



BLACKBUCKS INTERNSHIP REPORT

An Architecture of 6 **VPCS** with 6 **Instances** and
6 **Subnets** with **Peering Connections**
(6-TIER ARCHITECT)

SUBMITTED BY

Ms. DIVVELA DEVIKA RANI

RegdNo:21B91A5438

Ms. ELIKA LAVANYA

RegdNo:21B91A5442

Ms. AMBATI RAMA GAYATHRI

RegdNo:21B91A5407

UNDER THE GUIDANCE OF MR. AASHU DEV

B. Tech, AWS solution Architect (2x) certified,

AWS Academy Accredited Educator



BLACKBUCKS INTERNSHIP WORK

Title:

Connecting multi accounts using peering connections

(5.2-Tier Architecture)

Abstract:

The above Architecture is to create 6 VPCs with 6 Subnets and 6 Instances with required Peering Connections. And the creation of Instances is done by using the service Cloud Shell . And the termination of Instances is done by using Lambda function. The Architecture is done in two different accounts and two different regions (Mumbai, Stockholm).

Table of Contents

1. Introduction to Amazon web services (AWS)
2. Cloud Computing
3. What is AWS?
4. History of AWS
5. Why is AWS?
6. Applications of AWS
7. Companies using AWS
8. AWS global Infrastructure
9. List of AWS services
10. Implementation of the Architecture is to create 6 VPCs with 6 Subnets and 6 Instances With peering connections
 - 10.1. Introduction
 - 10.2. AWS services used
 - 10.3. Rough Architecture
 - 10.4. Final Architecture
11. Implementation of the project
 - 11.1. Service 1: VPC
 - Sub Service 1: peering connection
 - Sub Service 2: Endpoint
 - 11.2. Service 2: Cloud Shell
 - 11.3. Service 3: EC2
 - 11.4. Service 5: Lambda
 - 11.5. Service 6: IAM
 - 11.6. Conclusion

Introduction To AWS

AWS stands for Amazon Web Services, it needs no formal introduction, given its immense popularity. The leading cloud provider in the marketplace is Amazon Web Services. It provides over 170 AWS services to the developers so they can access them from anywhere at the time of need.

AWS has customers in over 190 countries worldwide, including 5000 ed-tech institutions and 2000 government organizations. Many companies like ESPN, Adobe, Twitter, Netflix, Facebook, BBC, etc., use AWS services.

For example, Adobe creates and updates software without depending upon the IT teams. It uses its services by offering multi-terabyte operating environments for its clients. By deploying its services with Amazon services, Adobe integrated and operated its software in a simple manner.

An analogy: think of electricity services....

You simply plug into a vast electrical grid managed by experts to get a low cost , reliable power supply – available to you with much greater efficiency than you could generate on your own.

Power is a utility service – available to you on-demand and pay only for what you use.



What is Cloud Computing?

Cloud Computing is the delivery of online services (such as servers, databases, software) to users. With the help of cloud computing, storing data on local machines is not required. It helps you access data from a remote server. Moreover, it is also used to store and access data from anywhere across the world.

What is AWS?

AWS Meaning: The Amazon Web Services (AWS) platform provides more than 200 fully featured services from data centers located all over the world, and is the world's most comprehensive cloud platform.

Amazon web service is an online platform that provides scalable and cost-effective cloud computing solutions.

AWS is a broadly adopted cloud platform that offers several on-demand operations like compute power, database storage, content delivery, etc., to help corporates scale and grow.

History of AWS

- In the year 2002 - AWS services were launched
- In the year 2006- AWS cloud products were launched
- In the year 2012 - AWS had its first customer event
- In the year 2015- AWS achieved \$4.6 billion
- In the year 2016- Surpassed the \$10 billion revenue target
- In the year 2016- AWS snowball and AWS snowmobile were launched
- In the year 2019- Released approximately 100 cloud services

Why AWS?

There are several reasons why AWS has become a popular choice for cloud computing:

❖ Easy to use

AWS is designed to allow application providers, ISVs, and vendors to quickly and securely host your applications – whether an existing application or a new SaaS-based application. You can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform.

❖ Flexible

AWS enables you to select the operating system, programming language, web application platform, database, and other services you need. With AWS, you receive a virtual environment that lets you load the software and services your application requires. This eases the migration process for existing applications while preserving options for building new solutions.

❖ Cost-Effective

You pay only for the compute power, storage, and other resources you use, with no long-term contracts or up-front commitments. For more information on comparing the costs of other hosting alternatives with AWS, see the AWS Economics Center.

❖ Reliable

With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure, the virtual backbone of Amazon .com 's multi – billion dollar online business that has been honed for over a decade.

❖ Scalable and high-performance

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

❖ Secure

AWS utilizes an end-to-end approach to secure and harden our infrastructure, including physical, operational, and software measures. For more information, see the AWS Security Center.

❖ Integration and Ecosystem

AWS integrates well with various third-party tools, technologies, and services. It offers extensive APIs and SDKs, making it easier to integrate AWS services into existing applications or build new solutions from scratch. The AWS ecosystem also includes a vibrant community, documentation, training resources, and support services, facilitating development and troubleshooting.

These factors, among others, contribute to the popularity and success of AWS as a cloud computing provider. However, it's important to note that the choice of cloud provider should be based on your specific needs, requirements, and preferences. It's worth evaluating multiple cloud platforms to determine the best fit for your organization.

Advantages of AWS:

1. AWS provides a user-friendly programming model, architecture, database as well as operating system that has been already known to employers.
2. AWS is a very cost-effective service. There is no such thing as long-term commitments for anything you would like to purchase.
3. It offers billing and management for the centralized sector, hybrid computing, and fast installation or removal of your application in any location with few clicks.
4. There is no need to pay extra money on running data servers by AWS.

- AWS offers a total ownership cost at very reasonable rates in comparison to other private cloud servers.

Disadvantages of AWS:

- AWS has supportive paid packages for intensive or immediate response. Thus, users might need to pay extra money for that.
- There might be some cloud computing problems in AWS especially when you move to a cloud Server such as backup protection, downtime, and some limited control.
- From region to region, AWS sets some default limitations on resources such as volumes, images, or snapshots.
- If there is a sudden change in your hardware system, the application on the cloud might not offer great performance.

Applications of AWS

The most common applications of AWS are storage and backup, websites, gaming, mobile, web, and social media applications. Some of the most crucial applications in detail are as follows:

Storage and Backup

One of the reasons why many businesses use AWS is because it offers multiple types of storage to choose from and is easily accessible as well. It can be used for storage and file indexing as well as to run critical business applications.

Websites

Businesses can host their websites on the AWS cloud, similar to other web applications.

Gaming

There is a lot of computing power needed to run gaming applications. AWS makes it easier to provide the best online gaming experience to gamers across the world.

Mobile, Web and Social Applications

A feature that separates AWS from other cloud services is its capability to launch and scale mobile, e-commerce, and SaaS applications. API-driven code on AWS can enable companies to build uncompromisingly scalable applications without requiring any OS and other systems.

Big Data Management and Analytics (Application)

- Amazon Elastic Map Reduced to process large amounts of data via the Hadoop framework.
- Amazon Kinesis to analyze and process the streaming data.
- AWS Glue to handle, extract, transform and load jobs.
- Amazon Elasticsearch Service to enable a team to perform log analysis, and tool monitoring with the help of the open source tool ,Elastic-search.
- Amazon Athena to query data.
- Amazon Quick Sight to visualize data.

Artificial Intelligence

- Amazon Lex to offer voice and text chatbot technology.
- Amazon Polly to translate text-to-speech translation such as Alexa Voice Services and echo devices.
- Amazon Rekognition to analyze the image and face.

Messages and Notifications

- Amazon Simple Notification Service (SNS) for effective business or core communication.
- Amazon Simple Email Service (SES) to receive or send emails for IT professionals and marketers.
- Amazon Simple Queue Service (SQS) to enable businesses to subscribe or publish messages to end users.

Augmented Reality and Virtual Reality

- Amazon Sumerian service enables users to make the use of AR and VR development tools to offer 3D web applications, E-commerce & sales applications, Marketing, Online education, Manufacturing, Training simulations, and Gaming.

Game Development

- AWS game development tools are used by large game development companies that offer developer back-end services, analytics, and various developer tools.
- AWS allows developers to host game data as well as store the data to analyze the gamer's performance and develop the game accordingly.

Internet of Things

- AWS IoT service offers a back-end platform to manage IoT devices as well as data ingestion to database services and AWS storage.

- AWS IoT Button offers limited IoT functionality to hardware.
- AWS Greengrass offers AWS computing for IoT device installation.

Companies Using AWS

Whether it's technology giants, startups, government, food manufacturers or retail organizations, there are so many companies across the world using AWS to develop, deploy and host applications. According to Amazon, the number of active AWS users exceeds 1,000,000. Here is a list of companies using AWS:

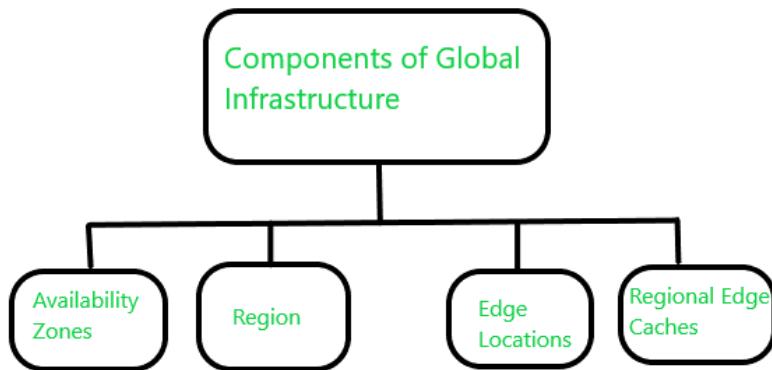
- Netflix
- Intuit
- Coinbase
- Finra
- Johnson & Johnson
- Capital One
- Adobe
- Airbnb
- AOL
- Hitachi

AWS Global Infrastructure

- AWS is a cloud computing platform which is globally available.

- Global infrastructure is a region around the world in which AWS is based. Global infrastructure is a bunch of high-level IT services which is shown below:
- AWS is available in 19 regions, and 57 availability zones in December 2018 and 5 more regions 15 more availability zones for 2019.

The following are the components that make up the AWS infrastructure:



The infrastructure of AWS contains regions, availability zones, data centers, edge locations, or points of presence.

AWS Regions

AWS region is a cluster of data centers. It has regions all around the world. Regions name can be like “us-east-1”, “EU-west-1” etc. Most AWS services are region-scoped.

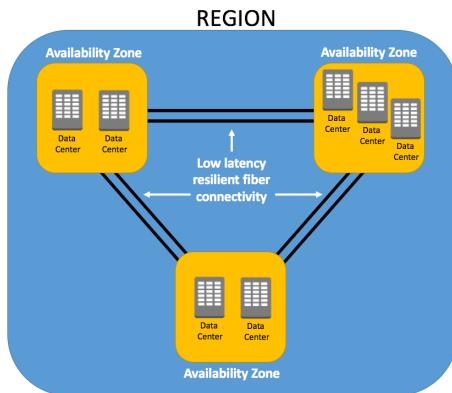
If you want to launch a new application, then you should consider the below parameters to choose a region:

- **Compliance with data governance and legal requirements:** Data never leaves a region without your explicit permission.
- **Proximity to customers:** Reduced latency.

US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
<hr/>	
Africa (Cape Town)	af-south-1
<hr/>	
Asia Pacific (Hong Kong)	ap-east-1
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2

- **Compliance with data governance and legal requirements:** Data never leaves a region without your explicit permission.
- **Proximity to customers:** Reduced latency.

AWS Availability Zones



Each region has many availability zones (usually 3, min is 2, max is 6). Example: ap-southeast-2a, ap-southeast-2b, ap-southeast-2c etc. Each availability zone (AZ) is one or more discrete data centers with redundant power, networking, and connectivity. They're separate from each other so that they're isolated from disasters. They're connected with high bandwidth, ultra-low latency networking.

AWS Points of Presence (Edge Locations)

- Edge locations are the endpoints for AWS used for caching content.
- Edge locations consist of CloudFront and Amazon's Content Delivery Network (CDN).
- Edge locations are more than regions. Currently, there are over 150 edge locations.
 - Edge location is not a region but a small location that AWS have. It is used for caching the content.
 - Edge locations are mainly located in most of the major cities to distribute the content to end users with reduced latency.
 - For example, some user accesses your website from Singapore; then this request would be redirected to the edge location closest to Singapore where cached data can be read.

Regional Edge Cache

- AWS announced a new type of edge location in November 2016, known as a Regional Edge Cache.
- Regional Edge cache lies between CloudFront Origin servers and the edge locations.
 - A regional edge cache has a larger cache than an individual edge location
 - Data is removed from the cache at the edge location while the data is retained at the Regional Edge Caches.
- When the user requests the data, then data is no longer available at the edge location. Therefore, the edge location retrieves the cached data from the regional edge cache instead of the Origin servers that have high latency.

List Of AWS Services:

❖ Amazon EC2 (Elastic Cloud Compute)



Amazon EC2 is the fastest cloud computing service provided by AWS. It offers virtual, secure, reliable, and resizable servers for any workload. Through this service, it becomes easy for developers to access resources and also facilitates web-scale cloud computing. This comes with the best suitable processors, networking facilities, and storage systems. Developers can quickly and dynamically scale capacities as per business needs. It has over 500 instances and you can also choose the latest processor, operating system, storage, and networking to help you choose according to the needs of the business. Also, with Amazon EC2, you only have to pay for what you use, and also as per the time period, scale with amazon EC2 auto-scaling has optimal storage and can optimize CPU configurations.

- **Amazon VPC (Virtual Private Cloud)**



Another AWS service is Amazon VPC (Virtual Private Cloud) which is an isolated cloud resource, it enables you to set up an isolated section where you can deploy AWS resources at scale in a virtual environment. This service is responsible to control the virtual networking environment such as resource placement, security, and connectivity. Security can be improved by applying rules for outbound and inbound connections. Also, it detects anomalies in the patterns, troubleshoots network connections, prevents data leakage, and handles configuration issues. Using VPC, you get complete access to control the environment, such as choosing IP address, subset creation, and route table arrangement.

- **Amazon S3 (Simple Storage Service)**



With Amazon, it has become easy to store data anytime, anywhere. Amazon S3 (Simple Storage Service), one of the best services provided by AWS is an object storage service offering scalability, availability, security, and high-performing. You can also retrieve data, data here is stored in “storage classes” where there’s no requirement of extra investment and you can also manage it well. Amazon S3 is the perfect fit for big businesses where a large amount of data is managed for varied purposes. It comes with handling any volume of data with its robust access controls, and replication tools prevent accidental deletion, and also maintains data version controls.

- **Amazon RDS (Relational Database Services)**



Amazon RDS (Relational Database Service) is another service provided by AWS which is a managed database for PostgreSQL, MariaDB, MySQL, and Oracle.

Using Amazon RDS, you can set up, operate, and scale databases in the cloud. It provides high performance by automating the tasks like database setup, hardware provisioning, patching, and backups. Also, it helps in cost optimization by providing high availability, compatibility, and security for resources, and there’s no need to install and manage the database software during its usage. As per the need, you can easily choose any engine out of 15+ engines some of them being MySQL, PostgreSQL, Oracle, etc. It is a highly secure and easily available AWS service.

- **Amazon IAM (Identity and Access Management)**



Amazon IAM (Identity and Access Management) allows users to securely access and manage resources. To achieve complete access to the tools and resources provided by AWS, AWS IAM is the best AWS service. It gives you the right to have control over who has authorization (signed in) and authentication (has permissions) access to the resources. It comes with attribute-based access control which helps you to create separate permissions on the basis of the user's attributes such as job role, department, etc. Through this, you can allow or deny access given to users. AWS IAM has complete access or is a central manager for refining permissions across AWS. He /She handles who can access what.

- **Amazon EBS (Elastic Block Store)**



Amazon EBS is the next service provided by AWS which is a block storage solution specifically designed for Amazon EC2. Throughout a workload of any size, Amazon EBS helps to securely manage transactions. You can handle diverse workloads, be it relational, non-relational, or business applications. You get to choose between five different volume types so as to achieve effectiveness and optimum cost. It helps to resize workloads for big data analytics engines such as Hadoop and Spark. Its lifecycle management creates policies to create and manage backups effectively. It supports high-performance scaling workloads such as Microsoft, and SAP products.

- **Amazon Lambda**



Another promising service by AWS is Amazon Lambda which is a serverless and event-driven computing service that lets you run code for virtual applications or backend services automatically. You need to worry about servers and clusters when working with solutions using Amazon Lambda. It is also cost-effective where you have to only pay for the services you use. As a user, your responsibility is to just upload the code and Lambda handles the rest. Using Lambda, you get precise software scaling and extensive availability. With hundreds to thousands of workloads per second, AWS Lambda responsibly handles code execution requests. It is one of the best services provided by AWS for developers.

- **Amazon EFS (Elastic File System)**



Amazon EFS (Elastic File System) is a simple and serverless system where you can create and configure file systems without provisioning, deploying, patching, and maintaining. It is a scalable NFS file system made for use in AWS cloud services and on-premises resources. Also, it has no minimum fee or setup charge. You pay for the storage you use such as –

- for provisioned throughput
- automatically expand and shrink as per the addition and removal of files
- read and write access to data stored in Infrequent Access storage classes

It is a scalable service where you can scale up to petabytes without thinking about the performance of the application.

- **Amazon SNS (Simple Notification Service)**



It is a web service provided by AWS, which is a fully managed solution for messaging having low-cost infrastructure. It is used for bulk message delivery and direct chat with the customers through system-to-system or app-to-person between decoupled microservice apps. It is used to easily set up, operate, and send notifications from the cloud. It is a messaging service between Application to Application (A2A) and Application to Person (A2Person), and sends notifications in two ways – A2A and A2P. A2P allows many-to-many messaging between microservices, distributed systems, and event-driven serverless applications, allowing you to send messages to customers with SMS texts, email, and push notifications.

- **Amazon Auto-Scaling**



Amazon Auto-Scaling is one of the best services provided by AWS, it examines the applications and adjusts its capacity accordingly which is to maintain predictable performance at the lowest possible cost. It becomes easy to scale up applications for multiple resources across services in seconds. To meet the demands of the business, it scales computing capacity and this can be achieved by adding or removing EC2 instances automatically. There are two types of scaling – dynamic (presently changing demands) and predictive (response based on predictions) scaling. It can be used with Amazon EC2 Auto Scaling to dynamically scale your Amazon EC2 instances and also it receives the right resource at the right time.

- **Dynamo DB**



DynamoDB is a serverless, document database key-value NoSQL database that is designed to run high-performance applications. It can manage up to 10 trillion requests on a daily basis and support thresholds of more than 20 million requests per second. DynamoDB has built-in security with a fully-managed multi-master, multi-region, durable database, and in-memory archiving for web-scale applications. It has in-built tools which are used to generate actionable insights, useful analytics, and monitor traffic trends. It offers built-in security, continuous backups, automated multi-region replication, data import and export, and in-memory caching.

- **Amazon Elastic Beanstalk**



Amazon Elastic Beanstalk is an AWS service used for deployment and scaling web applications developed using Java, PHP, Python, Docker, etc. It supports running and managing web applications. You just need to upload your code and the deployment part is handled by Elastic Beanstalk (from capacity provisioning, load balancing, and auto-scaling to application health monitoring). It is the best service for developers since it takes care of the servers, load balancers, and firewalls. Also, you can have control over AWS assets and the other resources required for the application. You get the benefit of paying for what you use, thus maintaining cost-effectiveness.

- **Amazon Cloud watch**



Amazon CloudWatch detects uncommon changes in the environment, set an alert, troubleshoots issues, and take automated actions. With this, you can track the complete stack, and use logs, alarms, and events data to take actions and thereby focusing on building the application, resulting in the growth of the business. It is the best service designed for developers, DevOps engineers, site reliability engineers, and IT managers. With Amazon CloudWatch, you can detect anomalies in the environment. With this single platform, you can monitor all AWS resources and applications quickly. It monitors application performance and optimizes resources.

- **Amazon Cloud Shell**



Cloud Shell is a lightweight, award-winning infrastructure automation tool. Cloud Shell is offered on the AWS Marketplace to accelerate and simplify environment provisioning by enabling your teams to create self-service, on-demand replicas of full-stack infrastructure environments for AWS and hybrid cloud in just one click.

Innovators everywhere use Cloud Shell solutions to:

- Accelerate time to market up to 50% faster by automating the set-up and tear-down of complex hybrid environments on AWS.

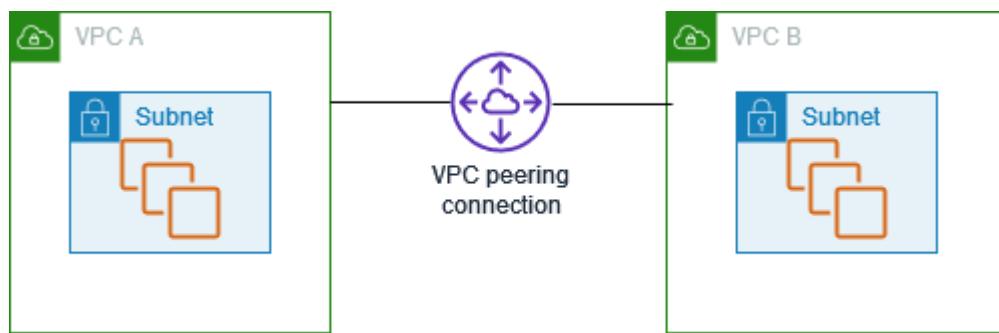
- Lift productivity by providing a self-service catalog of standardized environment blueprints used for development and sales demos.
- Improve utilization of cloud resources to attain 3-5x ROI.

Implementation of the Architecture of 6 VPCs with Peering Connection(6TIER-ARCHITECT):

Introduction:

Peering Connection:

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account. The VPCs can be in different Regions (also known as an inter-Region VPC peering connection).



Use of IAM in Lambda Function:

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Lambda resources. IAM is an AWS service that you can use with no additional charge.

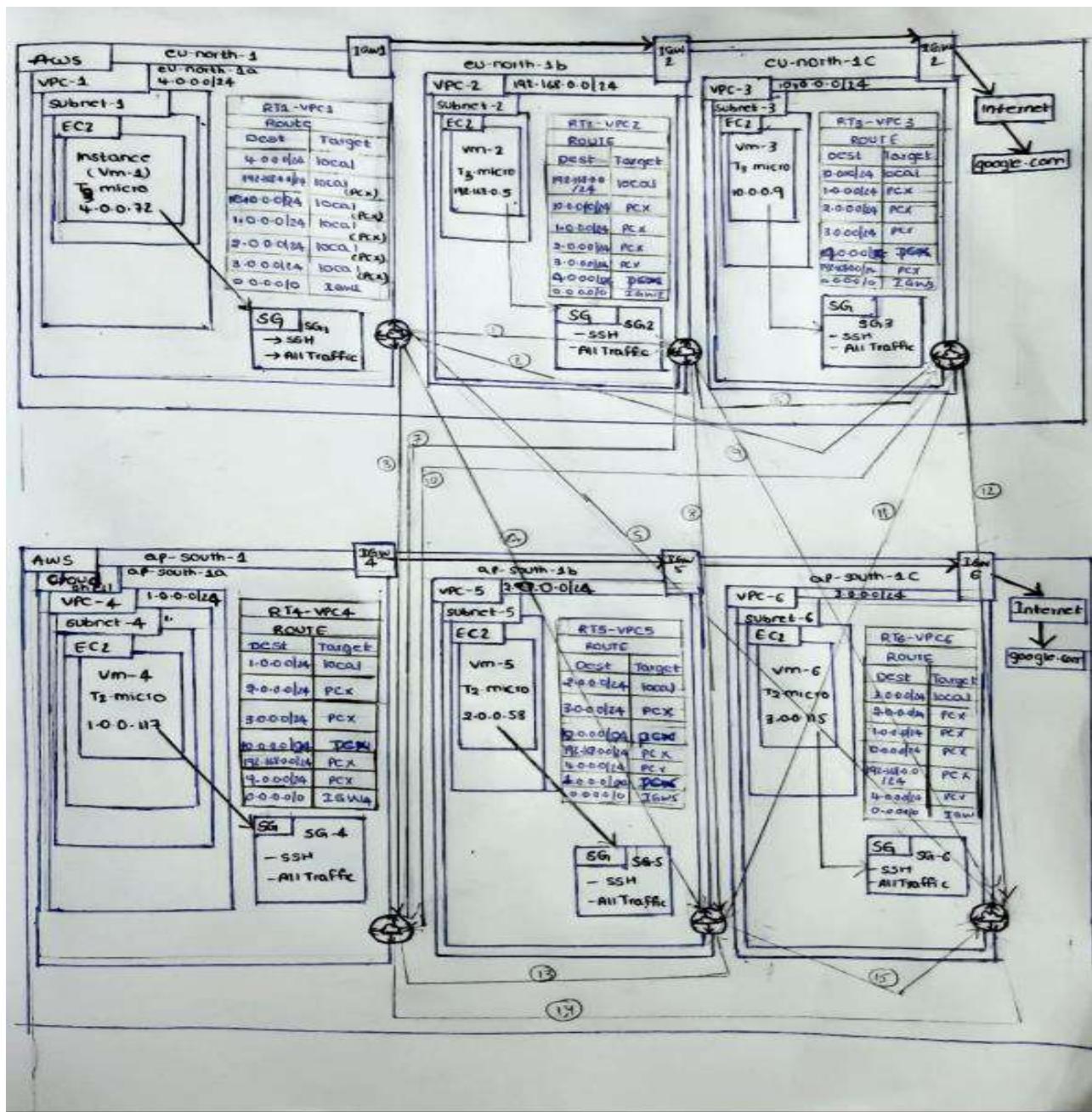
CLOUD SHELL:

AWS Cloud Shell is a browser-based shell that makes it easier to securely manage, explore, and interact with your AWS resources. Cloud Shell is pre-authenticated with your console credentials. Whilst Cloud Shell permits outbound connections to the internet, it does not support inbound connections. Furthermore, it does not allow connections to resources within private VPC subnets.

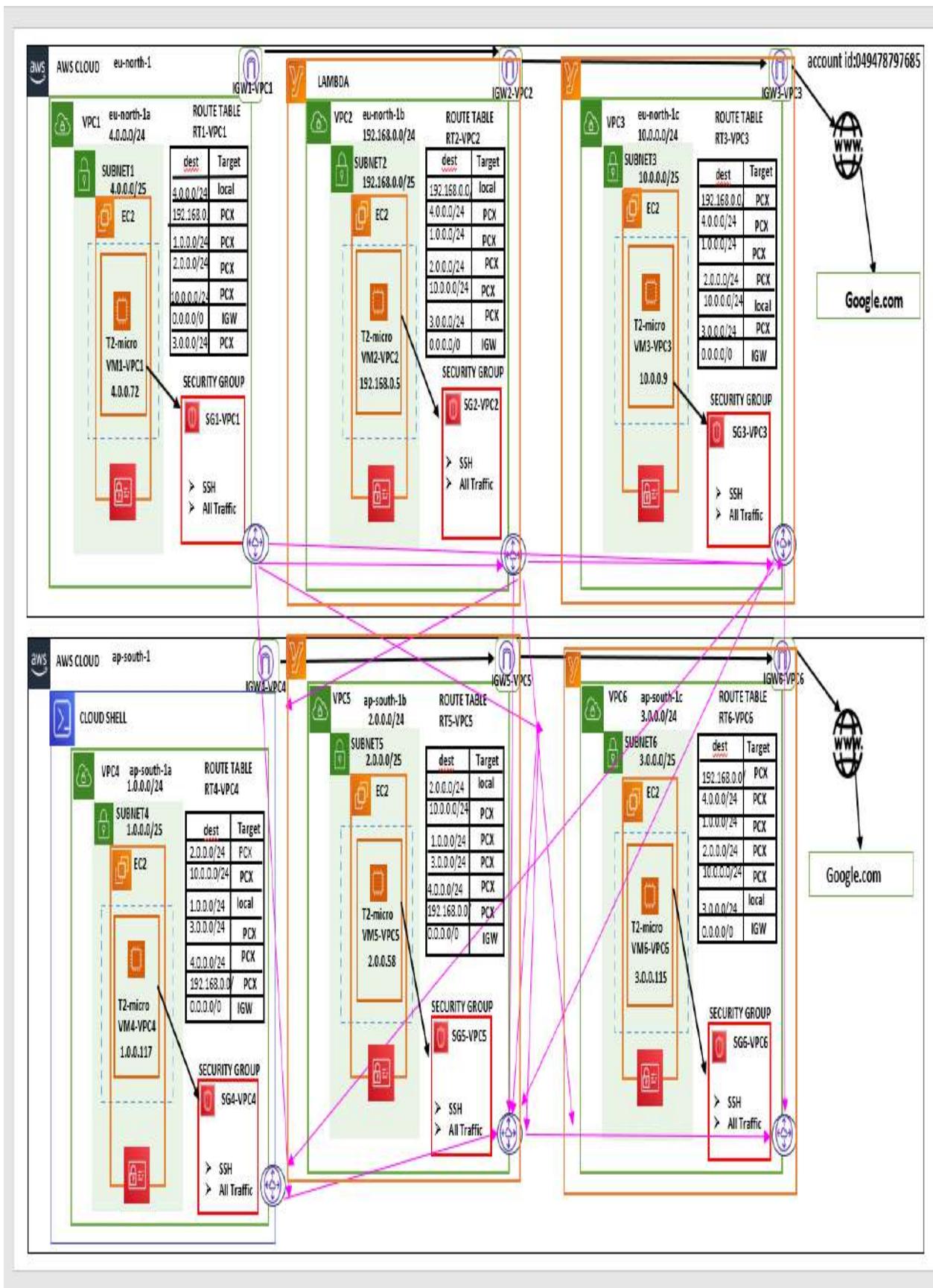
Services used:

- 1.VPC (virtual private cloud)
- 2.EC2 (Elastic compute cloud)
- 3.Lambda function
- 4.IAM (Identity and access management)
- 5.Cloud shell

ROUGH ARCHITECT:



FINAL ARCHITECT:



Workflow of the Project:

- Create a 3VPCs (VPC1 , VPC2 ,VPC3) in One account and edit the route tables associated with it. Create a subnets and name it as public subnet. Edit the corresponding subnet associations, create an internet gateways and attach it to custom VPCs. Now launch an ec2 instance with created VPCs.
- Create a 3VPCs (VPC4 , VPC5 ,VPC6) in another account and edit the route tables associated with it. Create a subnets and name it as public subnet. Edit the corresponding subnet associations, create an internet gateways and attach it to custom VPCs. Now launch an ec2 instance with created VPCs.
- **POSSIBLE PEERING CONNECTIONS**

ACCOUNT1 (eu-north-1)	ACCOUNT2 (ap-south-1)	Different accounts and Cross Regions
VPC1-VPC2	VPC4-VPC5	VPC1-VPC4
VPC1-VPC3	VPC4-VPC6	VPC1-VPC5
VPC2-VPC3	VPC5-VPC6	VPC1-VPC6
		VPC2-VPC4
		VPC2-VPC5
		VPC2-VPC6
		VPC3-VPC4
		VPC3-VPC5
		VPC3-VPC6

Implementation of the Project

Service 1: VPC



To create a VPC with the name "VPC1," follow these steps:

- Sign into the AWS Management Console.
- Open the Virtual Private Cloud Service.
- Click on the "Create VPC" option.
- Give the VPC name as `vpcl` and CIDR value as `10.0.0.0/24`.
- Click on create VPC.
- And create `vpcl-1`.

The screenshot shows the AWS VPC Dashboard. At the top, there are navigation links for Services, Search, and various AWS services like EC2, S3, IAM, Billing, RDS, DynamoDB, Simple Notification Service, CloudShell, Lambda, AWS Auto Scaling, and EFS. Below the navigation bar, the main area is titled "VPC dashboard". It features a "Create VPC" button and a "Launch EC2 Instances" button. A note says "Note: Your Instances will launch in the Asia Pacific region." On the left, there's a sidebar with sections for "Virtual private cloud" (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections) and "Security" (Network ACLs, Security groups). The main content area is titled "Resources by Region" and shows a table of resources across the Asia Pacific region. The table includes:

Resource Type	Count	Region
VPCs	1	Asia Pacific
Subnets	3	Asia Pacific
Route Tables	1	Asia Pacific
Internet Gateways	1	Asia Pacific
Egress-only Internet Gateways	0	Asia Pacific
DHCP option sets	1	Asia Pacific
Elastic IPs	0	Asia Pacific
Managed prefix lists	-	-
Endpoints	-	-
Endpoint services	-	-
NAT gateways	-	-
Peering connections	-	-
NAT Gateways	0	Asia Pacific
VPC Peering Connections	0	Asia Pacific
Network ACLs	1	Asia Pacific
Security Groups	6	Asia Pacific
Customer Gateways	0	Asia Pacific
Virtual Private Gateways	0	Asia Pacific
Site-to-Site VPN Connections	0	Asia Pacific

The screenshot shows the AWS VPC Management Console interface. The top navigation bar includes tabs for Instances, EC2 Management Console, Your VPCs (VPC Management), AWS CloudShell, and Subnets | VPC Management Console. The main content area is titled "Your VPCs (1)" and displays a table with one row of data:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP o
vpc-0b6bcfb0c357b11ff	Available	172.31.0.0/16	-	-	opt-0a

The left sidebar contains navigation links for Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only Internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections) and Security (Network ACLs). The bottom status bar shows the date and time (22-07-2023, 14:38), weather (32°C, Mostly cloudy), and system icons.

The screenshot shows the "Create VPC" wizard in the AWS VPC Management Console. The top navigation bar includes tabs for Instances, EC2 Management Console, VPC Management Console, AWS CloudShell, and Subnets | VPC Management Console. The main content area is titled "Create VPC" and displays the "VPC settings" step.

VPC settings

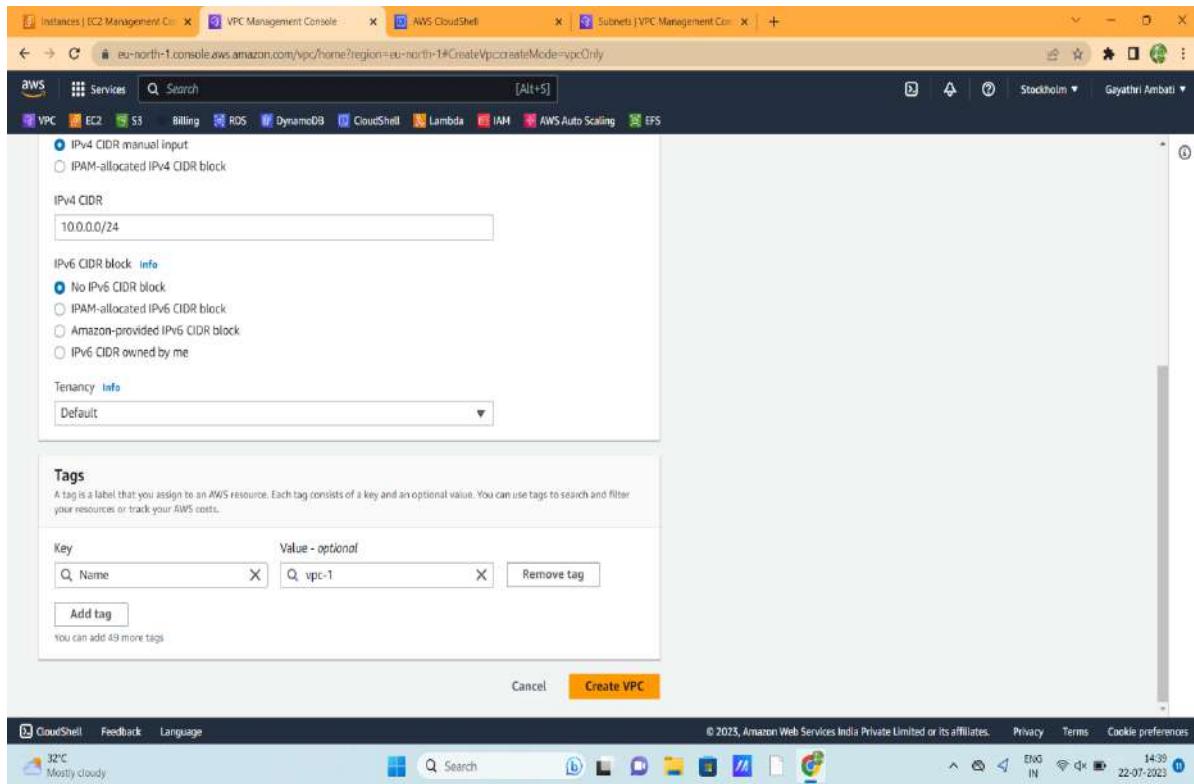
Resources to create: VPC only VPC and more

Name tag - optional: Creates a tag with a key of 'Name' and a value that you specify.
Name tag value:

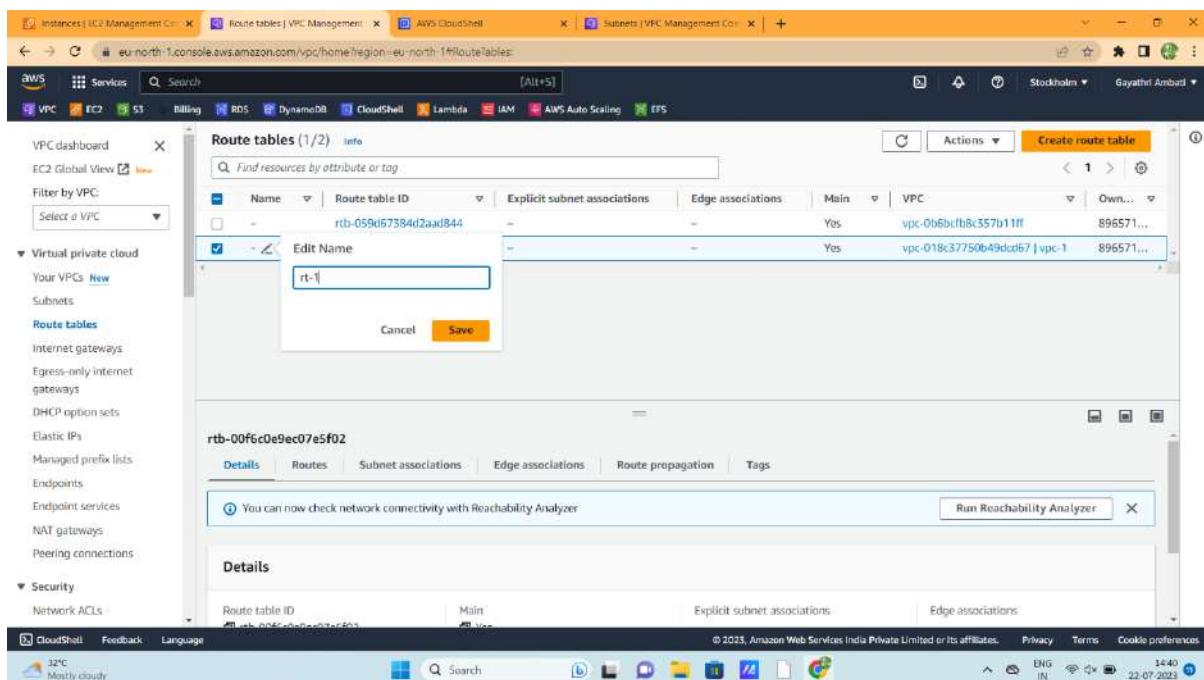
IPv4 CIDR block: IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block
IPv4 CIDR:

IPv6 CIDR block: No IPv6 CIDR block IPAM-allocated IPv6 CIDR block Amazon-provided IPv6 CIDR block
IPv6 CIDR:

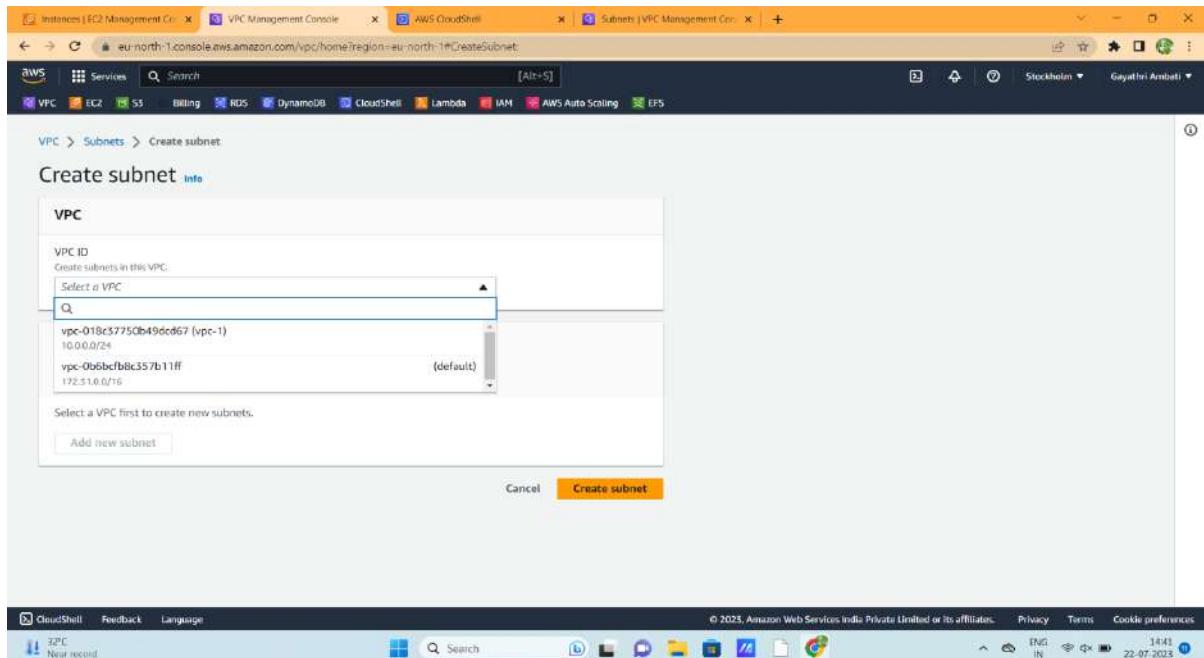
The bottom status bar shows the date and time (22-07-2023, 14:39), weather (32°C, Mostly cloudy), and system icons.



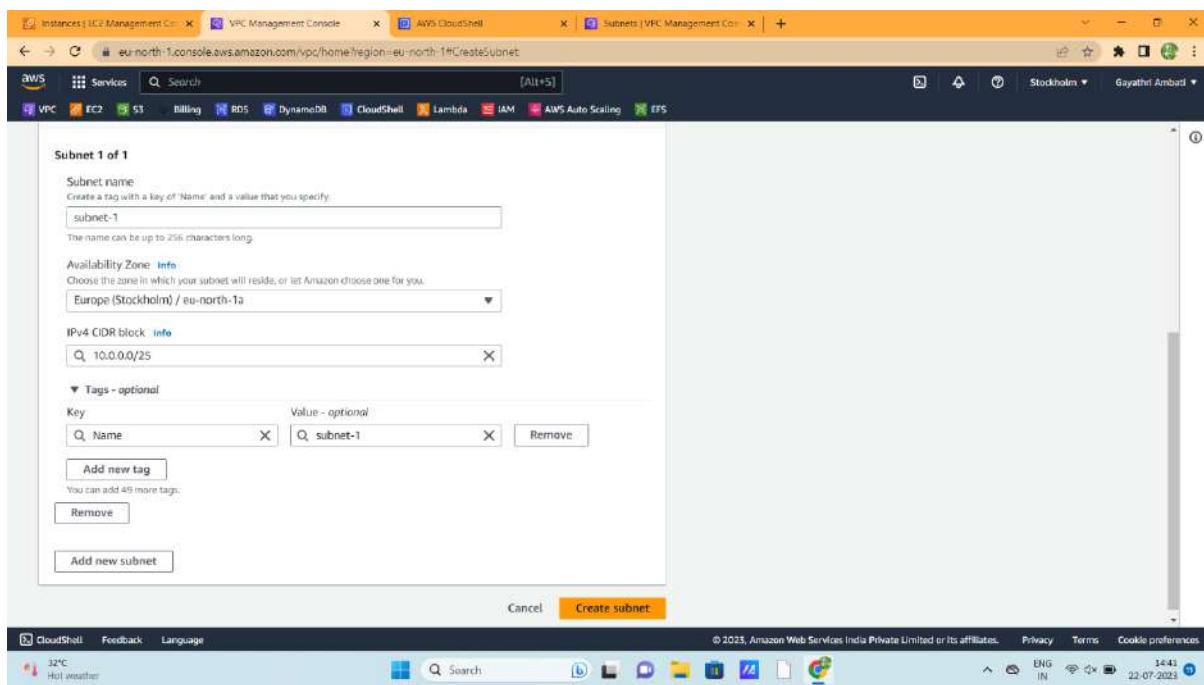
- Go to route table and edit the name of route table as rt1 public that is created along with vpc1.



- Go to subnets and create a subnet and name it as public subnet. Select the availability zone as North-1a, CIDR as 10.0.0.0/25 and select create subnet.
- we initially have 3 default Subnets.



- select VPC ID as vpc-1.



- Go to route tables, select the public route table, and go to subnet associations and select public subnet and click on save associations.

VPC > Route tables > rtb-00f60e9ec07e5f02 > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1/1)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
subnet-1	subnet-0bf9c810898c0b196	10.0.0.0/25	-	Main (rtb-00f60e9ec07e5f02 / rt-1)

Selected subnets

subnet-0bf9c810898c0b196 / subnet-1

Cancel **Save associations**

- Create an internet gateway with the name igw1.

VPC > Internet gateways > Create internet gateway

Create internet gateway

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

Name tag
Creates a tag with a key of 'Name' and a value that you specify.

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="lgw1"/> Remove

Add new tag
You can add 49 more tags.

Create internet gateway

- Attach custom IGW to custom VPC.

The screenshot shows the AWS VPC Management Console. In the center, there's a detailed view of an Internet Gateway named `igw-0445c80ecfb9f474e`. The gateway has an ID of `igw-0445c80ecfb9f474e`, is in a `Detached` state, and is owned by user `896571820628`. It has one tag named `Name` with the value `igw1`. On the left sidebar, under the `Internet gateways` section, there's a note: `The following internet gateway was created: igw-0445c80ecfb9f474e - igw1. You can now attach to a VPC to enable the VPC to communicate with the internet.`

- Internet Gate Way is successfully attached to VPC.
- Now go to route tables, select `rt1-vpc1-public`, click on edit routes, go to add routes, and add `0.0.0.0/0` at destination and select our internet gateway in destination and save it.

The screenshot shows the AWS VPC Management Console with the `Edit routes` dialog open for a route table. The table has two entries:

Destination	Target	Status	Propagated
<code>10.0.0.0/24</code>	<code>local</code>	<code>Active</code>	<code>No</code>
<code>0.0.0.0/0</code>	<code>igw-0445c80ecfb9f474e (igw1)</code>	<code>No</code>	<code>Remove</code>

At the bottom of the dialog, there are buttons for `Cancel`, `Preview`, and `Save changes`.

- Similarly create another two VPCs (VPC-2, VPC-3) using the above steps.

- Give the VPC name as `vpc2` and CIDR value as `192.168.0.0/24`.
- Click on create VPC and create `vpc-2`.
- create subnet to `vpc-2` with CIDR- `192.168.0.0/25`.
- Go to route tables, select the public route table, and go to subnet associations and select public subnet and click on save associations.
- Create an internet gateway with the name `igw2`.
- Attach custom IGW to custom VPC.
- Internet Gate Way is successfully attached to VPC.
- Now go to route tables, select `rt2 public`, click on edit routes, go to add routes, and add `0.0.0.0/0` at destination and select our internet gateway in destination and save it.
- Give the VPC name as `vpc3` and CIDR value as `4.0.0.0/24`.
- Click on create VPC and create `vpc-3`.
- create subnet to `vpc-2` with CIDR- `4.0.0.0/25`.
- Go to route tables, select the public route table, and go to subnet associations and select public subnet and click on save associations.
- Create an internet gateway with the name `igw3`.
- Attach custom IGW to custom VPC.
- Internet Gate Way is successfully attached to VPC.
- Now go to route tables, select `rt2 public`, click on edit routes, go to add routes, and add `0.0.0.0/0` at destination and select our internet gateway in destination and save it.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options
vpc-1	vpc-018c37750b49dd67	Available	10.0.0.0/24	-	dopt-0a
vpc-2	vpc-04c756fa5ed5a14b0	Available	192.168.0.0/24	-	dopt-0a
vpc-3	vpc-00c2e58b6e415d1d	Available	4.0.0.0/24	-	dopt-0a
vpc-4	vpc-066cfb8c357b11ff	Available	172.31.0.0/16	-	dopt-0a

- Now create an instance using Cloud Shell and remaining two instances in normal process.
- To create an instance using cloud shell we have some commands.

```
[cloudshell]: user@ip-10-4-33-72 ~]$ aws ec2 run-instances --image-id ami-0716e5989a4e4fa52 --count 1 --instance-type t3.micro --key-name gayathri --security-group-ids sg-08885de166a92efb --subnet-id subnet-0a9f2810ff8cb919c
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-0716e5989a4e4fa52",
      "InstanceId": "i-0811ade9ac98c17",
      "InstanceType": "t3.micro",
      "KeyName": "gayathri",
      "LaunchTime": "2023-07-22T09:24:55+00:00",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "eu-north-1a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-10-0-0-18.eu-north-1.compute.internal",
      "PrivateIpAddress": "10.0.0.18",
      "ProductCodes": [],
      "PublicDnsName": "",
      "State": {
        "Code": 0,
        "Name": "pending"
      },
      "StateTransitionReason": "",
      "SubnetId": "subnet-089fc810ff8cb919c",
      "VpcId": "vpc-018c37750b49dd67",
      "VpcOptions": {
        "Ipv4CidrBlock": "172.31.0.0/16"
      },
      "ClientToken": "0b31fd92-dfea-4510-8cd0-40c8d8896d7",
      "HasOptInSupport": false,
      "EndsSupport": true,
      "Hyperervisor": "xen",
      "NetworkInterfaces": [
        {
          "Attachment": {
            "AttachmentId": "eni-attach-018c37750b49dd67",
            "DeviceIndex": 0,
            "InstanceId": "i-0811ade9ac98c17",
            "InterfaceType": "interface",
            "MacAddress": "00:0c:29:00:00:02",
            "NetworkInterfaceId": "eni-018c37750b49dd67",
            "OwnerId": "116546166466",
            "PrivateDnsName": "eni-018c37750b49dd67.eni-attach-018c37750b49dd67.vpc-018c37750b49dd67.eu-north-1.amazonaws.com",
            "PrivateIpAddress": "172.31.0.2",
            "Status": "attached"
          }
        }
      ]
    }
  ]
}
```

- Now the Instance is created.

The screenshot shows the AWS EC2 Management Console. On the left, there's a sidebar with various navigation options like EC2 Dashboard, EC2 Global View, Events, Instances, Images, and Elastic Block Store. The main area displays a table of instances. One instance, 'vm-1', is selected and has a modal overlay where its name is being changed from 'vm-1' to 'vm-1'. Below the table, the details for instance 'i-04166d4e9ac790cc1' are shown, including its public and private IP addresses and state.

- Try to connect the Instance Vm-1. It will show error.
- create an end point to the Instance and then try to connect it.

This screenshot shows the 'Connect to instance' page for the instance 'i-04166d4e9ac790cc1'. A prominent warning message states: 'The instance does not have a public IPv4 address. To connect using the EC2 Instance Connect browser-based client, the instance must have a public IPv4 address.' Below this, there are two connection options: 'Connect using EC2 Instance Connect' (selected) and 'Connect using EC2 Instance Connect Endpoint'. The page also includes fields for 'Public IP address' (empty), 'User name' (set to 'ec2-user'), and a 'Password' input field.

Connection Type

- Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.
- Connect using EC2 Instance Connect Endpoint

Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Private IP address
10.0.0.18

User name
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ec2-user.
ec2-user

Max tunnel duration (seconds)
The maximum allowed duration of the SSH connection. Must comply with the maxTunnelDuration condition (if specified) in the IAM policy.
3600
Min 1 second. Max 3600 seconds (1 hour).

EC2 Instance Connect Endpoint
Only endpoints that have completed the creation process can be selected.
 Select an endpoint
 Create an endpoint

Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

https://eu-north-1.console.aws.amazon.com/vpc/home?region=eu-north-1#CreateVpcEndpoint

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

32°C Mostly cloudy

VPC > Endpoints > Create endpoint

Create endpoint Info

There are three types of VPC endpoints – Interface endpoints, Gateway Load Balancer endpoints, and Gateway endpoints. Interface endpoints and Gateway Load Balancer endpoints are powered by AWS PrivateLink, and use an Elastic Network Interface (ENI) as an entry point for traffic destined to the service. Interface endpoints are typically accessed using the public or private DNS name associated with the service, while Gateway endpoints and Gateway Load Balancer endpoints serve as a target for a route in your route table for traffic destined for the service.

Endpoint settings

Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.
end1

Service category
Select the service category

<input type="radio"/> AWS services Services provided by Amazon	<input type="radio"/> PrivateLink Ready partner services Services with an AWS Service Ready designation	<input type="radio"/> AWS Marketplace services Services that you've purchased through AWS Marketplace
<input checked="" type="radio"/> EC2 Instance Connect Endpoint An elastic network interface that allows you to connect to resources in a private subnet	<input type="radio"/> Other endpoint services Find services shared with you by service name	

CloudShell Feedback Language

32°C Mostly cloudy

VPC

Select the VPC in which to create the endpoint

The VPC in which to create your endpoint:

vpc-018c37750b49dc67 (vpc-1)

► Additional settings

Security groups (1/2) [Info](#)

Find resources by attribute or tag

Group ID	Group name	VPC ID
sg-0762152f0c26bc1cf	default	vpc-018c37750b49c
<input checked="" type="checkbox"/> sg-088850efd66b52efb	sg1-vpc1	vpc-018c37750b49c

sg-088850efd66b52efb X

Subnet

Select the Subnet in which to create the endpoint

Subnet

Select the subnets in which to create the endpoint.

Select a subnet

https://eu-north-1.console.aws.amazon.com/vpc/home?region=eu-north-1#CreateVpcEndpoint

32°C Mostly cloudy

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

ENG IN 14:57 22-07-2023

Security groups (1/2) [Info](#)

Find resources by attribute or tag

Group ID	Group name	VPC ID
sg-088850efd66b52efb		

sg-088850efd66b52efb X

Subnet

Select the Subnet in which to create the endpoint

Subnet

Select the subnets in which to create the endpoint.

subnet-0bf9c810898c0b196 (subnet-1)

Tags

Key Value - optional

Name	X	end1	X	Remove
Add new tag				

You can add 49 more tags.

Create endpoint

CloudShell Feedback Language

https://eu-north-1.console.aws.amazon.com/vpc/home?region=eu-north-1#CreateVpcEndpoint

32°C Mostly cloudy

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

ENG IN 14:57 22-07-2023

The screenshot shows the AWS VPC Endpoints console. On the left, there's a sidebar with options like 'Virtual private cloud' (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists), 'Endpoints' (selected), 'Endpoint services', 'NAT gateways', and 'Peering connections'. The main area is titled 'Endpoints (1) info' and shows a table with one row:

Name	VPC endpoint ID	VPC ID	Service name	Endpoint type	Status
end1	eice-029ace3103acbcb...	vpc-018c37750b49cd67 vp...	eice-029ace31...	EC2 Instance Connect Endp...	Available

Below the table, there's a section titled 'Select an endpoint' with three icons: a magnifying glass, a plus sign, and a minus sign.

- Now select the VM-1 and connect it will be connected.

The screenshot shows an EC2 Instance Connect terminal session. The title bar indicates the URL is <https://eu-north-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=eu-north-1&connType=eice&instanceId=i-0861f72a92d5e41f5&User=ec2-user&Port=22&instanceConnectEndpoint=end1>. The terminal window displays a root shell prompt for an Amazon Linux 2023 instance. The output includes:

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

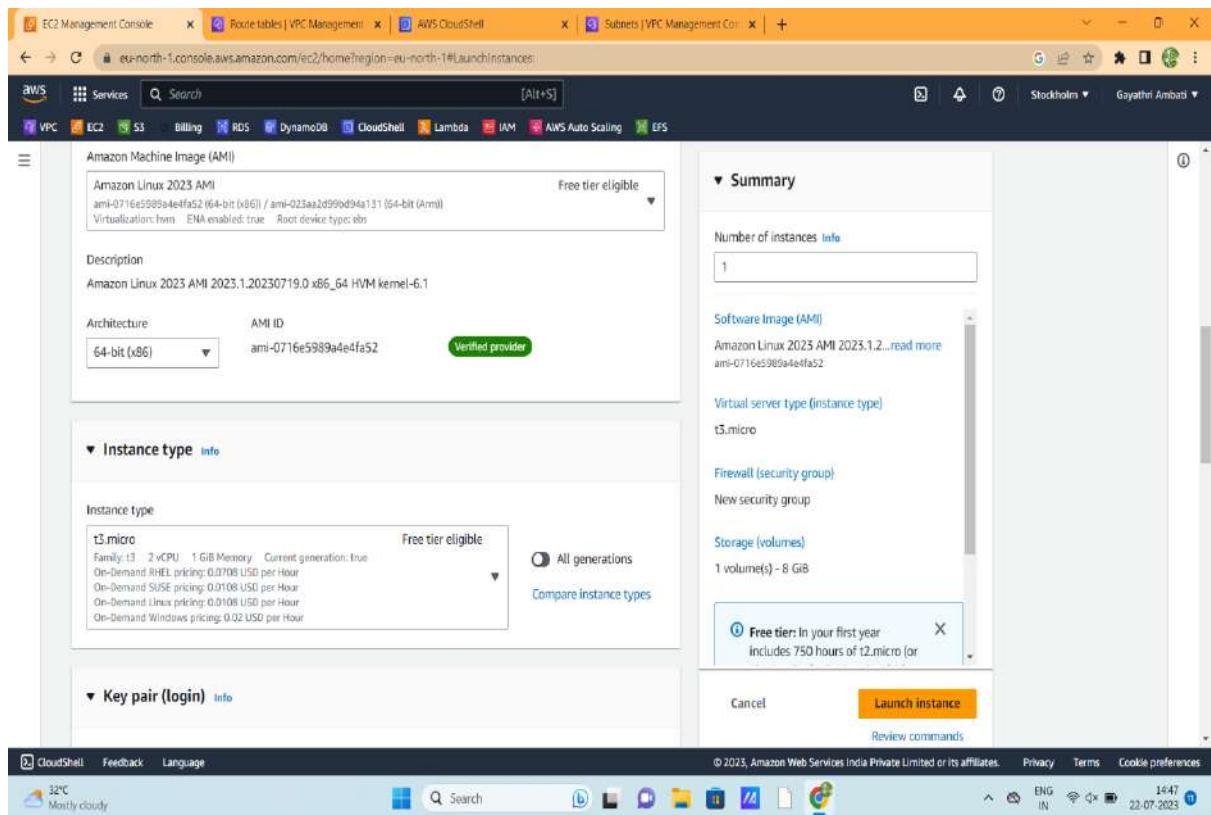
Last login: Sat Jul 22 09:53:40 2023 from 10.0.0.85
[ec2-user@ip-10-0-0-9 ~]$

```

At the bottom of the terminal window, it says 'i-0861f72a92d5e41f5 (vm-1)' and 'PrivateIPs: 10.0.0.9'.

- Now create another two Instances using EC2.
- Creating an EC2 instance in a public subnet:

- Name your instance as VM-2.
- Select “Amazon Linux 2 AMI”.
- Instance type “t3. micro”.
- Select your existing key pair.
- Select your custom VPC, public subnet and enable the auto-assign public IP.
- In the security group section, select availability zone as North-1a.
- create a new security group, add security groups that supports SSH and All Traffic.
- Similarly create the third Instance naming it as VM-3.
- Launch your Instance.



Screenshot of the AWS EC2 Management Console showing the 'Launch instances' wizard.

Key pair (login) [Info]

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required
gayathri

Network settings [Info]

VPC - required [Info]
vpc-018c37750b49dd67 (vpc-1)
10.0.0.0/24

Subnet Info
subnet-0bf9c810898c0b196 subnet-1
VPC: vpc-018c37750b49dd67 Owner: 896571820628 Availability Zone: eu-north-1a IP addresses available: 125 CIDR: 10.0.0.0/25

Create new subnet

Auto-assign public IP [Info]
Enable

Firewall (security groups) [Info]
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group
Select existing security group

Summary

Number of instances [Info]
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.1.2...read more
ami-0716e5989a4e4fa52

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year
includes 750 hours of t2.micro (or

Cancel Launch instance Review commands

CloudShell Feedback Language 32°C Mostly cloudy © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 1447 22-07-2023

Screenshot of the AWS EC2 Management Console showing the 'Launch instances' wizard.

Create security group

Security group name - required
sg1-vpc1

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _/-[!@#\$%^&_-=~`]{}

Description - required [Info]
launch-wizard-4 created 2023-07-22T09:17:07.974Z

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type [Info] Protocol [Info] Port range [Info]
ssh TCP 22

Source type [Info] Source [Info] Description - optional [Info]
Anywhere e.g. SSH for admin desktop
0.0.0.0/0 ::/0

Free tier: In your first year
includes 750 hours of t2.micro (or

Add security group rule

Summary

Number of instances [Info]
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.1.2...read more
ami-0716e5989a4e4fa52

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Cancel Launch instance Review commands

CloudShell Feedback Language 32°C Mostly cloudy © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 1449 22-07-2023

The screenshot shows the AWS EC2 Management Console with the URL eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#LaunchInstances. The main content area displays a green success message: "Successfully initiated launch of instance (i-0d27da7b68c754152)". Below this is a link to "Launch log". The "Next Steps" section contains several links: "Create billing and free tier usage alerts", "Connect to your instance", "Connect an RDS database", and "Create EBS snapshot policy". A search bar at the top asks, "What would you like to do next with this instance, for example 'create alarm' or 'create backup'?".

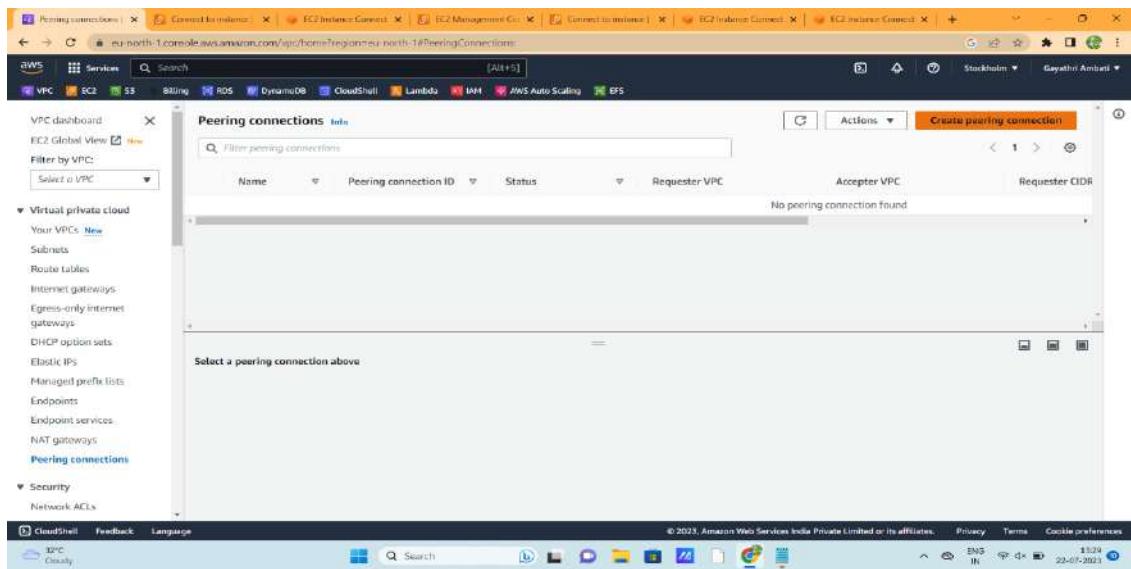
- Finally the Instance is launched.

The screenshot shows the AWS EC2 Management Console with the URL ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances. The left sidebar shows navigation options like "New EC2 Experience", "EC2 Dashboard", "EC2 Global View", "Events", "Instances", "Images", "Elastic Block Store", and "CloudShell". The main content area displays a table titled "Instances (5) info" with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. The table lists three instances: VM-1, VM-2, and VM-3, all in the "Running" state. Below the table is a modal window titled "Select an instance" with a dropdown menu.

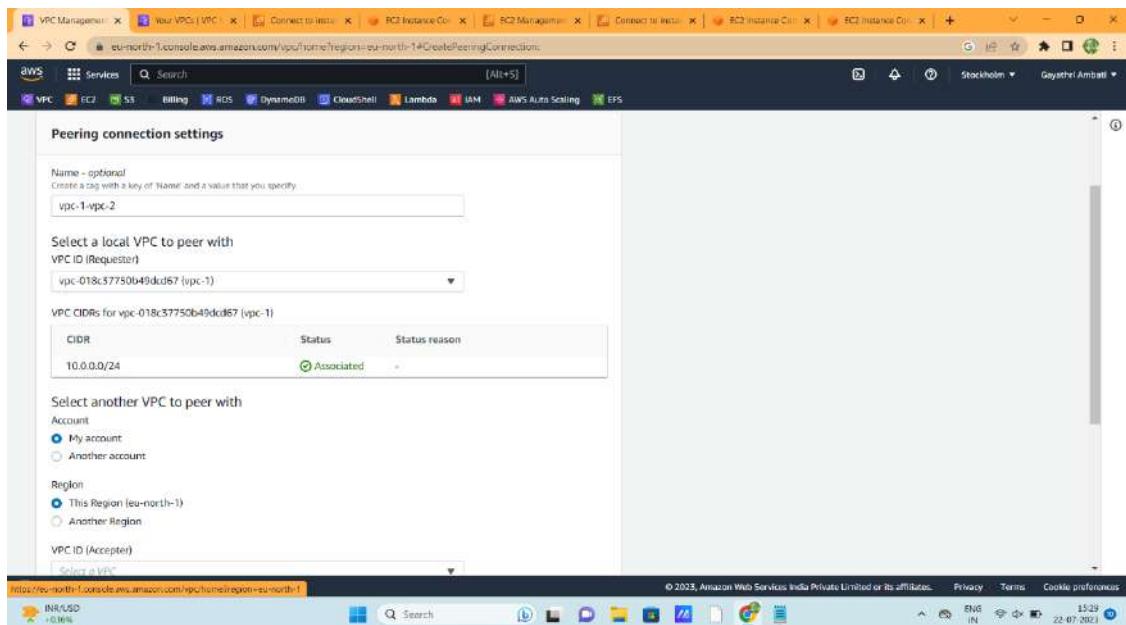
Peering connection

- Now we have to create peering connection between these 3 VPCs
- Go to peering connections in VPC Service

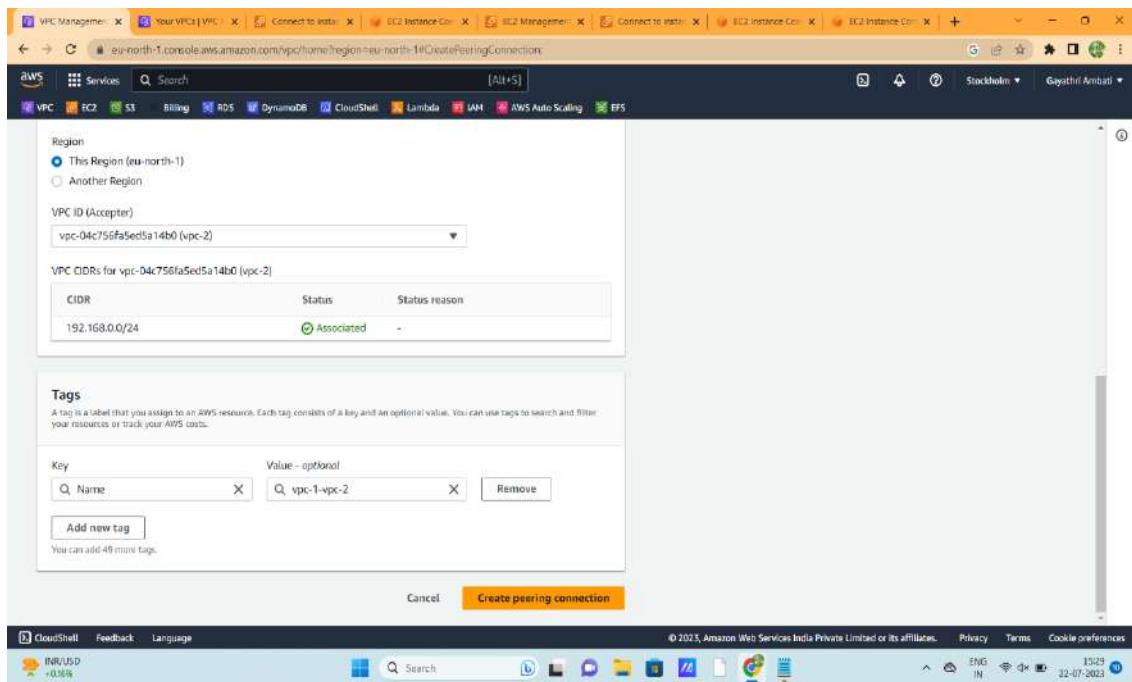
- Click on create peering connection



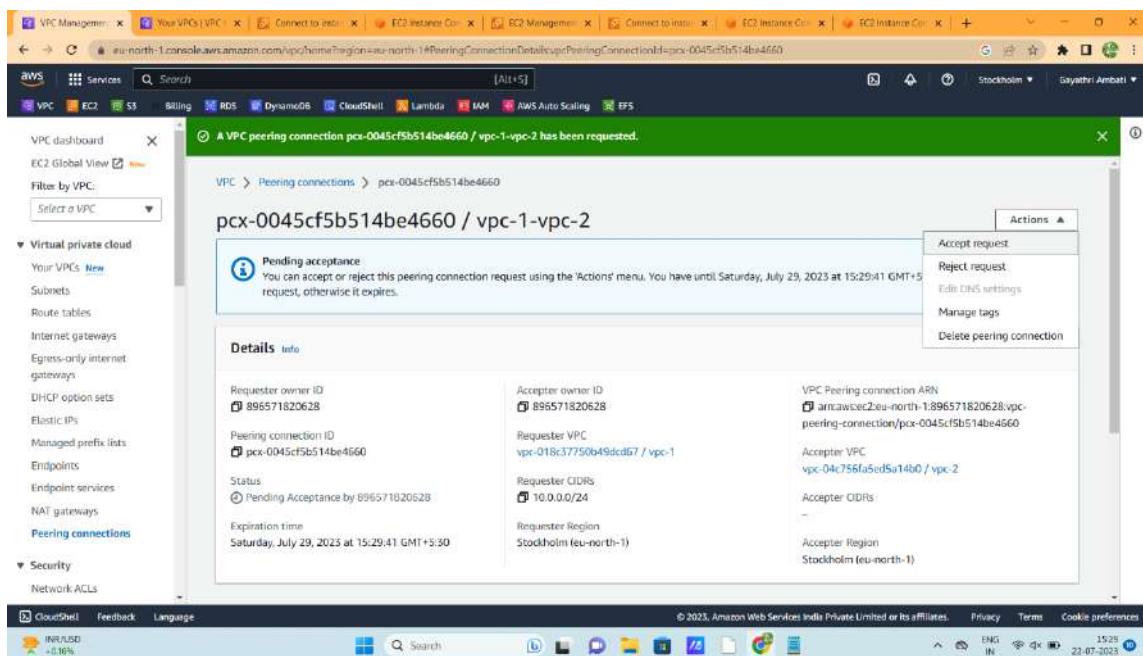
- For creating peering connection between VPC1 and VPC2. Give name as vpc-1-vpc-2 and select VPC(requester) as VPC1
- As these VPCs belongs to same region and same account so, select This region and My account



- Select VPC (acceptor) as VPC2 and click on create peering connection.



- After creating peering connection
- Click on Actions
- Click on Accept Request.



- Click on modify route tables now.

The screenshot shows the AWS VPC Management console. In the left sidebar, under 'Virtual private cloud', 'Peering connections' is selected. The main content area displays a single peering connection entry:

- peering connection ID:** ppx-0045cf5b514be4660
- Status:** Active
- Requester owner ID:** 896571820628
- Requester VPC:** vpc-018c37750b49cd67 / vpc-1
- Requester CIDR:** 10.0.0.0/24
- Requester Region:** Stockholm (eu-north-1)
- Acceptor owner ID:** 896571820629
- Acceptor VPC:** vpc-04c756fa5ed5a14b0 / vpc-2
- Acceptor CIDR:** 192.168.0.0/24
- Acceptor Region:** Stockholm (eu-north-1)

Below the details, there are tabs for DNS, Route tables, and Tags. The Route tables tab is currently selected.

- Go to route tables and select rt1.
- Click on routes and then click on edit routes.
- In edit route dialogue box , click on add route and add CIDR (192.168.0.0/24)of VPC2 in rt1 and In target select peering connection(vpc1-vpc2).
- Click on save changes

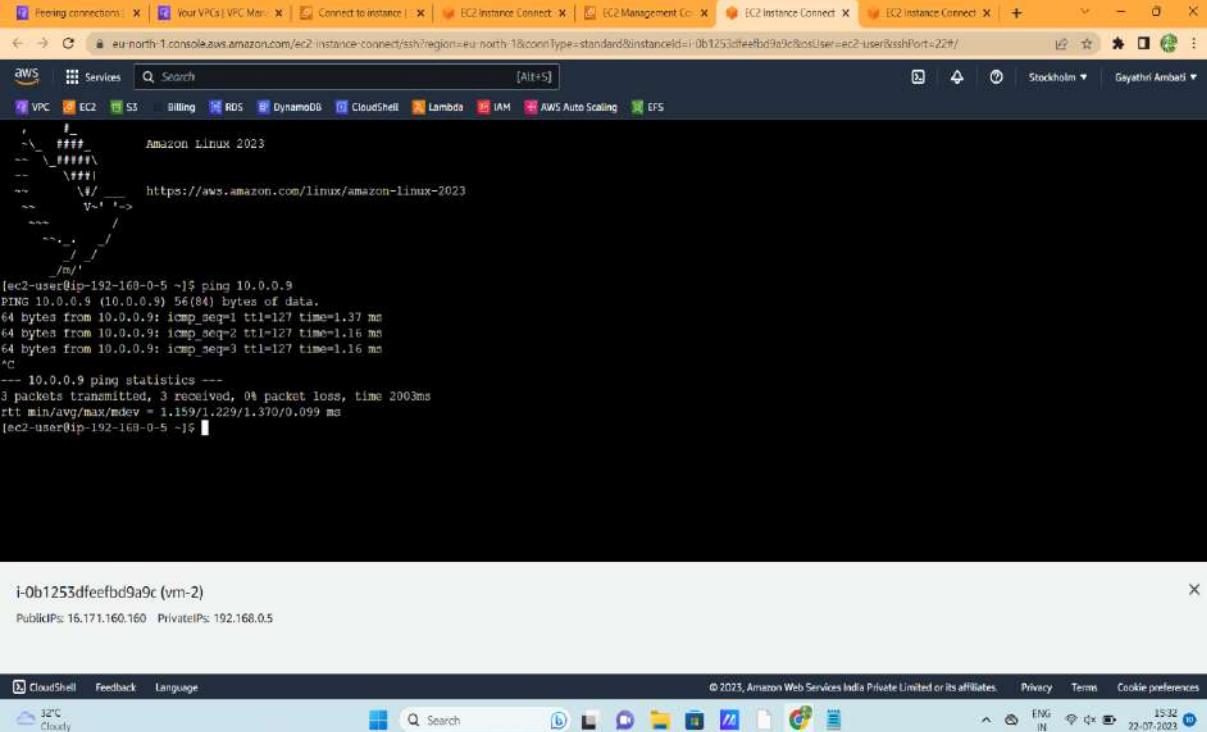
The screenshot shows the AWS VPC Management console. In the left sidebar, 'Route Tables' is selected. The main content area shows a route table with the following configuration:

Destination	Target	Status	Propagated
10.0.0.0/24	local	Active	No
0.0.0.0/0	igw-0445c80ecfb9f474e	Active	No
192.168.0.0/24	(Search dropdown)	-	No

A modal dialog titled 'Edit routes' is open for the last row. The 'Destination' field contains '192.168.0.0/24'. The 'Target' field has a dropdown menu open, showing options like 'Instance', 'Internet Gateway', 'local', 'NAT Gateway', etc. At the bottom right of the dialog are 'Cancel', 'Preview', and 'Save changes' buttons.

- Now connect Instance1(vm1) and Instance2(vm2) to the console.
- Ping the Instances with Private IP addresses.

- Type Ping 10.0.0.9 which is IP address of vm1 in console of vm1

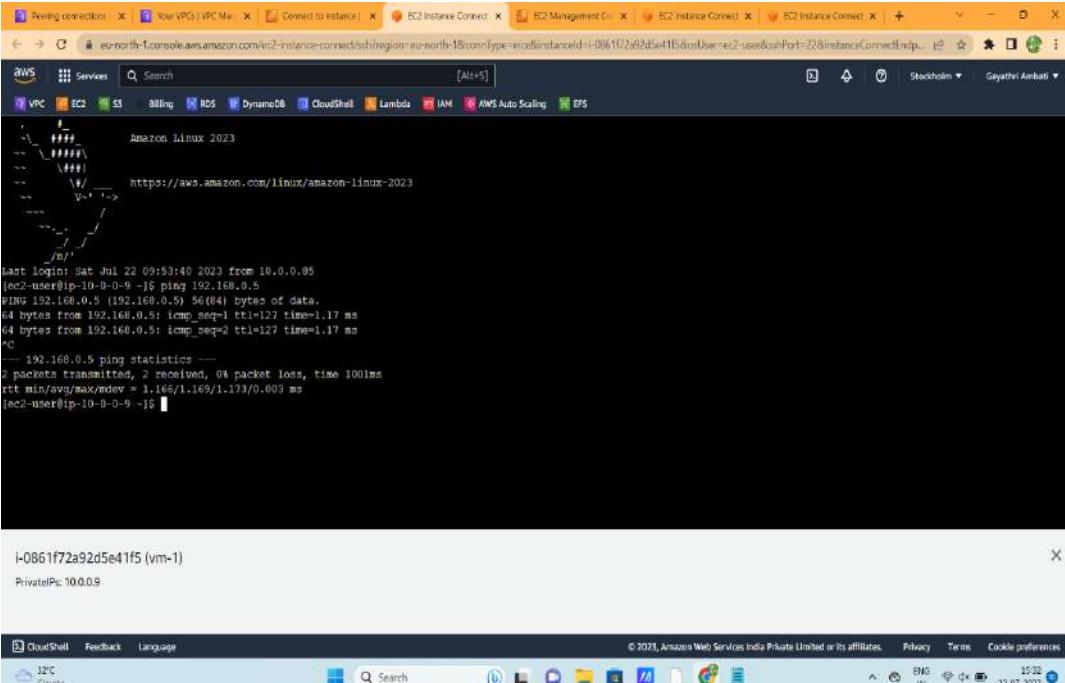


```
[ec2-user@ip-192-168-0-5 ~]$ ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=1.37 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=1.16 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=127 time=1.16 ms
^C
--- 10.0.0.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.159/1.229/1.370/0.099 ms
[ec2-user@ip-192-168-0-5 ~]$
```

i-0b1253dfeefbd9a9c (vm-2)

PublicIPs: 16.171.160.160 PrivateIPs: 192.168.0.5

- Now Type ping 192.168.0.5 on VM-1 console



```
[ec2-user@ip-10-0-0-9 ~]$ ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=127 time=1.17 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=127 time=1.17 ms
^C
--- 192.168.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 100ms
rtt min/avg/max/mdev = 1.165/1.169/1.173/0.003 ms
[ec2-user@ip-10-0-0-9 ~]$
```

i-0861f72a92d5e41f5 (vm-1)

PrivateIPs: 10.0.0.9

- Now create peering connection between VPC1 and VPC3 and ping private IP of VM1 in Console of VM3 and ping private IP of VM3 in Console of VM1
- Consider VPC1 as requester and VPC3 as accepter for creating peering connection.

```

aws [Services] Search [Alt+S]
VPC EC2 S Billing RDS DynamoDB CloudShell Lambda IAM AWS Auto Scaling EFS

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

(amazonlinux) [~] ~$ ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=1.24 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=1.10 ms
^C
--- 10.0.0.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.099/1.171/1.244/0.072 ms
(amazonlinux) [~] ~$ 

```

i-0cf2b7535472c212a (vm-3)

PublicIPs: 16.171.68.5 PrivateIPs: 4.0.0.72

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 32°C Cloudy 13:34 ENG IN 22-07-2023

```

aws [Services] Search [Alt+S]
VPC EC2 S Billing RDS DynamoDB CloudShell Lambda IAM AWS Auto Scaling EFS

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

Last login: Sat Jul 22 09:53:40 2023 from 10.0.0.85
(amazonlinux) [~] ~$ ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=127 time=1.17 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=127 time=1.17 ms
^C
--- 192.168.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.166/1.169/1.173/0.003 ms
(amazonlinux) [~] ~$ ping 4.0.0.72
PING 4.0.0.72 (4.0.0.72) 56(84) bytes of data.
64 bytes from 4.0.0.72: icmp_seq=1 ttl=127 time=1.02 ms
64 bytes from 4.0.0.72: icmp_seq=2 ttl=127 time=1.03 ms
64 bytes from 4.0.0.72: icmp_seq=3 ttl=127 time=1.07 ms
^C
--- 4.0.0.72 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.024/1.042/1.072/0.021 ms
(amazonlinux) [~] ~$ 

```

i-0861f72a92d5e41f5 (vm-1)

PrivateIPs: 10.0.0.9

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 32°C Cloudy 13:35 ENG IN 22-07-2023

- Similarly, create peering connection between VPC2 and VPC3 and name it as VPC-2-VPC-3.
Consider VPC2 as requester and VPC3 as accepter.
- After creating peering connection and editing route tables . Now ping Private IP addresses of VM2(192.168.0.5) in VM3 Console.

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-4-0-0-72 ~]$ ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=1.24 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=1.10 ms
^C
--- 10.0.0.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.099/1.171/1.244/0.072 ms
[ec2-user@ip-4-0-0-72 ~]$ ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=127 time=1.71 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=127 time=1.48 ms
^C
--- 192.168.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.481/1.594/1.703/0.113 ms
[ec2-user@ip-4-0-0-72 ~] $
```

i-0cf2b7535472c212a (vm-3)

PublicIPs: 16.171.68.5 PrivateIPs: 4.0.0.72

- Now ping Private IP addresses of VM3 (4.0.0.72) in VM2 Console

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

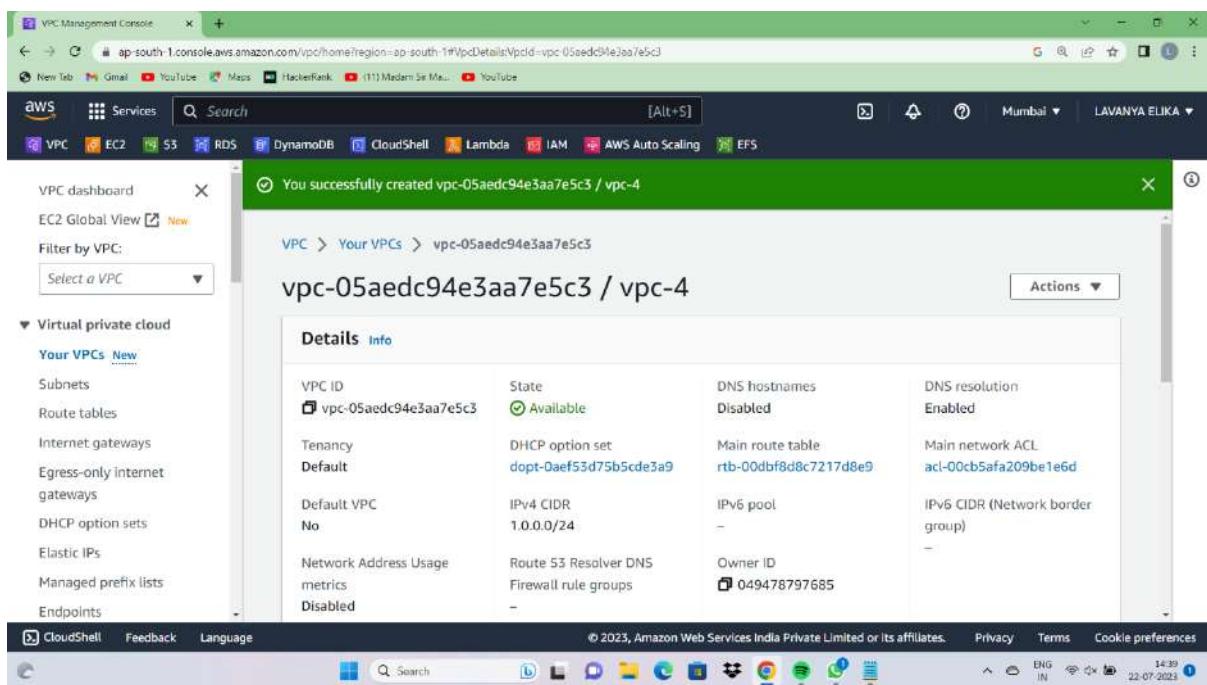
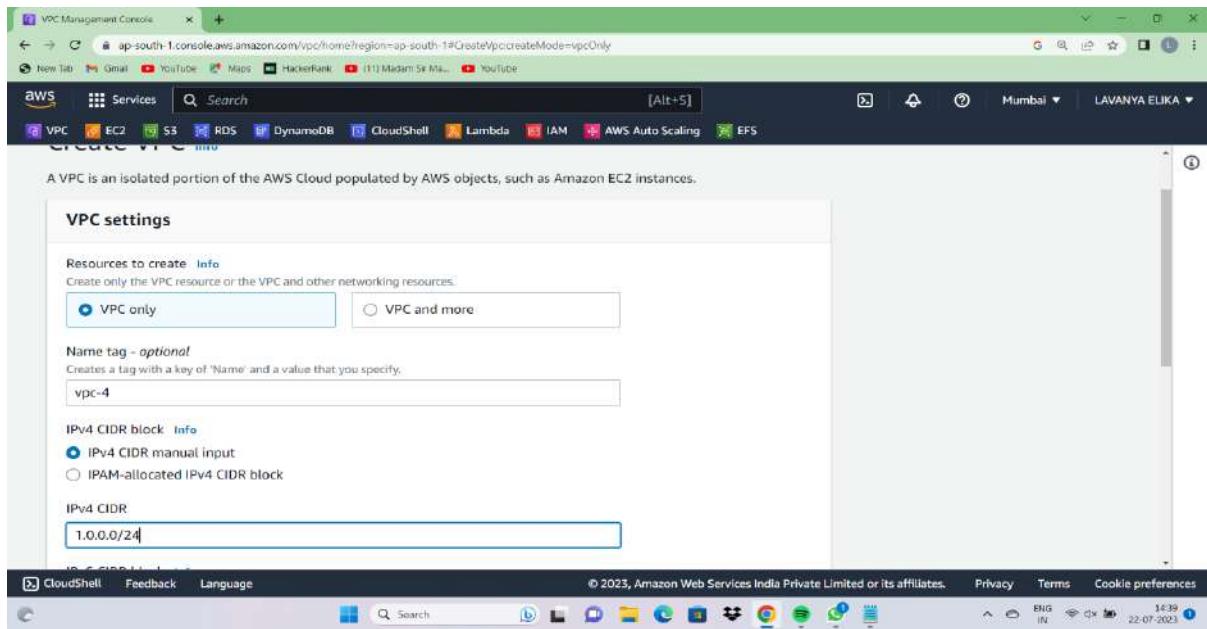
[ec2-user@ip-192-168-0-5 ~]$ ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=1.07 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=1.16 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=127 time=1.16 ms
^C
--- 10.0.0.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.159/1.229/1.370/0.099 ms
[ec2-user@ip-192-168-0-5 ~]$ ping 4.0.0.72
PING 4.0.0.72 (4.0.0.72) 56(84) bytes of data.
64 bytes from 4.0.0.72: icmp_seq=1 ttl=127 time=1.48 ms
64 bytes from 4.0.0.72: icmp_seq=2 ttl=127 time=1.49 ms
^C
--- 4.0.0.72 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.482/1.483/1.485/0.001 ms
[ec2-user@ip-192-168-0-5 ~] $
```

i-0b1253dfeefbd9a9c (vm-2)

PublicIPs: 16.171.160.160 PrivateIPs: 192.168.0.5

- Upto here we completed 3 peering connections in stockholm.
- Now we create another 3 Peering connections in different account and different region(Mumbai).
- Now create VPC.
- Click on the "Create VPC" option.
- Give the VPC name as **vpc4** and CIDR value as **1.0.0.0/24**.

- Click on create VPC.
- And create vpc-4.



- Go to route table and edit the name of route table as rt1 public that is created along with vpc1.

Name	Route table ID	Explicit subnet associations	Edge associations
-	-00dbfb8d8c7217d8e9	-	-
-	-08ae9309e75613e48	-	-

rtb-00dbfb8d8c7217d8e9

- Details
- Routes
- Subnet associations
- Edge associations
- Route propagation
- Tags

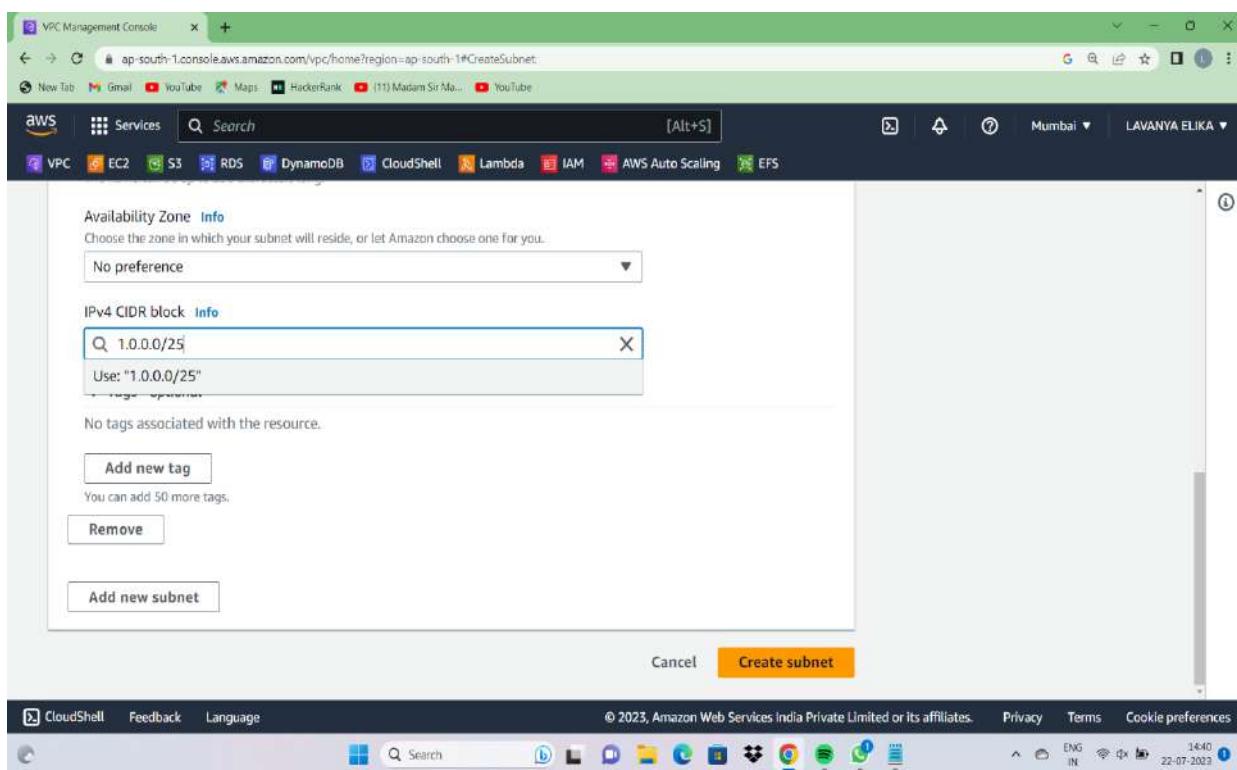
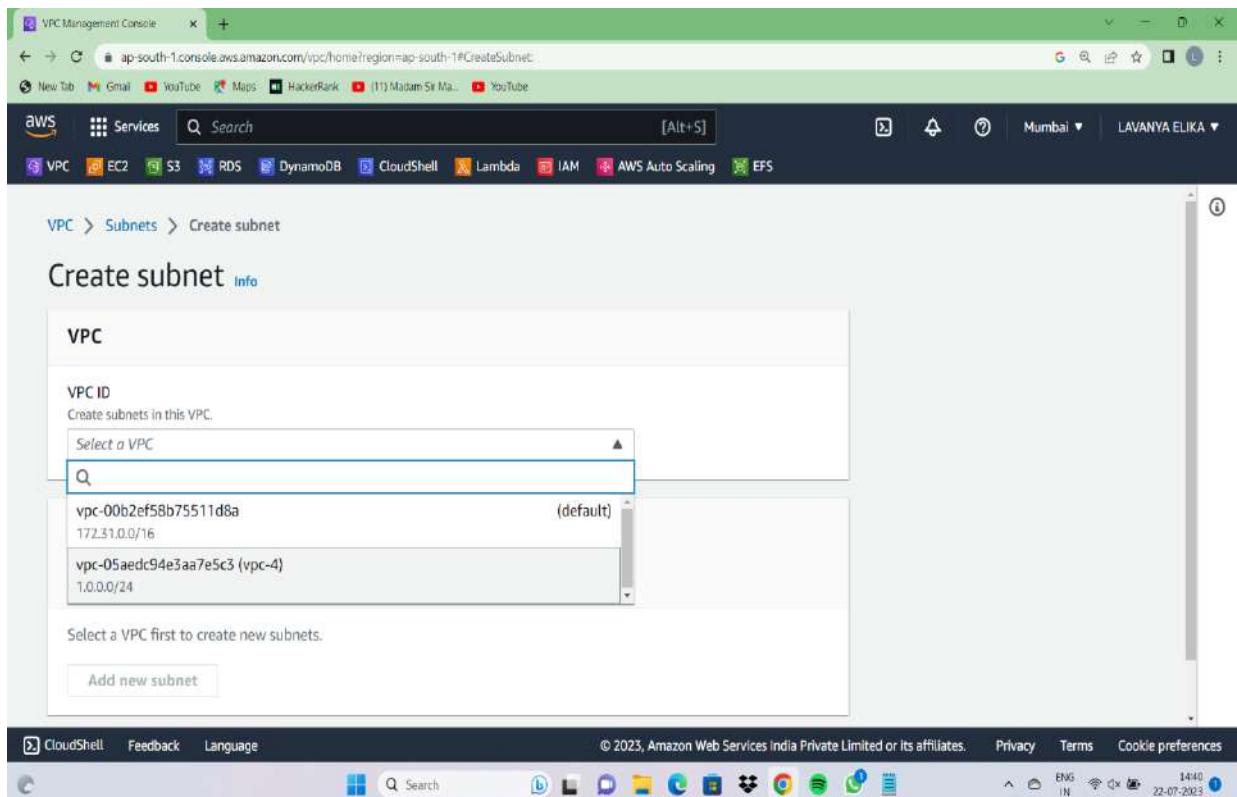
You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer

- Go to subnets and create a subnet and name it as public subnet. Select the availability zone as South-1a, CIDR as 1.0.0.0/25 and select create subnet.
- we initially have 3 default Subnets.

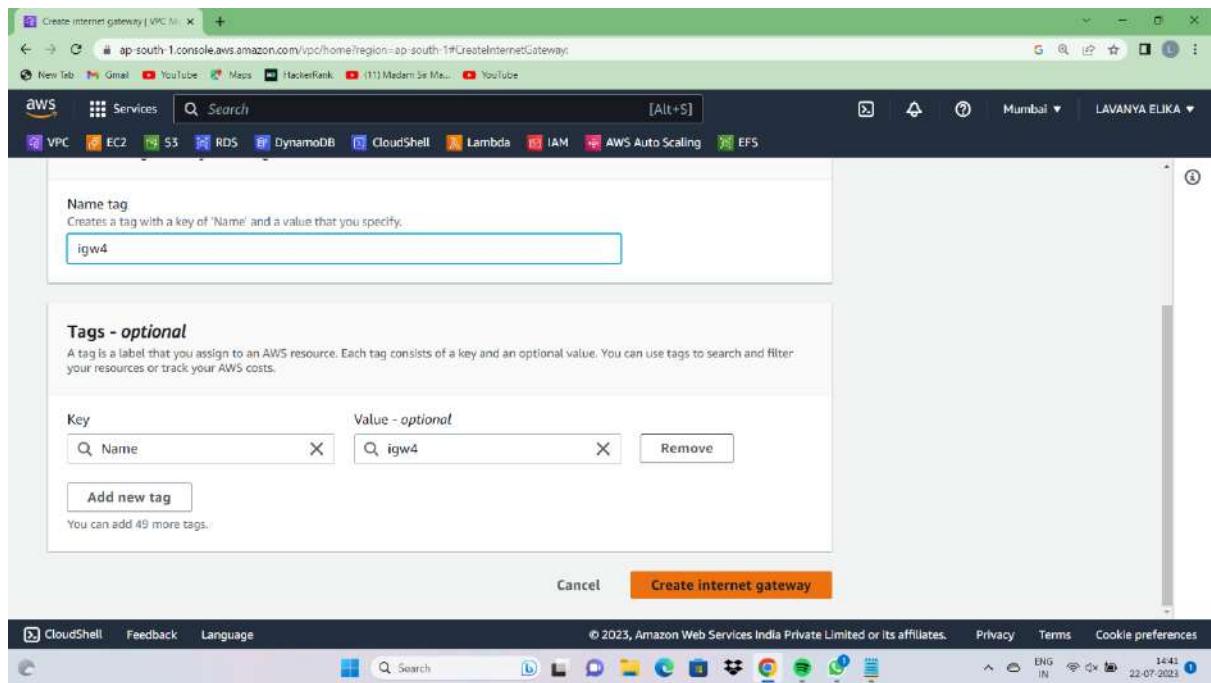
Name	Subnet ID	State	VPC
-	subnet-07568f47c6deb09eb	Available	vpc-00b2ef58b75511d8a
-	subnet-0ee95d0ba96358f82	Available	vpc-00b2ef58b75511d8a
-	subnet-039c77ef3f57ec3d3	Available	vpc-00b2ef58b75511d8a

Select a subnet

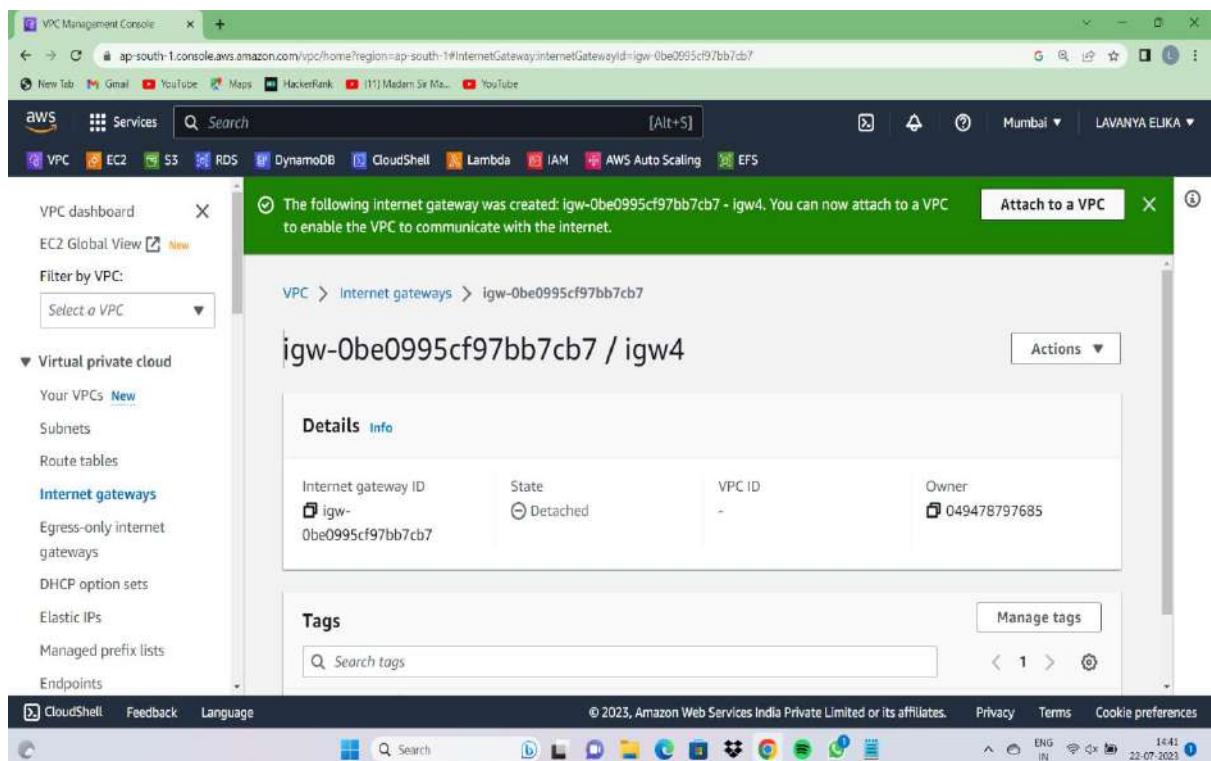


- Go to route tables, select the public route table, and go to subnet associations and select public subnet and click on save associations.

- Create an internet gateway with the name igw4.

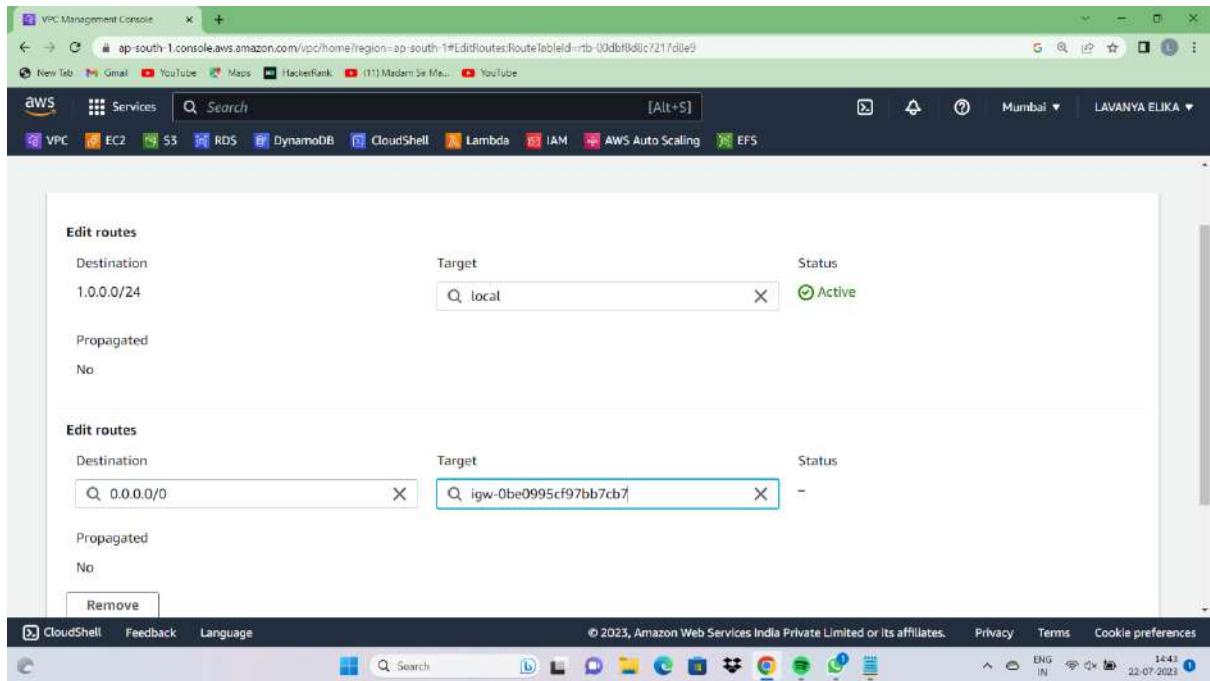


- Attach custom IGW to custom VPC.



- Internet Gate Way is successfully attached to VPC.

- Now go to route tables, select rt4 public, click on edit routes, go to add routes, and add 0.0.0.0/0 at destination and select our internet gateway in destination and save it



- Similarly create another two VPCs (VPC-5, VPC-6) using the above steps.
- Give the VPC name as `vpc5` and CIDR value as `2.0.0.0/24`.
- Click on create VPC and create `vpc-5`.
- create subnet to `vpc-2` with CIDR- `2.0.0.0/25`.
- Go to route tables, select the public route table, and go to subnet associations and select public subnet and click on save associations.
- Create an internet gateway with the name `igw5`.
- Attach custom IGW to custom VPC.
- Internet Gate Way is successfully attached to VPC.
- Now go to route tables, select `rt5` public, click on edit routes, go to add routes, and add `0.0.0.0/0` at destination and select our internet gateway in destination and save it.
- Give the VPC name as `vpc-6` and CIDR value as `3.0.0.0/24`.
- Click on create VPC and create `vpc-6`.

Name	VPC ID	State	IPv4 CIDR
-	vpc-00b2ef58b75511d8a	Available	172.31.0.0/16
vpc-4	vpc-05edc94e3aa7e5c3	Available	1.0.0.0/24
vpc-5	vpc-0c76c41134d2e668d	Available	2.0.0.0/24
vpc-6	vpc-03863ead6933e9556	Available	3.0.0.0/24

- Creating an EC2 instance in a public subnet:
- Name your instance as VM-4.
- Select “Amazon Linux 2 AMI”.
- Instance type “t2. micro”.
- Select your existing key pair.
- Select your custom VPC, public subnet and enable the auto-assign public IP.
- In the security group section, select availability zone as South-1a.
- create a new security group, add security groups that supports SSH and All Traffic.
- Similarly create the third Instance and fourth instances and name them as VM-5 , VM-6.
- Launch your Instance.

The screenshot shows the AWS EC2 Management Console interface. In the top navigation bar, the URL is `ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances`. The main content area is titled "Name and tags". A "Name" field contains the value "vm-4". Below this, there is a section titled "Application and OS Images (Amazon Machine Image)" with a search bar. A "Quick Start" button is also visible.

The screenshot shows the AWS EC2 Management Console interface. In the top navigation bar, the URL is `ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances`. The main content area is titled "Network settings". It includes fields for "VPC - required" (set to "vpc-05aedc94e3aa7e5c3 (vpc-4) 1.0.0.0/24"), "Subnet info" (set to "subnet-05a02139b2b962ef6 subnet4 VPC: vpc-05aedc94e3aa7e5c3 Owner: 049478797685 Availability Zone: ap-south-1c IP addresses available: 123 CIDR: 1.0.0.0/25"), and "Auto-assign public IP" (set to "Enable"). A "Create new subnet" button is also present.

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Security group name - required

sg4

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-/.@#[]+=&.;!\$^*

Description - required

launch-wizard-3 created 2023-07-22T09:15:05.756Z

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type **Info** Protocol **Info** Port range **Info**

ssh TCP 22

Remove

Security group rule 2 (All, All, Multiple sources)

Type **Info** Protocol **Info** Port range **Info**

All traffic All All

Source type **Info** Source **Info** Description - optional **Info**

Anywhere Add CIDR, prefix list or security group e.g. SSH for admin desktop

0.0.0.0/0 X ::/0 X

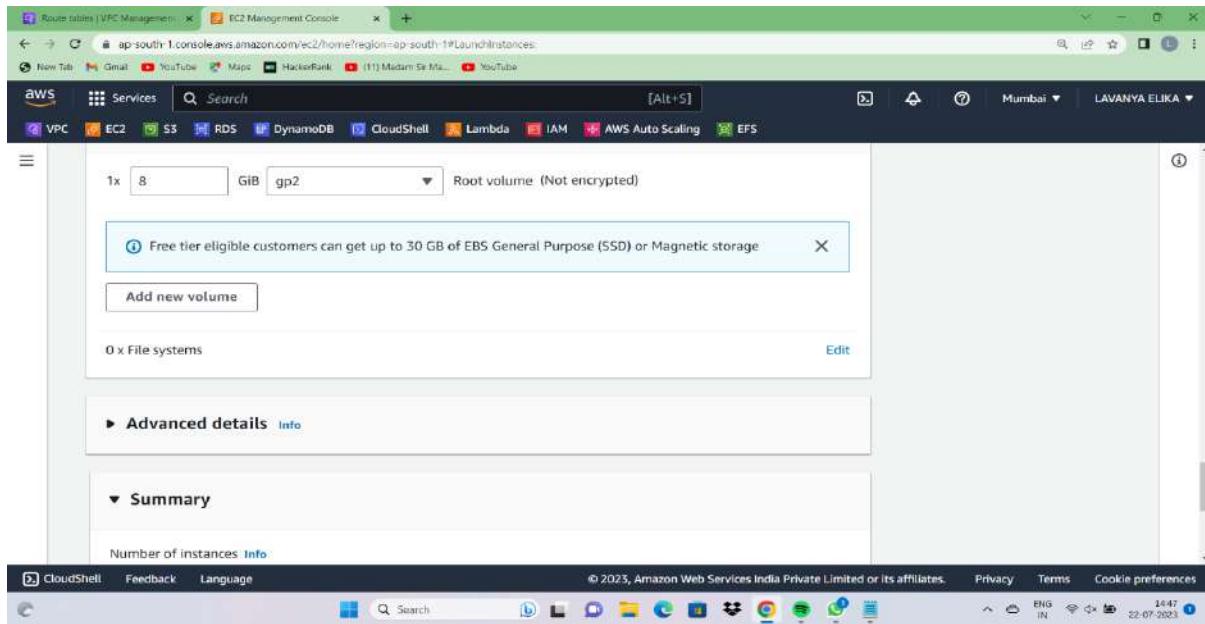
⚠️ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. X

Add security group rule

Advanced network configuration

Configure storage **Info**

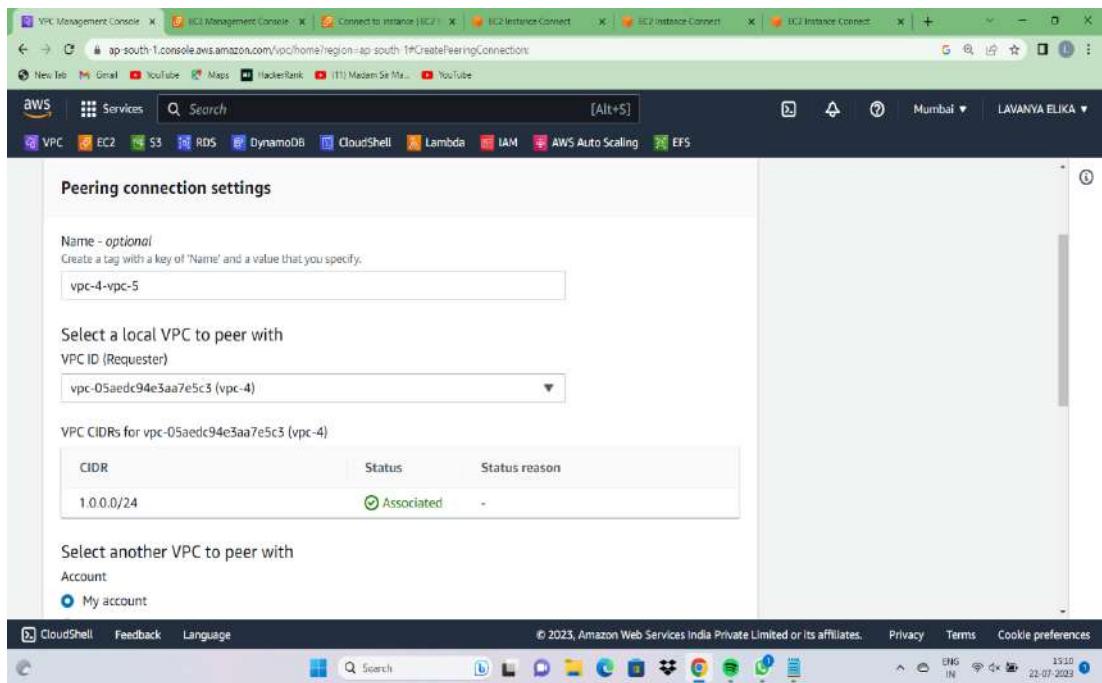
Advanced



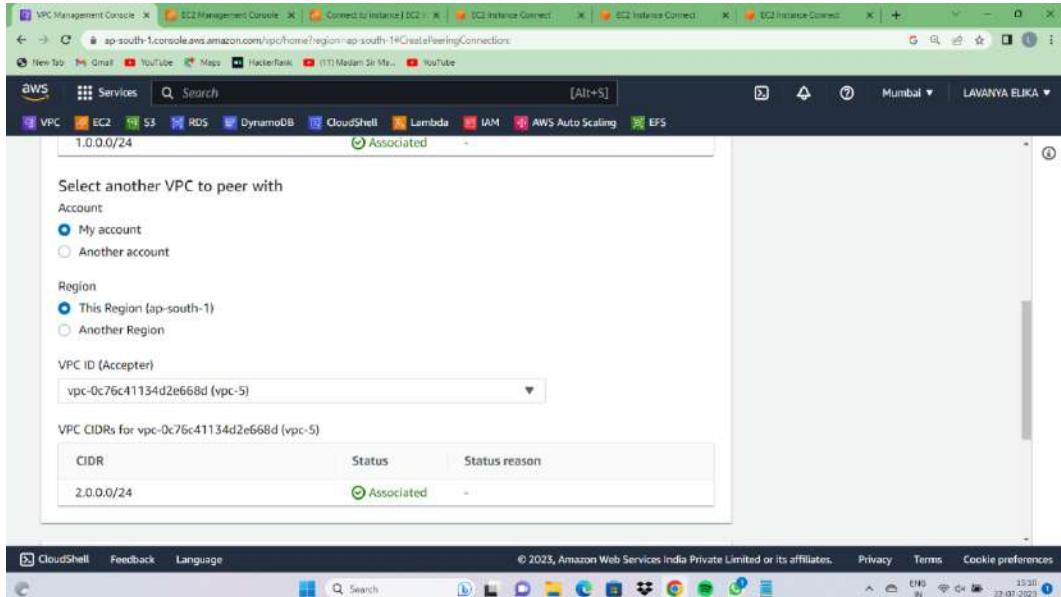
- Finally the instances are launched connect them by going actions option.

Peering Connection

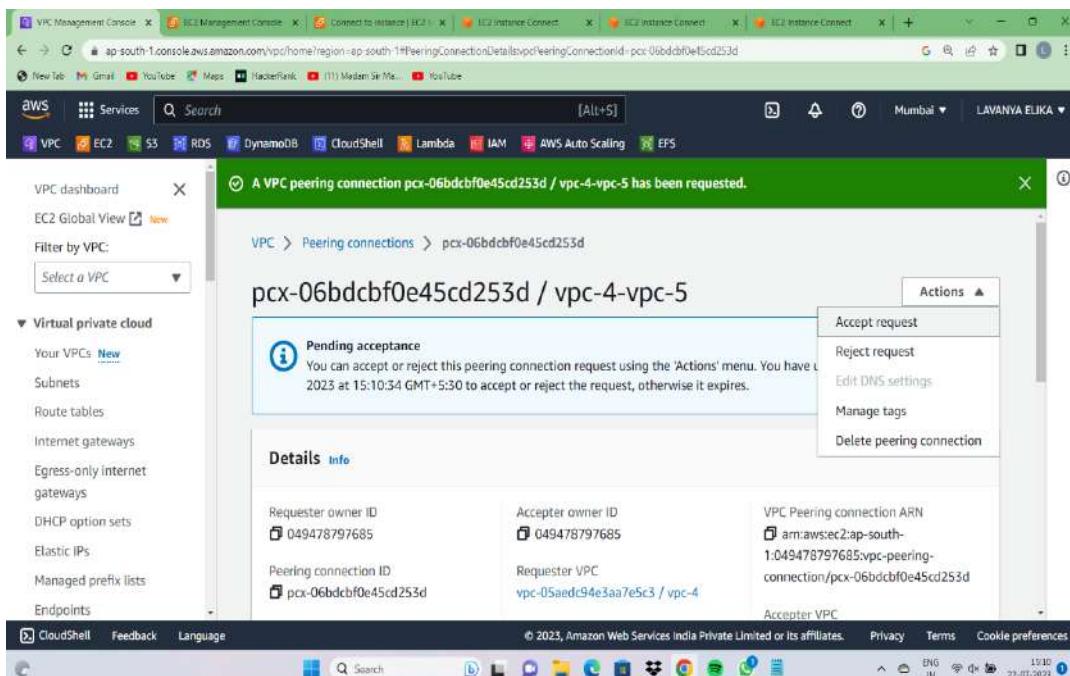
- Now we have to create peering connection between these 3 VPCs
- Go to peering connections in VPC Service
- Click on create peering connection



- For creating peering connection between VPC4 and VPC6 .Give name as vpc-4-vpc-6 and select VPC(requester) as VPC4
- Select VPC (acceptor) as VPC5and click on create peering connection.



- After creating peering connection
- Click on Actions
- Click on Accept Request.



- Click on modify route tables now.
- Go to route tables and select rt4.

- Click on routes and then click on edit routes.
- In edit route dialogue box, click on add route and add CIDR (2.0.0.0/ 24) of VPC5 in rt4 and In target select peering connection(vpc4-vpc5).

The screenshot shows the AWS VPC Management Console. The URL is <https://ap-south-1.console.aws.amazon.com/vpc/home?region=ap-south-1#EditRoutes:RouteTableId=rtb-00dbfb0dc7217d8e9>. The page displays the Route Table configuration for VPC5. It shows two route entries:

- Propagated**: No
- Edit routes**:

Destination	Target	Status
Q 0.0.0.0/0	Q igw-0be0995cf97bb7cb7	Active
- Propagated**: No
- Edit routes**:

Destination	Target	Status
Q 2.0.0.0/24	Q pccx-06bdcbf0e45cd253d	-

- Now connect Instance1(vm4) and Instance2(vm5) to the console
- Ping the Instances with Private IP addresses.
- Type Ping 2.0.0.0.9 which is IP address of vm1 in console of vm4.

The screenshot shows the AWS CloudShell terminal window. The URL is <https://ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=ap-south-1&connType=standard&instanceId=i-023dc28e7a05e2f7e&User=ec2-user&Port=224/>. The terminal session is connected to instance **i-023dc28e7a05e2f7e (vm-4)**. The user is running the command **ping 2.0.0.0.9**. The output shows the ping results between the two instances:

```

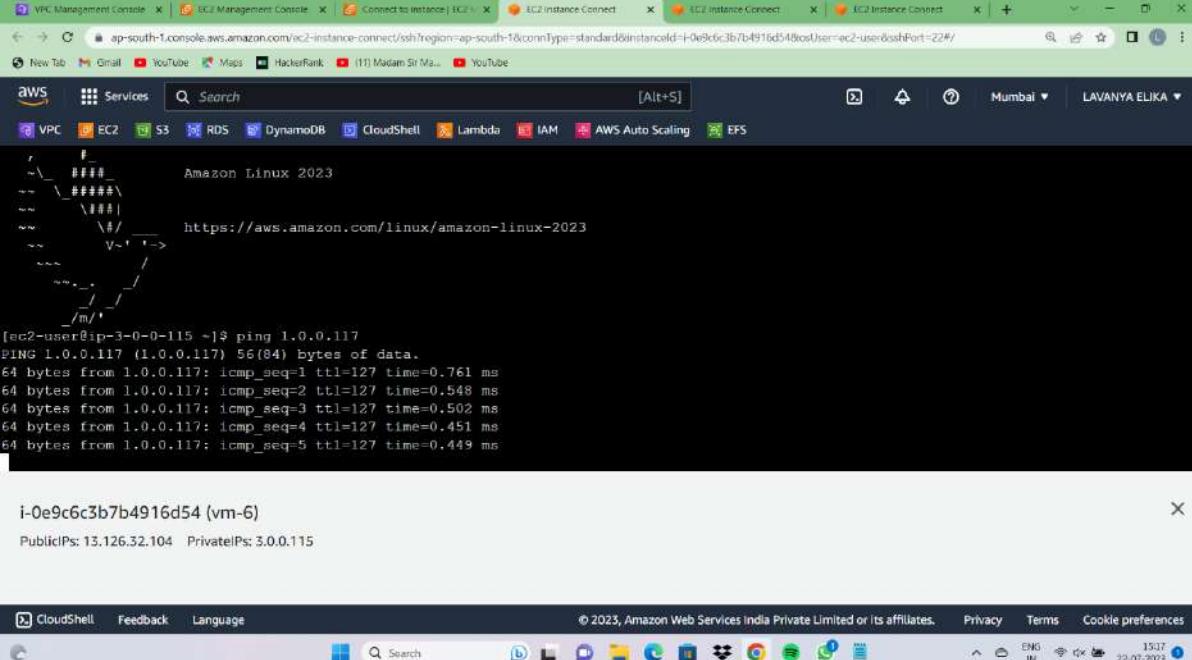
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-1-0-0-117 ~]$ ping 2.0.0.58
PING 2.0.0.58 (2.0.0.58) 56(84) bytes of data.
64 bytes from 2.0.0.58: icmp_seq=1 ttl=127 time=0.774 ms
64 bytes from 2.0.0.58: icmp_seq=2 ttl=127 time=0.524 ms
64 bytes from 2.0.0.58: icmp_seq=3 ttl=127 time=0.433 ms
64 bytes from 2.0.0.58: icmp_seq=4 ttl=127 time=0.485 ms

```

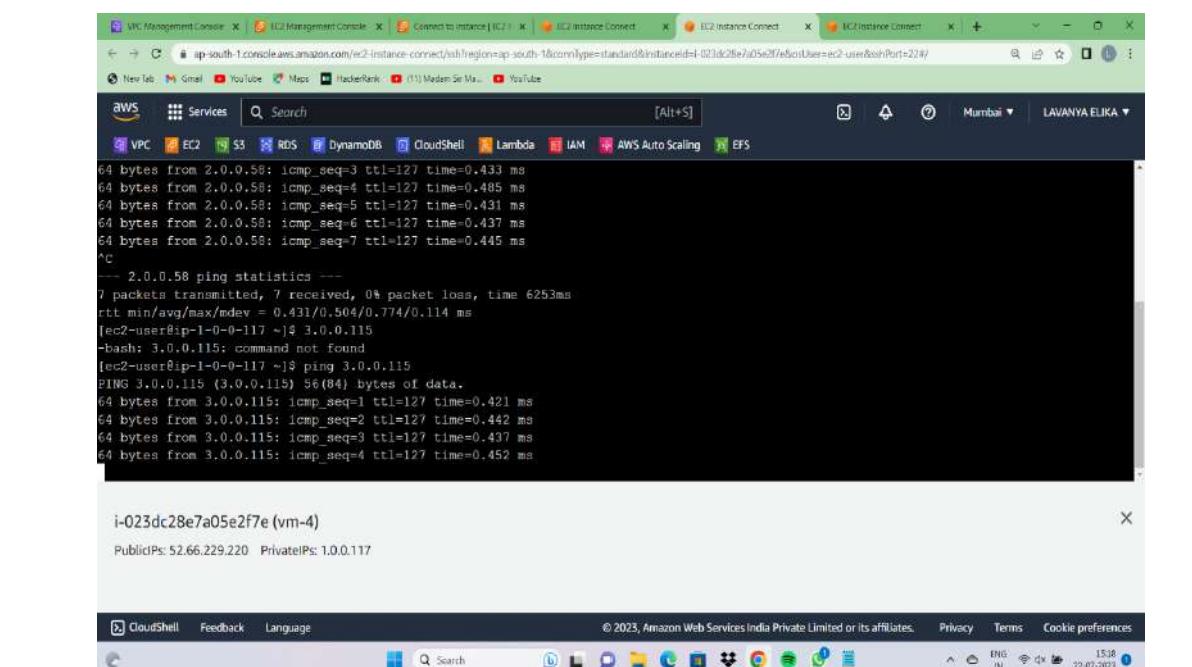
Below the terminal, the instance details are shown: **i-023dc28e7a05e2f7e (vm-4)**, **PublicIPs: 52.66.229.220**, and **PrivateIPs: 1.0.0.117**.

- Now Type ping 1.0.0.117 on VM-5 console



```
[ec2-user@ip-3-0-0-115 ~]$ ping 1.0.0.117
PING 1.0.0.117 (1.0.0.117) 56(84) bytes of data.
64 bytes from 1.0.0.117: icmp_seq=1 ttl=127 time=0.761 ms
64 bytes from 1.0.0.117: icmp_seq=2 ttl=127 time=0.548 ms
64 bytes from 1.0.0.117: icmp_seq=3 ttl=127 time=0.502 ms
64 bytes from 1.0.0.117: icmp_seq=4 ttl=127 time=0.451 ms
64 bytes from 1.0.0.117: icmp_seq=5 ttl=127 time=0.449 ms
```

i-0e9c6c3b7b4916d54 (vm-6)
PublicIPs: 13.126.32.104 PrivateIPs: 3.0.0.115



```
64 bytes from 2.0.0.58: icmp_seq=3 ttl=127 time=0.433 ms
64 bytes from 2.0.0.58: icmp_seq=4 ttl=127 time=0.495 ms
64 bytes from 2.0.0.58: icmp_seq=5 ttl=127 time=0.431 ms
64 bytes from 2.0.0.58: icmp_seq=6 ttl=127 time=0.437 ms
64 bytes from 2.0.0.58: icmp_seq=7 ttl=127 time=0.445 ms
^C
--- 2.0.0.58 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6253ms
rtt min/avg/max/mdev = 0.431/0.504/0.774/0.114 ms
[ec2-user@ip-1-0-0-117 ~]$ ping 3.0.0.115
-bash: 3.0.0.115: command not found
[ec2-user@ip-1-0-0-117 ~]$ ping 3.0.0.115
PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=0.421 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=0.442 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=0.437 ms
64 bytes from 3.0.0.115: icmp_seq=4 ttl=127 time=0.452 ms
```

i-023dc28e7a05e2f7e (vm-4)
PublicIPs: 52.66.229.220 PrivateIPs: 1.0.0.117

- Now create peering connection between VPC4 and VPC6 and ping private IP of VM4 in Console of VM6 and ping private IP of VM6 in Console of VM4
- Consider VPC4 as requester and VPC6 as accepter for creating peering connection.

```

./m/
[ec2-user@ip-2-0-0-58 ~]$ ping 3.0.0.115
PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=0.400 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=0.188 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=0.193 ms
64 bytes from 3.0.0.115: icmp_seq=4 ttl=127 time=0.167 ms
64 bytes from 3.0.0.115: icmp_seq=5 ttl=127 time=0.157 ms
^C
--- 3.0.0.115 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4170ms
rtt min/avg/max/mdev = 0.157/0.221/0.400/0.090 ms
[ec2-user@ip-2-0-0-58 ~]$ ping 1.0.0.117
PING 1.0.0.117 (1.0.0.117) 56(84) bytes of data.
64 bytes from 1.0.0.117: icmp_seq=1 ttl=127 time=0.444 ms
64 bytes from 1.0.0.117: icmp_seq=2 ttl=127 time=0.421 ms
64 bytes from 1.0.0.117: icmp_seq=3 ttl=127 time=0.423 ms

```

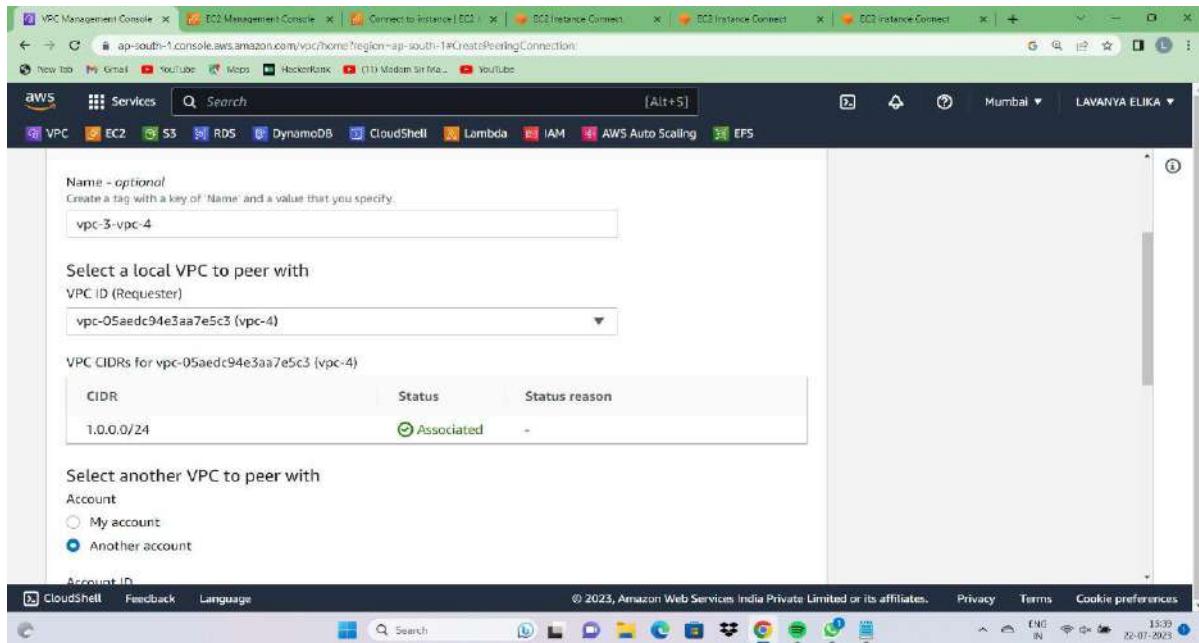
i-03f287ca5f65b4c59 (vm-5)
PublicIPs: 13.235.95.153 PrivateIPs: 2.0.0.58

- Similarly, create peering connection between VPC5 and VPC6 and name it as VPC-5-VPC-6.
Consider VPC5 as requester and VPC6 as accepter.
- After creating peering connection and editing route tables . Now ping Private IP addresses of VM2(2.0.0.5) in VM3 Console.
- Now ping Private IP addresses of VM3 (3.0.0.115) in VM2 Console
- Upto here, we completed 3 peering connections in ap-south-1 belongs to 049478797685 and another 3 peering connections in eu-north-1 belongs to 049478797685
- Until now we completed peering connections vpc1-vpc2, vpc1-vpc3, vpc2-vpc-3, vpc4-vpc5, vpc4-vpc6,vpc5-vpc6

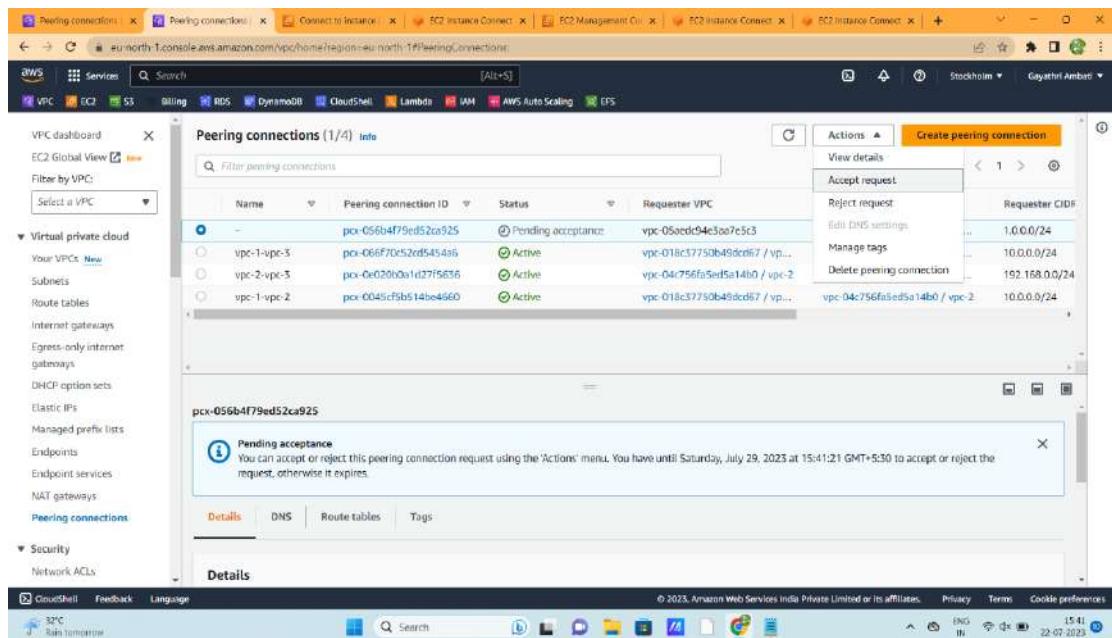
Peering connection between two cross regions and two different accounts

- For creating peering connection between two different accounts
- Consider one account as requester and another as accepter.
- For creating peering connection between vpc3 and vpc4
- Create peering connection in ap-south-1 belongs to account 049478797685.

- Select another account option in vpc peering connection vpc peering
- Consider vpc4 as requester and vpc3 as accepter and send request to vpc3



- In Account ID give 896571820628 and give accepter vpc ID and create peering connection
- Then request send to the another account .routes can be edited only after request is accepted.



The screenshot shows the AWS VPC Peering Connections page. A green banner at the top states: "Your VPC peering connection (pxc-056b4f79ed52ca925) has been established. To send and receive traffic across this VPC peering connection, you must add a route to the peered VPC in one or more of your VPC route tables." Below this, a table lists four peering connections:

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDR
vpc-3-vpc-4	pxc-056b4f79ed52ca925	Active	vpc-05aed94e3aa7e5c3	vpc-00c2e58bd5e415d1d / vpc-2	1.0.0.0/24
vpc-1-vpc-2	pxc-0045cf5b514be4660	Active	vpc-018c37750b49cd67 / vpc-3	vpc-04c756fa5ed5a14b0 / vpc-2	10.0.0.0/24
vpc-1-vpc-3	pxc-066f70c52cd5454a6	Active	vpc-018c37750b49cd67 / vpc-3	vpc-00c2e58bd5e415d1d / vpc-2	10.0.0.0/24
vpc-2-vpc-3	pxc-0e020b0a1d27f5636	Active	vpc-04c756fa5ed5a14b0 / vpc-2	vpc-00c2e58bd5e415d1d / vpc-3	102.168.0.0/24

Below the table, a specific peering connection is selected: "pxc-056b4f79ed52ca925 / vpc-3-vpc-4". The "Details" tab is active, showing the peering connection ID and VPC IDs.

- After request is accepted in stockholm then edit the routes in the both the accounts.

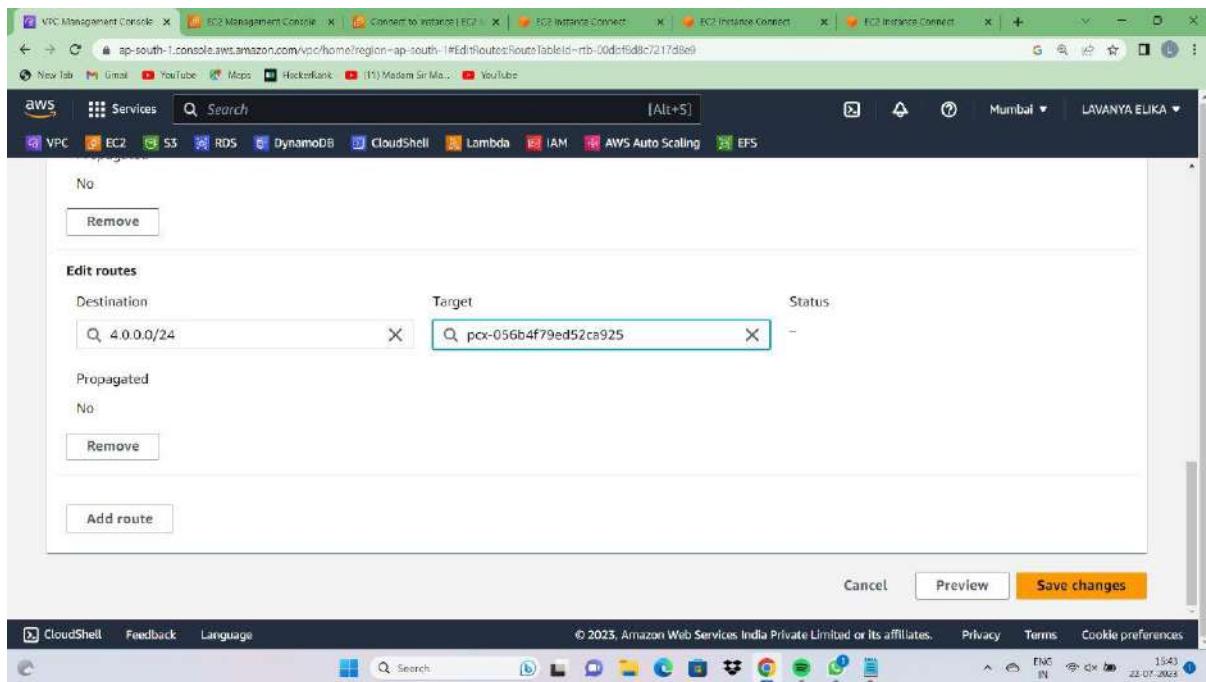
The screenshot shows the AWS Route Tables page. The URL indicates the route table being edited is "rtb-09993147638952ac9". The page title is "Edit routes".

The "Edit routes" table lists the following routes:

Destination	Target	Status	Propagated
4.0.0.0/24	local	Active	No
192.168.0.0/24	pxc-0e020b0a1d27f5636	Active	No
10.0.0.0/24	pxc-066f70c52cd5454a6	Active	No
0.0.0.0/0	igw-054b1046ffa406e70	Active	No
1.0.0.0/24	pxc-056b4f79ed52ca925	-	No

At the bottom right, there are "Cancel", "Preview", and "Save changes" buttons.

- in edit routes give CIDR of VPC-4 in RT-3 of VPC-3 and select target as peering connection as (VPC3-VPC4)
- in edit routes give CIDR of VPC-3 in RT-4 of VPC-4 and select target as peering connection as (VPC3-VPC4)



- after editing the routes now ping private ip address of VPC-3 belongs to stockholm into vm-4 in mumbai.
- now ping private ip address of VPC-4 belongs to mumbai into vm-3 in Stockholm.

```

PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=0.421 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=0.442 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=0.437 ms
64 bytes from 3.0.0.115: icmp_seq=4 ttl=127 time=0.452 ms
64 bytes from 3.0.0.115: icmp_seq=5 ttl=127 time=0.433 ms
64 bytes from 3.0.0.115: icmp_seq=6 ttl=127 time=0.435 ms
^C
--- 3.0.0.115 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5180ms
rtt min/avg/max/mdev = 0.421/0.436/0.452/0.009 ms
[ec2-user@ip-1-0-0-117 ~]$ ping 4.0.0.72
PING 4.0.0.72 (4.0.0.72) 56(84) bytes of data.
64 bytes from 4.0.0.72: icmp_seq=1 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=3 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=4 ttl=127 time=135 ms

```

i-023dc28e7a05e2f7e (vm-4)
PublicIPs: 52.66.229.220 PrivateIPs: 1.0.0.117

```

i-0cf2b7535472c212a user@ip-4-0-0-72: ~ % ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=1.24 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=1.10 ms
^C
--- 10.0.0.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 100ms
rtt min/avg/max/mdev = 1.099/1.171/1.244/0.072 ms
i-0cf2b7535472c212a user@ip-4-0-0-72: ~ % ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=127 time=1.71 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=127 time=1.46 ms
^C
--- 192.168.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 100ms
rtt min/avg/max/mdev = 1.481/1.594/1.708/0.113 ms
i-0cf2b7535472c212a user@ip-4-0-0-72: ~ % ping 1.0.0.117
PING 1.0.0.117 (1.0.0.117) 56(84) bytes of data.
64 bytes from 1.0.0.117: icmp_seq=1 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=3 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=4 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=5 ttl=127 time=135 ms
^C
--- 1.0.0.117 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 200ms
rtt min/avg/max/mdev = 134.663/134.676/134.687/0.008 ms
i-0cf2b7535472c212a user@ip-4-0-0-72: ~ %

```

i-0cf2b7535472c212a (vm-3)
PublicIP: 16.171.68.5 PrivateIP: 4.0.0.72

- For creating peering connection between vpc3 and vpc5 . Now consider Stockholm as requester and Mumbai as accepter.
- Create peering connection in eu-north-1 belongs to account 896571820628.
- Select another account option in vpc peering connection vpc peering
- Consider vpc3 as requester and vpc5 as accepter and send request to vpc5

Create peering connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them

Peering connection settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.
vpc-3-vpc-5

Select a local VPC to peer with
VPC ID (requester)
vpc-00c2e58bd6e415d1d (vpc-3)

CIDR	Status	Status reason
4.0.0.0/24	Associated	-

Select another VPC to peer with
Account
 My account
 Another account
Account ID:

- In Account ID give 049478797685 and give accepter vpc ID of vpc5 and create peering connection

Name	Peering connection ID	Status	Requester VPC	Accepter VPC	Requester CIDR
vpc-3-vpc-4	pcx-056b4f79ed52ca925	Active	vpc-05aedc94e3aa7e5c3	vpc-002e58bd6e415d1d / vp...	1.0.0.0/24
vpc-1-vpc-3	pcx-06970c52e0545a6	Active	vpc-016c37750b49dd67 / vp...	vpc-002e58bd6e415d1d / vp...	10.0.0.0/24
vpc-2-vpc-3	pcx-0e020b01d2715636	Active	vpc-04c756fa5ed5a14b0 / vp...	vpc-002e58bd6e415d1d / vp...	192.168.0.0/24
vpc-3-vpc-5	pcx-0d5988e84fa8d9195	Pending acceptance	vpc-002e58bd6e415d1d / vp...	vpc-0c76c41134d2e668d	4.0.0.0/24
vpc-1-vpc-2	pcx-0045ef5b514b4660	Active	vpc-016c37750b49dd67 / vp...	vpc-04c756fa5ed5a14b0 / vp...	10.0.0.0/24

Then request send to the another account . peering connection remains in pending until accepter accepts the requests.we can edit routes only after request is accepted (peering connection status becomes active)

Name	Peering connection ID	Status	Requester	Accepter
vpc1-vpc2	pcx-0a142f4efe74a13d0	Deleted		
vpc-4-vpc-6	pcx-090d3554f829bea06	Active		vpc-C
vpc-4-vpc-5	pcx-06bdcfb0e45cd253d	Active		vpc-C
vpc-3-vpc-4	pcx-056b4f79ed52ca925	Active		vpc-C
vpc-5-vpc-6	pcx-061319add75587aad	Active		vpc-C
-	pcx-0d5988e84fa8d9195	Pending acceptance	vpc-002e58bd6e415d1d	vpc-C

The screenshot shows the AWS VPC Peering Connections page. A green banner at the top indicates a new peering connection request from VPC 5 to VPC 3. The main table lists five existing peering connections:

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDR
vpc-3-vpc-4	pcx-056b4f79d52ca925	Active	vpc-05endk94e3aa7e5c3	vpc-00c2e58bd6e415d1d / vp...	1.0.0.0/24
vpc-2-vpc-3	pcx-0e020b0a1d27f5636	Active	vpc-04756fa5ed5a14b0 / vpc-2	vpc-00c2e58bd6e415d1d / vp...	192.168.0.0/24
vpc-3-vpc-5	pcx-0d5988e84fa8d9195	Active	vpc-002e58bd6e415d1d / vp...	vpc-076c41134d2c668d	4.0.0.0/24
vpc-1-vpc-2	pcx-0045dc5b514be4660	Active	vpc-018c7750b49dd67 / vp...	vpc-04756fa5ed5a14b0 / vpc-2	10.0.0.0/24
vpc-1-vpc-3	pcx-066f70c52cf545465	Active	vpc-018c37750b49dd67 / vp...	vpc-00c2e58bd6e415d1d / vp...	10.0.0.0/24

- Now peering connection is active for vpc3-vpc5, so now we can edit routes in for both vpc3 and vpc5 and give respective CIDRs in routes and save changes
- Now ping private address of VPC-3 belongs to stockholm into vm-5 in mumbai.
- now ping private ip address of VPC-5 belongs to mumbai into vm-3 in Stockholm

```

64 bytes from 192.168.0.5: icmp_seq=1 ttl=127 time=1.71 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=127 time=1.48 ms
^C
--- 192.168.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.481/1.594/1.708/0.113 ms
[ec2-user@ip-4-0-0-72 ~]$ ping 1.0.0.117
PING 1.0.0.117 (1.0.0.117) 56(84) bytes of data.
64 bytes from 1.0.0.117: icmp_seq=1 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=3 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=4 ttl=127 time=135 ms
64 bytes from 1.0.0.117: icmp_seq=5 ttl=127 time=135 ms
^C
--- 1.0.0.117 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 134.663/134.678/134.685/0.008 ms
[ec2-user@ip-4-0-0-72 ~]$ ping 2.0.0.58
PING 2.0.0.58 (2.0.0.58) 56(84) bytes of data.
64 bytes from 2.0.0.58: icmp_seq=1 ttl=127 time=138 ms
64 bytes from 2.0.0.58: icmp_seq=2 ttl=127 time=138 ms
64 bytes from 2.0.0.58: icmp_seq=3 ttl=127 time=138 ms
64 bytes from 2.0.0.58: icmp_seq=4 ttl=127 time=138 ms
^C
--- 2.0.0.58 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4007ms
rtt min/avg/max/mdev = 139.669/137.685/137.700/0.011 ms
[ec2-user@ip-4-0-0-72 ~] $ i-0cf2b7535472c212a (vm-3)
PublicIPs: 16.171.68.5 PrivateIPs: 4.0.0.72

```

```

aws Management Console EC2 Management Console Connect to instance (EC2) EC2 Instance Connect EC2 Instance Connect EC2 Instance Connect
ap-south-1.console.aws.amazon.com/ec2-instance-connect/vch?region=ap-south-1&connType=standard&instanceId=i-03f287ca5f65b4c59&ec2User=ec2-user&dashPort=22#
New Tab Gmail YouTube Maps HackerRank (11) Madm Sir Ma... YouTube
Services Search [Alt+S]
VPC EC2 S3 RDS DynamoDB CloudShell Lambda IAM AWS Auto Scaling EPS
64 bytes from 1.0.0.117: icmp_seq=3 ttl=127 time=0.423 ms
64 bytes from 1.0.0.117: icmp_seq=4 ttl=127 time=0.475 ms
^C
--- 1.0.0.117 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3121ms
rtt min/avg/max/mdev = 0.421/0.440/0.475/0.021 ms
[ec2-user@ip-2-0-0-58 ~]$ ping 4.0.0.72
PING 4.0.0.72 (4.0.0.72) 56(84) bytes of data.
64 bytes from 4.0.0.72: icmp_seq=1 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=3 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=4 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=5 ttl=127 time=135 ms
^C
--- 4.0.0.72 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 134.901/134.934/134.993/0.039 ms
[ec2-user@ip-2-0-0-58 ~]$

```

i-03f287ca5f65b4c59 (vm-5)
PublicIPs: 13.235.95.153 PrivateIPs: 2.0.0.58

Create peering connection for vpc3-vpc6 considering vpc3 as requester and vpc-6 accepter.

A VPC peering connection pxc-02d445c445aa04b0a / vpc-3-vpc-6 has been requested.
The owner of vpc-03863ead6933e9556 must accept the peering connection.

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDR
vpc-1-vpc-2	pxc-0045cf5b514be4660	Active	vpc-018c377501d9dc0d67 / vpc-1	vpc-04c756f5ed5a1ab0 / vpc-2	10.0.0.0/24
vpc-1-vpc-3	pxc-066f70c52cd5454a6	Active	vpc-018c377501d9dc0d67 / vpc-1	vpc-00c2e58bd6e415d1d / vpc-3	10.0.0.0/24
vpc-2-vpc-3	pxc-0e20bb0a1d27f5636	Active	vpc-04c756fa5ed5a1ab0 / vpc-2	vpc-00c2e58bd6e415d1d / vpc-3	192.168.0.0/24
vpc-3-vpc-4	pxc-056bf4f79ed52ca925	Active	vpc-05aedc94e5aa7e5c5	vpc-00c2e58bd6e415d1d / vpc-4	1.0.0.0/24
vpc-3-vpc-5	pxc-0d5988a84fa8d9195	Active	vpc-00c2e58bd6e415d1d / vpc-5	vpc-076c41134d2e668d	4.0.0.0/24
vpc-3-vpc-6	pxc-02d445c445aa04b0a	Active	vpc-00c2e58bd6e415d1d / vpc-3	vpc-03863ead6933e9556	4.0.0.0/24

Select a peering connection above

```

64 bytes from 1.0.0.117: icmp seq=4 ttl=127 time=0.451 ms
64 bytes from 1.0.0.117: icmp_seq=5 ttl=127 time=0.449 ms
64 bytes from 1.0.0.117: icmp_seq=6 ttl=127 time=0.436 ms
64 bytes from 1.0.0.117: icmp_seq=7 ttl=127 time=0.447 ms
...
-- 1.0.0.117 ping statistics --
7 packets transmitted, 0% packet loss, time 6243ms
rtt min/avg/max/mdev = 0.436/0.513/0.761/0.107 ms
[ec2-user@ip-3-0-0-115 ~]$ ping 3.0.0.115
PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=135 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=135 ms
...
-- 3.0.0.115 ping statistics --
3 packets transmitted, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 135.315/135.319/135.323/0.003 ms
[ec2-user@ip-3-0-0-115 ~]

```

i-0e9c6c3b7b4916d54 (vm-6)
PublicIPs: 13.126.32.104 PrivateIPs: 3.0.0.115

```

64 bytes from 1.0.0.117: icmp seq=4 ttl=127 time=0.451 ms
64 bytes from 1.0.0.117: icmp_seq=5 ttl=127 time=0.449 ms
64 bytes from 1.0.0.117: icmp_seq=6 ttl=127 time=0.436 ms
64 bytes from 1.0.0.117: icmp_seq=7 ttl=127 time=0.447 ms
...
-- 1.0.0.117 ping statistics --
7 packets transmitted, 0% packet loss, time 6243ms
rtt min/avg/max/mdev = 0.436/0.513/0.761/0.107 ms
[ec2-user@ip-3-0-0-115 ~]$ ping 3.0.0.115
PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=135 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=135 ms
...
-- 3.0.0.115 ping statistics --
3 packets transmitted, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 135.315/135.319/135.323/0.003 ms
[ec2-user@ip-3-0-0-115 ~]

```

i-0e9c6c3b7b4916d54 (vm-6)
PublicIPs: 13.126.32.104 PrivateIPs: 3.0.0.115

Create peering connection for vpc1-vpc4 considering vpc1 as requester and vpc-4 as accepter.

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDR
vpc-1-vpc-4	pxc-013e1a65e87413a28	Pending acceptance	vpc-011c57750b49dd67 / vp...	vpc-05aecd94e5aa7e5c5	10.0.0.0/24
vpc-3-vpc-5	pxc-0d5988e84fa8d9195	Active	vpc-002e5bd6e415d1d / vp...	vpc-0c76e41134d2e668d	4.0.0.0/24
vpc-3-vpc-4	pxc-056bf479ed52ca925	Active	vpc-05aecd94e5aa7e5c5	vpc-002e5bd6e415d1d / vp...	1.0.0.0/24
vpc-1-vpc-2	pxc-0045cf5b514be4660	Active	vpc-011c57750d49dd67 / vp...	vpc-04c756f5ed5a14b0 / vpc-2	10.0.0.0/24
vpc-2-vpc-3	pxc-0e20bd1d2715636	Active	vpc-04c756f5ed5a14b0 / vpc-2	vpc-002e5bd6e415d1d / vp...	192.168.0.0/24
vpc-3-vpc-6	pxc-02d445c445aa04b0a	Active	vpc-002e5bd6e415d1d / vp...	vpc-03863ea6d6933e9556	4.0.0.0/24
vpc-1-vpc-3	pxc-066f70c52cd5454a6	Active	vpc-011c57750b49dd67 / vp...	vpc-002e5bd6e415d1d / vp...	10.0.0.0/24

i-0861f72a92d5e41f5 (vm-1)

```

i-0861f72a92d5e41f5 (vm-1)
PrivateIPs: 10.0.0.9

CloudShell Feedback Language
3°C
Now record
Search
EN IN 22-07-2023

```

i-023dc28e7a05e2f7e (vm-4)

```

i-023dc28e7a05e2f7e (vm-4)
PublicIPs: 52.66.229.220 PrivateIPs: 1.0.0.117

CloudShell Feedback Language
3°C
Now record
Search
EN IN 22-07-2023

```

Create peering connection for vpc1-vpc5 considering vpc1 as requester and vpc5 as accepter.

A VPC peering connection pcr-02586d362e8cfc has been requested.
The owner of vpc-0c76c41134d2e4668d must accept the peering connection.

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDR
vpc-3-vpc-4	pcx-01604f79ed52ca925	Active	vpc-058bd6e415d1d / vpc-0	vpc-002e58bd6e415d1d / vpc-0	1.0.0.0/24
vpc-1-vpc-5	pcx-023884fd62e8cfc	Active	vpc-011c87750b45ddcf7 / vpc-0	vpc-076d41134d2e668d	10.0.0.0/24
vpc-3-vpc-6	pcx-02d445c445au4060a	Active	vpc-002e58bd6e415d1d / vpc-0	vpc-03865eadb935e9556	4.0.0.0/24
vpc-1-vpc-4	pcx-013e1a65e87415a28	Active	vpc-011c87750b45ddcf7 / vpc-0	vpc-05865eadb945aae7e43	10.0.0.0/24
vpc-1-vpc-3	pcx-066f70c52cd545a6	Active	vpc-011c87750b45ddcf7 / vpc-0	vpc-002e58bd6e415d1d / vpc-0	10.0.0.0/24
vpc-1-vpc-2	pcx-0045cf5b514be4600	Active	vpc-011c87750b45ddcf7 / vpc-0	vpc-04c756faed5a1fb0 / vpc-2	10.0.0.0/24
vpc-3-vpc-5	pcx-045988efaf8a89195	Active	vpc-002e58bd6e415d1d / vpc-0	vpc-0c76d41134d2e668d	4.0.0.0/24
vpc-2-vpc-3	pcx-0e02050a1d27f5636	Active	vpc-04c756faed5a1fb0 / vpc-2	vpc-002e58bd6e415d1d / vpc-0	192.168.0.0/24

```

64 bytes from 4.0.0.72: icmp_seq=2 ttl=127 time=1.03 ms
64 bytes from 4.0.0.72: icmp_seq=3 ttl=127 time=1.07 ms
^C
--- 4.0.0.72 ping statistics ---
3 packets transmitted, 0 received, 0% packet loss, time 200ms
rtt min/avg/max/mdev = 134.092/134.101/134.106/0.000 ms
[ec2-user@ip-10-0-0-9 ~]$ ping 2.0.0.58
PING 2.0.0.58 (2.0.0.58) 56(44) bytes of data.
64 bytes from 2.0.0.58: icmp_seq=1 ttl=127 time=137 ms
64 bytes from 2.0.0.58: icmp_seq=2 ttl=127 time=137 ms
64 bytes from 2.0.0.58: icmp_seq=3 ttl=127 time=137 ms
64 bytes from 2.0.0.58: icmp_seq=4 ttl=127 time=137 ms
64 bytes from 2.0.0.58: icmp_seq=5 ttl=127 time=137 ms
^C
--- 2.0.0.58 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 400ms
rtt min/avg/max/mdev = 136.595/136.650/136.715/0.041 ms
[ec2-user@ip-10-0-0-9 ~]$ 

```

i-0861f72a92d5e41f5 (vm-1)

PrivateIPs: 10.0.9


```

64 bytes from 4.0.0.72: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=3 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=4 ttl=127 time=135 ms
64 bytes from 4.0.0.72: icmp_seq=5 ttl=127 time=135 ms
^C
--- 4.0.0.72 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 134.901/134.934/134.993/0.039 ms
[ec2-user@ip-2-0-0-58 ~]$ ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(44) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=137 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=137 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=127 time=137 ms
^C
--- 10.0.0.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 136.605/136.622/136.647/0.018 ms
[ec2-user@ip-2-0-0-58 ~]$ 

```

i-03f287ca5f65b4c59 (vm-5)

PublicIPs: 13.235.95.153 PrivateIPs: 2.0.0.58

Create peering connection for vpc1-vpc6 considering vpc1 as requester and vpc-6 as accepter.

Name	Peer connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDR
vpc-3-vpc-5	pxc-0d5988e84fa8d9195	Active	vpc-0c2e58bd6e415d1d / v...	vpc-0c7641134d2e66bd	4.0.0.0/24
vpc-1-vpc-6	pxc-018c00b4bf6a7f4d	Pending acceptance	vpc-018c37750d40dc0d67 / v...	vpc-03863ead6933e9556	10.0.0.0/24
vpc-1-vpc-3	pxc-066f70c52cd5454a6	Active	vpc-018c37750d49dd67 / v...	vpc-0c2e58bd6e415d1d / v...	10.0.0.0/24
vpc-1-vpc-5	pxc-03868d85629895e8c	Active	vpc-018c37750d49dd67 / v...	vpc-0c7641134d2e66bd	10.0.0.0/24
vpc-3-vpc-6	pxc-02d445c45454a010a	Active	vpc-00c2e58bd6e415d1d / v...	vpc-03863ead6933e9556	4.0.0.0/24
vpc-1-vpc-4	pxc-013e1a65a87413a28	Active	vpc-018c37750d49dd67 / v...	vpc-05a63ed94e5aa7e5c3	10.0.0.0/24
vpc-2-vpc-2	pxc-0045c5b514be4f660	Active	vpc-018c37750d49dd67 / v...	vpc-04c756fa5ed9a14b0 / vpc-2	10.0.0.0/24
vpc-2-vpc-3	pxc-0e020b0a1d227f5636	Active	vpc-00c2e58bd6e415d1d / v...	vpc-00c2e58bd6e415d1d / v...	192.168.0.0/24
vpc-3-vpc-4	pxc-056b4f79nid52ca925	Active	vpc-05a63ed94e5aa7e5c3	vpc-00c2e58bd6e415d1d / v...	1.0.0.0/24

```

[ec2-user@ip-10-0-0-9 ~]$ ping 3.0.0.115
PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=4 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=5 ttl=127 time=137 ms
...
--- 3.0.0.115 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 136.595/136.650/136.715/0.041 ms
[ec2-user@ip-10-0-0-9 ~]$ ping 3.0.0.119
PING 3.0.0.119 (3.0.0.119) 56(84) bytes of data.
...
--- 3.0.0.119 ping statistics ---
12 packets transmitted, 0 received, 100% packet loss, time 11427ms
[ec2-user@ip-10-0-0-9 ~]$ ping 3.0.0.115
PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=135 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=135 ms
64 bytes from 3.0.0.115: icmp_seq=4 ttl=127 time=135 ms
...
--- 3.0.0.115 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4007ms
rtt min/avg/max/mdev = 134.835/134.853/134.883/0.018 ms
[ec2-user@ip-10-0-0-9 ~]

```

i-0861f72a92d5e41f5 (vm-1)

PrivateIPs: 10.0.0.9

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 22-07-2023 16:07


```

[ec2-user@ip-3-0-0-115 ~]$ ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
...
--- 10.0.0.9 ping statistics ---
126 packets transmitted, 0 received, 100% packet loss, time 12996ms
[ec2-user@ip-3-0-0-115 ~]$ ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=137 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=137 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=127 time=137 ms
64 bytes from 10.0.0.9: icmp_seq=4 ttl=127 time=137 ms
...
--- 10.0.0.9 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 136.077/136.953/137.160/0.119 ms
[ec2-user@ip-3-0-0-115 ~]

```

i-0e9c6c3b7b4916d54 (vm-6)

PublicIPs: 13.126.32.104 PrivateIPs: 3.0.0.115

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 22-07-2023 16:07

Create peering connection for vpc2-vpc4 considering vpc2 as requester and vpc-4 as accepter.

Name	Peering connection ID	Status	Requester VPC	Accepter VPC	Requester CIDR
vpc-1-vpc-5	pxc-023868d562e8f5c68c	Active	vpc-018c37750b49dd67 / vp...	vpc-0c76c41134d2e668d	10.0.0.0/24
vpc-3-vpc-4	pxc-0568af79ed52ca925	Active	vpc-05aeed94e3aa7e5c3	vpc-0c2e58bd6e415d1d / vp...	1.0.0.0/24
vpc-1-vpc-3	pxc-0667f052cd5454a6	Active	vpc-018c37750b49dd67 / vp...	vpc-00c2e58bd6e415d1d / vp...	10.0.0.0/24
vpc-3-vpc-5	pxc-045988e8f4fa0ab9795	Active	vpc-00c2e58bd6e415d1d / vp...	vpc-0c76c41134d2e668d	4.0.0.0/24
vpc-1-vpc-6	pxc-02d445c4f5aa04b0a	Active	vpc-018c37750b49dd67 / vp...	vpc-03865ead933e9556	10.0.0.0/24
vpc-3-vpc-6	pxc-02d445c4f5aa04b0a	Active	vpc-00c2e58bd6e415d1d / vp...	vpc-03865ead933e9556	4.0.0.0/24
vpc-1-vpc-4	pxc-013e1a65e87413e28	Active	vpc-018c37750b49dd67 / vp...	vpc-05eedc94e3aa7e5c3	10.0.0.0/24
vpc-2-vpc-4	pxc-0a3609092235b31e3	Pending acceptance	vpc-04c756fa5ed5a14b0 / vp...	vpc-05eedc94e3aa7e5c3	192.168.0.0/16
vpc-2-vpc-3	pxc-0e0200e1d275f536	Active	vpc-04c756fa5ed5a14b0 / vp...	vpc-0c2e58bd6e415d1d / vp...	192.168.0.0/16
vpc-1-vpc-2	pxc-03865ead933e9556	Active	vpc-018c37750b49dd67 / vp...	vpc-0c2e58bd6e415d1d / vp...	10.0.0.0/24

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 22-07-2023 16:10

```

PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=127 time=1.37 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=127 time=1.16 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=127 time=1.16 ms
^C
--- 10.0.0.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.159/1.229/1.370/0.099 ms
PING 4.0.0.72 (4.0.0.72) 56(84) bytes of data.
64 bytes from 4.0.0.72: icmp_seq=1 ttl=127 time=1.48 ms
64 bytes from 4.0.0.72: icmp_seq=2 ttl=127 time=1.49 ms
^C
--- 4.0.0.72 ping statistics ---
3 packets transmitted, 2 received, 33% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.483/1.483/1.485/0.009 ms
[ec2-user@ip-192-168-0-5 ~]$ ping 1.0.0.117
PING 1.0.0.117 (1.0.0.117) 56(84) bytes of data.
64 bytes from 1.0.0.117: icmp_seq=1 ttl=127 time=137 ms
64 bytes from 1.0.0.117: icmp_seq=2 ttl=127 time=137 ms
64 bytes from 1.0.0.117: icmp_seq=3 ttl=127 time=137 ms
64 bytes from 1.0.0.117: icmp_seq=4 ttl=127 time=137 ms
^C
--- 1.0.0.117 ping statistics ---
6 packets transmitted, 6 received, 16.6667% packet loss, time 500ms
rtt min/avg/max/mdev = 137.151/137.160/137.215/0.032 ms
[ec2-user@ip-192-168-0-5 ~] $ i-Db1253dfeefbd9a9c (vm-2)
PublicIP: 16.171.160.160 PrivateIP: 192.168.0.5

CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
16:12 22-07-2023

AWS Services Search [Alt+S]
Peering connections Root tables | VPC M... Connect to instance EC2 Instance Connect EC2 Management Con... EC2 Instance Connect EC2 Instance Connect
New Tab Home YouTube Help HacketBash (T) Medium S... W... YouTube

PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=125 time=135 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=125 time=135 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=125 time=135 ms
^C
--- 10.0.0.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 135.40/135.069/135.124/0.038 ms
[ec2-user@ip-1-0-0-117 ~]$ ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=127 time=139 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=127 time=139 ms
64 bytes from 192.168.0.5: icmp_seq=3 ttl=127 time=139 ms
^C
--- 192.168.0.5 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3005ms
rtt min/avg/max/mdev = 138.650/138.658/138.662/0.005 ms
[ec2-user@ip-1-0-0-117 ~] $ i-023dc28e7a05e2f7e (vm-4)
PublicIP: 52.66.229.220 PrivateIP: 1.0.0.117

CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
16:12 22-07-2023

```

Create peering connection for vpc2-vpc5 considering vpc2 as requester and vpc-5 accepter.

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDR
vpc-3-vpc-5	pxc-0d5988e04fa8d9195	Active	vpc-002e58bd6e415d1d / v...	vpc-0c76c41134d2e668d	4.0.0.0/24
vpc-1-vpc-6	pxc-018cd006bf67f74d	Active	vpc-018c7750b40ddcf67 / v...	vpc-03865cd6935e9556	10.0.0.0/24
vpc-1-vpc-2	pxc-0045fc5b14b4660	Active	vpc-018c7750b40ddcf67 / v...	vpc-04c756faSed5a14b0 / vpc-2	10.0.0.0/24
vpc-3-vpc-6	pxc-02d445c45aa0400a	Active	vpc-002e58bd6e415d1d / v...	vpc-03865ead933e9556	4.0.0.0/24
vpc-1-vpc-5	pxc-023866d162e85c6c	Active	vpc-018c7750b40ddcf67 / v...	vpc-0c76c41134d2e668d	10.0.0.0/24
vpc-1-vpc-4	pxc-013e1a6587413c28	Active	vpc-018c7750b40ddcf67 / v...	vpc-05seed94e3aa7e5c3	10.0.0.0/24
vpc-1-vpc-3	pxc-068170c52cd545a6	Active	vpc-018c7750b40ddcf67 / v...	vpc-00c2e58bd6e415d1d / v...	10.0.0.0/24
vpc-2-vpc-4	pxc-0a3509092235031e3	Active	vpc-04c756faSed5a14b0 / vpc-2	vpc-05seed94e3aa7e5c3	192.168.0.0/24
vpc-2-vpc-5	pxc-0275285ad5d0c1b03	Pending acceptance	vpc-04c756faSed5a14b0 / vpc-2	vpc-0c76c41134d2e668d	192.168.0.0/24
vpc-3-vpc-4	pxc-056b4f79ed52ea925	Active	vpc-05eed94e3aa7e5c3	vpc-0c2e58bd6e415d1d / v...	1.0.0.0/24


```

^C
--- 1.0.0.117 ping statistics ---
6 packets transmitted, 5 received, 16.6667% packet loss, time 5006ms
rtt min/avg/max/mdev = 137.151/137.180/137.215/0.022 ms
[ec2-user@ip-192-168-0-5 ~]$ ping 2.0.0.58
PING 2.0.0.58 (2.0.0.58) 56(84) bytes of data.
64 bytes from 2.0.0.58: icmp_seq=1 ttl=127 time=136 ms
64 bytes from 2.0.0.58: icmp_seq=2 ttl=127 time=136 ms
64 bytes from 2.0.0.58: icmp_seq=3 ttl=127 time=136 ms
64 bytes from 2.0.0.58: icmp_seq=4 ttl=127 time=136 ms
64 bytes from 2.0.0.58: icmp_seq=5 ttl=127 time=136 ms
64 bytes from 2.0.0.58: icmp_seq=6 ttl=127 time=136 ms
^C
--- 2.0.0.58 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 136.290/136.352/136.414/0.038 ms
[ec2-user@ip-192-168-0-5 ~]$ ping 3.0.0.119
PING 3.0.0.115 (3.0.0.115) 56(84) bytes of data.
64 bytes from 3.0.0.115: icmp_seq=1 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=2 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=3 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=4 ttl=127 time=137 ms
64 bytes from 3.0.0.115: icmp_seq=5 ttl=127 time=137 ms
^C
--- 3.0.0.115 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 136.838/136.980/137.277/0.153 ms
[ec2-user@ip-192-168-0-5 ~]$ 

```

i-0b1253dfeefbd9a9c (vm-2)

PublicIPs: 16.171.160.160 PrivateIPs: 192.168.0.5

```

^C
--- 10.0.0.9 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 136.877/136.953/137.160/0.119 ms
[ec2-user@ip-3-0-0-115 ~]$ ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=127 time=137 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=127 time=137 ms
64 bytes from 192.168.0.5: icmp_seq=3 ttl=127 time=137 ms
64 bytes from 192.168.0.5: icmp_seq=4 ttl=127 time=137 ms
^C
--- 192.168.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 136.023/136.879/136.927/0.047 ms
[ec2-user@ip-3-0-0-115 ~]$ 

```

i-0e9c6c3b7b4916d54 (vm-6)

PublicIPs: 13.126.32.104 PrivateIPs: 3.0.0.115

TO TERMINATE THE INSTANCES USING LAMBDA FUNCTION

- open the lambda service and create a function.

The screenshot shows the AWS Lambda home page. At the top, there are several tabs: Peering connection, Lambda, Connect to instance, EC2 Instance Connect, EC2 Management Console, EC2 instance Connect, and EC2 instance Connect. Below the tabs, the AWS navigation bar includes Services, Search, and a list of services: VPC, EC2, S3, Billing, RDS, DynamoDB, CloudShell, Lambda, IAM, AWS Auto Scaling, and EFS. The Lambda service is selected. The main content area has a dark background with white text. It features the heading "AWS Lambda" and the subtext "lets you run code without thinking about servers." Below this, a note states: "You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration." To the right, a "Get started" box contains the text "Author a Lambda function from scratch, or choose from one of many preconfigured examples." A prominent orange "Create a function" button is at the bottom of the box. The browser's address bar shows the URL: <https://eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/begin>. The status bar at the bottom right indicates the date and time: 22-07-2023 16:21.

The screenshot shows the "Create function" wizard. The first step, "How it works", is displayed. It features a title "How it works" and a sub-section "Run". Below this is a code editor window showing a Node.js script:

```
1+ exports.handler = async (event) => {
2-     console.log(event);
3-     return 'Hello from Lambda!';
4-};
```

The browser's address bar shows the URL: <https://eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function?isFirst=true>. The status bar at the bottom right indicates the date and time: 22-07-2023 16:23.

- Now go to IAM role and create a role

The screenshot shows the AWS IAM Dashboard. On the left, there's a sidebar with options like 'Identity and Access Management (IAM)', 'Dashboard', 'Access management', 'Access reports', and 'Tools'. The main area has sections for 'Security recommendations' (with a red warning icon) and 'IAM resources' (User groups: 0, Users: 0, Roles: 16, Policies: 4, Identity providers: 0). A 'What's new' section lists recent updates. On the right, there's an 'AWS Account' summary with account ID, alias, sign-in URL, and quick links for security credentials and policy simulator.

- Specify permissions and select JSON
- Edit the code with EC2 start and stop policy code.
- Edit errors and click next.
- And create policy.

The screenshot shows the 'Create New Role' wizard at Step 2: 'Add permissions'. It displays a list of available policies under 'Permissions policies (864)'. The policies are filtered by name and include 'arn_for_ramagayathri...', 'startstop', 'AdministratorAccess', 'PowerUserAccess', 'ReadOnlyAccess', 'AWSCloudFormationFullAccess', 'CloudFrontFullAccess', 'AWSCloudHSMFullAccess', and 'AWSCloudHSMRea...'. There are also sections for 'Properties' (Type, Path, Used as) and 'Step 3: Name, review, and create'.

The screenshot shows the AWS IAM 'Create policy' wizard at Step 1: Specify permissions. The policy editor displays the following JSON:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Start",
                "ec2:Stop"
            ],
            "Resource": "*"
        }
    ]
}

```

The screenshot shows the AWS IAM 'Create policy' wizard at Step 1: Specify permissions. The policy editor displays the following JSON:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Start",
                "ec2:Stop"
            ],
            "Resource": "*"
        }
    ]
}

```

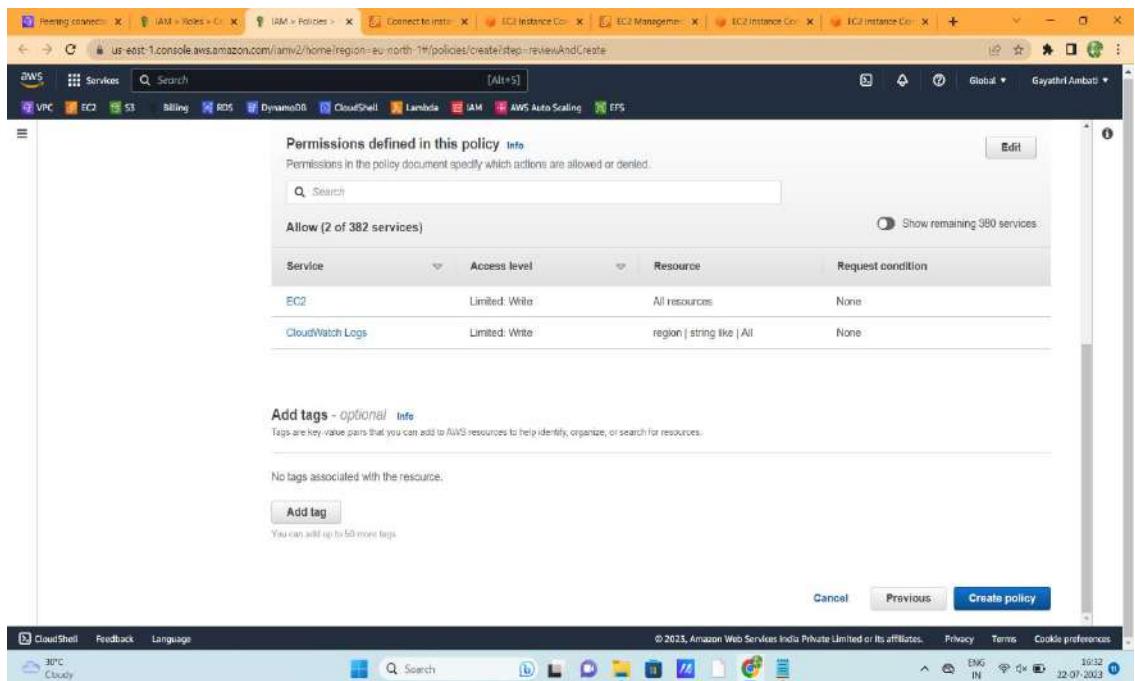
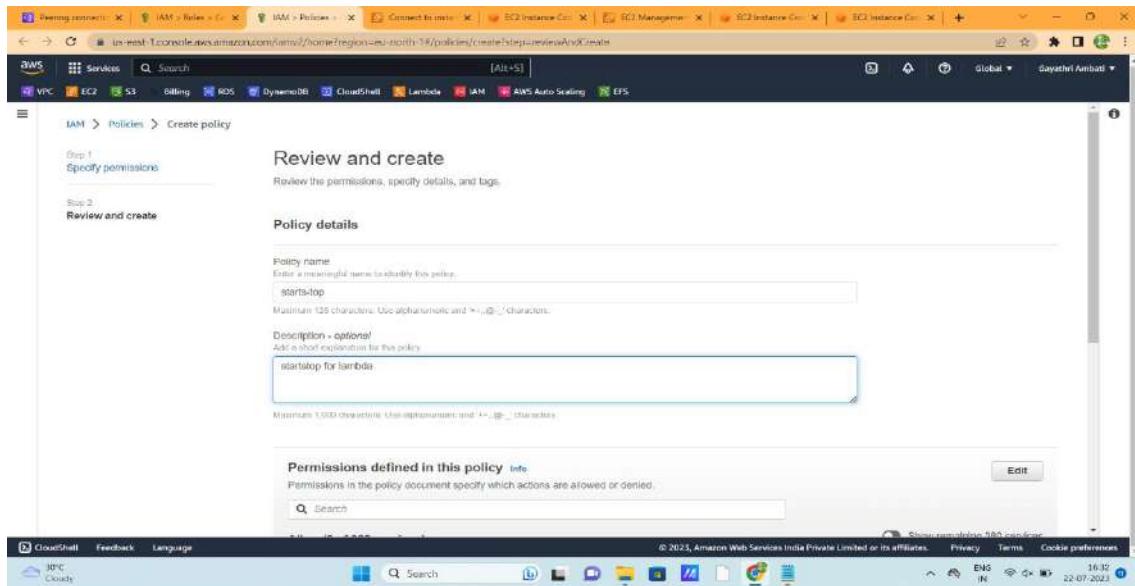
The screenshot shows the AWS IAM 'Create policy' wizard at Step 1: Specify permissions. The policy editor displays the following JSON:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Start",
                "ec2:Stop"
            ],
            "Resource": "*"
        }
    ]
}

```

- In create role page Add Permissions and select startstop policy and create role
- In lambda page select the role we created and create function
- In lambda function page go to code option give stopEC2lambdafunction code and change the region and instance id click on the deploy.
- In Test give event as mytest and then click on Test.
- Now the code start executing and after its execution we can check that the instance has been stopped.



Screenshot of the AWS IAM Policies page showing a list of existing policies:

Policy name	Type	Used as	Description
sdm_for_ramayayati_115780	Customer managed		Permissions policy (1)
s3or_for_ramayayati_583704	Customer managed		Permissions policy (1)
s3or_for_ramayayati_ae5c3e	Customer managed		Permissions policy (1)
startstop	Customer managed		startstop for lambda
startstop	Customer managed		Permissions policy (1)
AdministratorAccess	AWS managed - job function	None	Provides full access to AWS services as...
PowerUserAccess	AWS managed - job function	None	Provides full access to AWS services a...
ReadOnlyAccess	AWS managed - job function	None	Provides read-only access to AWS serv...
AWSCloudFormationReadOnlyAccess	AWS managed	None	Provides access to AWS CloudFormat...
CloudFrontFullAccess	AWS managed	None	Provides full access to the CloudFront...

Screenshot of the AWS IAM Create Policy Step 1: Specify permissions page. The JSON editor shows the following policy document:

```

1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Effect": "Allow",
6:       "Action": [
7:         "logs:CreateLogGroup",
8:         "logs:CreateLogStream",
9:         "logs:PutLogEvents"
10:      ],
11:      "Resource": "*arn:aws:logs:*:*"
12:    },
13:    {
14:      "Effect": "Allow",
15:      "Action": [
16:        "ec2:Start",
17:        "ec2:Stop",
18:        "ec2:Terminate"
19:      ],
20:      "Resource": "*"
21:    }
22:  ]
}

```

Screenshot of the AWS IAM Create Policy Step 2: Review and create page. The policy details are as follows:

- Policy name:** startstop
- Description (optional):** startstop
- Permissions defined in this policy:** (empty)

Screenshot of the AWS IAM Policy creation screen:

Permissions defined in this policy

Allow (2 of 382 services)

Service	Access level	Resource	Request condition
EC2	Limited; Write	All resources	None
CloudWatch Logs	Limited; Write	region string like All	None

Add tags – optional

No tags associated with the resource.

Add tag

You can add up to 50 more tags.

Create policy

Screenshot of the AWS IAM Role creation screen:

Add permissions

Permissions policies (Selected 1/865)

Policy name	Type	Description
s3or_for_ramagayathri...	Custom...	
s3or_for_ramagayathri...	Custom...	
s3or_for_ramagayathri...	Custom...	
startstop	Custom...	startstop for lambda
startstop	Custom...	startstop
AdministratorAccess	AWS m...	Provides full access to AWS services and resources.
PowerUserAccess	AWS m...	Provides full access to AWS services and resources, but does not allow management of Users and gr...
ReadOnlyAccess	AWS rn...	Provides read-only access to AWS services and resources.

Screenshot of the AWS IAM Roles list screen:

Role Iam created

Roles (17) Info

Role name: Iam

Trusted entities: AWS Service: lambda

Roles Anywhere - Info

Access AWS from your non AWS workloads

X.509 Standard

Temporary credentials

The screenshot shows three browser windows demonstrating the creation and deployment of an AWS Lambda function named "myfunc".

Top Window: AWS Lambda Functions page showing the function "myfunc" created 6 minutes ago.

Function name	Description	Package type	Runtime	Last modified
myfunc	-	Zip	Python 3.10	6 minutes ago

Middle Window: File Explorer showing the code source for "myfunc". The file "lambda_function.py" contains the following Python code:

```

1 #!/usr/bin/python3
2 import json
3 import boto3
4 region = "us-west-1"
5 instances = ["i-0861f72a93d5e41f5", "i-0c0b753547c213a", "i-0c0b753547c232a"]
6 w2 = boto3.client('ec2', region_name=region)
7
8 def lambda_handler(event, context):
9     ec2.terminate_instances(instance_ids=instances)
10    print("Terminated your instances. -> " + str(instances))

```

Bottom Window: AWS Lambda Test tab showing the successful update of the function "myfunc".

Screenshot of the AWS EC2 Management Console showing the Instances page. The page displays three instances: vm-4 (Running, t3.micro), vm-5 (Terminated, t3.micro), and vm-6 (Terminated, t3.micro). The sidebar shows navigation options like EC2 Dashboard, EC2 Global View, Events, and Instances.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
vm-4	i-023dc28e/a05e2f7e	Running	t3.micro	2/2 checks passed	No alarms
vm-5	i-03f287ca5f65b4c59	Terminated	t3.micro	-	No alarms
vm-6	i-0e9c6c3b7b4916d54	Terminated	t3.micro	-	No alarms

Select an instance

Screenshot of the AWS EC2 Management Console showing the Instances page. A message at the top indicates "Successfully terminated i-0efb2949d225558e4". The Instances table shows three instances: vm-1 (Shutting-down, t3.micro), vm-2 (Shutting-down, t3.micro), and vm-3 (Shutting-down, t3.micro). The sidebar shows navigation options like EC2 Dashboard, EC2 Global View, Events, and Instances.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
vm-1	i-0b8ae01020a67aad	Shutting-down	t3.micro	Initializing	No alarms	+ ap-south-1c	ec2-13-233-135-
vm-2	i-0e36d27e0b9e2d238	Shutting-down	t3.micro	Initializing	No alarms	+ ap-south-1c	ec2-52-66-230-1
vm-3	i-0efb2949d225558e4	Shutting-down	t3.micro	Initializing	No alarms	+ ap-south-1c	ec2-13-200-18-1

Instance: i-0efb2949d225558e4 (vm-3)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0efb2949d225558e4 (vm-3)	13.200.18.106 open address	172.31.29.80
IPv6 address	Instance state	Public IPv4 DNS
-	Shutting-down	ec2-13-200-18-106.ap-south-1.compute.amazonaws.com open address
Hostname type	Private IP DNS name (IPv4 only)	

CONCLUSION:

These architect helps to create a VPC peering connection is a networking connection between two VPCs that enables routing using each VPC's private IP addresses as if they were in the same network. VPC peering connections can be created between your own VPCs or with a VPC in another AWS account. VPC peering also supports inter-region peering.