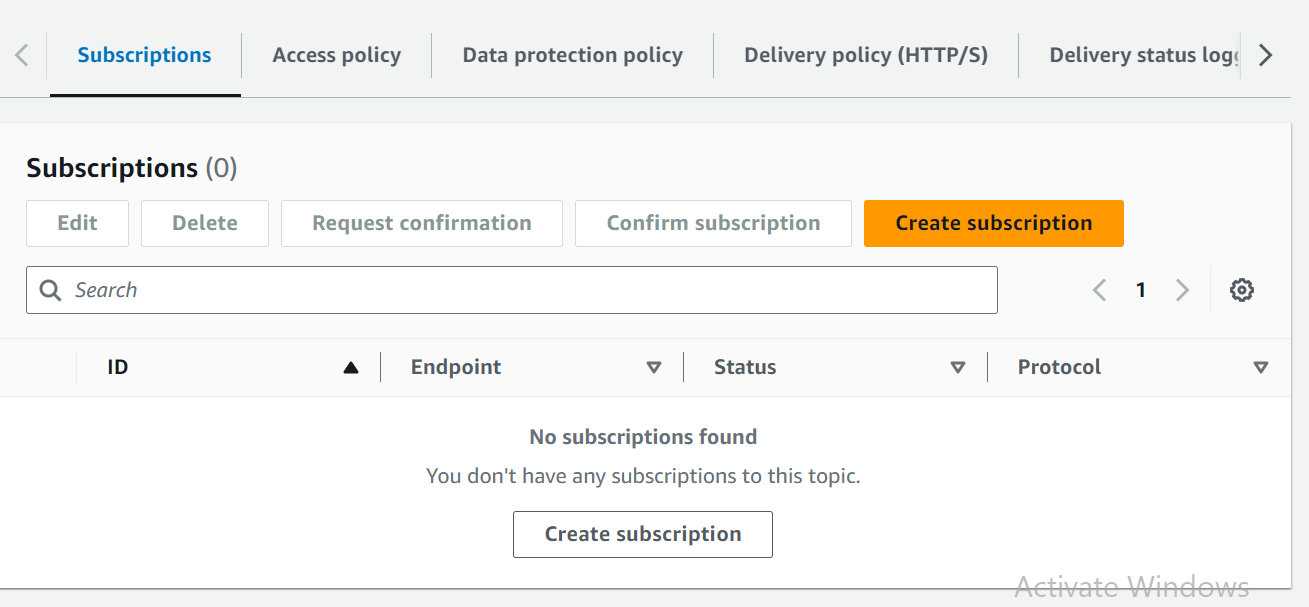
**Here select standard**



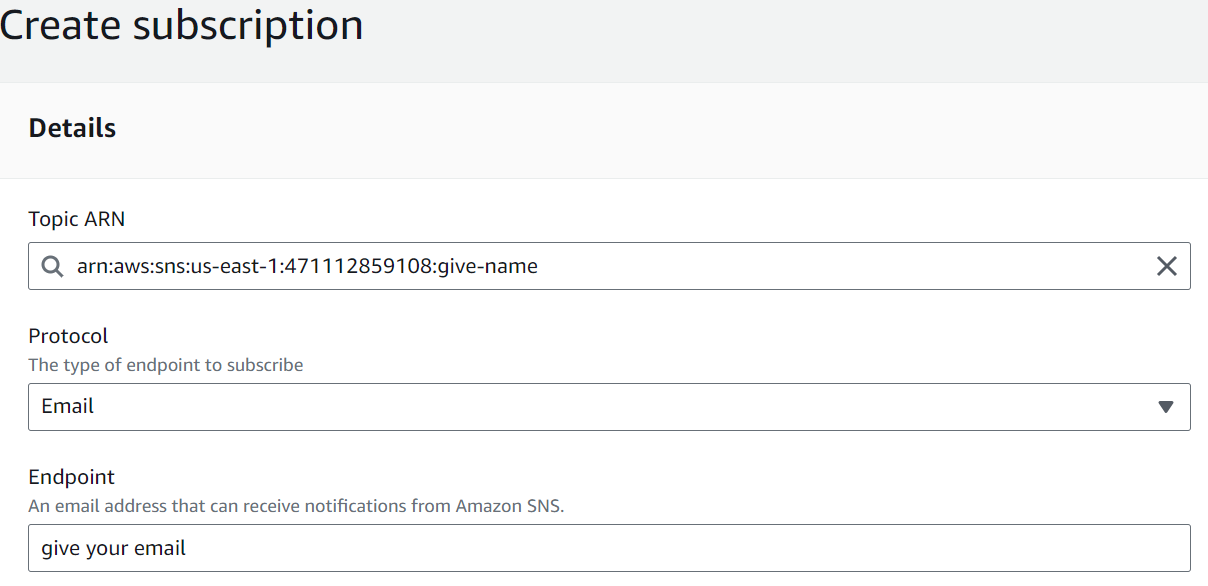


**Topic has been created**

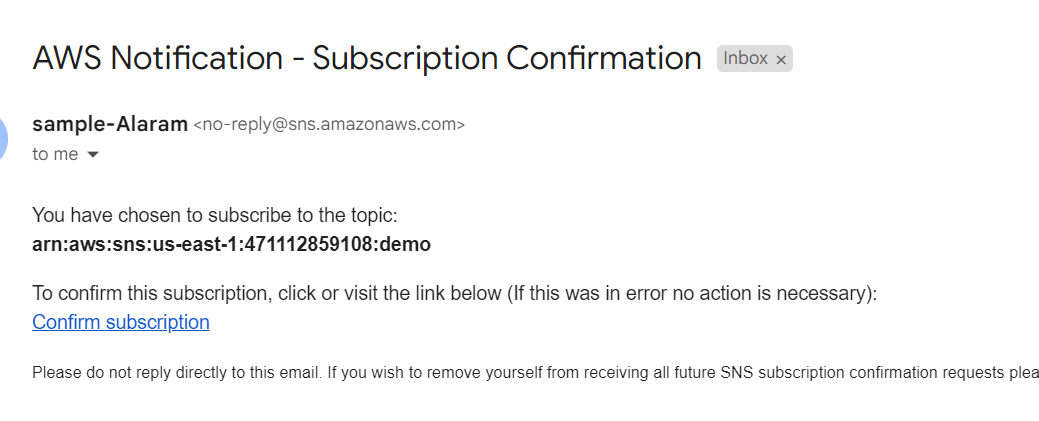




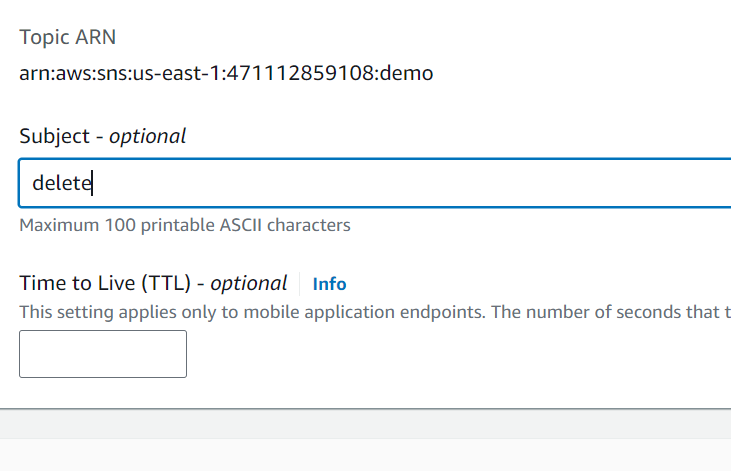
**Create a subscription and add a email.**



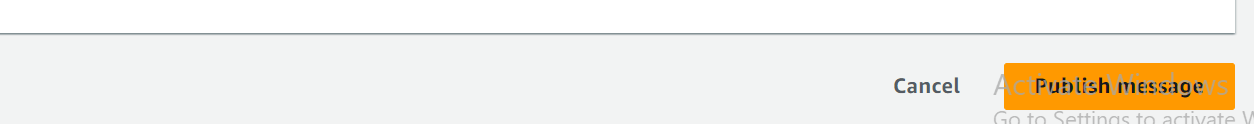
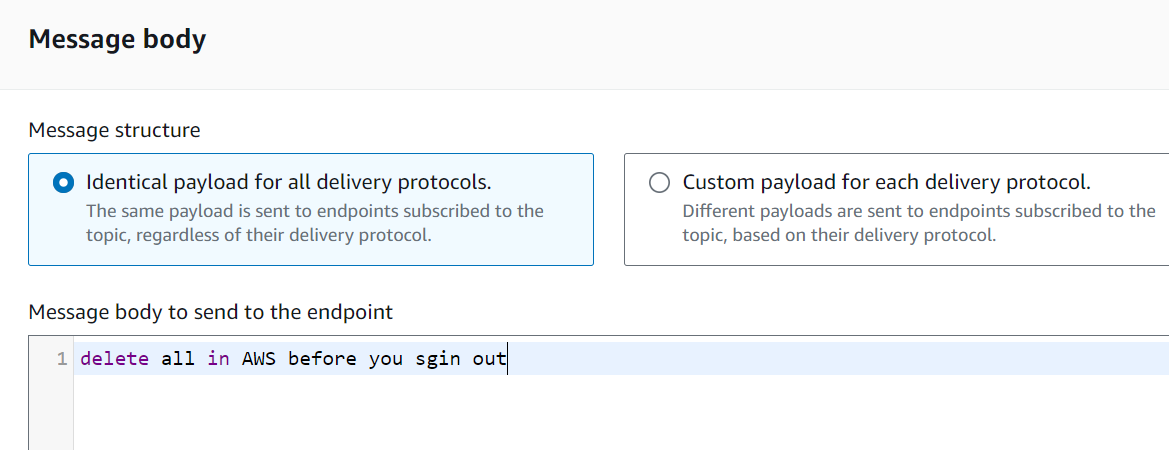
**Check the mail and subscribe the sns notification.**



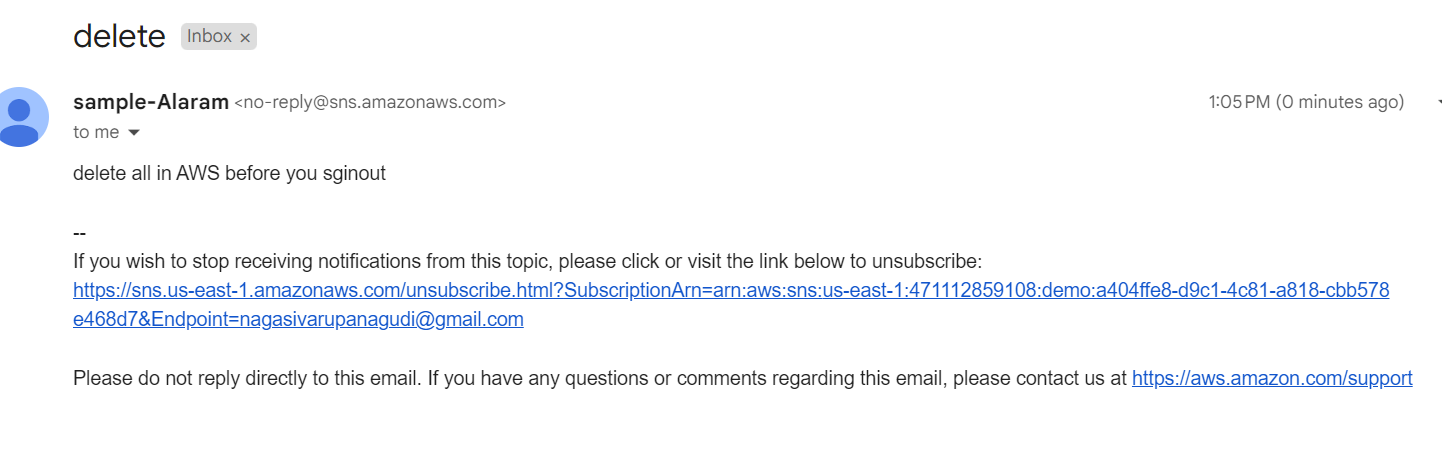
**Add the subject**



**Add the message body and publish the message.**



**Once notification is alert the message will popup on Email.**



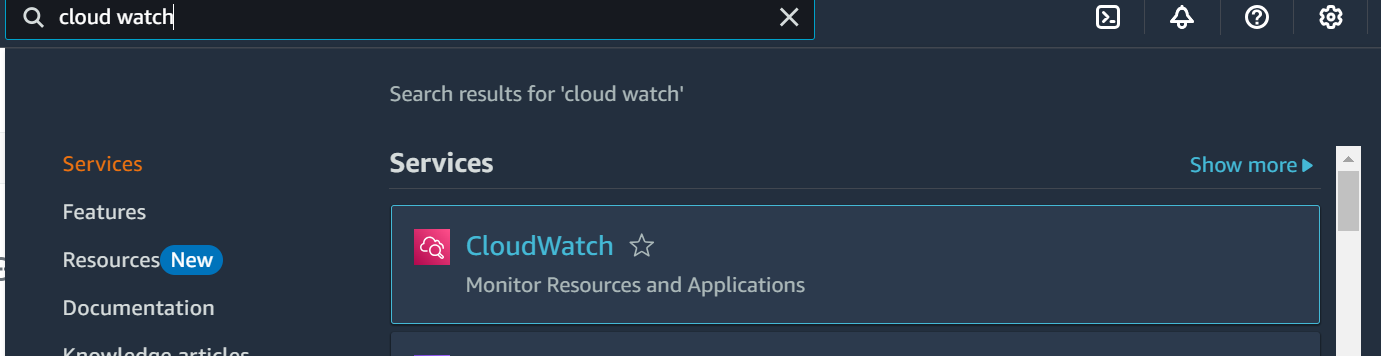
**Cloud watch**

Amazon CloudWatch is a monitoring and observability service provided by AWS that enables you to collect, analyze, and act on metrics and log data from your AWS resources and applications. Here’s an overview of its key features and functionalities:

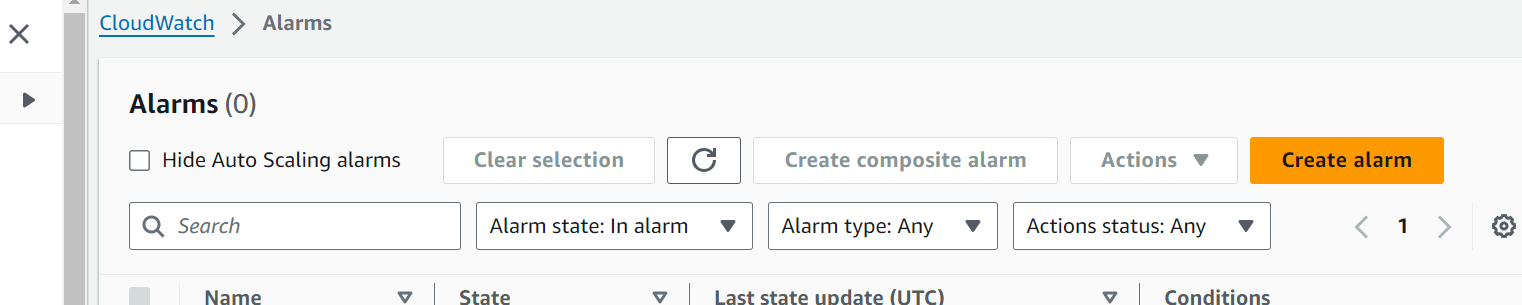
### Key Features

1. **Metrics Monitoring**: CloudWatch collects and monitors metrics from various AWS services (e.g., EC2, RDS, Lambda) and allows you to create custom metrics.
2. **Logs Management**: CloudWatch Logs lets you store, monitor, and access log files from your applications and AWS resources. You can search and filter log data, set alarms, and visualize log data.
3. **Alarms**: You can set alarms to monitor specific metrics and receive notifications when those metrics exceed predefined thresholds. Alarms can trigger actions, such as scaling resources or sending notifications via Amazon SNS.
4. **Dashboards**: Create custom dashboards to visualize your metrics and logs in real-time. Dashboards can include graphs, statistics, and other widgets to provide insights at a glance.
5. **Events**: CloudWatch Events (now part of EventBridge) helps you respond to state changes in your AWS resources by allowing you to create rules that trigger actions (like invoking a Lambda function) based on specific events.
6. **Insights and Analytics**: CloudWatch provides tools like CloudWatch Logs Insights for querying log data and gaining insights into your application’s performance and behavior.
7. **Service Integrations**: CloudWatch integrates with many AWS services, allowing you to monitor a wide array of resources and applications seamlessly.

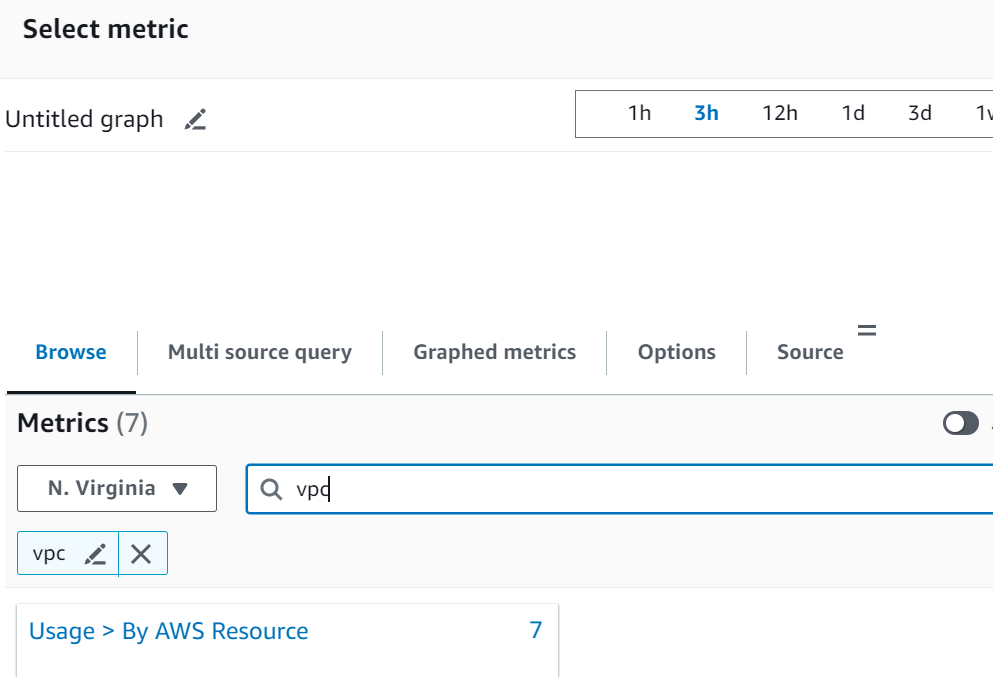
**Search and select cloud watch service.**



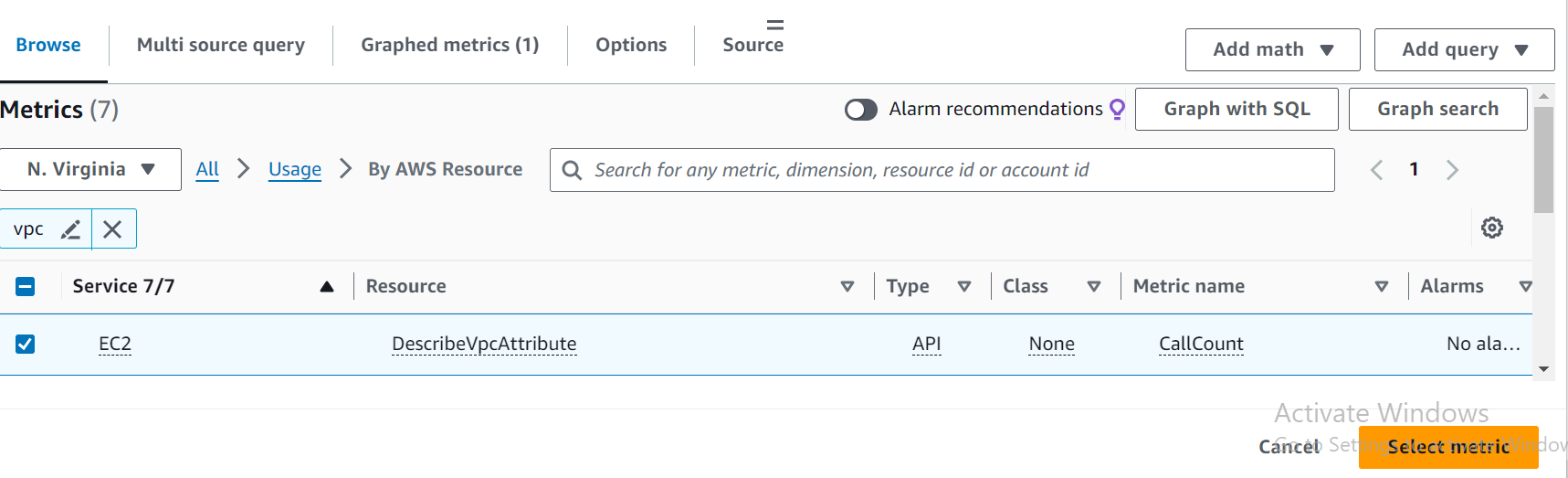
**Create a alarm**



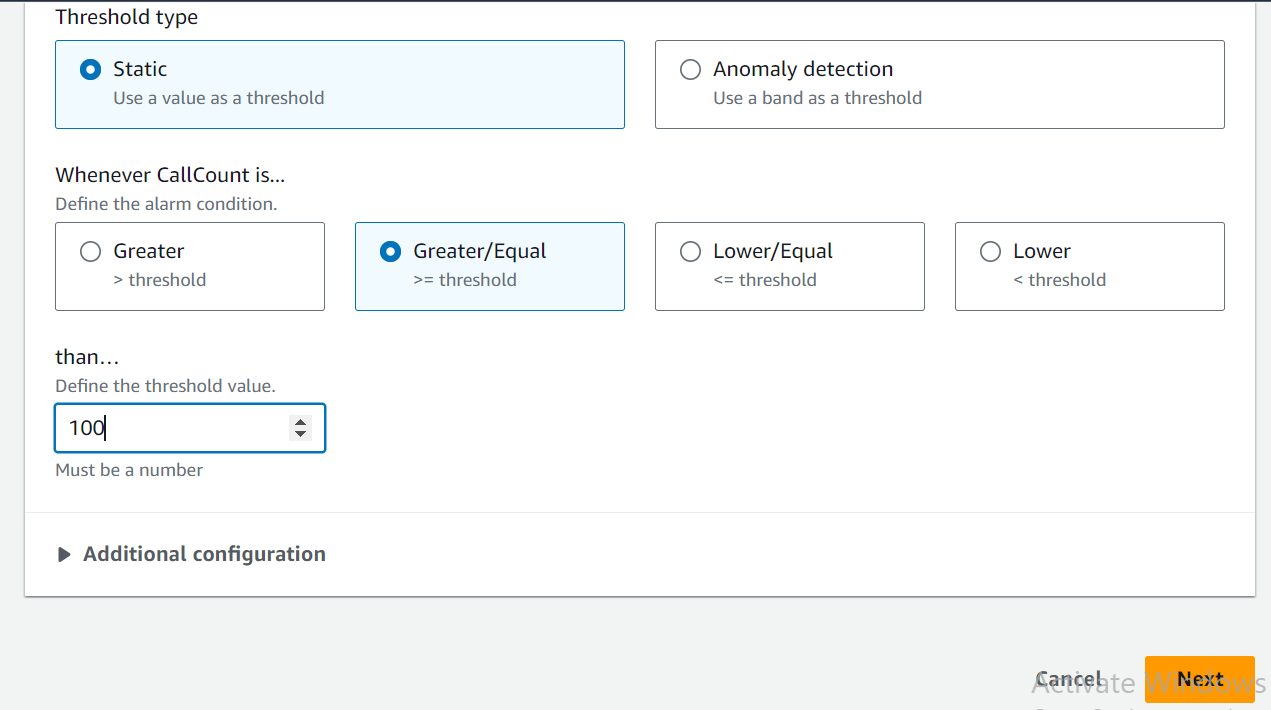
**Select metric**



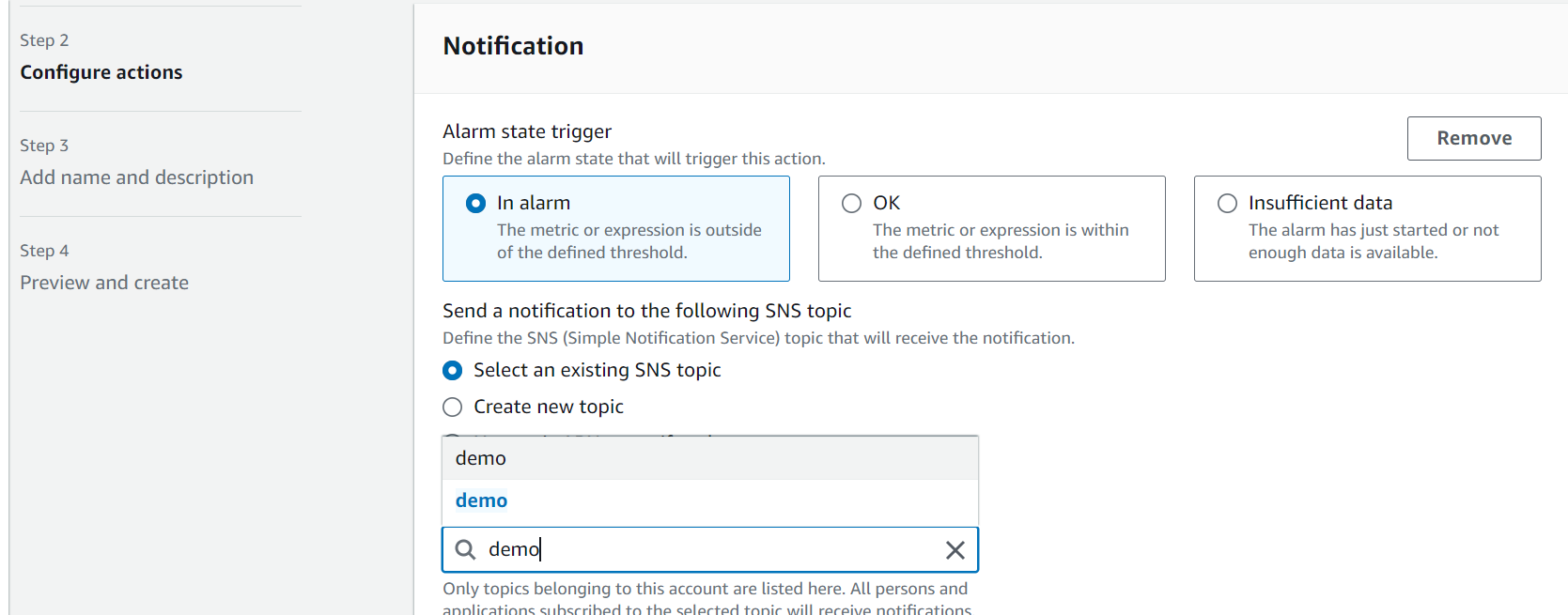
**Here select the metric as Ec2.**



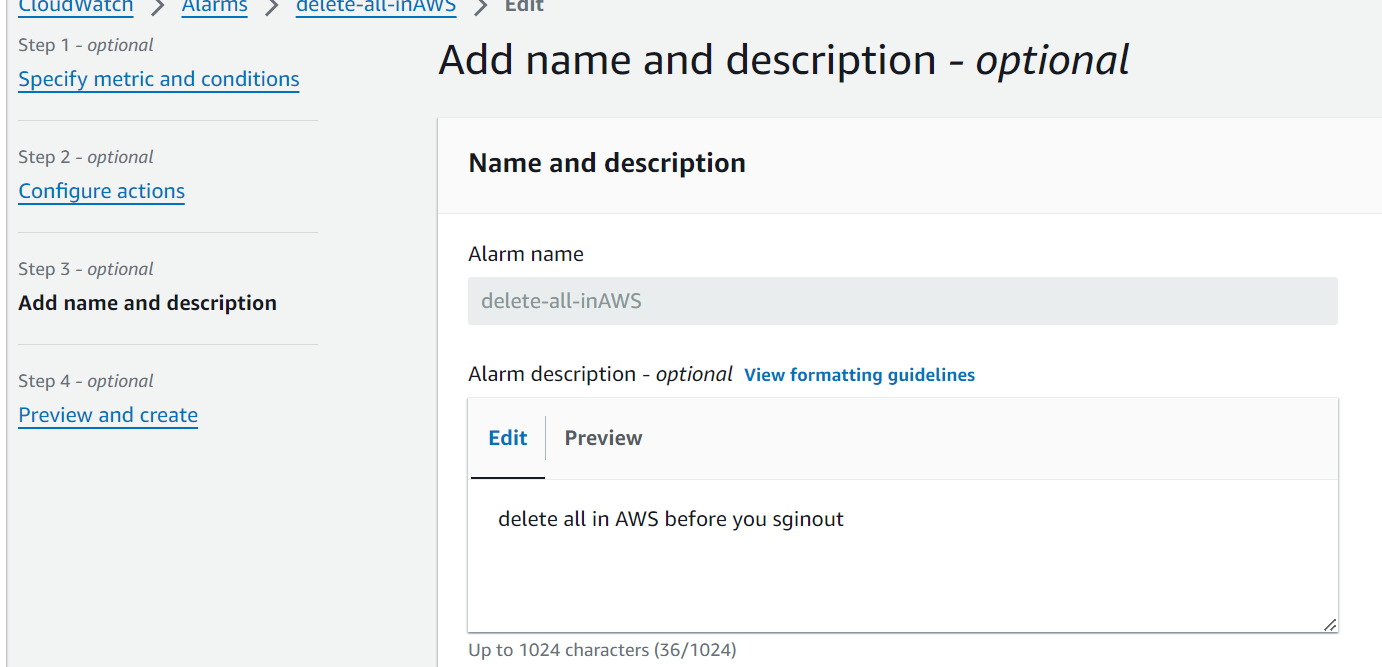
**Select the threshold type and define the value and click on next**

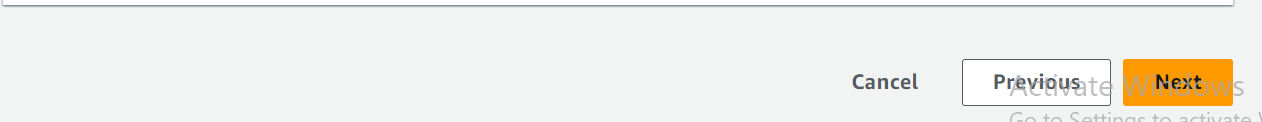


**Select the alarm in notification**

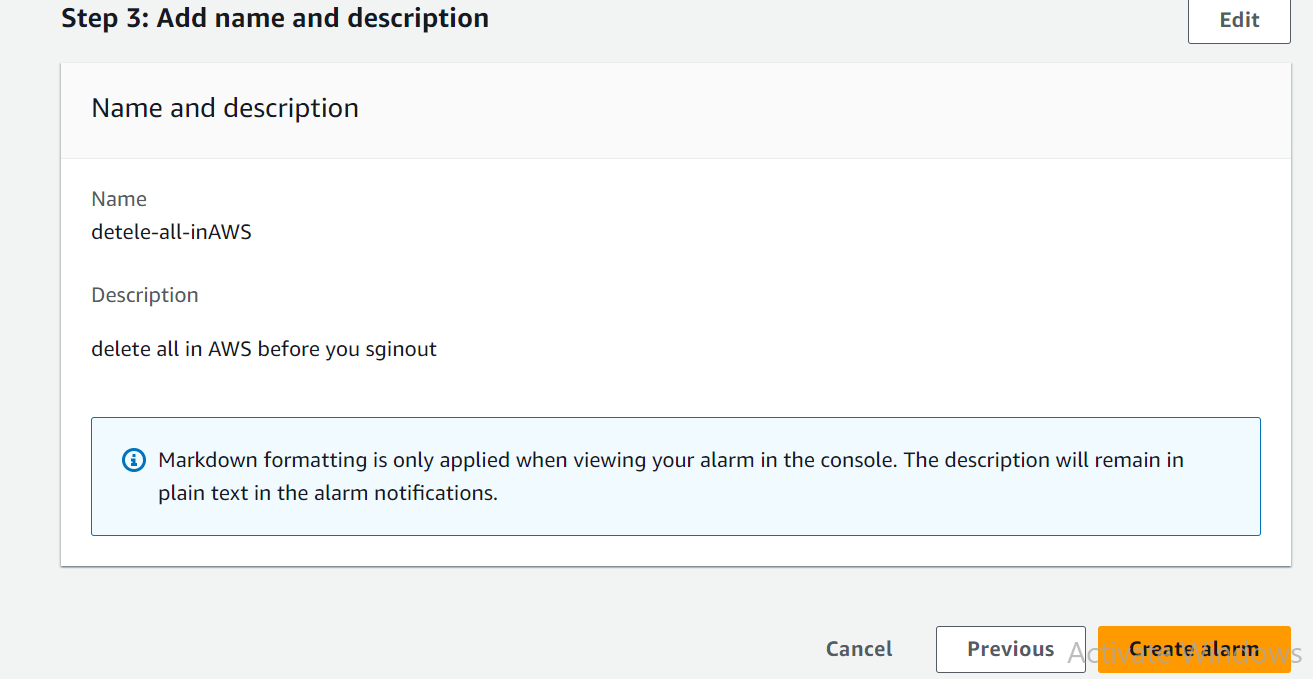


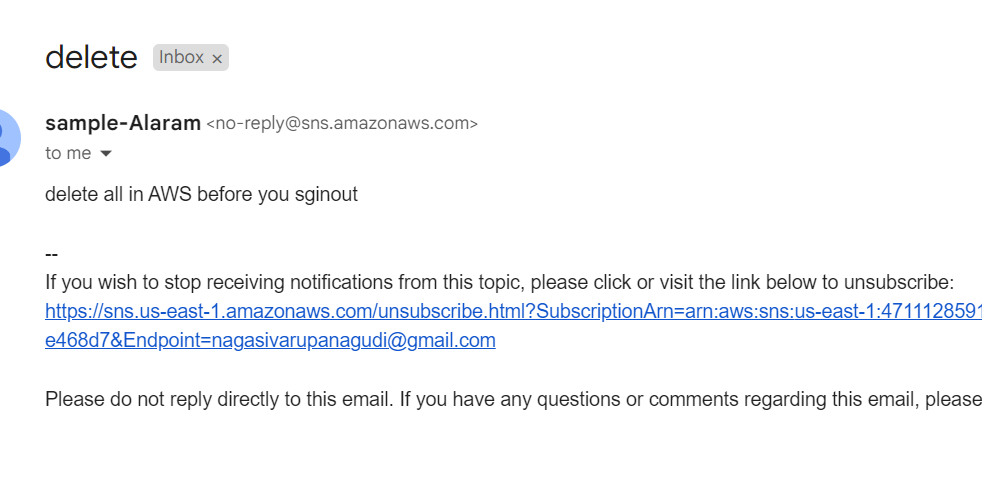
**Create a alarm.**





**Add a name and and description and then create a alarm**

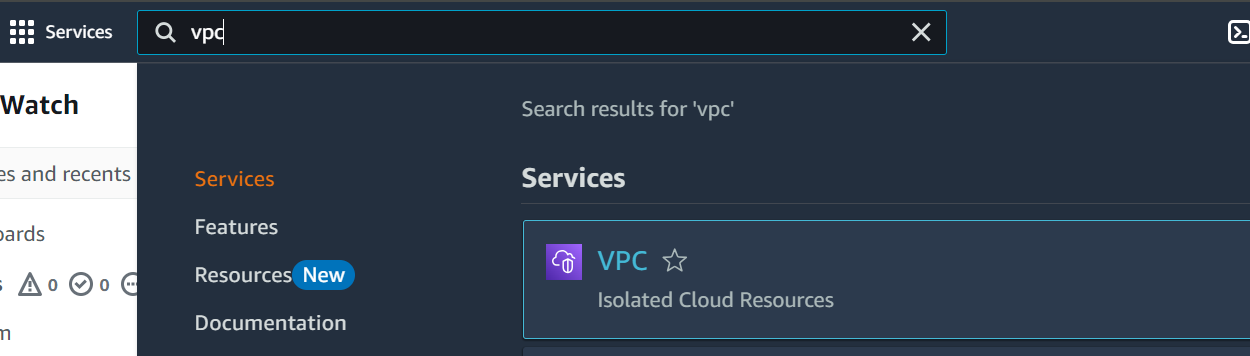




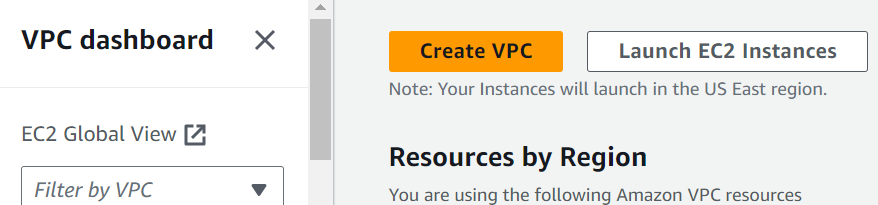
**VPC**

A virtual private cloud (VPC) is a secure, isolated [private cloud](https://www.cloudflare.com/learning/cloud/what-is-a-private-cloud/) hosted within a [public cloud](https://www.cloudflare.com/learning/cloud/what-is-a-public-cloud/). VPC customers can run code, store data, host websites, and do anything else they could do in an ordinary private cloud, but the private cloud is hosted remotely by a public cloud provider. (Not all private clouds are hosted in this fashion.) VPCs combine the scalability and convenience of public cloud computing with the data isolation of private cloud computing.

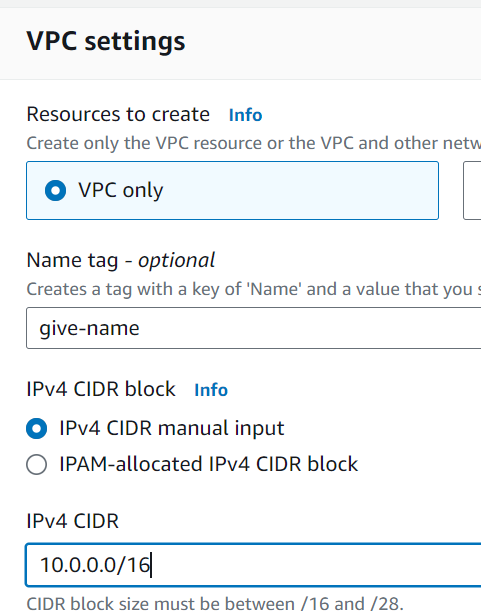
**Search and select vpc service.**



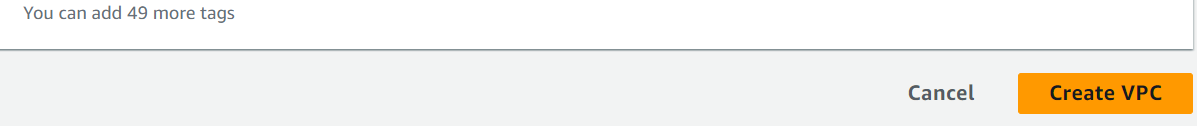
**Create VPC**



**Select vpc only option give CIDR Block.**

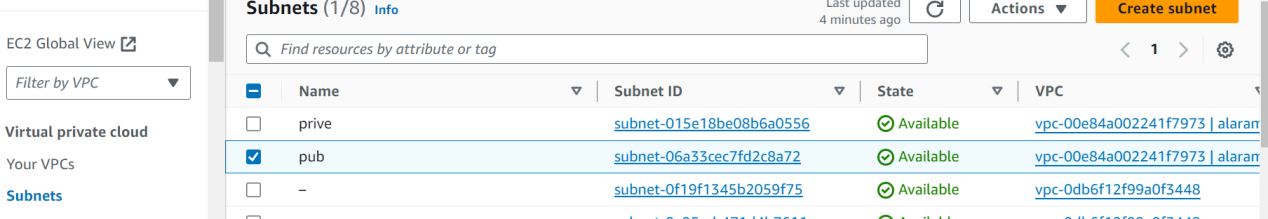


**Create a VPC**

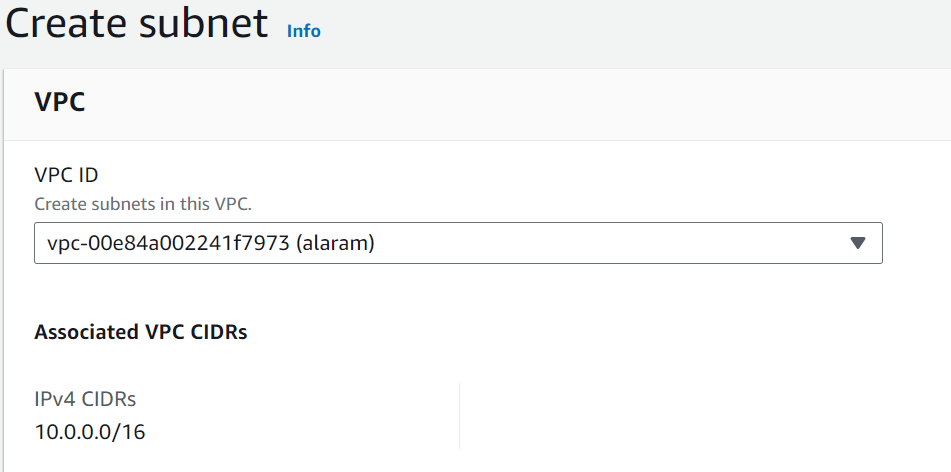




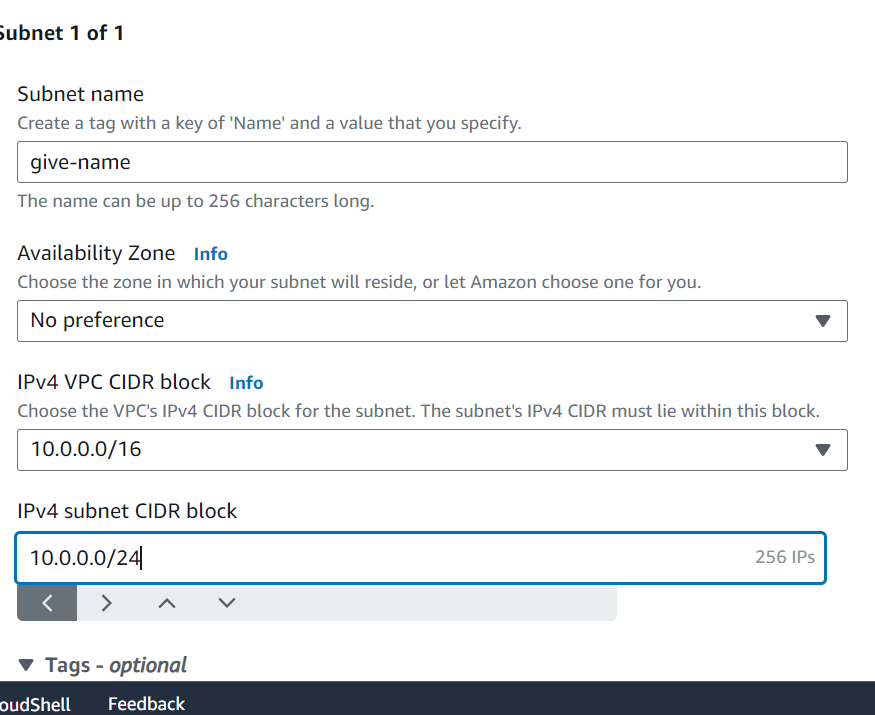
**List the VPC**



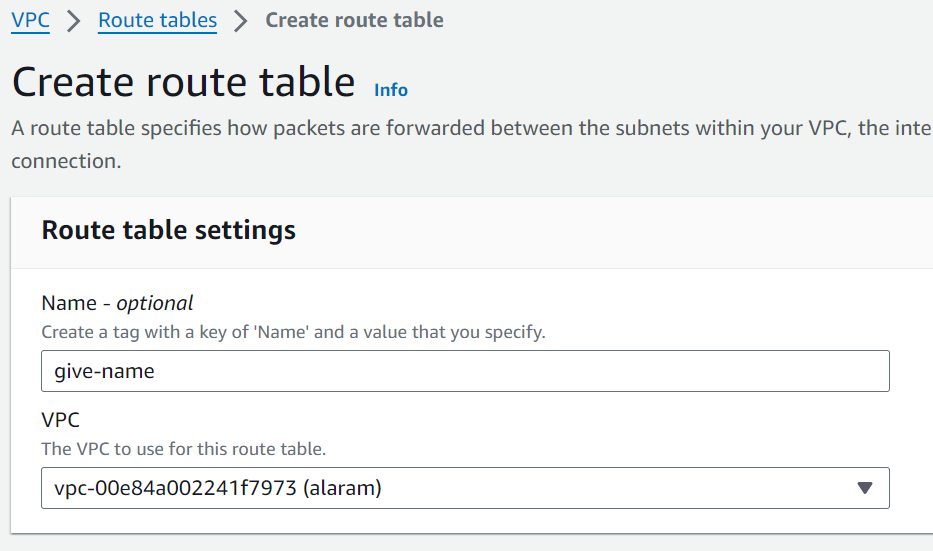
**Create a subnet using already created vpc.**



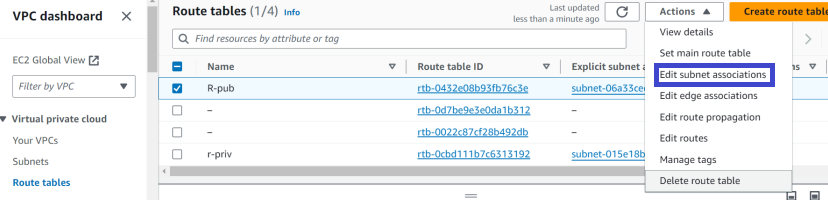
**Add the below details**



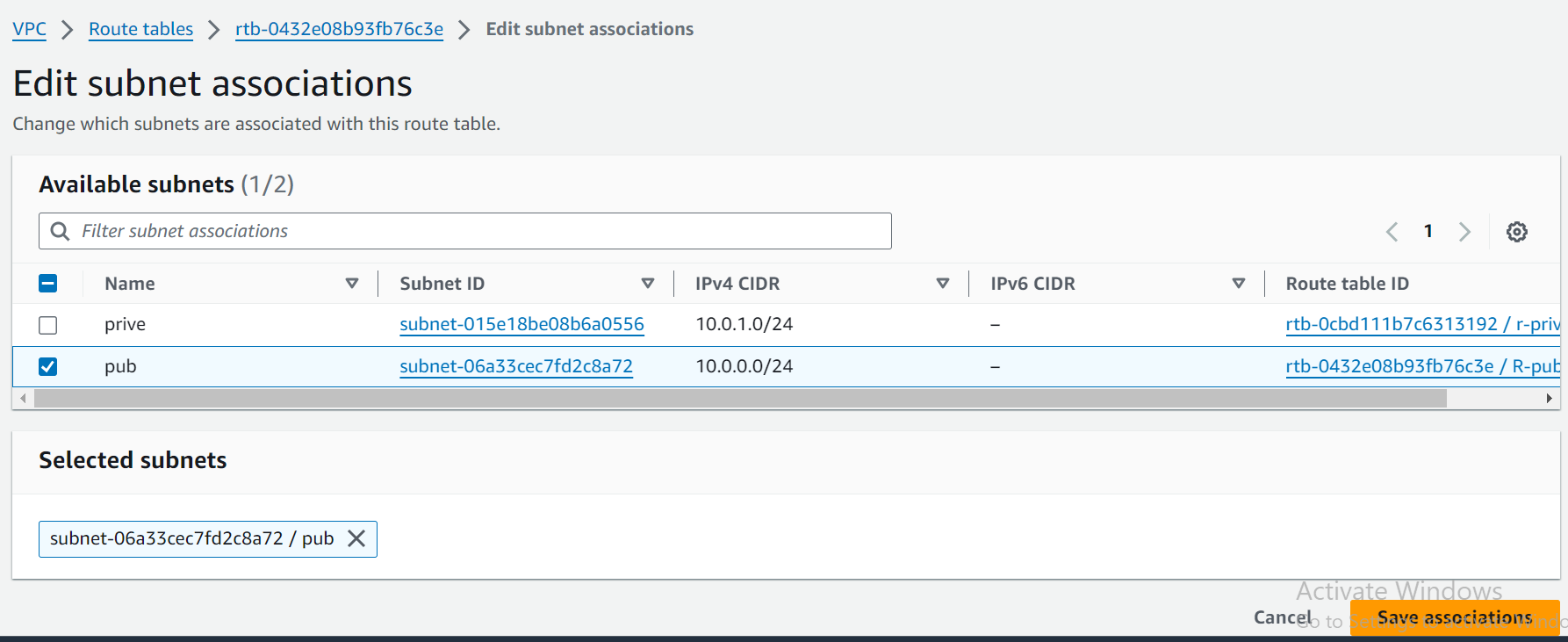
**Select the already created vpc**



**List the vpc**



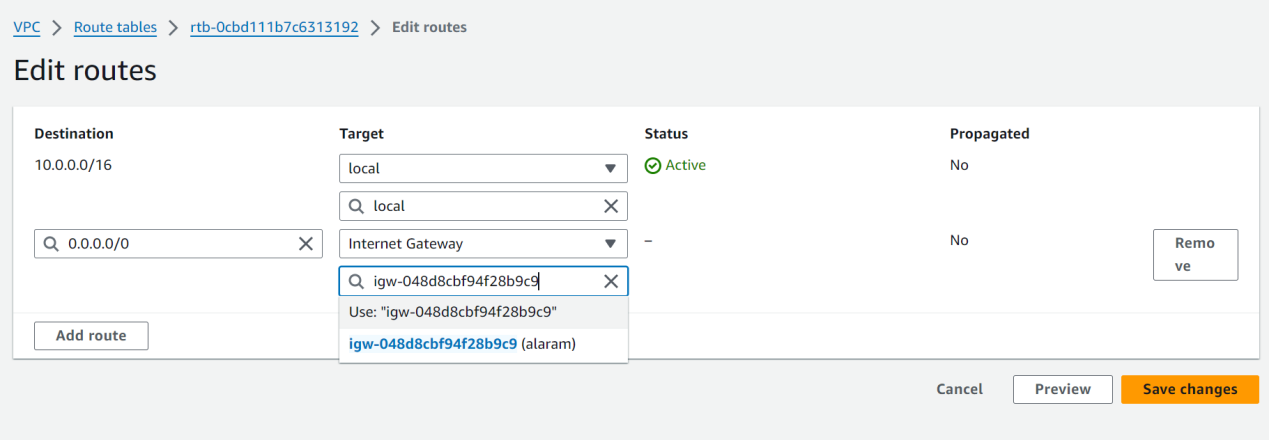
**Edit the subnet association**



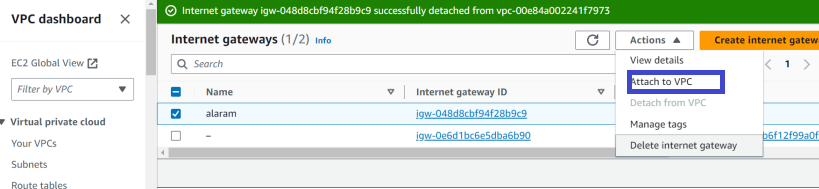
**Add the route table with subnet.**



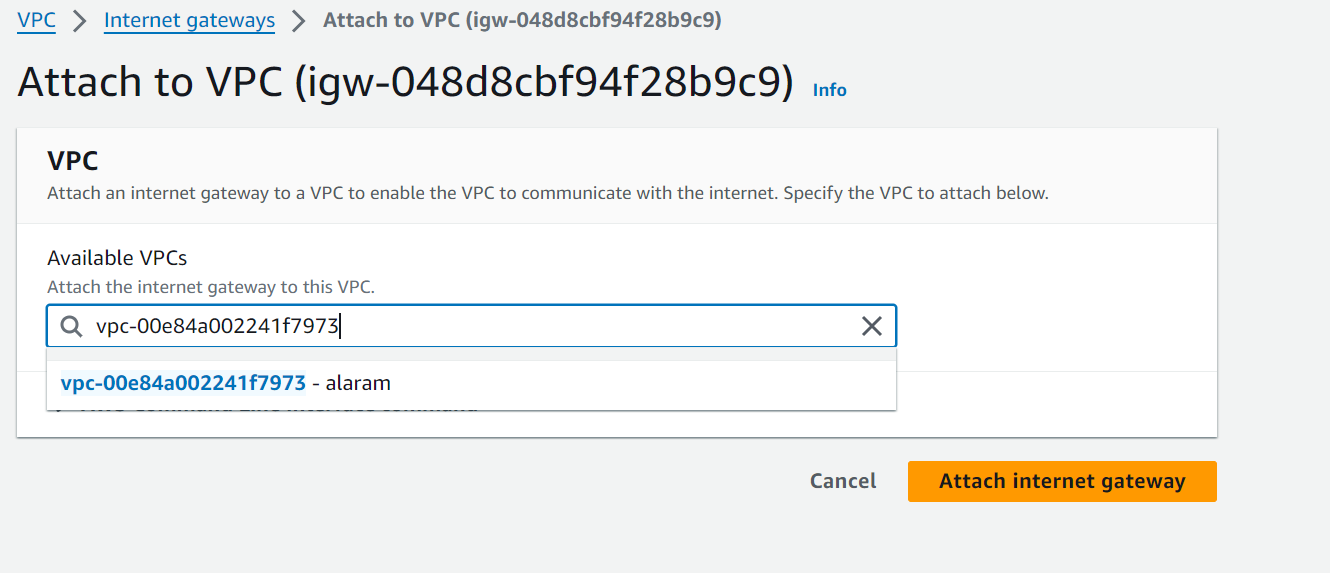
**Select internet gateway**

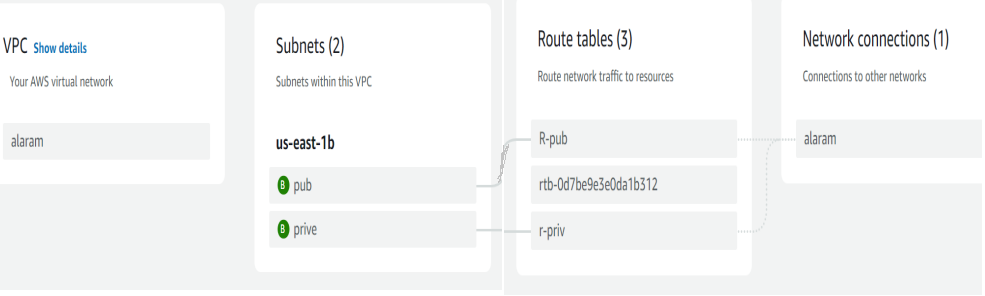


**Attach the vpc**

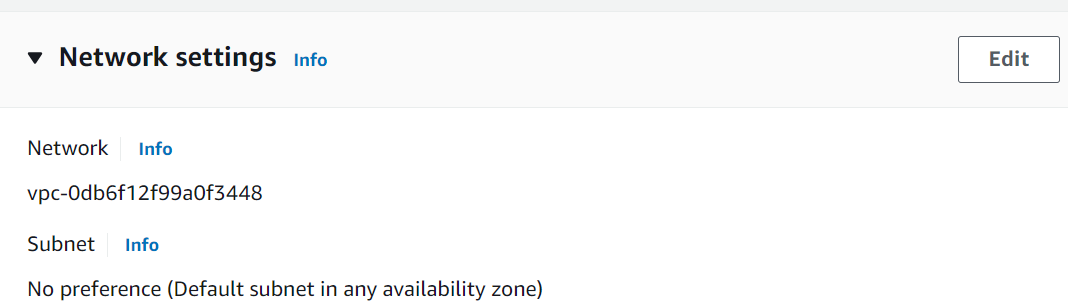


**Attach vpc to internet gateway**

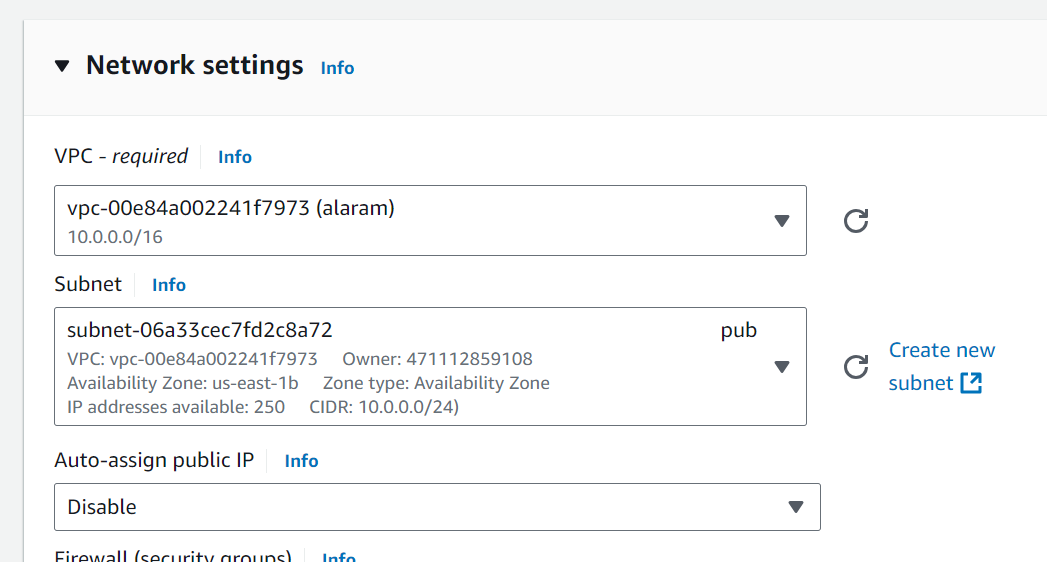


**Subnet route table and internet gateway is created.**  

**In Ec2 select the network setting and give an option edit**



**Here select already created vpc and subnet and create a EC2 instance**



**VPC**

Amazon VPC (Virtual Private Cloud) allows you to create a private network within the Amazon Web Services (AWS) cloud. Here are some key features and concepts associated with Amazon VPC:

**Key Features**

1. **Isolation**: Your VPC is isolated from other virtual networks in the AWS Cloud, ensuring that your resources are secure.
2. **Subnets**: You can divide your VPC into subnets (public and private) to organize resources and control traffic flow. Public subnets can access the internet, while private subnets are restricted.
3. **Route Tables**: You can create route tables to manage traffic between subnets and to/from the internet.
4. **Internet Gateway**: This allows resources in your public subnet to connect to the internet.
5. **NAT Gateway/Instance**: Enables instances in a private subnet to initiate outbound traffic to the internet while preventing inbound traffic.
6. **Security Groups and Network ACLs**: Security groups act as a virtual firewall for your EC2 instances, while Network ACLs provide an additional layer of security at the subnet level.
7. **Peering Connections**: You can connect multiple VPCs for resource sharing without using the public internet.
8. **VPN Connections**: Establish secure connections between your VPC and your on-premises network.
9. **VPC Endpoints**: These allow private connections to AWS services without needing an internet gateway.

**CODE**:

provider "aws" {

  region = "eu-west-1"

}

resource "aws\_vpc" "demo-vpc" {

  cidr\_block = "10.0.0.0/16"

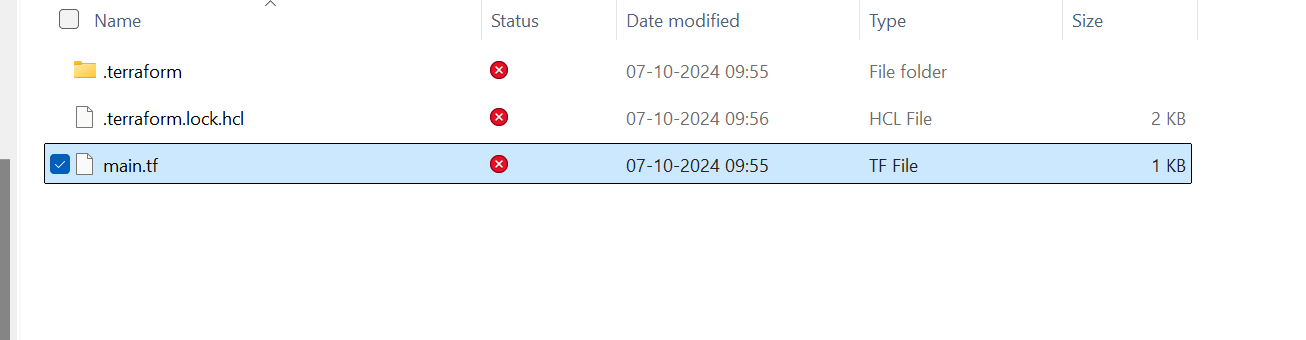
  tags = {

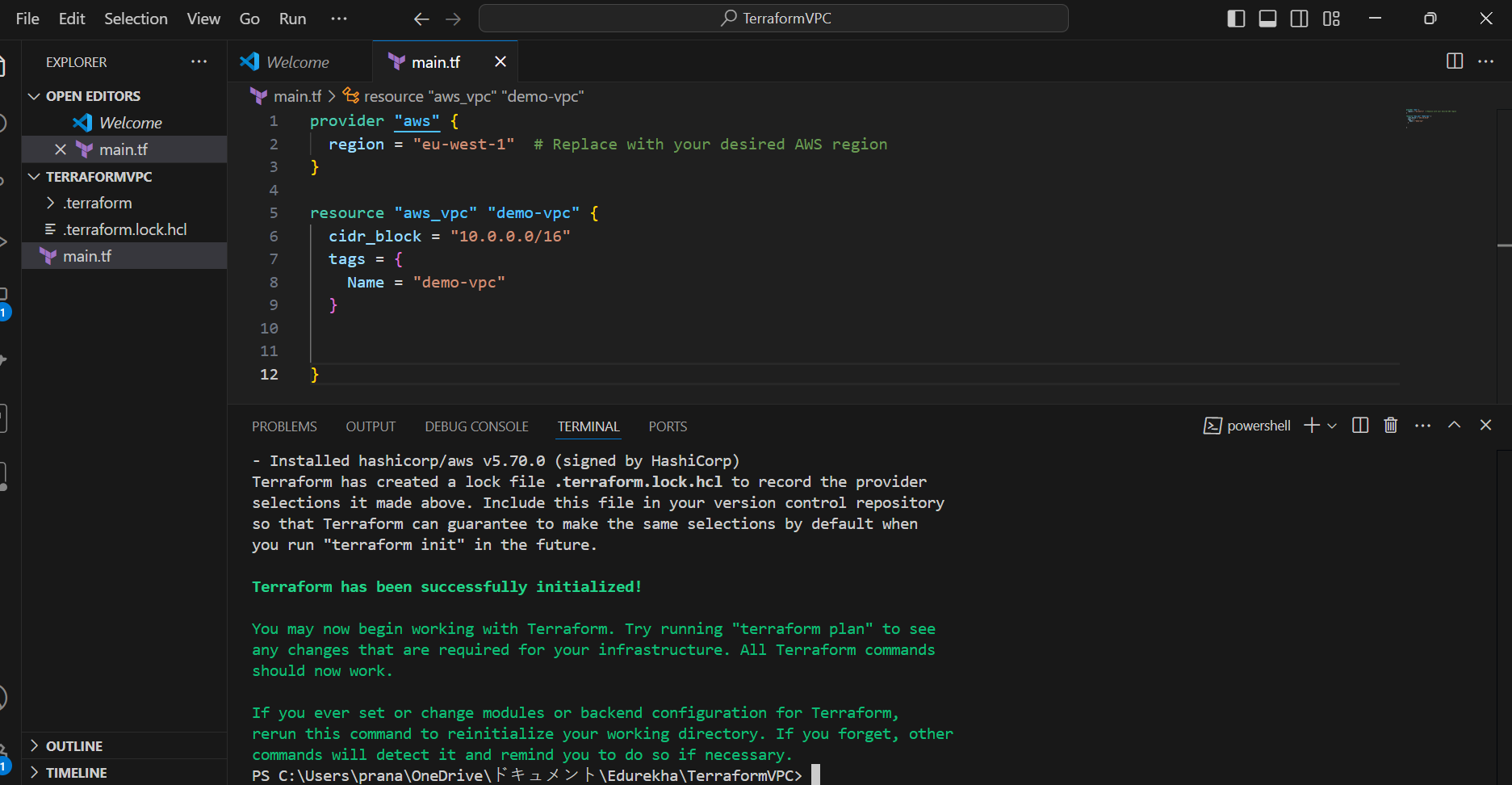
    Name = "demo-vpc"

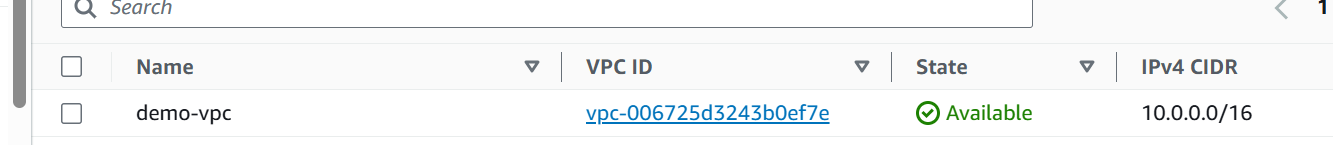
  }

}

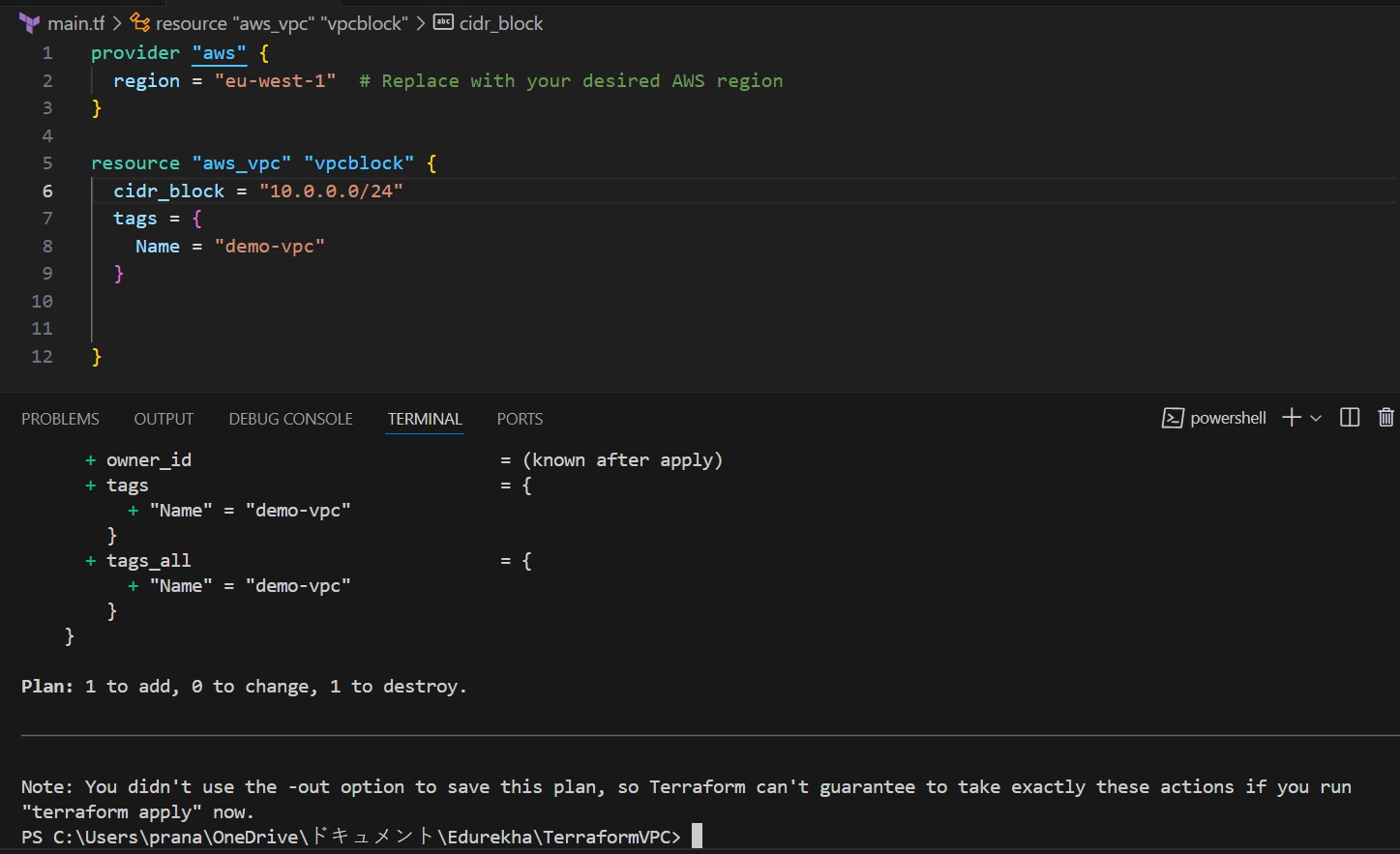
Check in the folder .tf file is created.

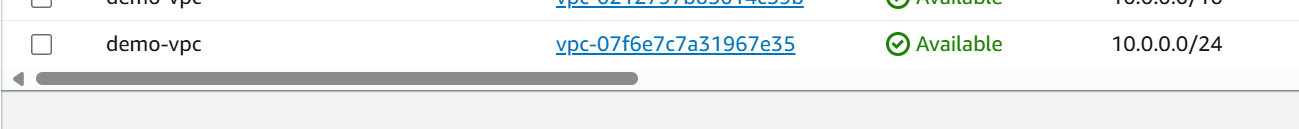






Change the CIDR number and check the status





**INTERNET GATEWAY**

An internet gateway is a logical connection between the vpc and internet. It allows communication between resources within the vpc and the internet. Each vpc has only one IGW and supports both IPv4 and IPv6 traffic.

**CODE**:

resource "aws\_internet\_gateway" "demo-igw" {

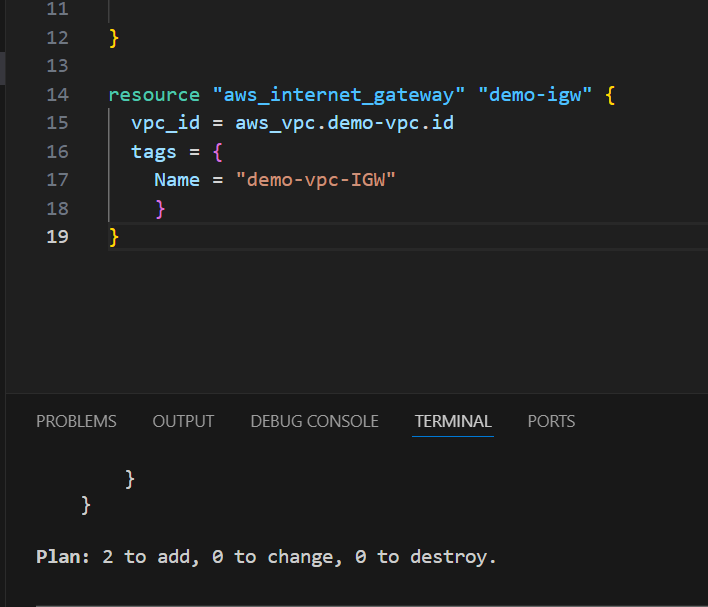
  vpc\_id = aws\_vpc.demo-vpc.id

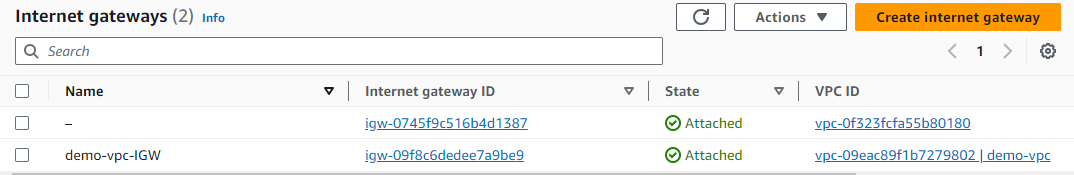
  tags = {

    Name = "demo-vpc-IGW"

    }

}





**SUBNET**

* A subnet is a range of IP addresses in your VPC.
* You launch AWS resources, such as Amazon EC2 instances, into your subnets.
* Subnets are regional resources.
* Each subnet defines a range of IPv4 addresses.
* Traffic to and from instances can be controlled with network firewall rules.

**CODE:**

  resource "aws\_subnet" "private-subnet-1" {

  vpc\_id     = aws\_vpc.demo-vpc.id

  cidr\_block = "10.0.1.0/24"

  availability\_zone = "eu-west-1a"

  tags = {

    Name = "private-subnet-1"

  }

}

resource "aws\_subnet" "private-subnet-2" {

  vpc\_id     = aws\_vpc.demo-vpc.id

  cidr\_block = "10.0.2.0/24"

  availability\_zone = "eu-west-1b"

  tags = {

    Name = "private-subnet-2"

  }

}

resource "aws\_subnet" "public-subnet-1" {

vpc\_id     = aws\_vpc.demo-vpc.id

cidr\_block = "10.0.3.0/24"

availability\_zone = "eu-west-1a"

tags = {

Name = "public-subnet-1"

}

}

resource "aws\_subnet" "public-subnet-2" {

  vpc\_id     = aws\_vpc.demo-vpc.id

  cidr\_block = "10.0.4.0/24"

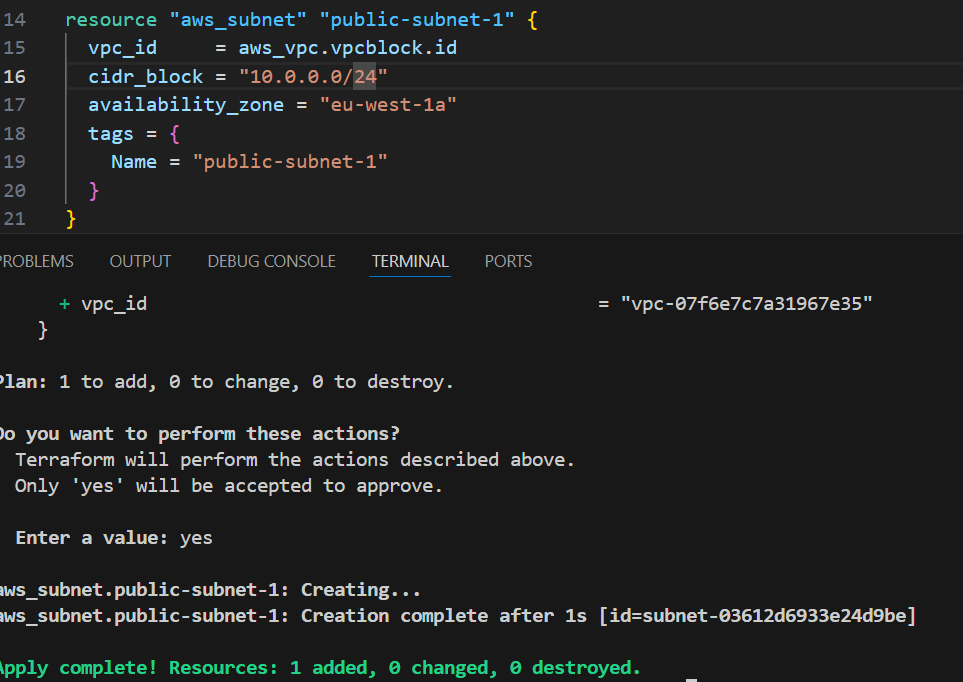
  availability\_zone = "eu-west-1b"

  tags = {

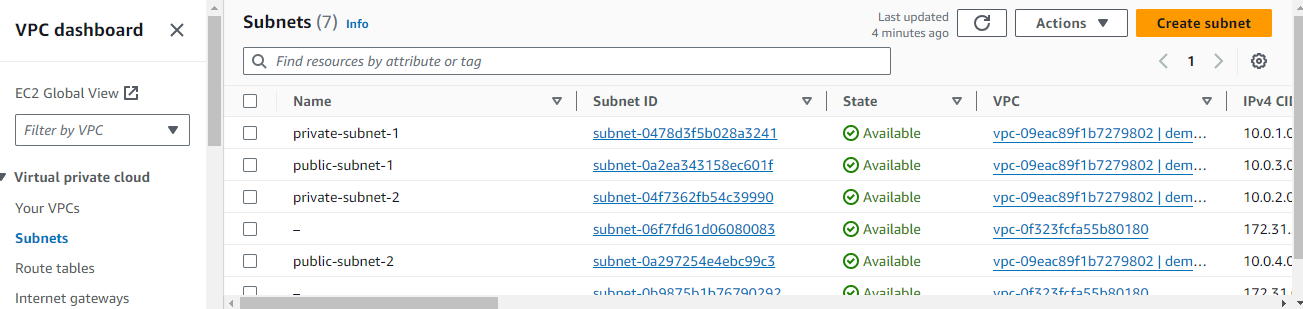
    Name = "public-subnet-2"

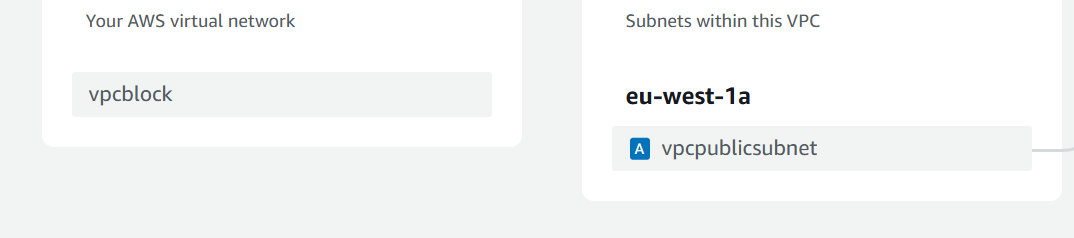
  }

}









**ROUTE TABLE**

In AWS, a route table is a set of rules that determines where neytwork traffic is directed. Each subnet in your aws virtual private cloud is associated with a route table traffis flow between subnets. The route tables includes details like the route table ID and the ID of its associated vpc.

**CODE:**

resource "aws\_route\_table" "public-route-table" {

  vpc\_id = aws\_vpc.demo-vpc.id

  tags = {

    Name = "public-route-table"

  }

}

resource "aws\_route" "public-route" {

  route\_table\_id         = aws\_route\_table.public-route-table.id

  destination\_cidr\_block = "0.0.0.0/0"

  gateway\_id             = aws\_internet\_gateway.demo-igw.id

}

resource "aws\_route\_table\_association" "public-subnet-1-association" {

  subnet\_id      = aws\_subnet.public-subnet-1.id

  route\_table\_id = aws\_route\_table.public-route-table.id

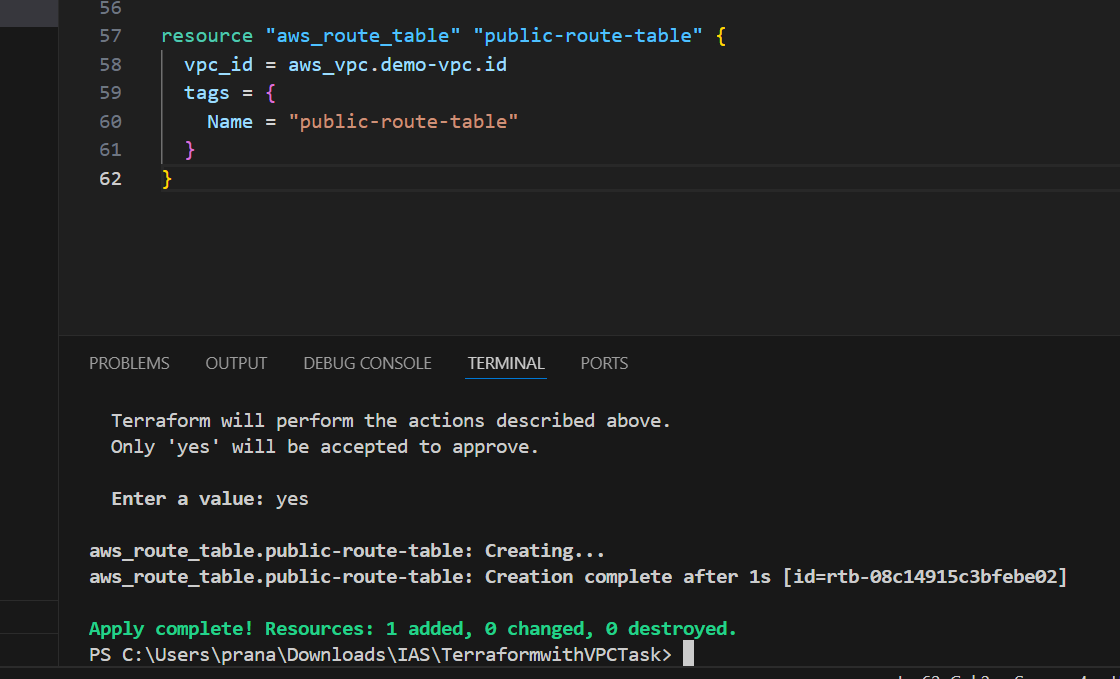
}

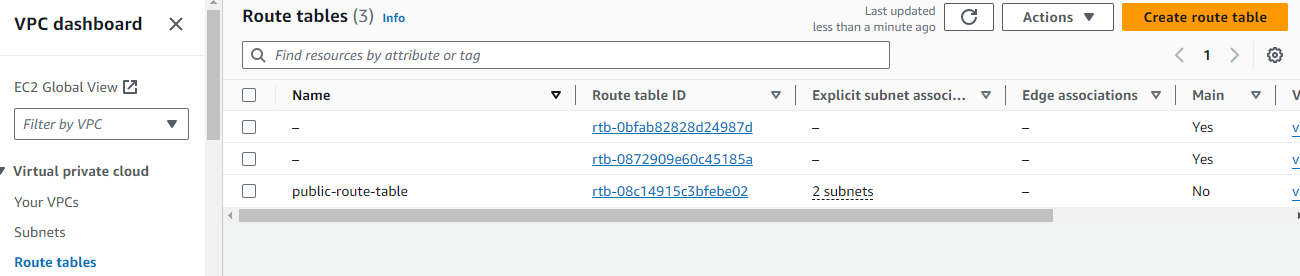
resource "aws\_route\_table\_association" "public-subnet-2-association" {

  subnet\_id      = aws\_subnet.public-subnet-2.id

  route\_table\_id = aws\_route\_table.public-route-table.id

}





**NAT GATEWAY**

AWS NAT Gateway – stands for Network Address Translation. It is a managed AWS service that is scaled based on your usage. NAT Gateway will help you to access the internet which instances are configured in the private subnet but without proper routing, no one can access that instance from outside.

**CODE:**

resource "aws\_nat\_gateway" "nat-gateway" {

  allocation\_id = aws\_eip.nat-eip.id

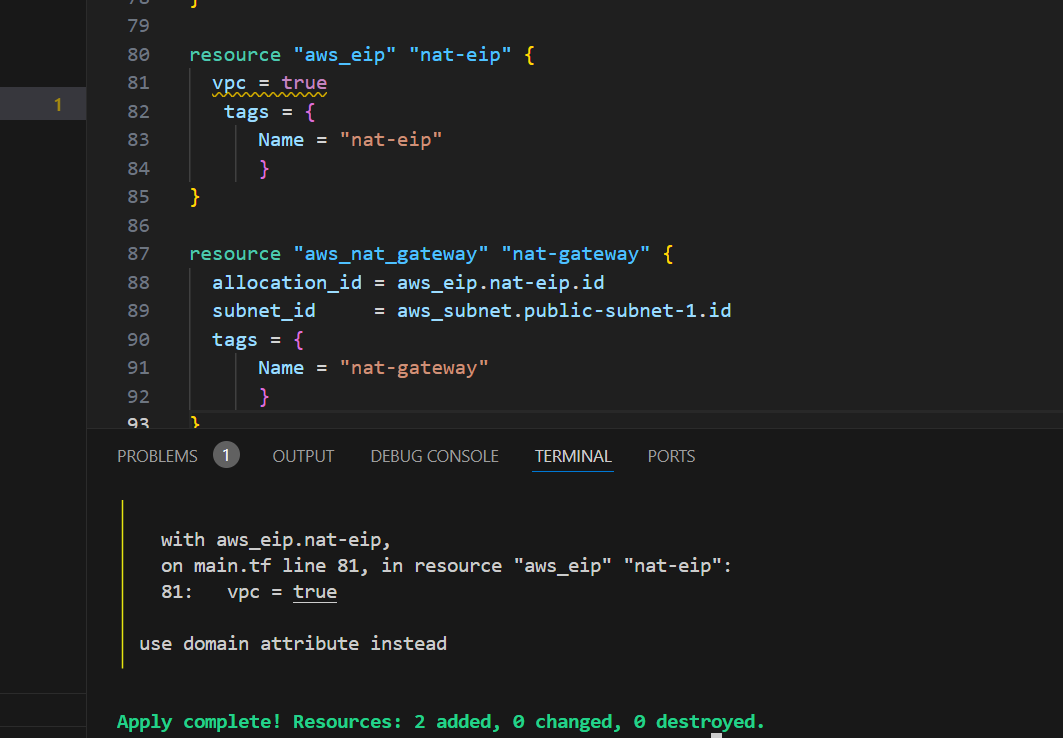
  subnet\_id     = aws\_subnet.public-subnet-1.id

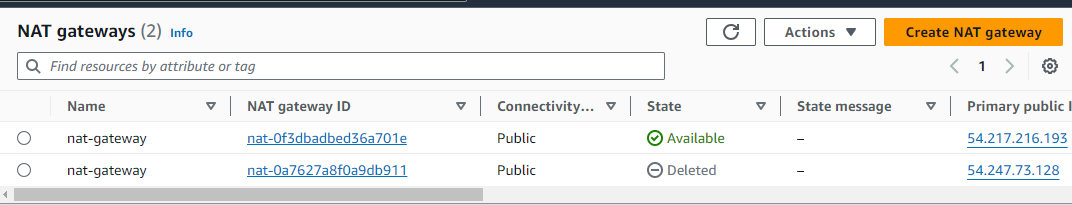
  tags = {

  Name = "nat-gateway"

      }

}





**SECURITY GROUPS**

Security group, which functions as a virtual firewall to regulate the inbound and outgoing traffic for AWS EC2 instances or other AWS resources in a VPC. We shall go over a security group’s definition and formation in this article.

**CODE:**

resource "aws\_security\_group" "secgroup" {

    name = "secgroup"

    description = "awssecuritygroup"

    vpc\_id = aws\_vpc.demo-vpc.id

       ingress {

        from\_port = 0

        to\_port = 65535

        protocol = "tcp"

        cidr\_blocks = ["0.0.0.0/0"]

       }

       egress {

          from\_port =0

          to\_port = 65535

          protocol ="tcp"

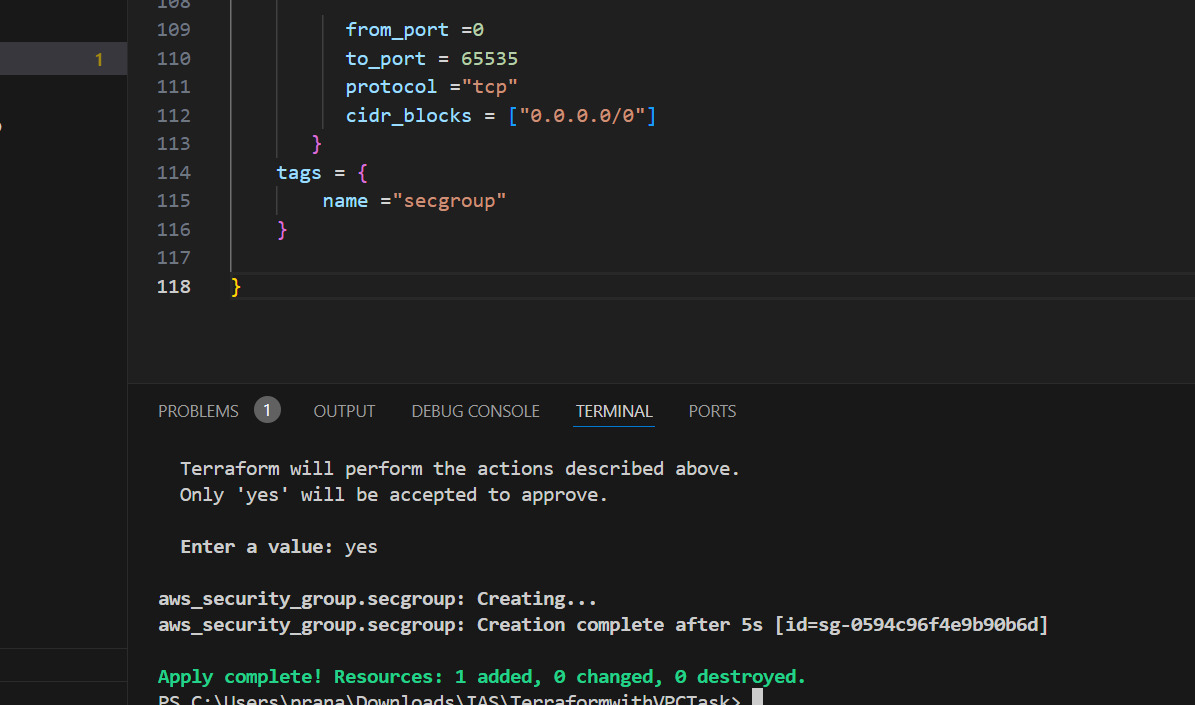
          cidr\_blocks = ["0.0.0.0/0"]

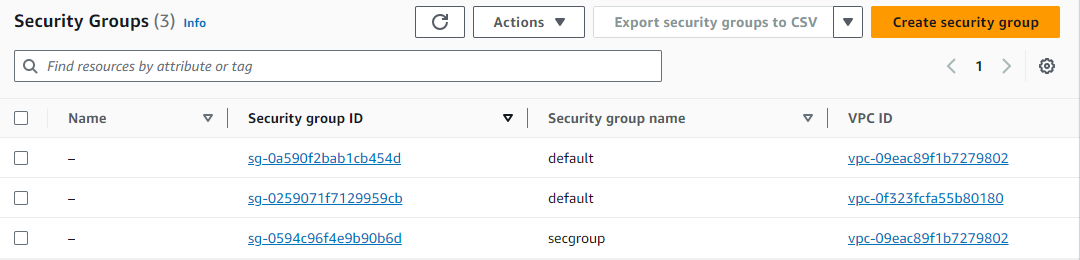
       }

    tags = {

        name ="secgroup"

    }





**PROCESS:**

Created vpc with region eu-west-1 with IP adddress 10.0.0.0/16 and named demo vpc.

Created a internet gateway for demo vpc.

Created 4 subnets; 2 private, 2 public –

* Private subnet-1, cidr 10.0.1.0/24, availability zone eu-west-1a
* Private subnet-2, cidr 10.0.2.0/24, availability zone eu-west-1b
* Public subnet-1, cidr 10.0.3.0/24, availability zone eu-west-1a
* Public subnet-1, cidr 10.0.4.0/24, availability zone eu-west-1b

Created route table for vpc named as public route table, connect to internet gateway through routes with cidr 0.0.0.0/0 and attaching public subnets to the route table.

Created NAT Gateway for the public subnet and named as nat gateway.

Created security group for vpc with inbound and outbound rules as port from port 0 to port 65535 and cidr blocks 0.0.0.0/0.

At the end we have allowed internet access to public subnets using IGW to private subnets using NGW.

**CONCLUSION:**

VPC provides a secure and flexible way to deploy applications in the cloud.