Table of contents

Introduction to VPC endpoints :

- VPC endpoints provide us with a secure way to connect to our AWS services from within a VPC without having to go on the internet .
- There are two types of VPC endpoints
  - Gatweay endpoint
  - Interface endpoint
- Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Gateway endpoint

- A gateway endpoint is a gateway that you specify as a target for a route in your route table for traffic destined to a supported AWS service. The following AWS services are supported:
  - Amazon S3
  - DynamoDB
- As iterated above , we do not need IGW or NAT to access the endpoints

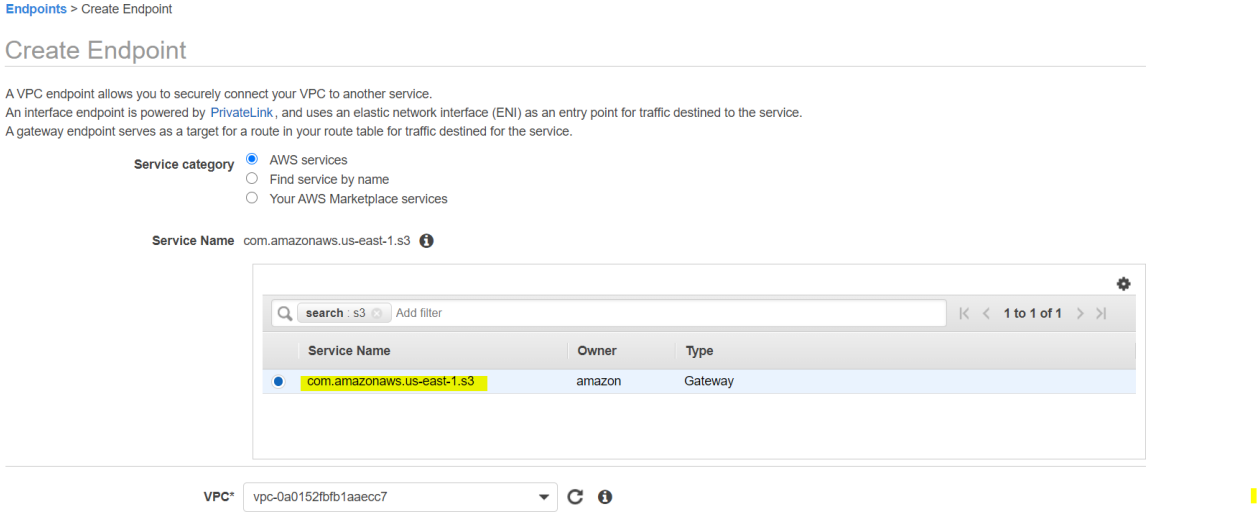
Creating gateway endpoint

- In order to test gateway endpoint we will require a private subnet
- Do not connect the NAT gateway to the associated route table as we need to test the theory of endpoint being available regardless of internet
- Connect to the private ec2 instance using a jump box . Make sure it has IAM privileges to access S3 . You can use IAM role here
- Once connected try below commands to check if we are able to access s3

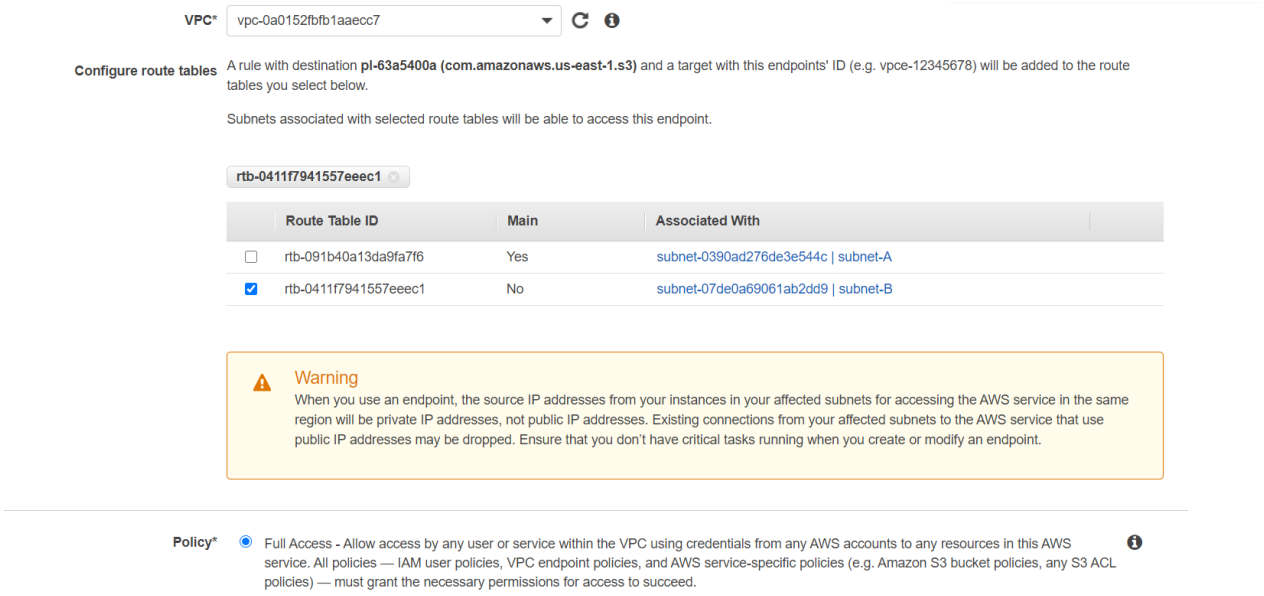
```
aws s3 ls

aws s3 ls bucketname
```

- Above commands ideally will not work as there is no internet access to the private instance as we did not attach a NAT gateway to the route table

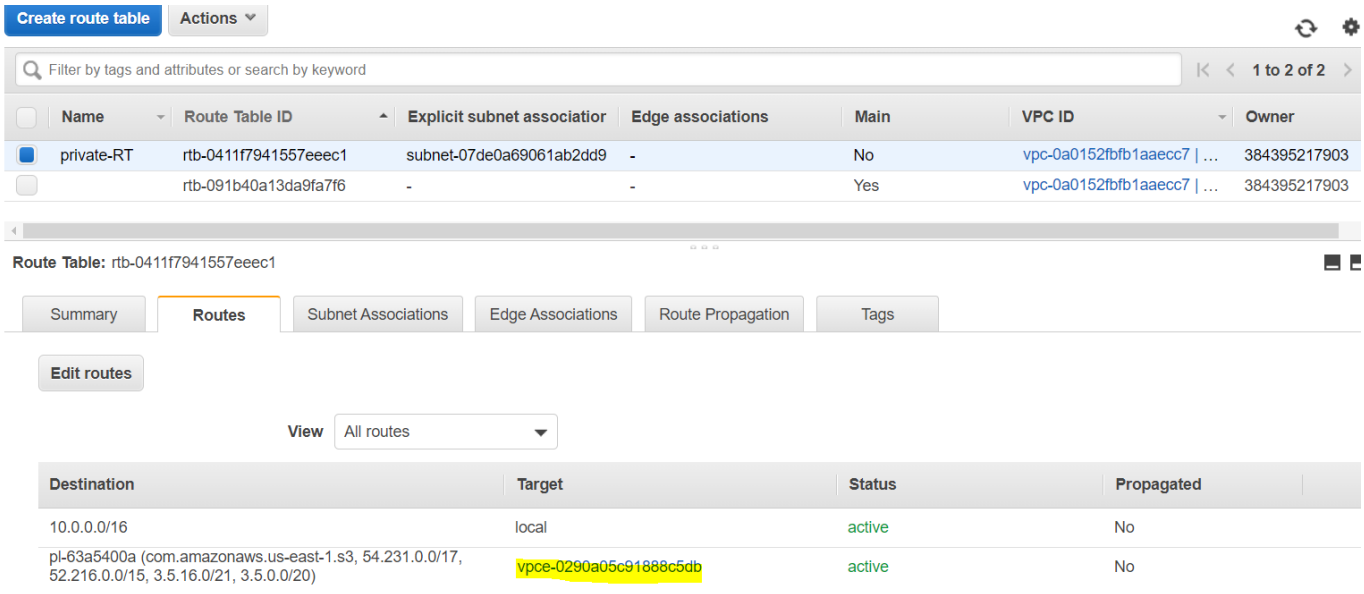
- Let us now navigate to the VPC . Click on endpoints and select create endpoint
- Keep the default option of **AWS service** as it is . Under service name let us select S3 .
- Select our VPC



- Select our private route table which is being used by private subnet
- Under policy select **Full access** , with this option we can restrict users to certain level of access but it is not recomended as it restricts users at subnet level itself. We can rather use IAM for the same



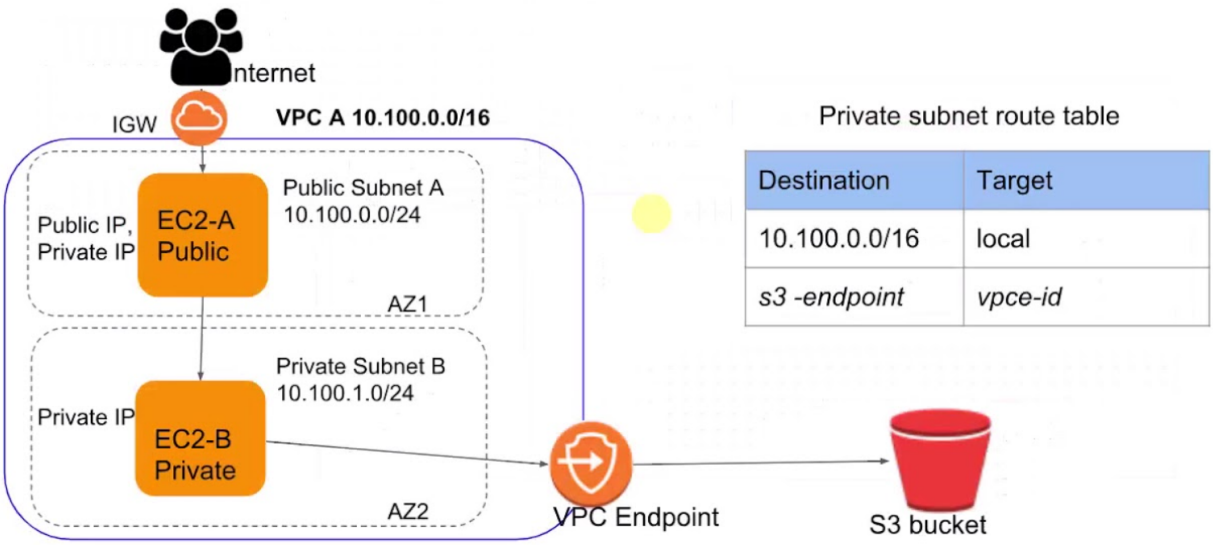
- Click on create endpoint
- We can go and check in pur private route table , and can observe an entry has been made in the routes automatically



- Let us test if we can access our S3

```
aws s3 ls
aws s3 ls bucketname
aws s3 cp s3://bucketname/objectname /home/ec2-user
```

- Above access should work now as we have amazons own internal network bringing us access to S3
- Even if we have NAT attached to the route table , the traffic to s3 will flow from the endpoint itself .



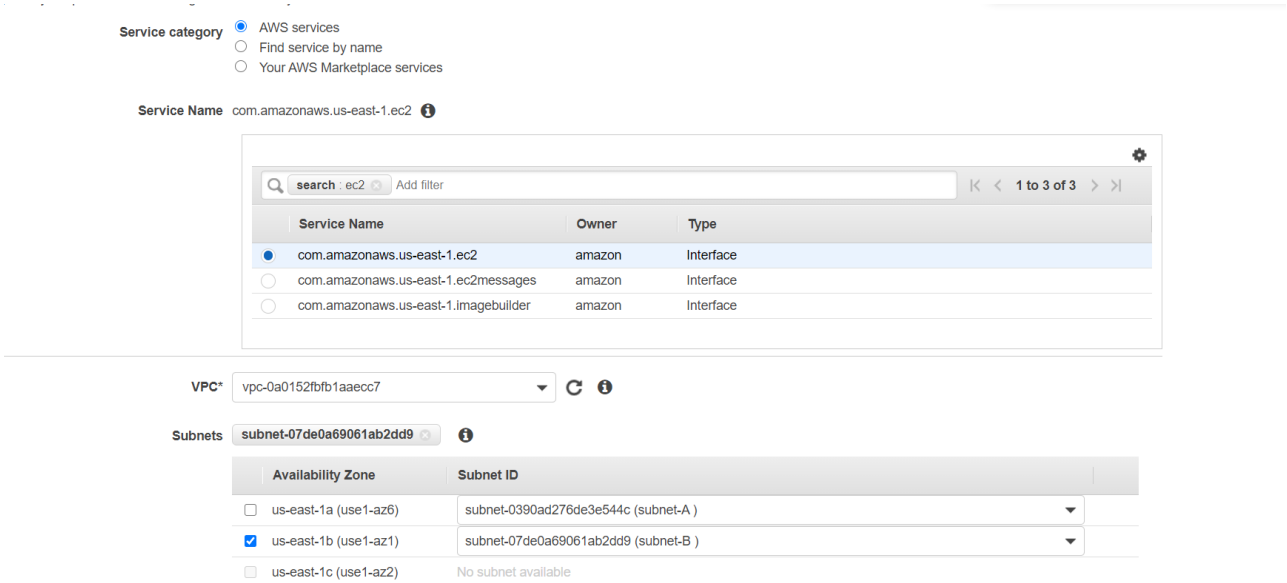
Interface endpoint

- An interface endpoint is an elastic network interface with a private IP address from the IP address range of your subnet that serves as an entry point for traffic destined to a supported service. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access services by using private IP addresses. AWS PrivateLink restricts all network traffic between your VPC and services to the Amazon network. You do not need an internet gateway, a NAT device, or a virtual private gateway.
- what is Elastic network interface

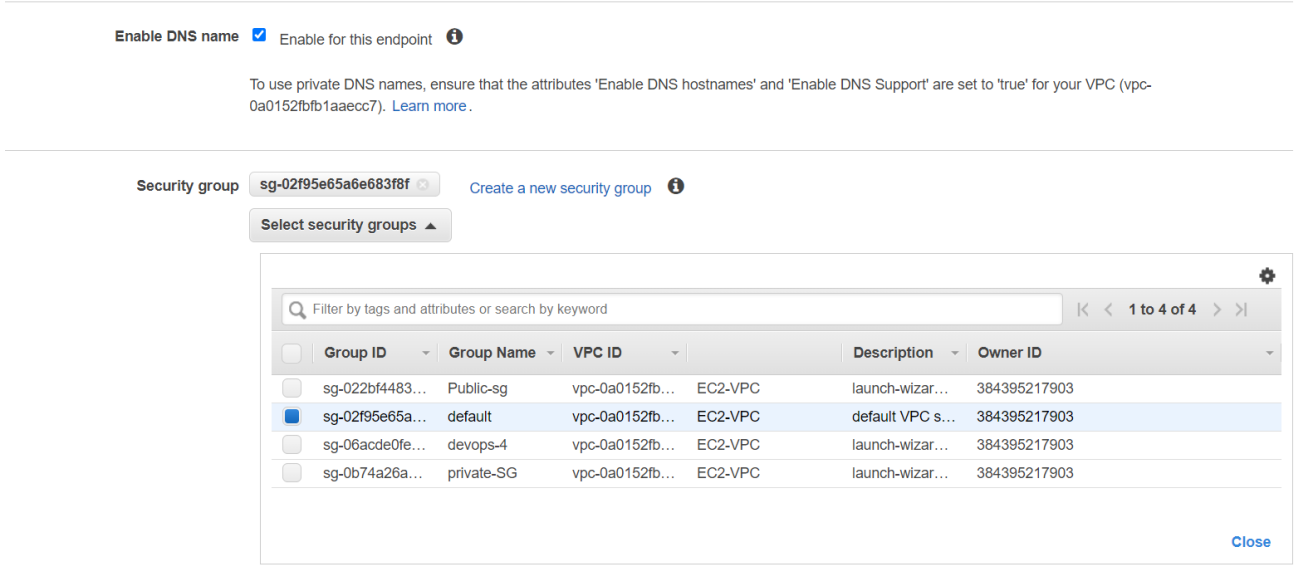
- You can create a network interface, attach it to an instance, detach it from an instance, and attach it to another instance. The attributes of a network interface follow it as it's attached or detached from an instance and reattached to another instance. When you move a network interface from one instance to another, network traffic is redirected to the new instance.
- Whenever we create an Instance , an eni gets created by default. we can also create additional eni and attach it to our instances or other components which use VPC
- ENIs can have their own ip address and security groups

Creating a interface endpoint

- For practice purposes we'll create a interface endpoint for EC2
- Similar to what we performed with S3 , you can use a privae subnet which does not have NAT attached to it so that we can confirm that the request is not going over internet
- Navigate to endpoint and click on create endpoint
- Keep the default option of **AWS service** as it is . Under service name let us select EC2 .
- Select our VPC and select the private subnet that we have
- An ENI wil be created in that subnet



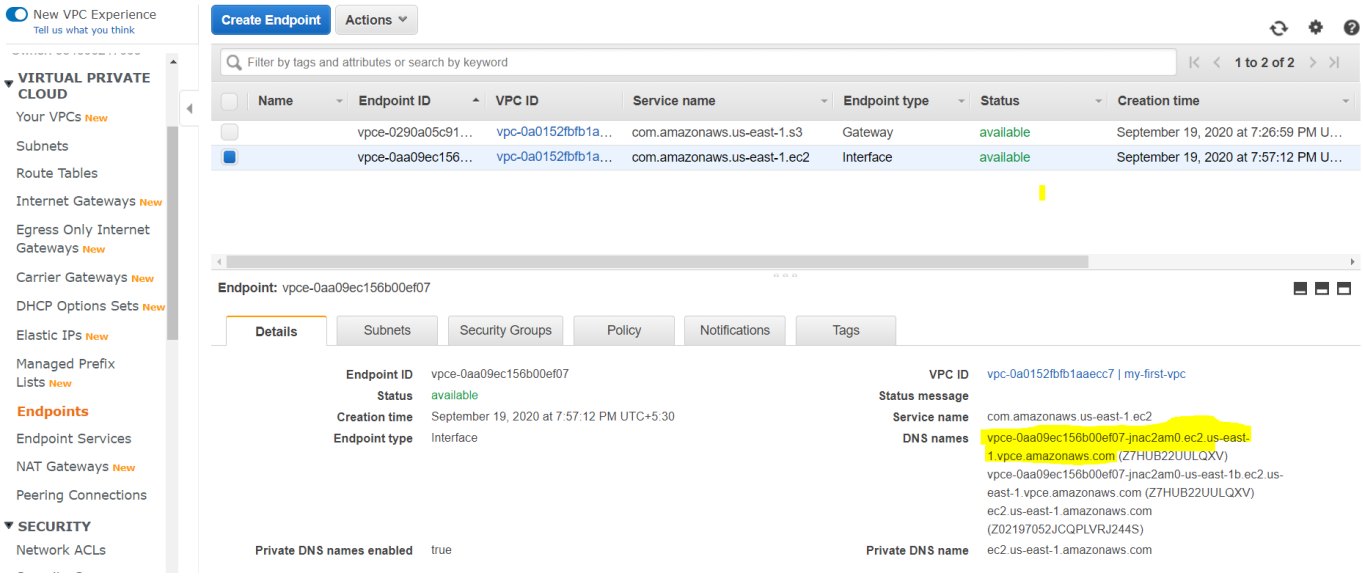
- Tick on enable dns hostname , this will help us resolve the endpoint later on while testing commands .
- Select a security group . This will be attached to our ENI that will get created . Make sure this security group allows traffic from our private EC2 instance
- Similar to S3 endpoint , lets keep the policy as full access



- Click on create endpoint
- Note
    - If you are facing error while selecting it , navigate to your VPC and make 'Enable DNS hostnames' and 'Enable DNS Support' to 'true'
    - You can find both options in actions once you select the VPC
- Once the endpoint is created, login to your private instance using jump host . Make sure this instance has access all the permissions to interact with Ec2 service
- Let us try below commands

```
aws ec2 describe-instances --region us-east-1
```

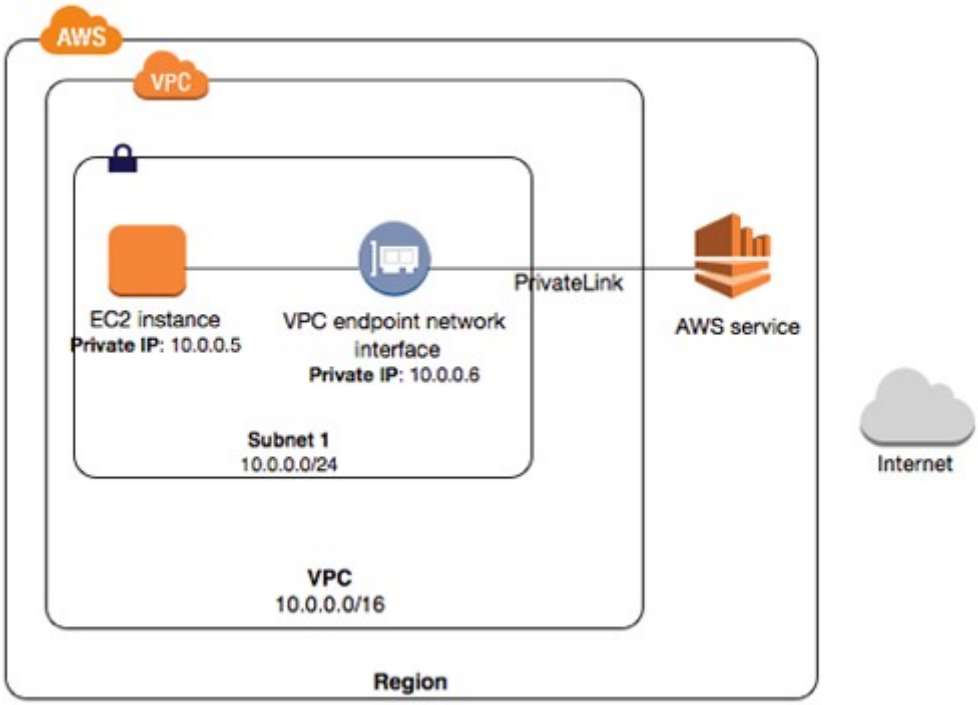
- Above command will not work as the private instance does not have access to internet
- In order to utilize the interface endpoint we need to include the endpoint-url in the command . We can find it as below



- Copy the dns name as highlighted above , it will be something like "vpce-0aa09ec156b00ef07-jnac2am0.ec2.us-east-1.vpce.amazonaws.com"
- now let us go back to the instance and try the command in below format

```
aws ec2 describe-instances --region us-east-1 --endpoint-url https://**vpce-0aa09ec156b00ef07-jnac2am0.ec2.us-east-1.vpce.amazonaws.com**
```

- As you can notice we entered the url that we copied in the argument "--endpoint-url " in the format of https://url
- Observe that now the access is working .
- Note
    - If you are not receiving any result , verify the security group attached to the endpoint interface . This security group should allow access from your private ec2 instance over https .



VPC peering

- VPC peering is the most common way to create a bridge between two VPCs on AWS
- This bridge allows two VPCs to connect to each other over a private network
- VPC peering is non transitive i.e. if VPC A has peering with VPC B , and VPC B has peerig with VPC C this does not mean VPC A has automatic peering with VPC C
- VPC peering can be done with VPCs in the same region , cross region or cross account as well .
- Pre-requisite to having VPC peering is the CIDR of the VPCs should not be overlapping with each other

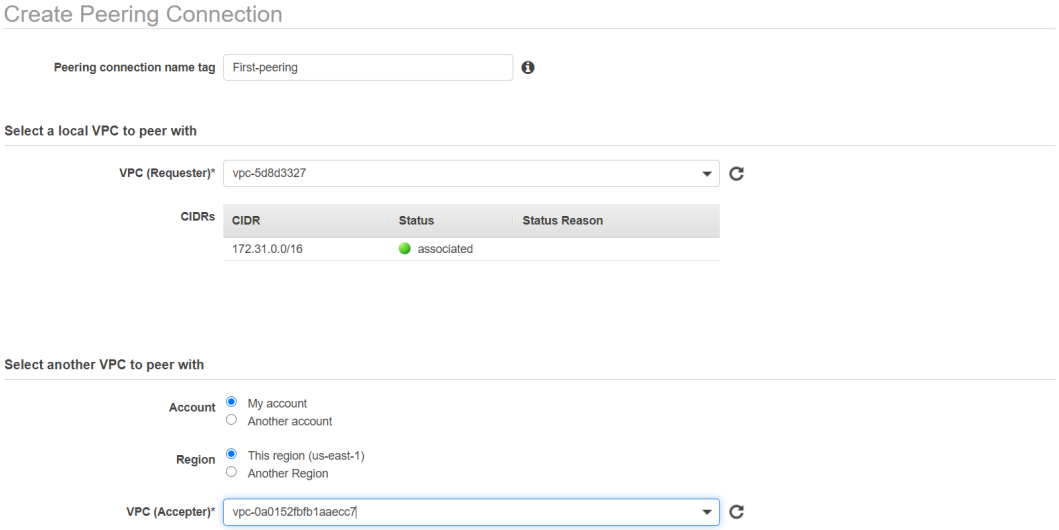
Creating VPC peering connection between 2 VPCs

- In order to test VPC peering , we need 2 vpcs which do not have overlapping CIDRs .

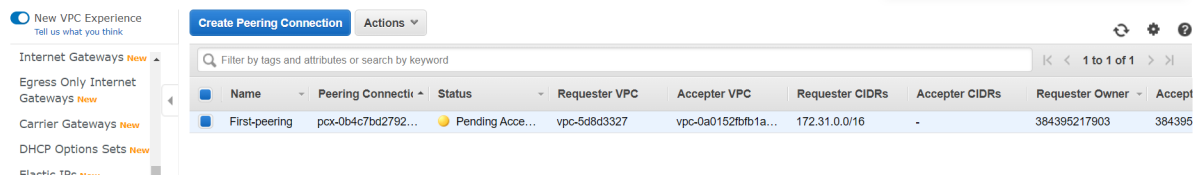
- Before we proceed to peering , in order to test our principal of peering , let us launch 2 instances in 2 seperate VPCs and try and ping each other using private ip. This will not be successful as there is no bridge. post peering try the same steps again

- There are 2 primary steps we need to perform in order to establish a peering connection . 1. Send peering request from requester VPC to accepter VPC , and accept that request 2. Enter the CIDR of VPC A in route table of VPC B . And smilarly CIDR of VPC B is to be entered in route table of VPC A.

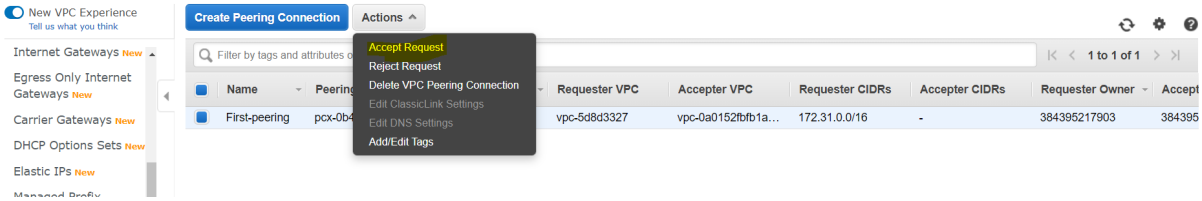
    1. Establishing the peering request
    - Navigate to VPC . Select peering connections from the left index
    - Click on create peering connection
    - Give an appropriate name to the peering request
    - First we have to select a VPC which will send the request . This has to be within the account and region from which you are trying to send the request .
    - While selecting the accepter VPC , we can chose VPC from other region or other aws account as well. As of now we'll stick to our own account and vpc . Select a VPC which does not have overlapping CIDR with the requester VPC



    - Click on create peering connection .
    - You will see that when you go to peering connections , you will be able to see the request in pending state



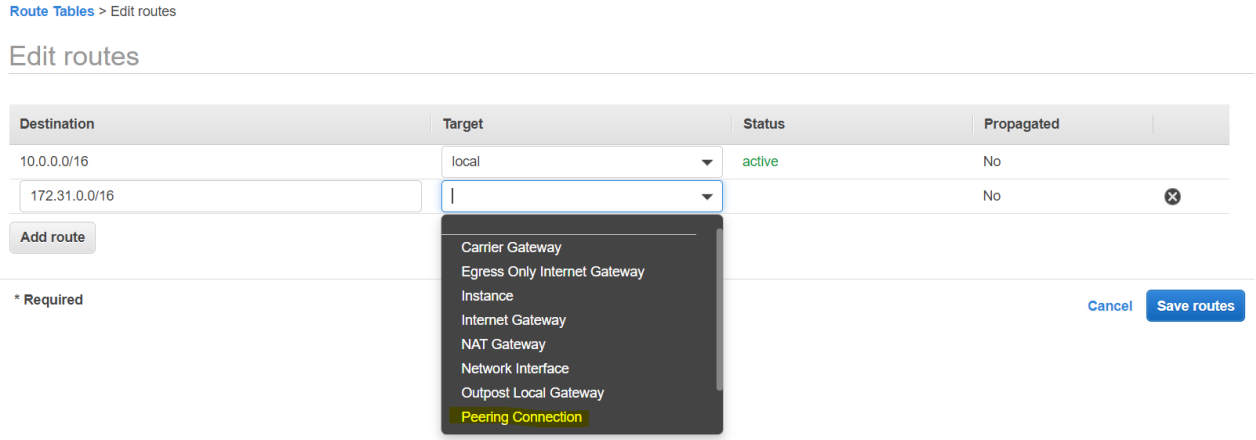
- Go to actions and click on accept



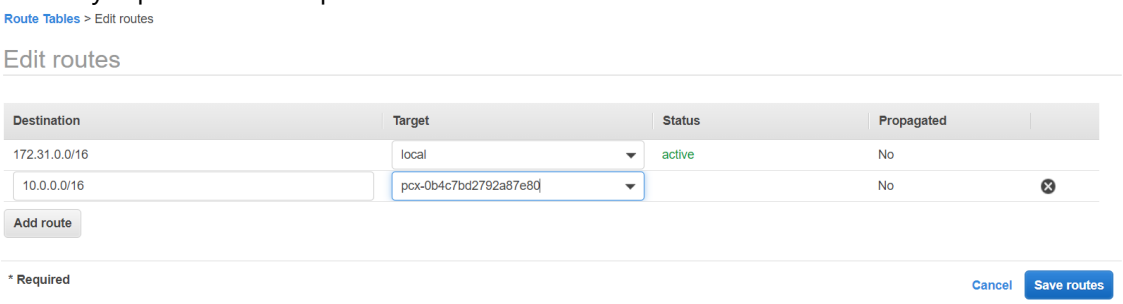
- Our first step is completed

2. Making entries in Route table

- As we know , route tables decide the route for traffic flow from out of the subnet
- Since now we want to travel to another network , we need to make an entry of the VPC CIDR of the VPC B under routes for route table of VPC A .
- this route table can be any route table that your instance is currently using . In most of the cases it is generally private route table



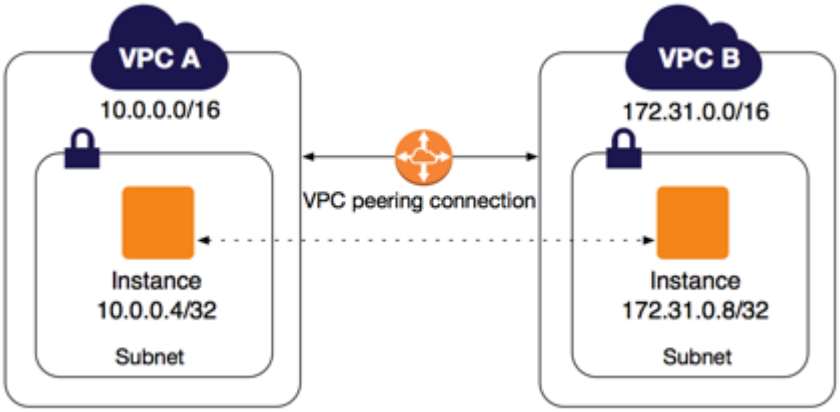
- As seen above , in destination enter CIDR of other VPC and in target select peering connection and select the peering connection id we just created .
- Similarly repeat the same process for other VPC



- Our 2nd step is completed
- In order to test if the VPC peering is successful , launch 2 instances in seperate VPCs
- Try and ping these instances over private ip
- Make sure security groups of these instances allow entry from private ip address of each other

```
ping 10.0.0.23
ping 172.0.0.4
```

- If the setup is correct , you will be able to ping and connect over private ip address/network
- Note
    - If the the connectivity is not working , check below things
        - Check if VPC peering has been accepted
        - CIDR of the VPC has been added in routes of respective route tables
        - Security groups allow connection over private ip



Points to consider

- VPC endpoint and peering are one of the most common features used in the industry
- It is always a best practice to utilize them for private subnets
- Post practice make sure to delete elastic ip , endpoints as these are chargable