BLACKBUCKS AWS INTERNSHIP REPORT

MULTI-TIER SCALABLE AWS ARCHITECT USING DIFFERENT ACCOUNTS

(- 12 TIER ARCHITECT)

SUBMITTED BY

1. A.SHYAM SATYA SIVA PRASAD REG:21B91A5403

2. A.VENKATA SRIRAM REG:21B91A5406

3. B.GNANA SATYA SWAROOP REG:21B91A5448

UNDER THE GUIDANCE OF "*MR. AASHU DEV* "

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

AWS Academy Accredited Educator

BlackBucks Aws Internship Report

TEAM:

- **A.VENKATA SRIRAM - 21B91A5406**
- **A.SHYAM SATYA SIVA PRASAD - 21B91A5403**
- **B.GNANA SATYA SWAROOP - 21B91A5448**

TITLE:

MULTI-TIER SCALABLE AWS ARCHITECT USING DIFFERENT ACCOUNTS

(- 12 TIER ARCHITECT)

ABSTRACT:

This **PROJECT** contains three different architects containing three different accounts which are connected by peering connections.

In this project the services used are,

Vpc,Ec2,Iam role,Cloudshell,Cloudwatch,SNS console,Docker hub,

Dynamo db,Lambda,Snapshot,RDS,S3.

Using the above services we created a **12-TIER ARCHITECT.**

CONTENTS

	Pg.no
1.What is cloud computing?	3-4
2. List of cloud platforms	4-8
3.Introduction to AWS	9-10
4.Why AWS?	10-18
5.AWS Services used	19-37
6.Implementation of Project	38
6.1.Introduction	38
6.2.Rough architecture	39
6.3.Final architecture	40
7.Implementation of Architecture	41
7.1.Account-1 Architect	41-55
7.2.Account-2 Architect	56-77
7.3.Account-3 Architect	78-99
8.Peering Connection	100-104
9.Summary	104

WHAT IS CLOUD COMPUTING?

- Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

- Cloud computing has a rich history that extends back to the 1960s, with the initial concepts of time-sharing becoming popularized via Remote Job Entry (RJE). The "data center" model, where users submitted jobs to operators to run on mainframes, was predominantly used during this era. This was a time of exploration and experimentation with ways to make large-scale computing power available to more users through time-sharing, optimizing the infrastructure, platform, and applications, and increasing efficiency for end users.

- The use of the "cloud" metaphor to denote virtualized services traces back to 1994, when it was used by General Magic to describe the universe of "places" that mobile agents in the Telescript environment could go. This metaphor is credited to David Hoffman, a General Magic communications employee, based on its long-standing use in networking and telecom. The expression cloud computing became more widely known in 1996 when the Compaq Computer Corporation drew up a business plan for future computing and the Internet. The company's ambition was to supercharge sales with "cloud computing-enabled applications". The business plan foresaw that online consumer file storage would most likely be commercially successful. As a result, Compaq decided to sell server hardware to internet service providers.

- In the 2000s, the application of cloud computing began to take shape with the establishment of Amazon Web Services in 2002, which allowed developers to build applications independently. In 2006 the beta version of Google Docs was released, Amazon Simple Storage Service, known as Amazon S3, and the Amazon Elastic Compute Cloud (EC2), in 2008 NASA's development of the first open-source software for deploying private and hybrid clouds.

- The following decade saw the launch of various cloud services. In 2010, Microsoft launched Microsoft Azure, and Rackspace Hosting and NASA initiated an open-source cloud-software project, OpenStack. IBM introduced the IBM SmartCloud framework in 2011, and Oracle announced the Oracle Cloud in 2012. In December 2019, Amazon launched AWS Outposts, a service that extends AWS infrastructure, services, APIs, and tools to customer data centers, co-location spaces, or on-premises facilities.

- Since the global pandemic of 2020, cloud technology has surged in popularity due to the level of data security it offers and the flexibility of working options it provides for all employees, notably remote workers.

List of Cloud Computing Platforms

Top 7 Cloud computing platforms are:

1) Microsoft azure
2) Amazon Web services
3) Google cloud
4) IBM cloud
5) CloudLinux
6) Hadoop
7) force.com and Salesforce.com

1) Microsoft Azure

Azure has long been regarded as one of the greatest cloud services platforms accessible, given to Microsoft's extensive suite of services. The extensive list of offered services is sufficient to meet the demands of any company in any sector.

You may operate services on the cloud or mix them with any of your current infrastructures using Azure. Microsoft Azure was first published in 2010, and it has since shown to be a reliable solution for businesses trying to digitally change.

2) Amazon Web Services

Amazon Online Services (AWS) is a popular cloud computing platform for developing interactive web applications for your company. Elastic Cloud Compute (EC2), Elastic Beanstalk, Simple Storage Service (S3), and Relational Database Service are just a few of the IaaS and PaaS options available (RDS)

AWS' architecture is extremely adaptable, allowing you to save expenses by just using the services you want.

3) Google Cloud

Google Cloud is a dependable, user-friendly, and secure cloud computing solution from one of the world's most powerful IT companies.

Although Google Cloud's service offering isn't as extensive as Azure's, it's still sufficient to meet all of your IaaS and PaaS requirements. Its headlines include user-friendliness and security.

Your first 12 months of service are also free, much like Azure. In addition, Google boasts that its services are less expensive and more budget-friendly than others.

4) IBM Cloud

IBM Cloud is another cloud computing platform that focuses on IaaS (Infrastructure as a Service), SaaS (Software as a Service), and PaaS (Platform as a Service).

It's one of the more cost-effective pricing plans on the market, and it's totally configurable, so you may save even more money. Using their APIs, creating an account is a breeze.

5) CloudLinux

CloudLinux is the way to go if you wish to construct your own IT infrastructure rather than depending on a third-party service. It's not just another cloud provider; it's a cloud platform for setting up your own infrastructure. It is a Linux-based operating system, as indicated by its name.

Working with CloudLinux comes with a lot of obstacles, but it also comes with a lot of benefits and advantages, such as total control, flexibility, security, and deep customization.

6) Hadoop

Apache Hadoop is a free and open source framework for processing massive amounts of data on commodity hardware. Hadoop is a Google-developed implementation of MapReduce, an application programming model. This paradigm includes two basic data processing operations: map and reduce.

Yahoo! is the Apache Hadoop project's sponsor, and it has invested a lot of work into making it an enterprise-ready cloud computing platform for data processing.

Hadoop is a key component of the Yahoo! Cloud architecture, supporting a variety of corporate business activities.

Yahoo! now manages the largest Hadoop cluster in the world, which is also open to academic institutions.

7) Force.com and Salesforce.com

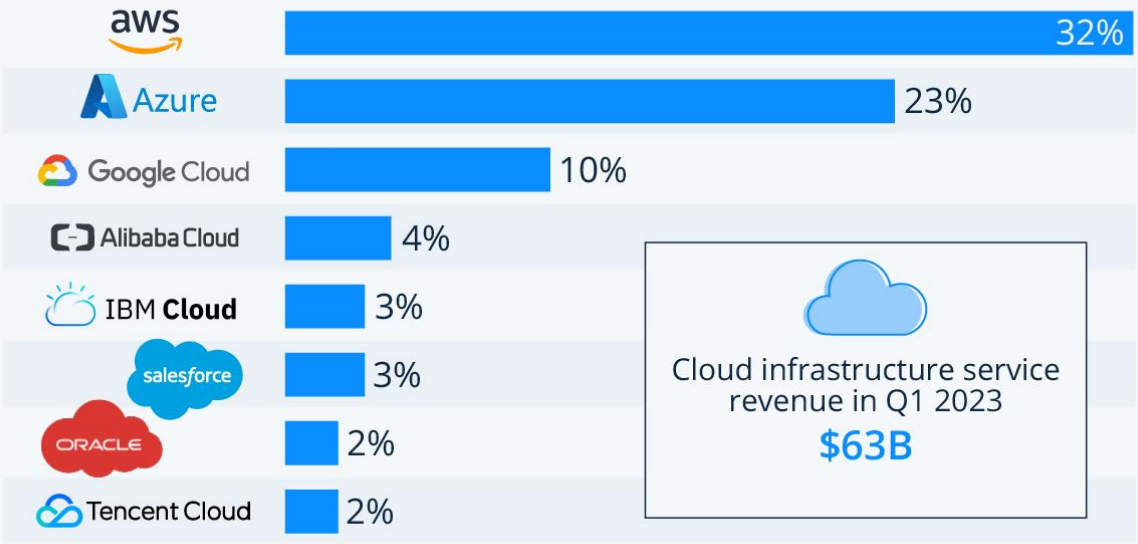
Force.com is a cloud computing platform that allows users to create social enterprise apps. SalesForce.com, a Software-as-a-Service solution for customer relationship management, is built on the platform.

Force.com enables the creation of applications by assembling ready-to-use blocks: a comprehensive collection of components covering all aspects of an organization's operations is accessible.

Force.com assists with everything from data layout design to business rule creation and user interface design. This platform is entirely hosted in the cloud, and it allows full access to all of its features, as well as those incorporated in the hosted apps, using Web services technologies.

Big Three Dominate the Global Cloud Market

Worldwide market share of leading cloud infrastructure service providers in Q1 2023*



* includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services

Source: Synergy Research Group

statista

INTRODUCTION TO AMAZON WEB SERVICES(AWS)

- Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.

- AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS) and packaged-software-as-a-service (SaaS) offerings. AWS services can offer an organization tools such as compute power, database storage and content delivery services.

- Amazon.com Web Services launched its first web services in 2002 from the internal infrastructure that Amazon.com built to handle its online retail operations. In 2006, it began offering its defining IaaS services. AWS was one of the first companies to introduce a pay-as-you-go cloud computing model that scales to provide users with compute, storage or throughput as needed.

- AWS offers many different tools and solutions for enterprises and software developers that can be used in data centers in up to 190 countries. Groups such as government agencies, education institutions, non-profits and private organizations can use AWS services.

- AWS provides services from dozens of data centers spread across 87 availability zones (AZs) in regions across the world. An AZ is a location that contains multiple physical data centers. A region is a collection of AZs in geographic proximity connected by low-latency network links.

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?

1) Free Tier

The biggest reason many people do not use AWS is lack of knowledge. EC2 is not like a traditional hosted solution, as it's designed to bring servers online and offline very quickly as needed. Because of this, many IT professionals were leery of using EC2 (or the rest of the AWS suite) because of the cost associated with "playing around" to figure it out.

The free tier, which provides enough credit to run an EC2 micro instance 24/7 all month, resolves this. It comes with S3 storage, EC2 compute hours, Elastic Load Balancer time, and much more. This gives developers a chance to try out AWS's API in their software, which not only enhances their software, but also ties them to AWS, which benefits Amazon in the long run.

2) Performance

There's no denying the speed of AWS. The Elastic Block Storage is nearly as fast as S3, but provides different features. EC2 Compute Units give Xeon-class performance on an hourly rate. The reliability is better than most private data centers in the world, and if there is a problem, you're usually still online , but with reduced capacity.

This is tested using a beautiful application Chaos Monkey, where by using this application it randomly powers down a component in your cloud environment. Then you could whether your application is still running or if it is brought down entirely. So in our case the chaos monkey brought down our database and a web server. The database which was a RDS service immediately switched to another database using the Multi AZ feature as promised by AWS. In the web server scenario, when one web server was down then another web server was launched using the autoscaling feature, so we finally concluded that AWS delivers High Availability Performance as promised by them.

In a traditional traditional hosting environments, this probably would have meant downtime and 404 errors as the websites would have just gone dark. But in a truly cloud-hosted environment like AWS, there's enough separation between processing and storage that sites can remain online and continue generating revenue even with reduced functionality. We host our sites out of the Northern Virginia cluster and Oregon cluster, and experienced no problems.

But the performance power of AWS is in the storage. The distributed nature of EBS and S3 yields millions of input/output operations per second to all instances. Think of it like having a raid array of SSDs attached to a particular computer. Add in incredible bandwidth, and you have a storage system that is capable of vast scaling, with the reliability of 99.999999999%.

3) Deployment Speed

If you've ever had to provision a hosted web service, you know this pain very well. Traditional providers take anywhere from 48-96 hours to provision a

server. Then you have to spend a few hours tweaking it and getting everything tested.

AWS shrinks that deployment time to minutes. If you utilize their Amazon Machine Images, you can have a machine deployed and ready to accept connections in that short amount of time. This is important when, for example, you are running a promotion that generates tons of traffic at specific intervals, or just need the flexibility to handle the demand when a new product launches.

The Cloudformation Templates is a gift from the AWS which can be used to roll out multiple environments at the click of button and as well can be rolled down at the click of a button when the requirement recedes.

4) Security

Access to the AWS resources can be restricted using the IAM(Identity and Access Management), using the roles in IAM we can define the privileges for user actions which greatly reduces any malpractices.

AWS also provides VPC, which can be used to host our services on a private network which is not accessible from the internet, but can communicate with the resources in the same network. This restricts the access to the resources such that any ill intentioned user from the internet.

These resources hosted in the private network can be accessed using the Amazon VPN or some open source service like OpenVPN.

5) Flexibility

The most important feature in AWS is its flexibility. All the services work and communicate together with your application to automatically judge demand and handle it accordingly.

Combined with the fantastic API and the Amazon Machine Images you create, you can have a completely customized solution that provisions a server instance in under 10 minutes, and is ready to to accept connections once it comes online. Then you can quickly shut down instances when they are no longer needed, making server management a thing of the past.

Benefits of Amazon Web Services

1)AWS Pricing

Whether you are a small scale startup or a full-fledged enterprise, Amazon Web Services has you covered when it comes to pricing. Firstly it offers 'pay as you go model', that means you pay for resources in volumes and duration you use them for. It charges you on a per-minute basis. Meaning if a resource is used for 30 minutes you be charged only for those 30 minutes and not more. It also offers a calculator that lets you track your expenses.

2)Zero Commitment

Whether you need to host a website, or even a high traffic hosting content delivery network. Amazon Web Services keeps you covered. You spawn a virtual machine, a database service or a data warehouse. This happens with you not requiring to be in an upfront commitment. This is because Amazon Web Services charges you on per minute and for some resources per hour basis. This means you are not tied with any yearly, quarterly or even monthly commitments.

3)Scalability and Procurement

If your applications lie on-premise, procuring your servers may take a lot of time. It can be a few hours to even 1- weeks. This holds true for your software licenses. Amazon Web Services paints a very different picture when it comes to procurement. You can launch new virtual machines or instances in a matter of minutes and save a lot of time and effort.

When it comes to scalability AWS ensures you can scale up and down instantly to adjust to spikes your infrastructure may face. This is something that can be difficult to achieve on your on-premise infrastructure.

4)Security

Amazon Web Services takes Cloud Security to the next level. It ensures your infrastructure is secure physically and also over the network, that consumers use to access it.

It supports shared security model. This means the consumer can control security at the consumer end and AWS at data centre end.

Physical security of data centre can be ensured by the fact that there is around the cloud physical security across all the data centres that Amazon Web Services owns

Its Global infrastructure ensures your data is well distributed and accessible to you across the globe and is highly resilient, available and safe from disasters

AWS provides firewalls to man your data at the entry points of the network and also ensures encryption of data that moves over the network, ensuring end to end security

Amazon IAM is a service that lets you identify user who can access your resources and control who get to access what and when

5)Flexible

Not a lot needs to be said about the flexibility, when a platform offers, 200+ services in 245 countries. But to point out few key pointers, Amazon Web Services offers flexibility in terms of pricing, security, and even when it comes to automating the process scaling your devices. It offers, IaaS, PaaS and even serverless computing. This means from configuring everything from a scratch to directly using a platform everything is flexible for a consumer. So much so that a user can just put his code in a serverless computing service and the service takes care of everything else.

6)PaaS Offerings

AWS offers an infrastructure that is scalable and also covers core domains, like compute storage, databases, networking. In the process, it takes care of configuring and managing platforms. Hence it provides good options when it comes to providing PaaS services to people. Meaning people do not have to worry about setting up infrastructures.

7)Adaptable

They say Amazon Web Services is everyone and that is very correct. Because it gives plenty of options when you want to set up your business on the cloud. If you are starting fresh with cloud or even if you have an infrastructure that needs to move to the cloud, AWS takes care of both situations. Amazon Web Services most types of migrations and license support for a smooth transition to AWS cloud

Your Scaling up and scaling down concerns are also nullified because AWS let us you handle data and applications in different volumes.

There are services that automate scaling and configuration processes. There are services like AWS EC2 that let you spawn instances in minutes and even create copies and backups of these instances ensuring you get adaptability that you were looking for.

8)API

API give us programmatic control over the resources we use. It comes to taking data backup, or even launching instances this all can be done API's and in short, it gives us more power compared to AWS management console.

AWS Supports plenty of API's and SDK's that let you have control over these resources.

So this was about the benefits that AWS has to offer to us. Let us go ahead and take a look at some popular use cases.

Use Cases of Amazon Web Services

1)Repp Health

We all hate waiting in hospitals for a turn to come up. As a patient, we do not have patience and want to get done with our diagnosis at the earliest. And that is understandable behaviour. However, the number of people visiting a hospital and the staff having to set up all the equipment, are some reasons that cause these situations.

Amazon Web Services provides some relief here. With the help of AWS Repp has come with cloud-based tracking solutions. It helps keep track of patients and assets that reside in the vicinity and it makes use of IoT to do so. It also updates, electronic health records considering the data from sensors that capture the movements of patients in the room.

AWS offers Server-less IoT infrastructure to this achieve this and help save as much as it can, for speeding up the overall process at the hospital

2)McDonald's

It is certain that most of us know what McDonald's is? McDonald's is a highly popular Burger and Fast chain across the globe. They have more than 37000 outlets across the globe and serve more than 60 million people every day. This gives you some idea as to the volume of customers they handle. It is no different for home delivery they provide.

However, having a home delivery platform for such a big vendor. Amazon Web Services ensured their home delivery platform was set up in just 4 months. It is a cloud-native microservices platform. It can scale up to 20000 orders per

second and the latency is less than 100 seconds. This platform ensures high integration with delivery platforms and ROI even for minimally charged orders.

3)WeWork

Here is a case study where an application was moved from one service AWS to the other service on AWS. WeWork has built a tool for Project management called FieldLens. Initially, this was set up on Amazon Elastic Cloud Compute. This was a monolith project that functioned well for 2 and a half years. However, it needed expansion and improvement. This came in the form of Docker and AWS Elastic Container Service. On moving this project to containers the performance improved immensely.

Talking of numbers FieldLens, now hosts 80,000 users and 110,000 projects that use containers, and are deployed through AWS Codepipeline and secured using AWS Service. It offers high integration and scalability.

There are plenty of use cases and applications that Amazon Web Services offers. And there are many customers that have invested billions in this cloud platform and continue to do so on a monthly basis. That tell you how reliable this cloud platform is. To note a few popular customers, here are some names that you may want to know about:

Popular Customers of Amazon Web Services

1)McDonald's

2)Netflix

3)Unilever

4)Samsung

5)MI

6)Airbnb

7)BMW

8)ESPN


Compute
IOT
Storage
aws
Application
Networking
Management Tools
Database
Security & Identity
Management

AWS Services used in this project:

1.VPC

2.EC2

3.IAM ROLE

4.DYNAMO DB

5.CLOUD WATCH

6.SNS CONSOLE

7.S3

8.LAMBDA

9.SNAPSHOT

10.CLOUDSHELL

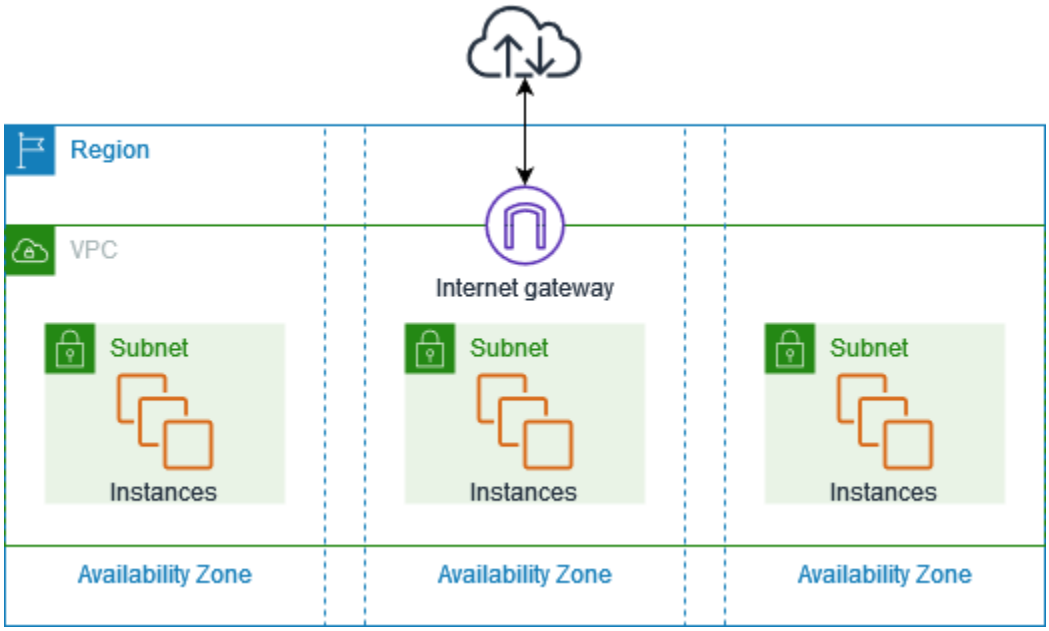
11.DOCKERHUB

12.RDS

1.VPC:

With Amazon Virtual Private Cloud (Amazon VPC), you can launch AWS resources in a logically isolated virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

The following diagram shows an example VPC. The VPC has one subnet in each of the Availability Zones in the Region, EC2 instances in each subnet, and an internet gateway to allow communication between the resources in your VPC and the internet.



Features

The following features help you configure a VPC to provide the connectivity that your applications need:

Features

The following features help you configure a VPC to provide the connectivity that your applications need:

Virtual private clouds (VPC)

A VPC is a virtual network that closely resembles a traditional network that you'd operate in your own data center. After you create a VPC, you can add subnets.

Subnets

A subnet is a range of IP addresses in your VPC. A subnet must reside in a single Availability Zone. After you add subnets, you can deploy AWS resources in your VPC.

IP addressing

You can assign IP addresses, both IPv4 and IPv6, to your VPCs and subnets. You can also bring your public IPv4 and IPv6 GUA addresses to AWS and

allocate them to resources in your VPC, such as EC2 instances, NAT gateways, and Network Load Balancers.

Routing

Use route tables to determine where network traffic from your subnet or gateway is directed.

Gateways and endpoints

A gateway connects your VPC to another network. For example, use an internet gateway to connect your VPC to the internet. Use a VPC endpoint to connect to AWS services privately, without the use of an internet gateway or NAT device.

Peering connections

Use a VPC peering connection to route traffic between the resources in two VPCs.

Traffic Mirroring

Copy network traffic from network interfaces and send it to security and monitoring appliances for deep packet inspection.

Transit gateways

Use a transit gateway, which acts as a central hub, to route traffic between your VPCs, VPN connections, and AWS Direct Connect connections.

VPC Flow Logs

A flow log captures information about the IP traffic going to and from network interfaces in your VPC.

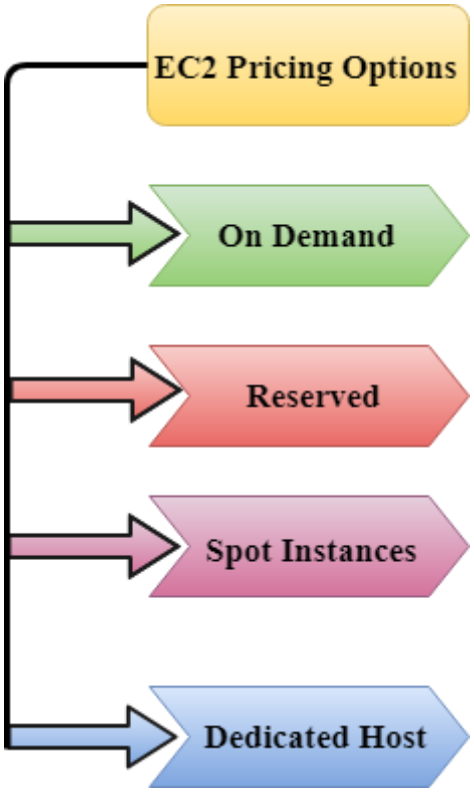
VPN connections

Connect your VPCs to your on-premises networks using AWS Virtual Private Network (AWS VPN).

2.EC2:

- EC2 stands for Amazon Elastic Compute Cloud.
- Amazon EC2 is a web service that provides resizable compute capacity in the cloud.
- Amazon EC2 reduces the time required to obtain and boot new user instances to minutes rather than in older days, if you need a server then you had to put a purchase order, and cabling is done to get a new server which is a very time-consuming process. Now, Amazon has provided an EC2 which is a virtual machine in the cloud that completely changes the industry.
- You can scale the compute capacity up and down as per the computing requirement changes.
- Amazon EC2 changes the economics of computing by allowing you to pay only for the resources that you actually use. Rather than you previously buy physical servers, you would look for a server that has more CPU capacity, RAM capacity and you buy a server over 5 year term, so you have to plan for 5 years in advance. People spend a lot of capital in such investments. EC2 allows you to pay for the capacity that you actually use.
- Amazon EC2 provides the developers with the tools to build resilient applications that isolate themselves from some common scenarios.

EC2 Pricing Options



On Demand

- It allows you to pay a fixed rate by the hour or even by the second with no commitment.
- Linux instance is by the second and windows instance is by the hour.
- On Demand is perfect for the users who want low cost and flexibility of Amazon EC2 without any up-front investment or long-term commitment.
- It is suitable for the applications with short term, spiky or unpredictable workloads that cannot be interrupted.
- It is useful for the applications that have been developed or tested on Amazon EC2 for the first time.
- On Demand instance is recommended when you are not sure which instance type is required for your performance needs.

Reserved

- It is a way of making a reservation with Amazon or we can say that we make a contract with Amazon. The contract can be for 1 or 3 years in length.
- In a Reserved instance, you are making a contract means you are paying some upfront, so it gives you a significant discount on the hourly charge for an instance.
- It is useful for applications with steady state or predictable usage.

- It is used for those applications that require reserved capacity.
- Users can make up-front payments to reduce their total computing costs. For example, if you pay all your upfronts and you do 3 years contract, then only you can get a maximum discount, and if you do not pay all upfronts and do one year contract then you will not be able to get as much discount as you can get If you do 3 year contract and pay all the upfronts.

3.IAM ROLE:

An IAM *role* is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but not want to embed AWS keys within the app (where they can be difficult to rotate and where users can potentially extract them). Sometimes you want to give AWS access to users who already have identities defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.

For these scenarios, you can delegate access to AWS resources using an *IAM role*. This section introduces roles and the different ways you can use them, when and how to choose among approaches, and how to create, manage, switch to (or assume), and delete roles.

Situations in which "IAM Roles" can be used:

- Sometimes you want to grant the users to access the AWS resources in your AWS account.
- Sometimes you want to grant the users to access the AWS resources in another AWS account.
- It also allows the mobile app to access the AWS resources, but not want to store the keys in the app.
- It can be used to grant access to the AWS resources which have identities outside of AWS.
- It can also be used to grant access to the AWS resources to the third party so that they can perform an audit on AWS resources.

Following are the important terms associated with the "IAM Roles":

- **Delegation:** Delegation is a process of granting the permissions to the user to allow the access to the AWS resources that you control. Delegation sets up the trust between a trusted account (an account that owns the resource) and a trusting account (an account that contains the users that need to access the resources).

  The trusting and trusted account can be of three types:

  - Same account
  - Two different accounts under the same organization control
  - Two different accounts owned by different organizations.

To delegate permission to access the resources, an IAM role is to be created in the trusting account that has the two policies attached.

Permission Policy: It grants the user with a role the needed permissions to carry out the intended tasks.

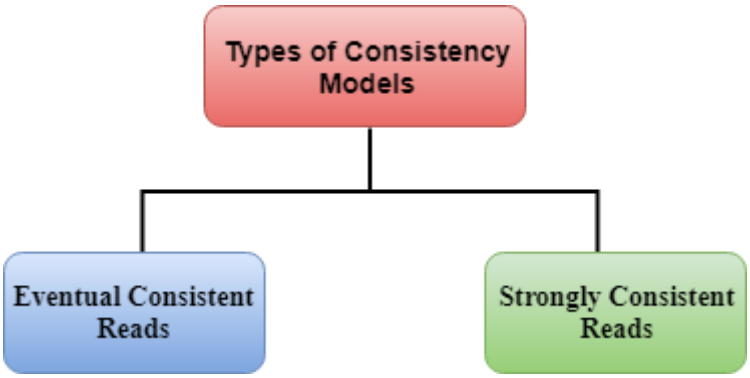
Trust Policy: It specifies which trusted account members can use the role.

- **Federation:** Federation is a process of creating the trust relationship between the external service provider and AWS. For example, Facebook allows the user to login to different websites by using their facebook accounts.

- **Trust policy:** A document was written in JSON format to define who is allowed to use the role. This document is written based on the rules of the IAM Policy Language.
- **Permissions policy:** A document written in JSON format to define the actions and resources that the role can use. This document is based on the rules of the IAM Policy Language.
- **Permissions boundary:** It is an advanced feature of AWS in which you can limit the maximum permissions that the role can have. The permission boundaries can be applied to IAM User or IAM role but cannot be applied to the service-linked role.
- **Principal:** A principal can be AWS root account user, an IAM User, or a role. The permissions that can be granted in one of the two ways:
  - Attach a permission policy to a role.
  - The services that support resource-based policies, you can identify the principal in the principal element of policy attached to the resource.
- **Cross-account access: Roles vs Resource-Based Policies:** It allows you to grant access to the resources in one account to the trusted principal in another account is known as cross-account access. Some services allow you to attach the policy directly, known as Resource-Based policy. The services that support Resource-Based Policy are Amazon S3 buckets, Amazon SNS, Amazon SQS Queues.

4.DYNAMO DB:

- Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that require consistent single-digit millisecond latency at any scale.
- It is a fully managed database that supports both document and key-value data models.
- Its flexible data model and performance makes it a great fit for mobile, web, gaming, ad-tech, IOT, and many other applications.
- It is stored in SSD storage.
- It is spread across three geographically data centres.



Because of its availability in three geographically data centres, It consists of two different types of consistency models:

- **Eventual Consistent Reads**
- **Strongly Consistent Reads**

Eventual Consistent Reads

It maintains consistency across all the copies of data which is usually reached within a second. If you read a data from **DynamoDB table**, then the response would not reflect the most recently completed write operation, and if you repeat to read the data after a short period, then the response would be the lattest update. This is the best model for Read performance.

5.CLOUD WATCH:

- CloudWatch is a service used to monitor your AWS resources and applications that you run on AWS in real time. CloudWatch is used to collect and track metrics that measure your resources and applications.
- It displays the metrics automatically about every AWS service that you choose.
- You can create the dashboard to display the metrics about your custom application and also display the metrics of custom collections that you choose.
- You can also create an alarm to watch metrics. For example, you can monitor CPU usage, disk read and disk writes of Amazon EC2 instance to determine whether the additional EC2 instances are required to handle the load or not. It can also be used to stop the instance to save money.

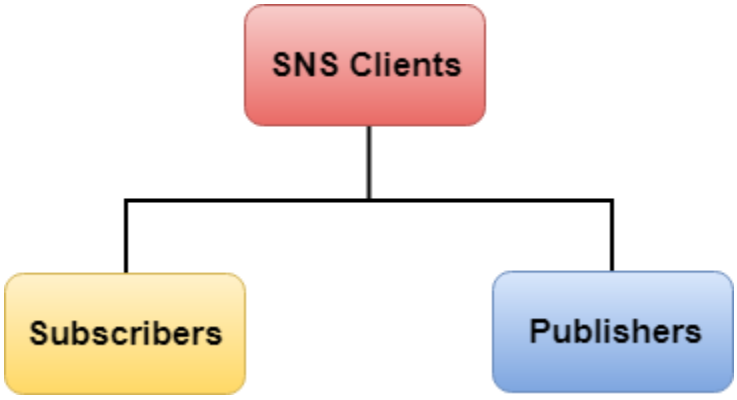
Following are the terms associated with CloudWatch:

- **Dashboards:** CloudWatch is used to create dashboards to show what is happening with your AWS environment.
- **Alarms:** It allows you to set alarms to notify you whenever a particular threshold is hit.
- **Logs:** CloudWatch logs help you to aggregate, monitor, and store logs
- **Events:** CloudWatch help you to respond to state changes to your AWS resources.

6.SNS CONSOLE:

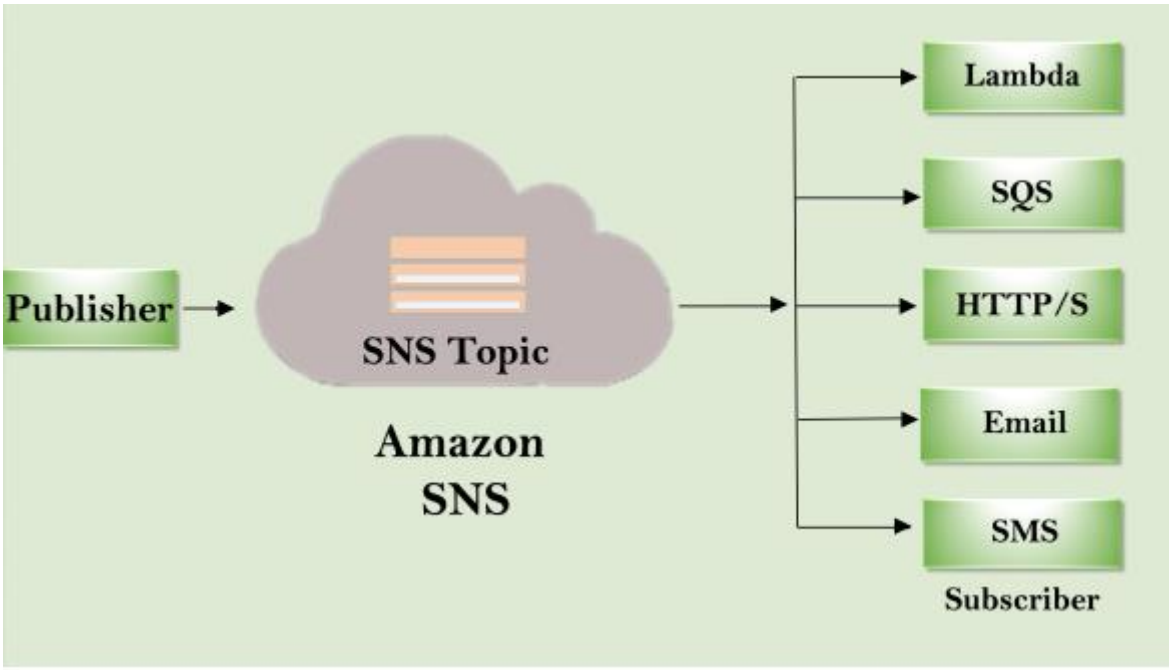
- SNS stands for Simple Notification Service.
- It is a web service which makes it easy to set up, operate, and send a notification from the cloud.
- It provides developers with the highly scalable, cost-effective, and flexible capability to publish messages from an application and sends them to other applications.
- It is a way of sending messages. When you are using AutoScaling, it triggers an SNS service which will email you that "your EC2 instance is growing".
- SNS can also send the messages to devices by sending push notifications to Apple, Google, Fire OS, and Windows devices, as well as Android devices in China with Baidu Cloud Push.
- Amazon SNS allows you to group multiple recipients using topics where the topic is a logical access point that sends the identical copies of the same message to the subscribe recipients.
- Amazon SNS supports multiple endpoint types. For example, you can group together IOS, Android and SMS recipients. Once you publish the message to the topic, SNS delivers the formatted copies of your message to the subscribers.
- To prevent the loss of data, all messages published to SNS are stored redundantly across multiple availability zones.

SNS Publishers and Subscribers



Amazon SNS is a web service that manages sending messages to the subscribing endpoint. There are two clients of SNS:

- Subscribers
- Publishers



7.S3:

- S3 is one of the first services that has been produced by aws.
- S3 stands for Simple Storage Service.
- S3 provides developers and IT teams with secure, durable, highly scalable object storage.
- It is easy to use with a simple web services interface to store and retrieve any amount of data from anywhere on the web.
- It is Object-based storage, i.e., you can store the images, word files, pdf files, etc.
- The files which are stored in S3 can be from 0 Bytes to 5 TB.
- It has unlimited storage means that you can store the data as much you want.
- Files are stored in Bucket. A bucket is like a folder available in S3 that stores the files.
- S3 is a universal namespace, i.e., the names must be unique globally. Bucket contains a DNS address. Therefore, the bucket must contain a unique name to generate a unique DNS address.

If you create a bucket, URL look like:

https://s3-eu-west-1.amazonaws.com/acloudguru

Region name **Bucket name**

- If you upload a file to S3 bucket, then you will receive an HTTP 200 code means that the uploading of a file is successful.

Advantages of Amazon S3

- **Create Buckets:** Firstly, we create a bucket and provide a name to the bucket. Buckets are the containers in S3 that stores the data. Buckets must have a unique name to generate a unique DNS address.
- **Storing data in buckets:** Bucket can be used to store an infinite amount of data. You can upload the files as much you want into an Amazon S3 bucket, i.e., there is no maximum limit to store the files. Each object can contain upto 5 TB of data. Each object can be stored and retrieved by using a unique developer assigned-key.

- **Download data:** You can also download your data from a bucket and can also give permission to others to download the same data. You can download the data at any time whenever you want.
- **Permissions:** You can also grant or deny access to others who want to download or upload the data from your Amazon S3 bucket. Authentication mechanism keeps the data secure from unauthorized access.
- **Standard interfaces:** S3 is used with the standard interfaces REST and SOAP interfaces which are designed in such a way that they can work with any development toolkit.
- **Security:** Amazon S3 offers security features by protecting unauthorized users from accessing your data

8.LAMBDA:

AWS Lambda is also a promising addition to the list of AWS services. Amazon Lambda is a serverless and event-driven computing AWS service. It helps to run codes automatically without worrying about servers and clusters. Simply put, codes can be uploaded directly to run without worrying about provisioning or managing infrastructure. So, this service automatically accepts 'code execution requests' irrespective of its scale. Besides, you can pay the price only for the computer time, so AWS Lambda makes effective cost-control.

9.SNAPSHOT:

- EBS provides the ability to create snapshots (backups) of any EBS volume and write a copy of the data in the volume to S3, where it is stored redundantly in multiple Availability Zones
- Snapshots are incremental backups and store only the data that was changed from the time the last snapshot was taken.
- Snapshots can be used to create new volumes, increase the size of the volumes or replicate data across Availability Zones.
- Snapshot size can probably be smaller than the volume size as the data is compressed before being saved to S3.
- Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.
- EBS Snapshots can be used to migrate or create EBS volumes in different AZs or regions.

10.CLOUDSHELL:

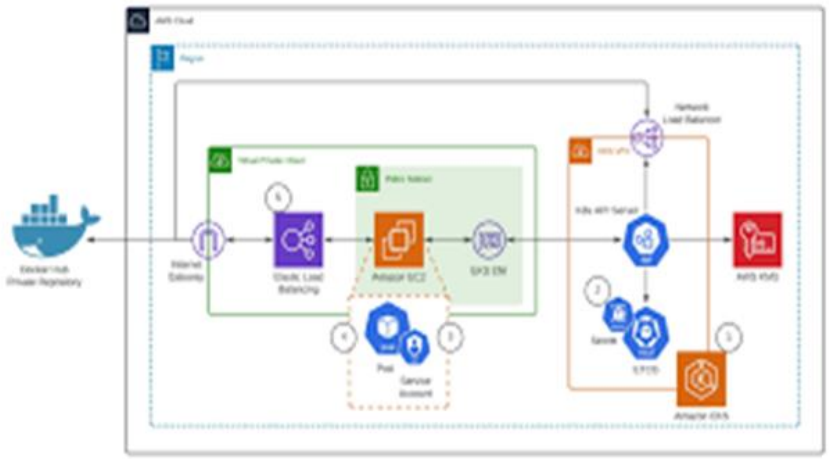
The AWS Management Console provides access to AWS CloudShell, a browser-based shell. Customers may immediately access an Amazon Linux 2 environment with the AWS Command Line Interface (CLI) pre-installed and pre-authenticated using the same credentials used to get into the Management Console by establishing a CloudShell session after logging into the console. CloudShell makes it simple to securely manage, interact with, and explore your resources from the command line. Standard tools, including AWS CLIs, are pre-installed, and root access allows you to install additional tools as needed. Bash, zsh, and PowerShell are all provided, so you may use whatever shell you choose.

AWS CloudShell may be launched with a single click from any supported AWS Region. Up to 1GB of data may be uploaded and downloaded to your CloudShell home directory ($HOME), and files, scripts, and tools saved there will persist across sessions. Per Region, CloudShell comes with 1 GB of persistent storage. Go to the top of any console page that shows when you're in a compatible Region and click the CloudShell symbol link.

11.DOCKER HUB:

Docker Hub is a hosted repository service provided by Docker for finding and sharing container images with your team. Key features include:

- Private Repositories: Push and pull container images
- Automated Builds: Automatically build container images from GitHub and Bitbucket and push them to Docker Hub
- Teams & Organizations: Manage access to private repositories
- Official Images: Pull and use high-quality container images provided by Docker
- Publisher Images: Pull and use high-quality container images provided by external vendors. Certified images also include support and guarantee compatibility with Docker Enterprise
- Webhooks: Trigger actions after a successful push to a repository to integrate Docker Hub with other services



12.RDS:

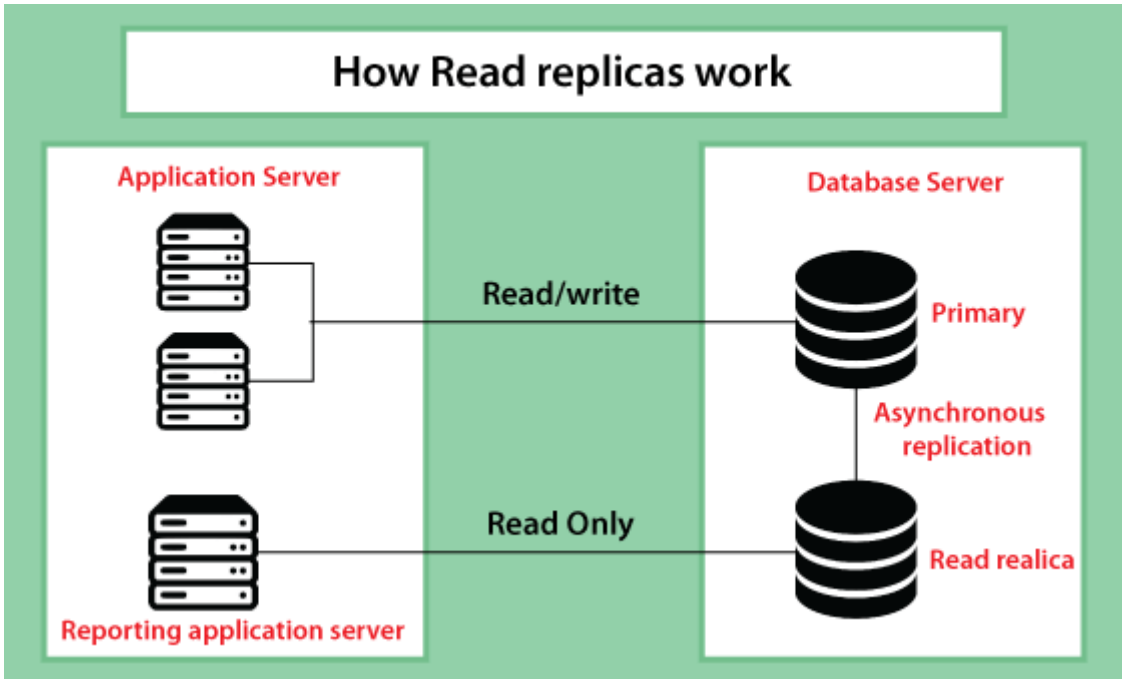
Amazon Relational Database Service (RDS) is a managed SQL database service provided by Amazon Web Services (AWS). Amazon RDS supports an array of database engines to store and organize data. It also helps in relational database management tasks like data migration, backup, recovery and patching.

Amazon RDS facilitates the deployment and maintenance of relational databases in the cloud. Cloud administrators use Amazon RDS to set up, operate, manage, and scale relational instances of cloud databases. Amazon RDS itself is not a database; It is a service used to manage relational databases.

Amazon RDS Features

Amazon RDS features include the following:

Replication. RDS uses the replication feature to create read replicas, and these are read-only copies of the database instances that the application uses without changing the original production database. Administrators can also enable automatic failover across multiple availability zones through RDS multi-edge deployment and synchronous data replication.



RDS provides three types of storage:

A general-purpose solid-state drive (SSD). Amazon recommends this storage as the default choice.

Provisioned input-output operations per second (IOPS). SSD storage for I/O-intensive workloads.

Magnetic. A lower-cost option.

Monitoring. The Amazon **CloudWatch** service enables managed monitoring, and it lets users view capacity and I/O metrics.

Patching. RDS provides patches for whichever database engine the user chooses.

Backups. Another feature is failure detection and recovery. RDS provides managed instance backups with transaction logs to enable point-in-time recovery. Users pick a retention period and restore databases to any time during that period. They also can manually take snapshots of instances that remain until they are manually deleted.

Amazon RDS database engines

An AWS customer can spin up to six types of database engines within Amazon RDS:

- **Amazon Aurora** is a proprietary AWS relational database engine. Amazon Aurora is compatible with MySQL and **PostgreSQL**.
- **RDS for MariaDB**is compatible with Maria DB, an open-source relational database management system (**RDBMS**) that's an offshoot of MySQL.
- **RDS for MySQL** is compatible with the MySQL open-source RDBMS.
- **RDS for Oracle Database**is compatible with several editions of Oracle Database, including bring-your-own-license and license-included versions.
- **RDS for PostgreSQL** is compatible with PostgreSQL open-source object-RDBMS.
- **RDS for SQL Server**is compatible with Microsoft SQL Server, an RDBMS.

Amazon RDS adds support for major and minor versions of database engines over time. It is designed to allow admins to specify an engine version when they create a database instance. In most cases, Amazon RDS can support developer code, applications, and tools already in use with existing databases.

IMPLEMENTATION OF ARCHITECTS OF MULTIPLE SERVICES

Introduction:

- This Project is a combination of multiple accounts with multiple services which form a multi-tier architecture.

- Services used in this project are

1.VPC

2.EC2

3.IAM ROLE

4.DYNAMO DB

5.CLOUD WATCH

6.SNS CONSOLE

7.S3

8.LAMBDA

9.SNAPSHOT

10.CLOUDSHELL

11.DOCKERHUB

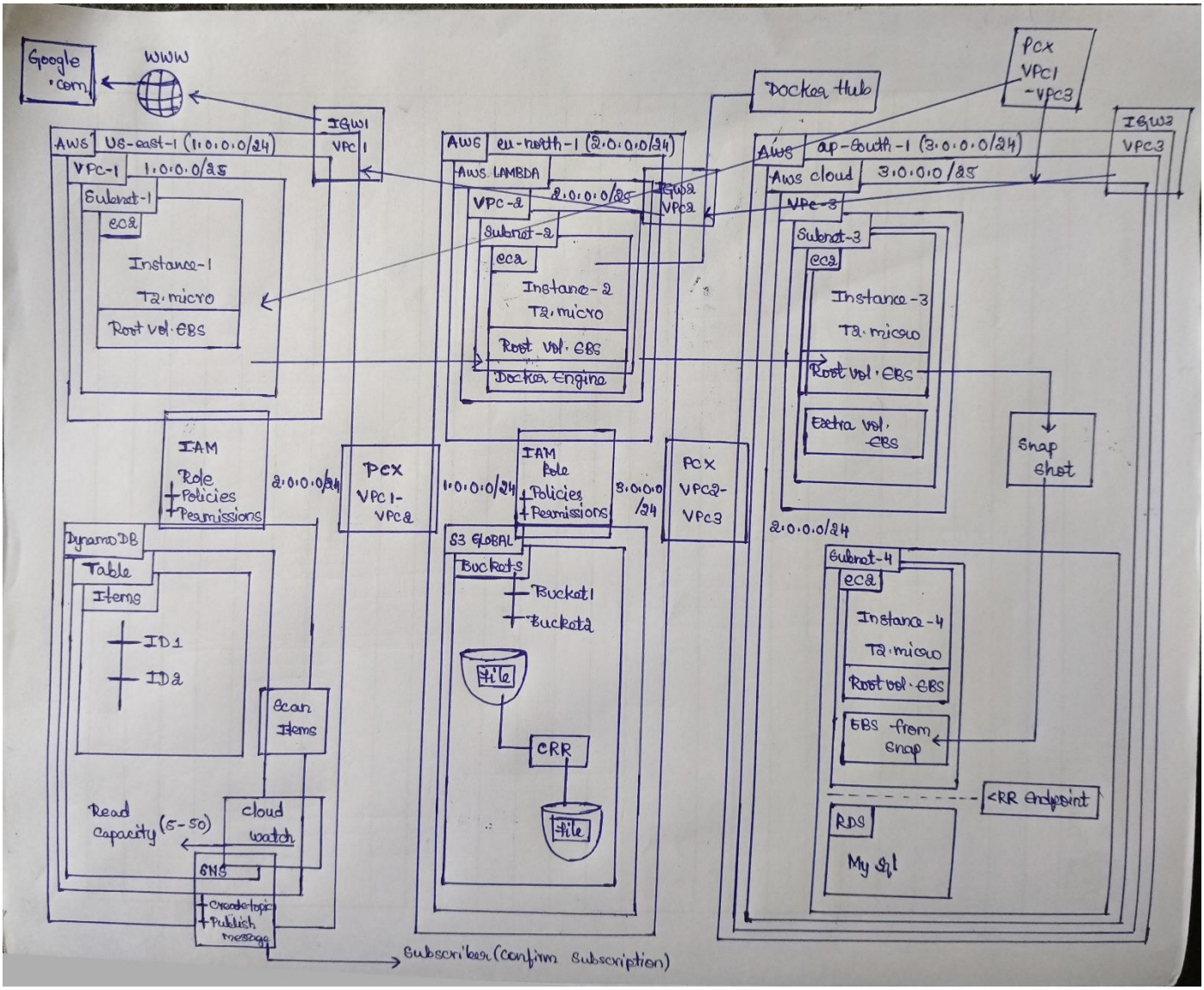
12.RDS

- Accounts used are

1. Sriram's AWS Account :- 0943-3026-3750 (us-east-1)
2. Shyam's AWS Account :- 6367-2738-5025 (eu-north-1)
3. Swaroop's AWS Account :- 4839-6869-6965 (ap-south-1)

- Using the above accounts and services three different Architectures are created and their connected by the peer connection

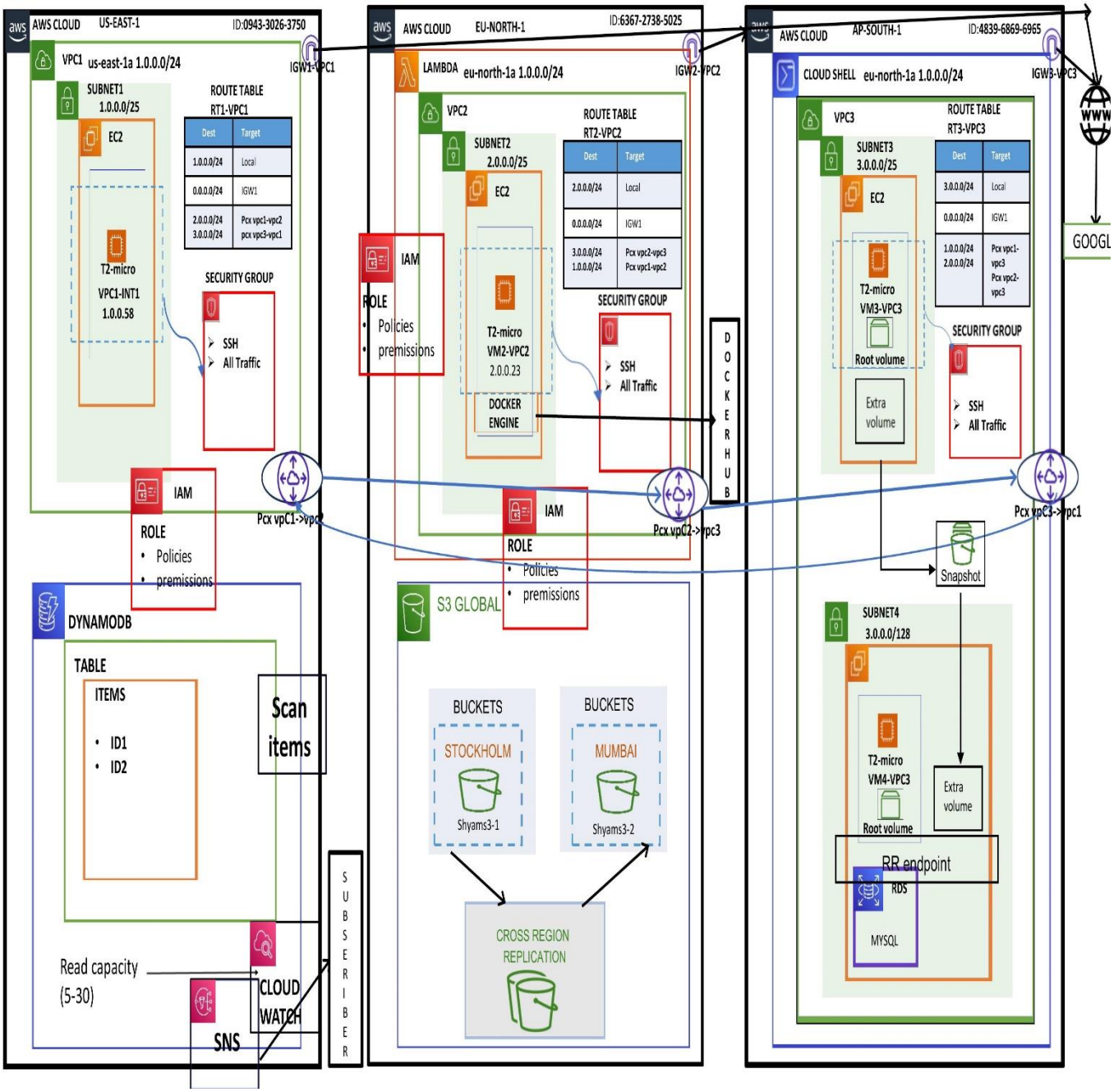
ROUGH ARCHITECTURE:



FINAL ARCHITECTURE:

MULTI-TIER SCALABLE AWS ARCHITECT USING DIFFERENT ACCOUNTS

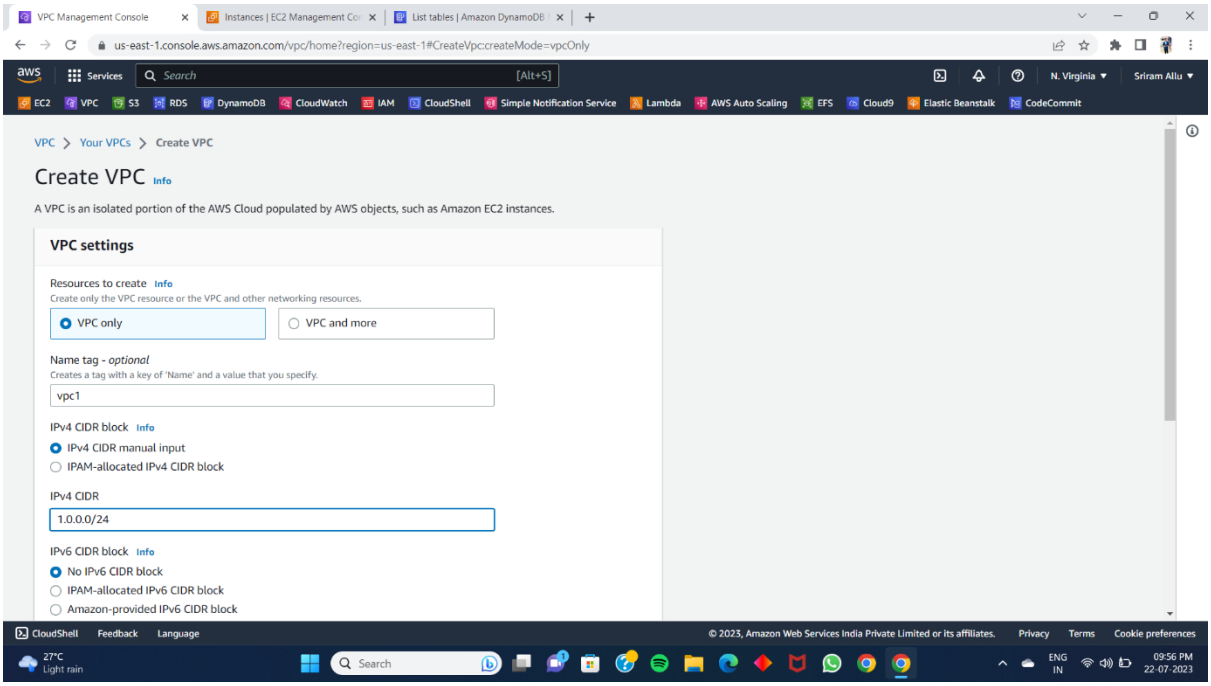
(- 12 TIER ARCHITECT)

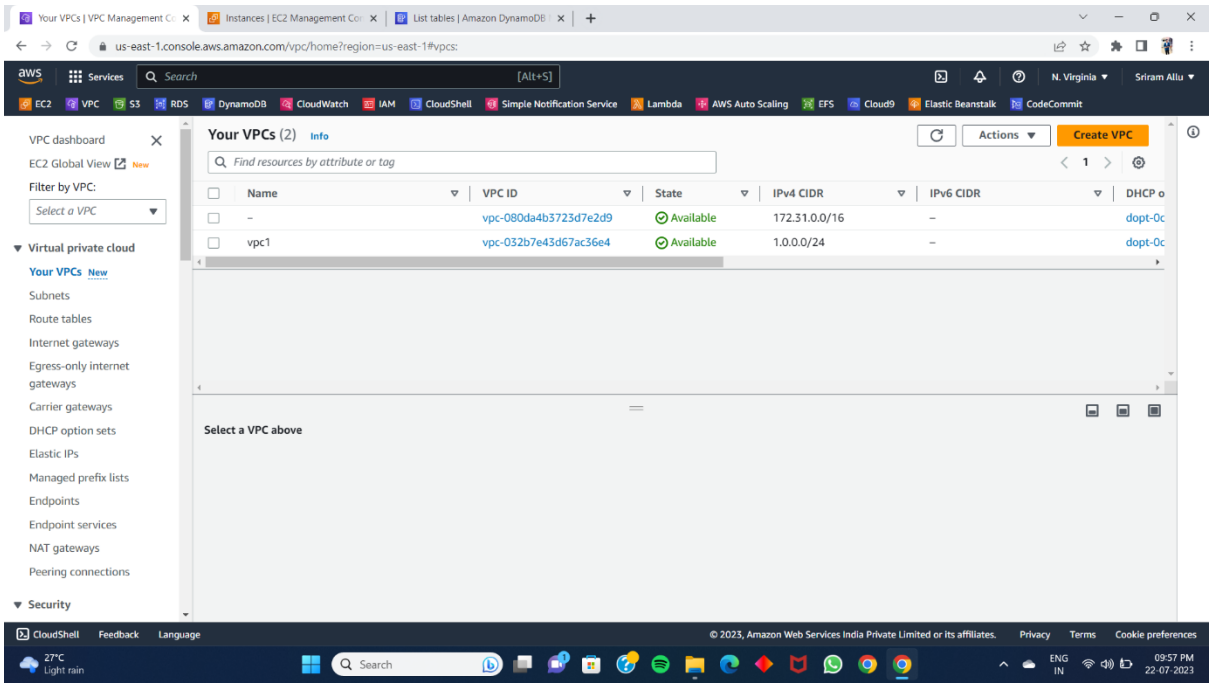


Implementation of the Project:

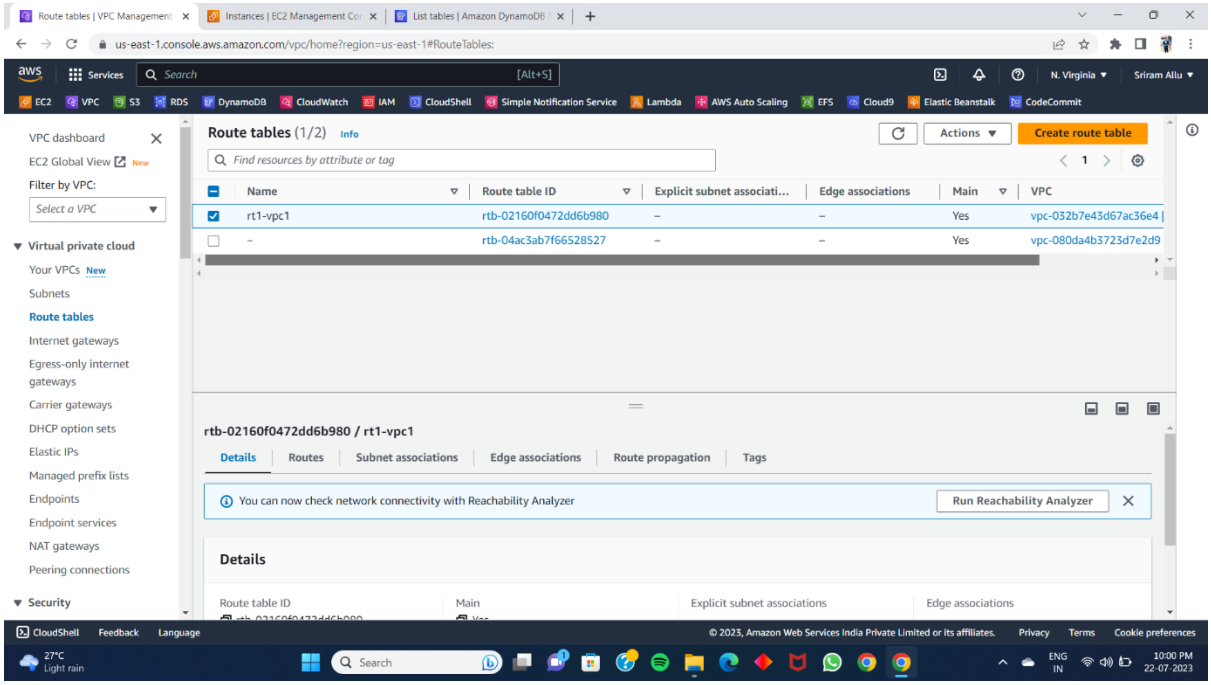
In us-east-1 Acc id: 0943-3026-3750:-

- First Create a Custom VPC in the region N.Virginia
- 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 vpc1 and CIDR value as 1.0.0.0/24.
- Click on **Create VPC**



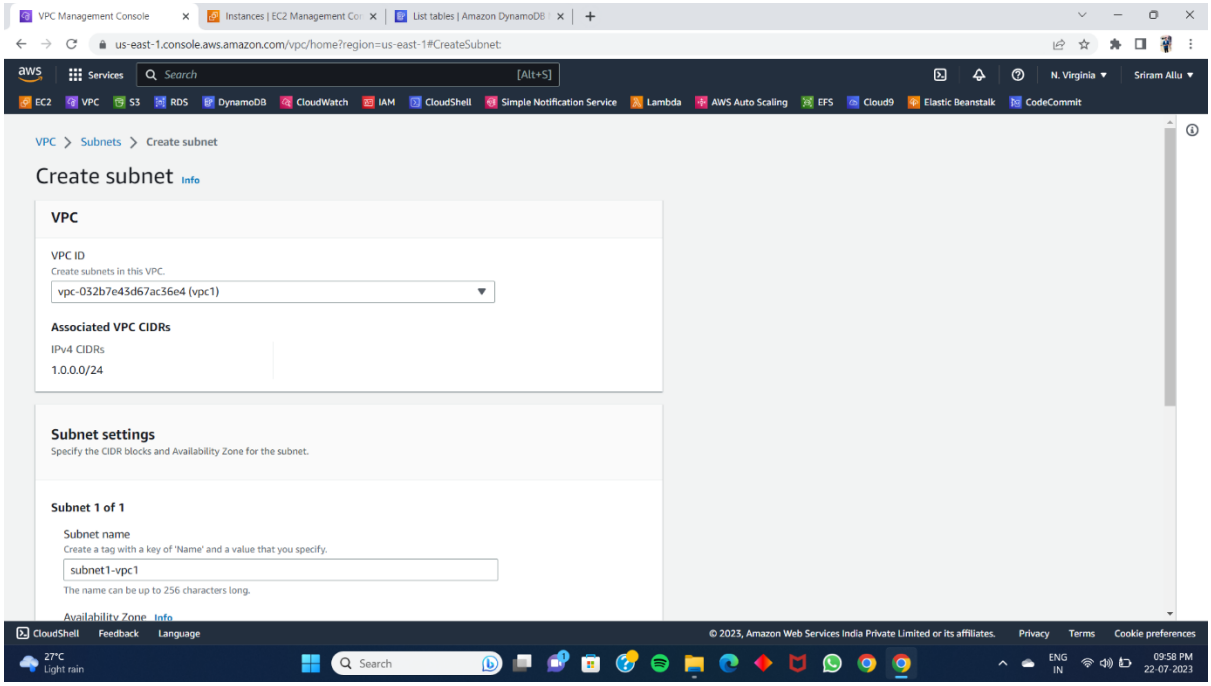


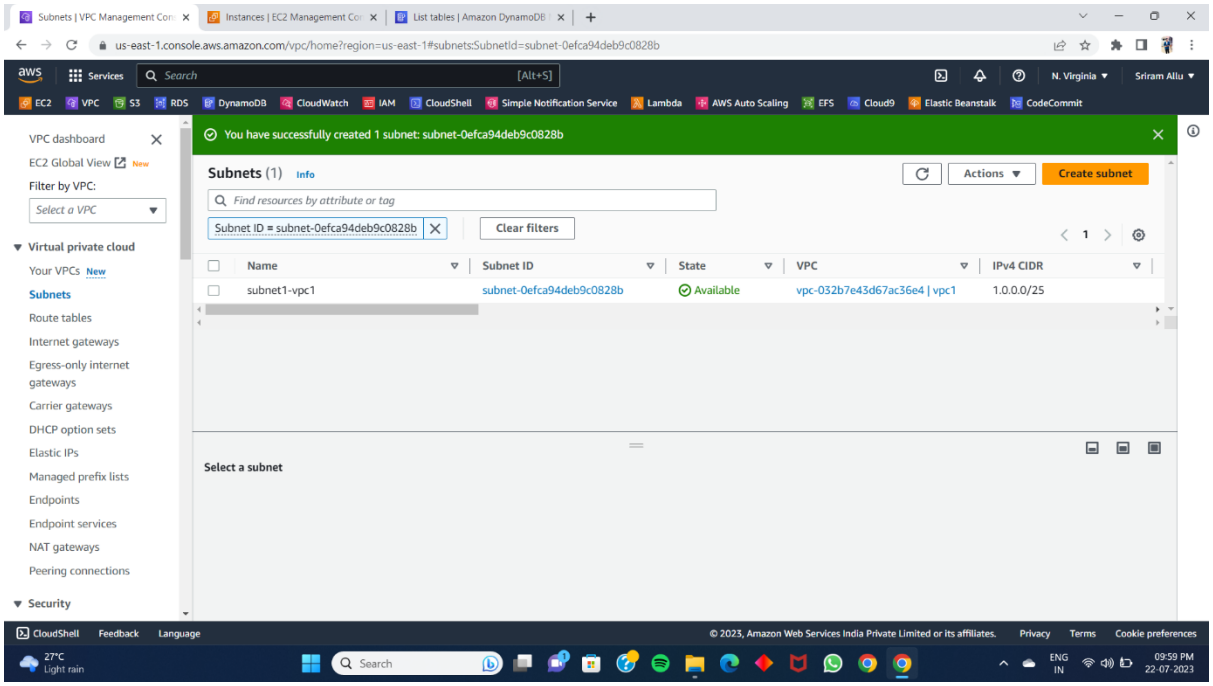
- Go to route table and edit the name of route table as rt1-vpc1 that is created along with vpc1.



CREATING A SUBNET:

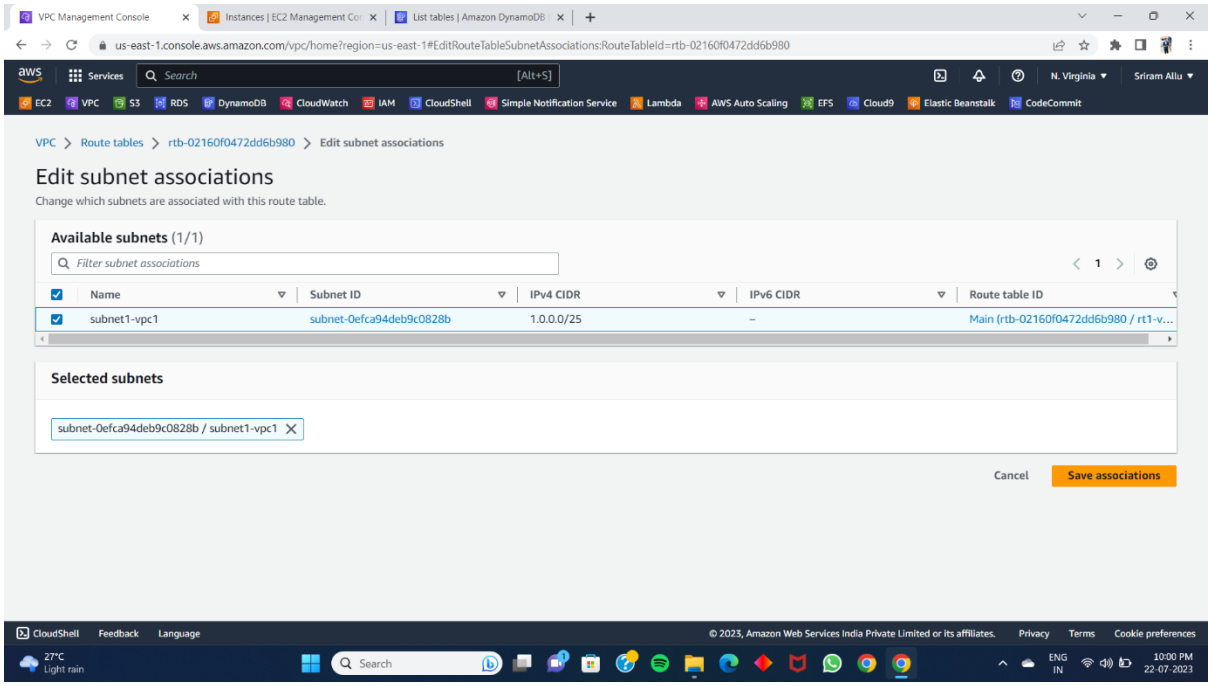
- Go to subnets and create a subnet and name it as subnet1-vpc1. Select the availability zone as us-east-1a, CIDR as 1.0.0.0/25 and select create subnet





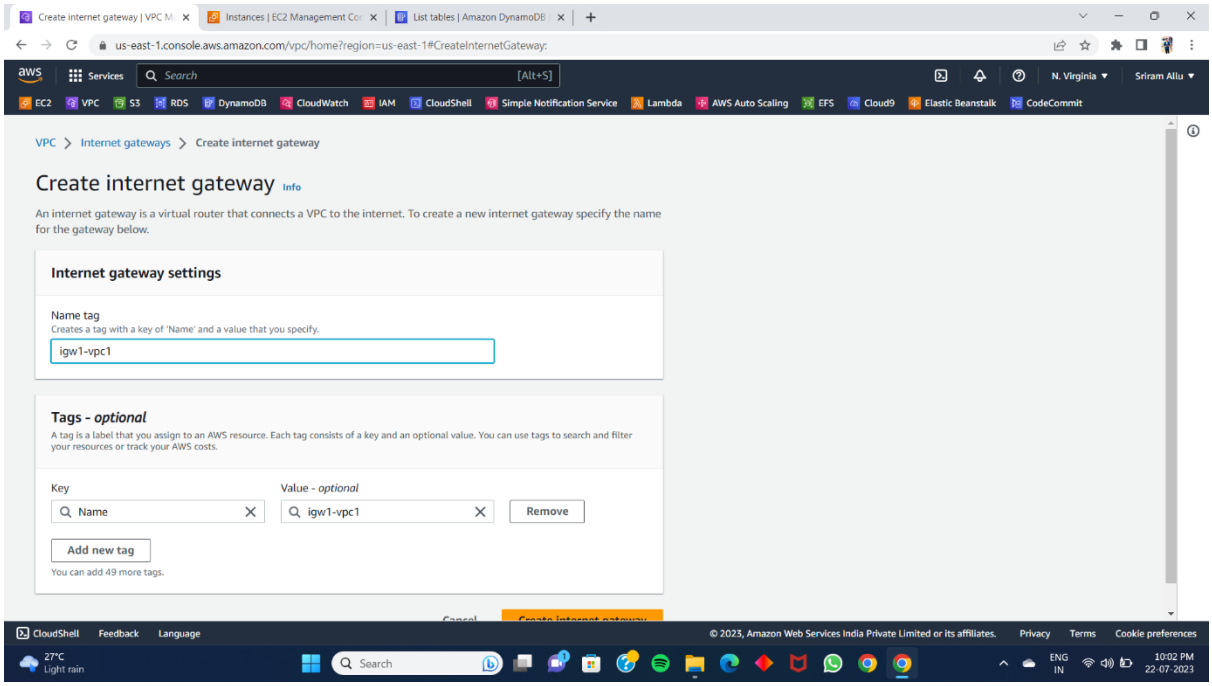
EDIT SUBNET ASSOCIATIONS:

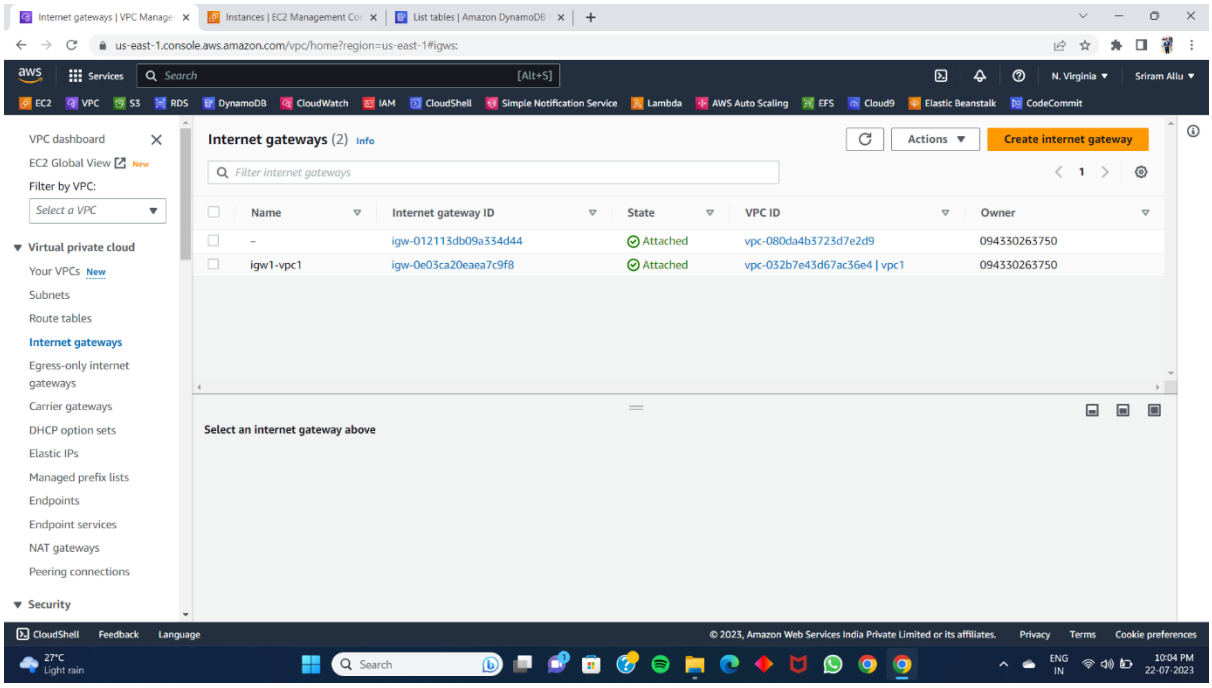
• Go to route tables, select the route table of vpc1, and go to subnet associations and select subnet of vpc1 and click on save associations.



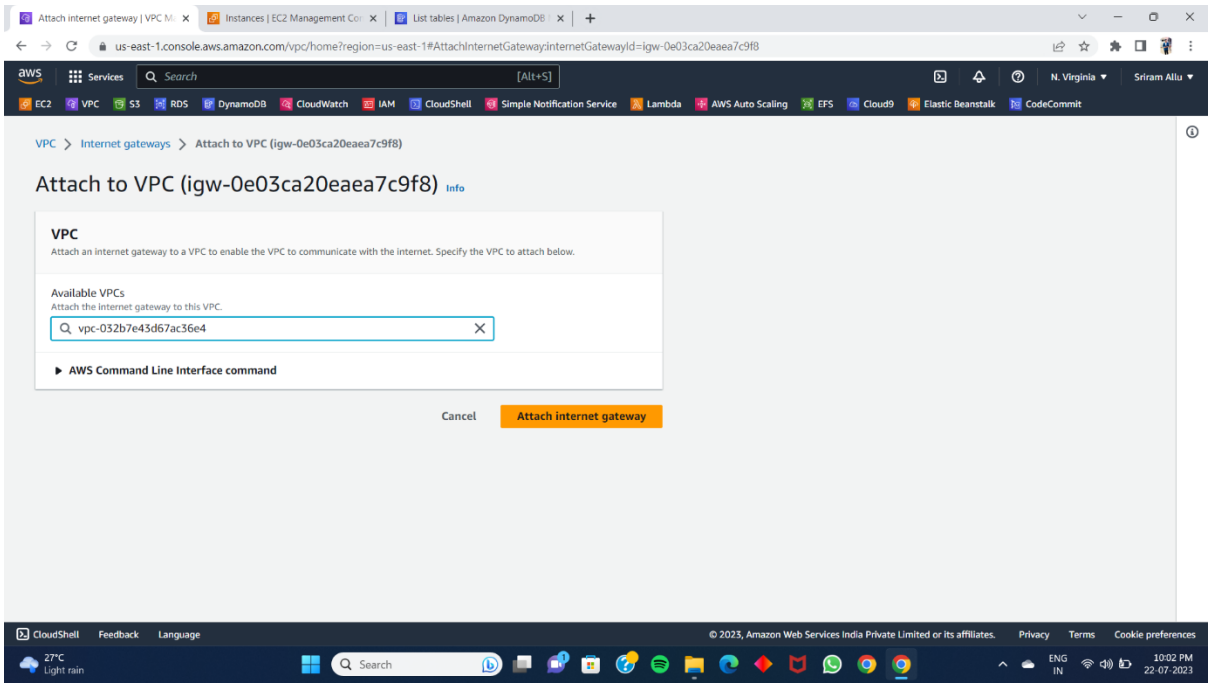
CREATING INTERNET GATEWAY:

• Create an internet gateway with the name igw1-vpc1.

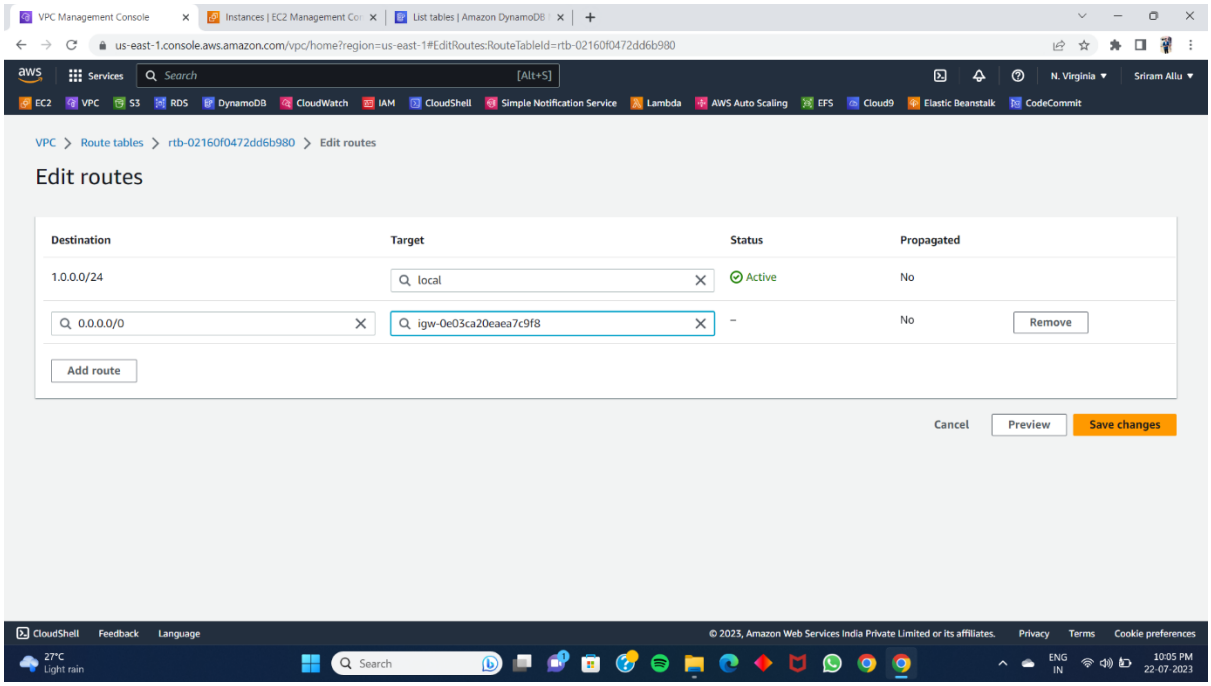




• Attach custom IGW to custom VPC

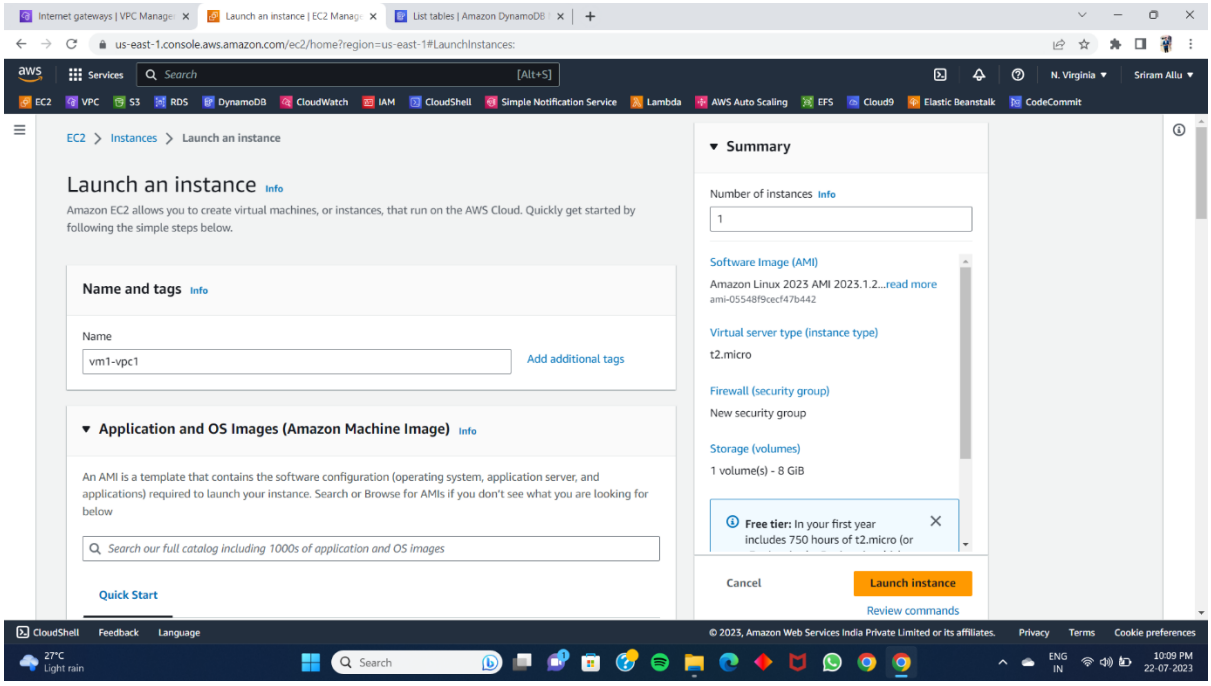


• go to route table edit routes add internet gateway.

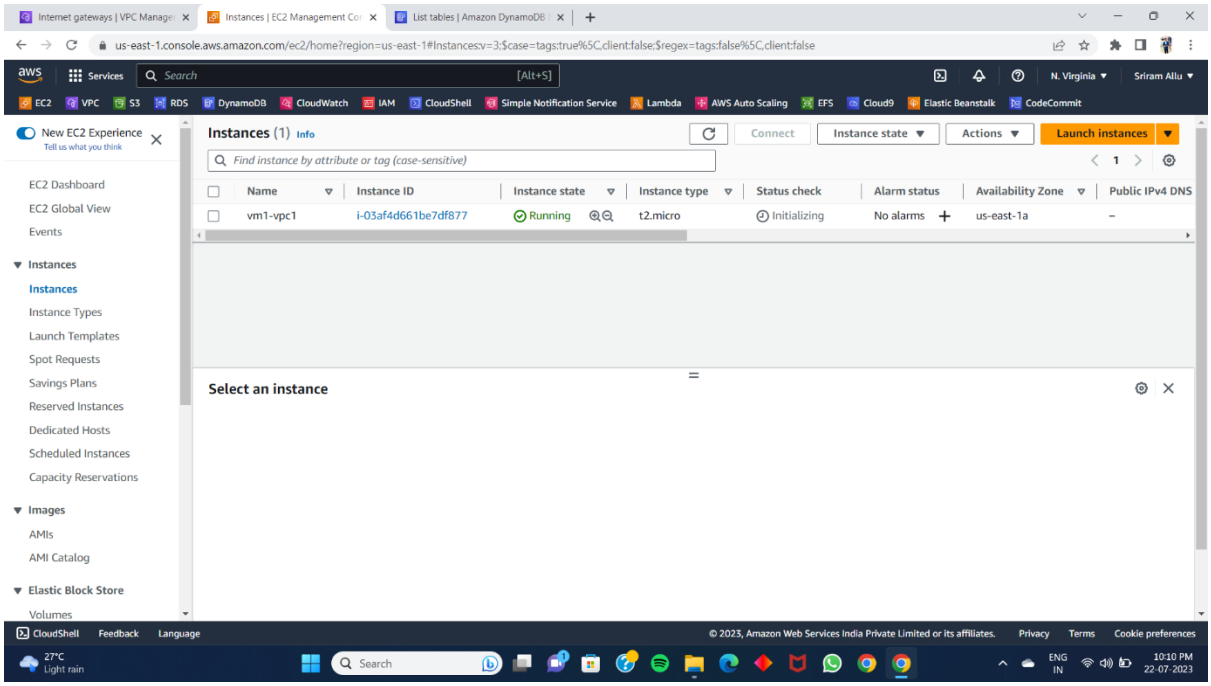


CREATING INSTANCE:

• Create a Ec2 instance with custom vpc1.

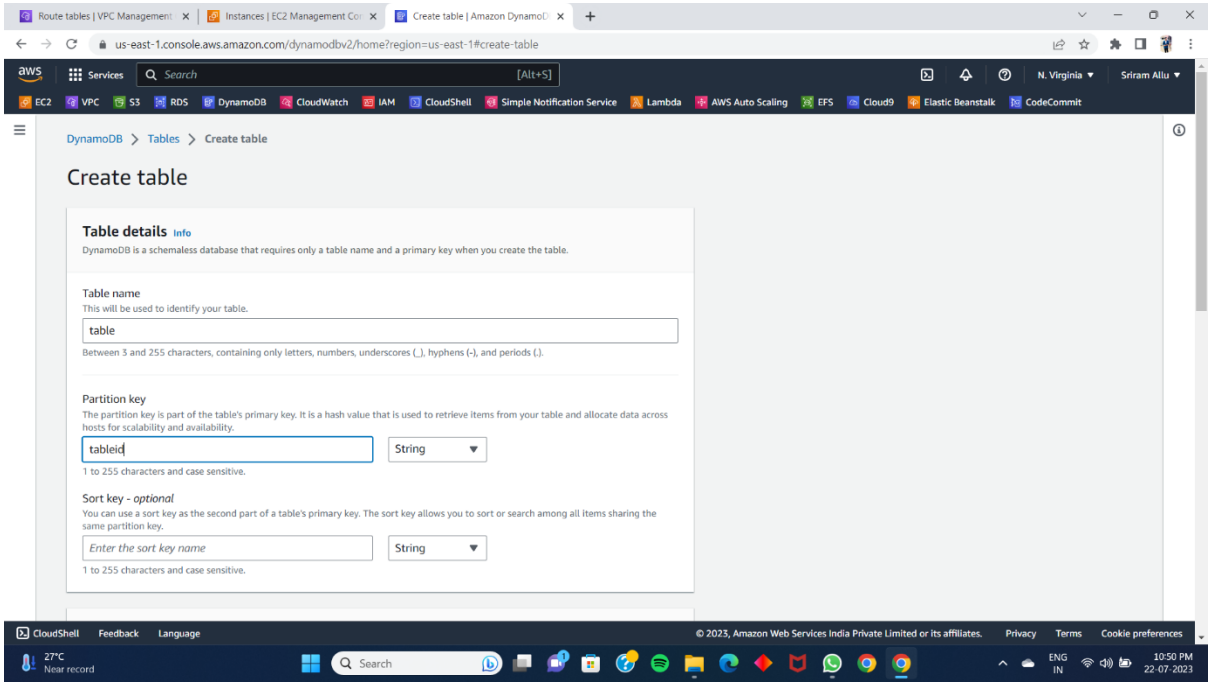


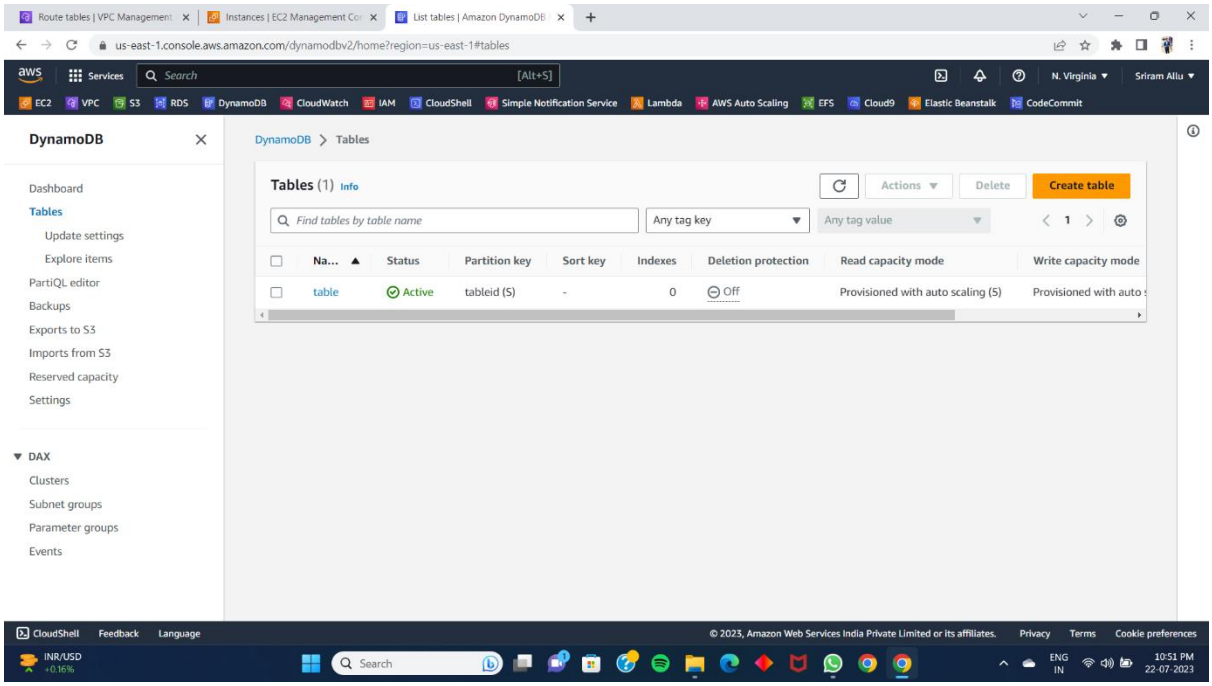
• Name the instance as vm1-vpc1.



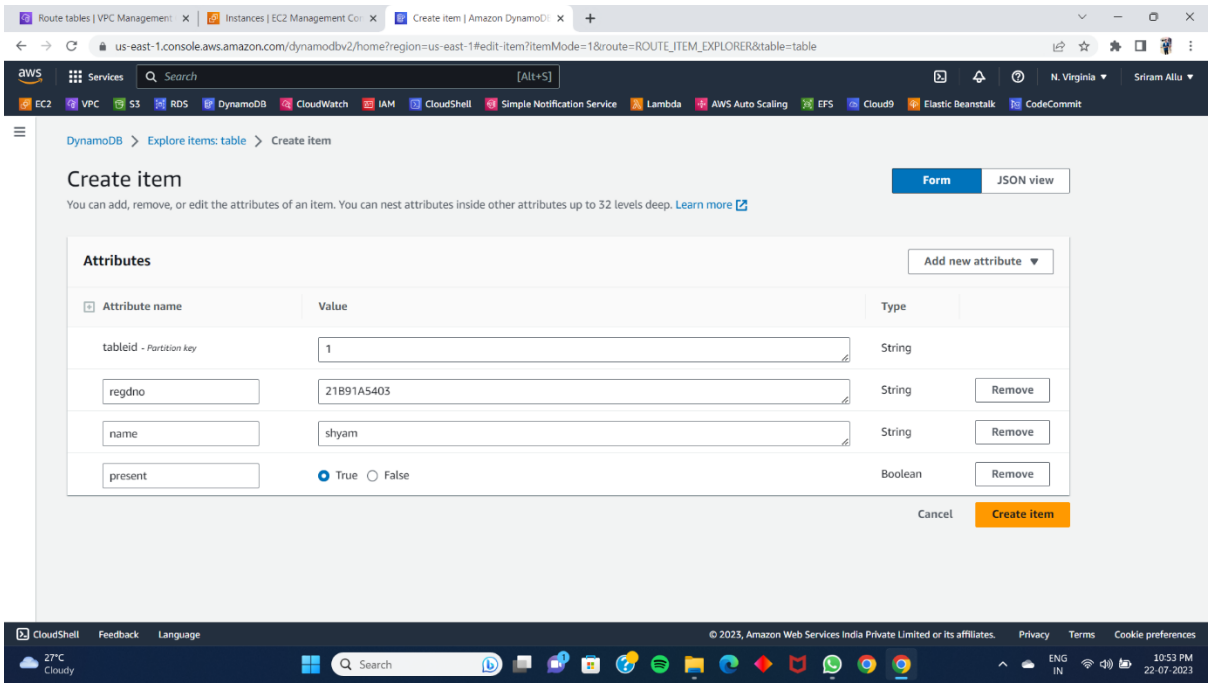
DYNAMODB SERVICE:

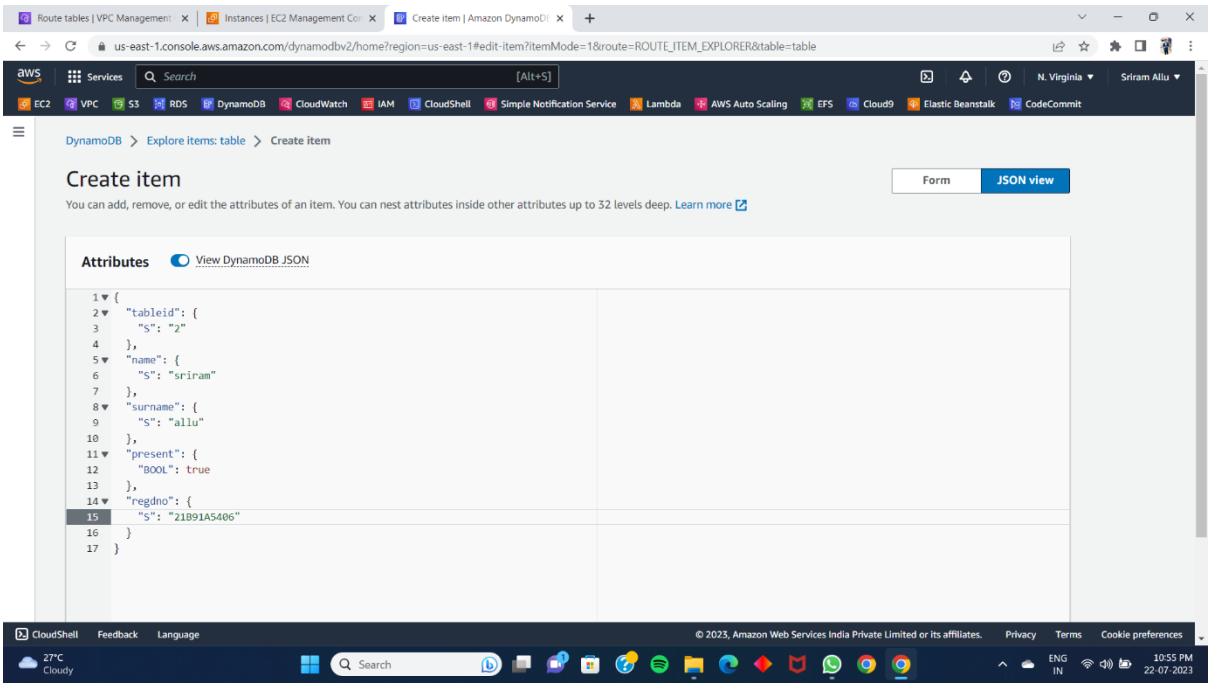
• Create Dynamo db table. Open Dynamodb Create table name it as table.



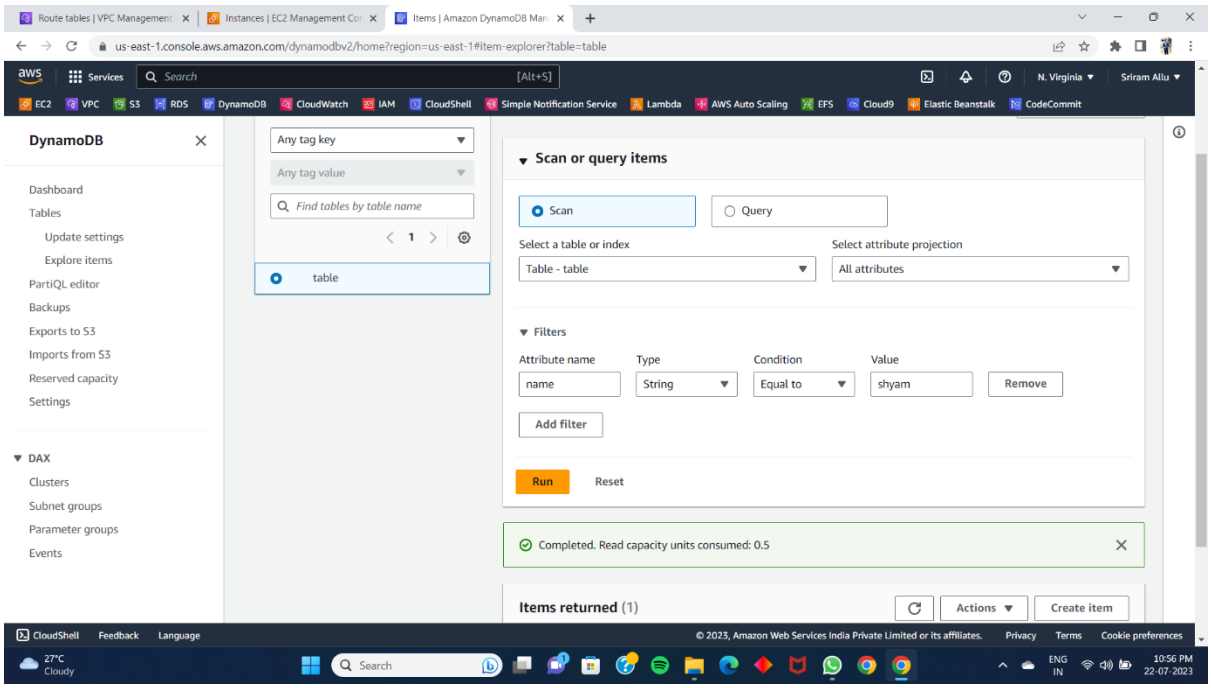


• Create 2 items in table.

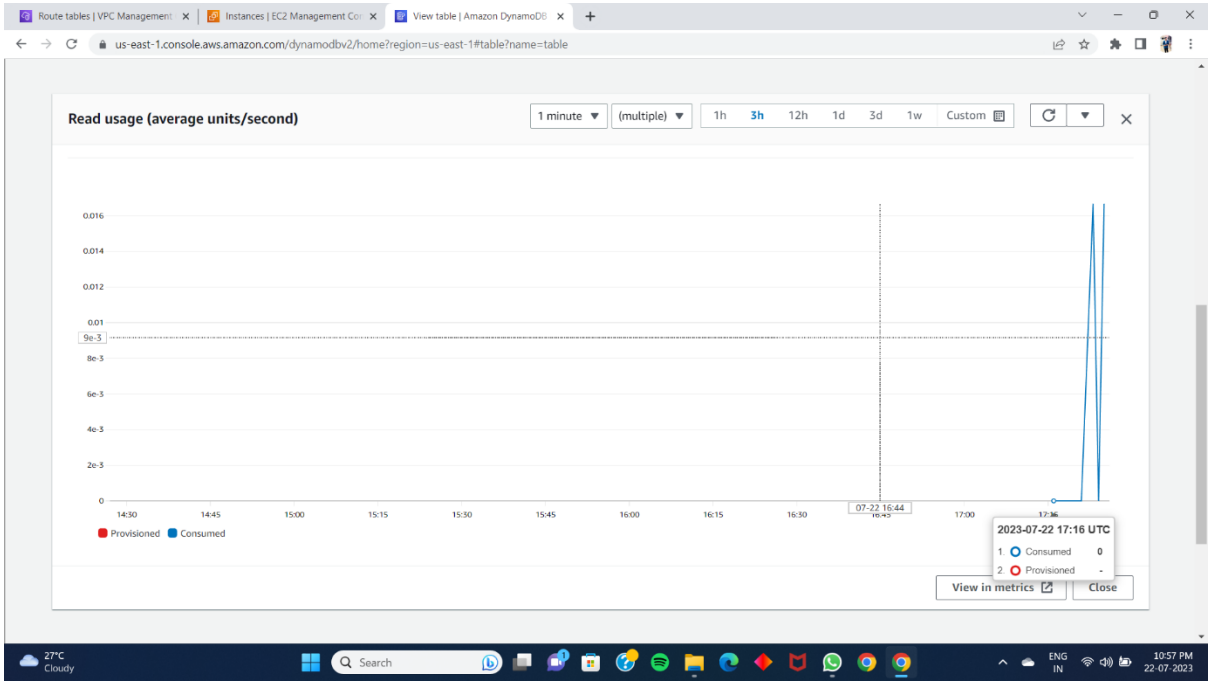




• Scan the items.

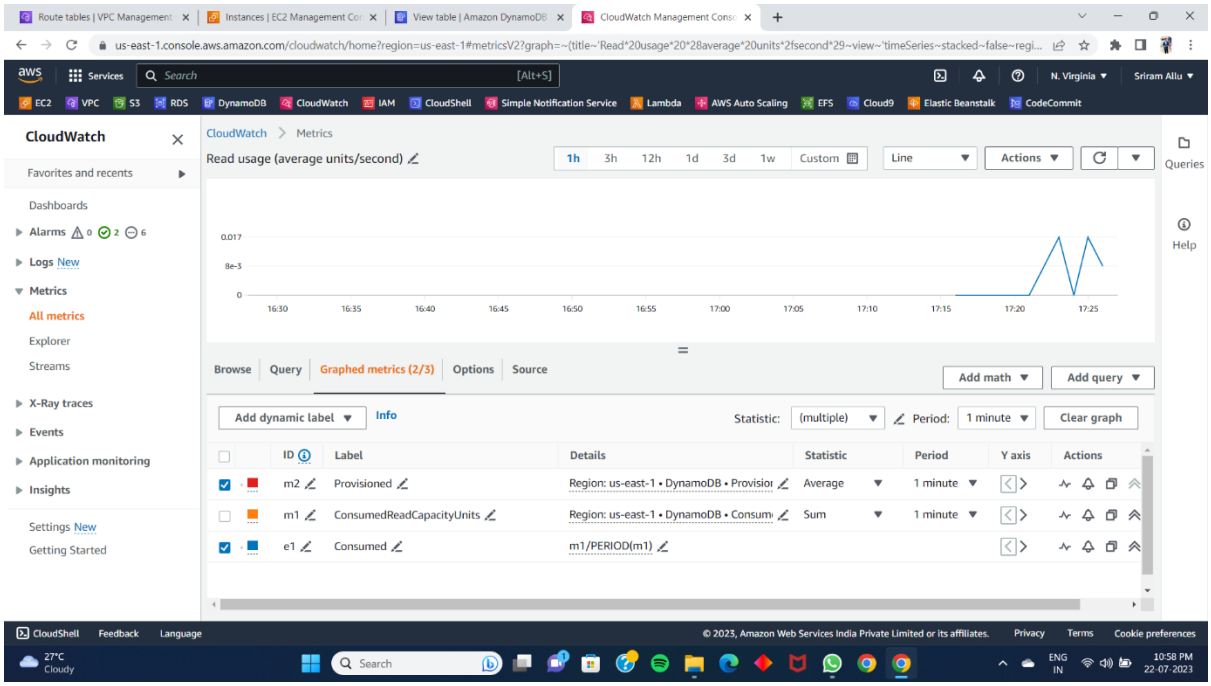


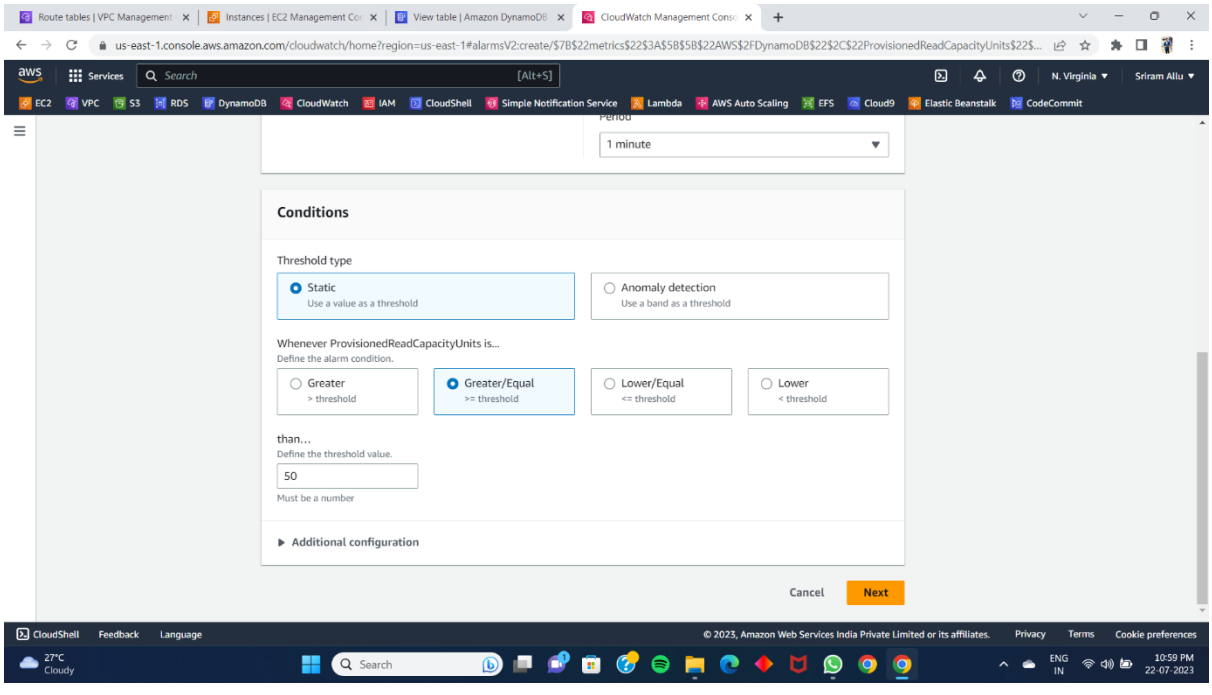
• Read usage of the items.



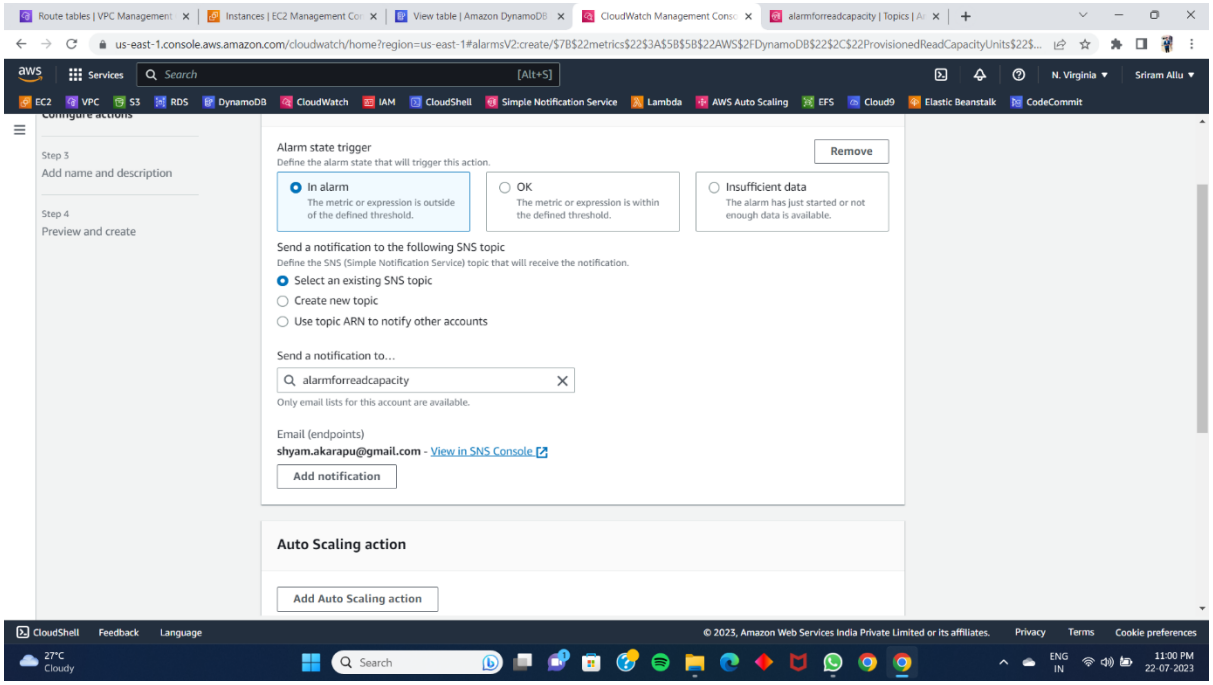
CLOUDWATCH SERVICE:

• Creating a alarm in cloud watch for readcapacity.

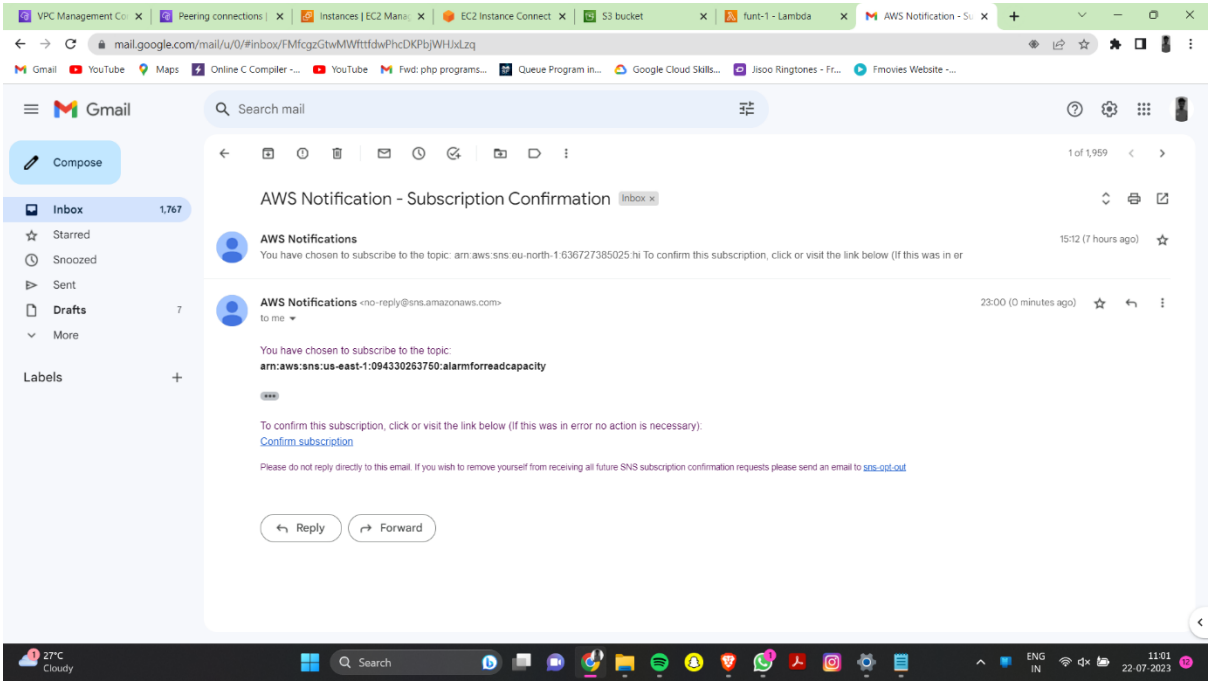




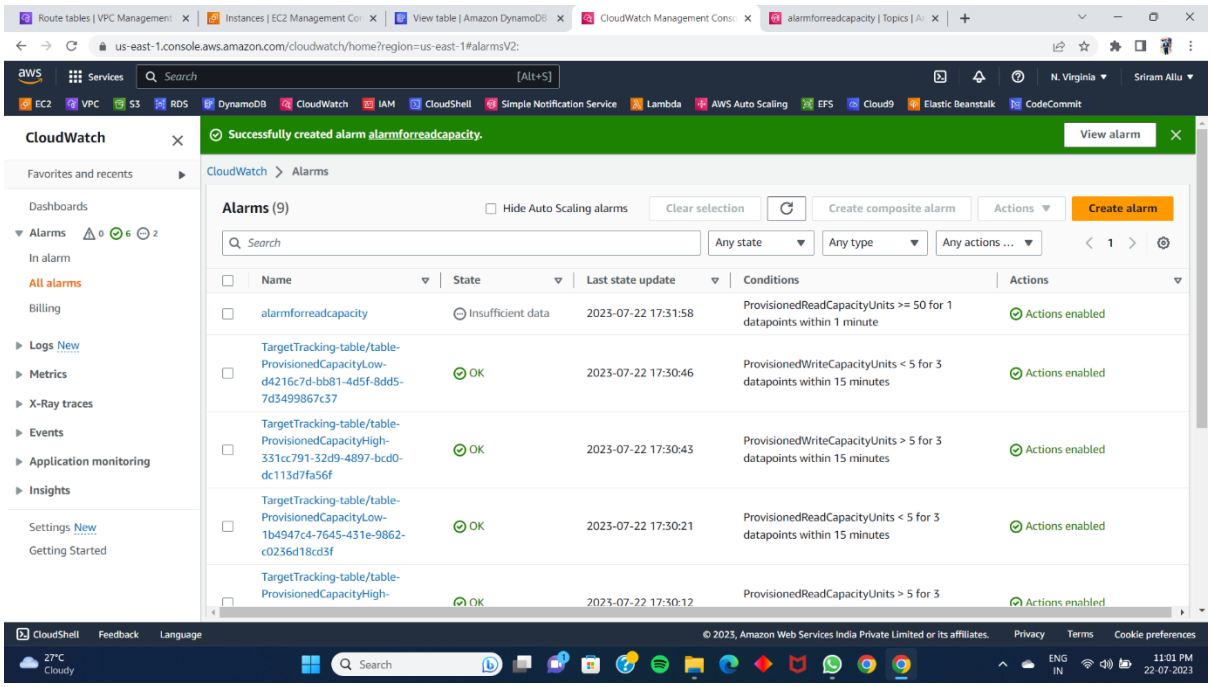
• Creating a topic and sending mail to the subscriber.



• Mail received by subscriber .

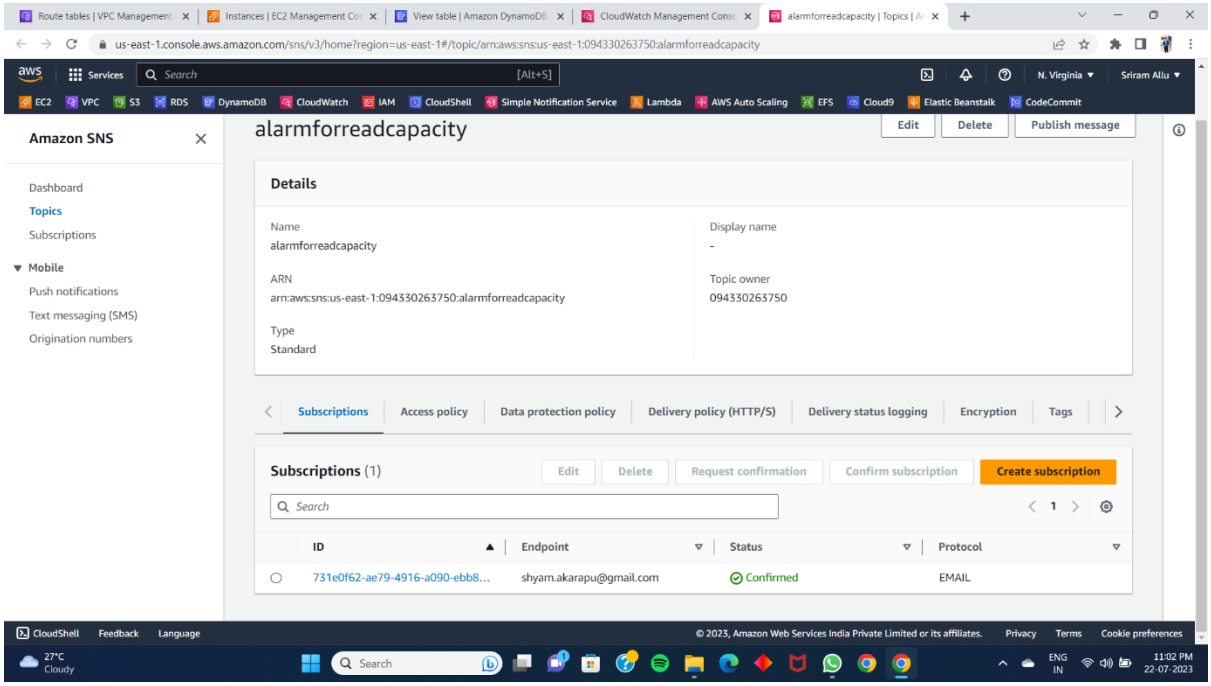


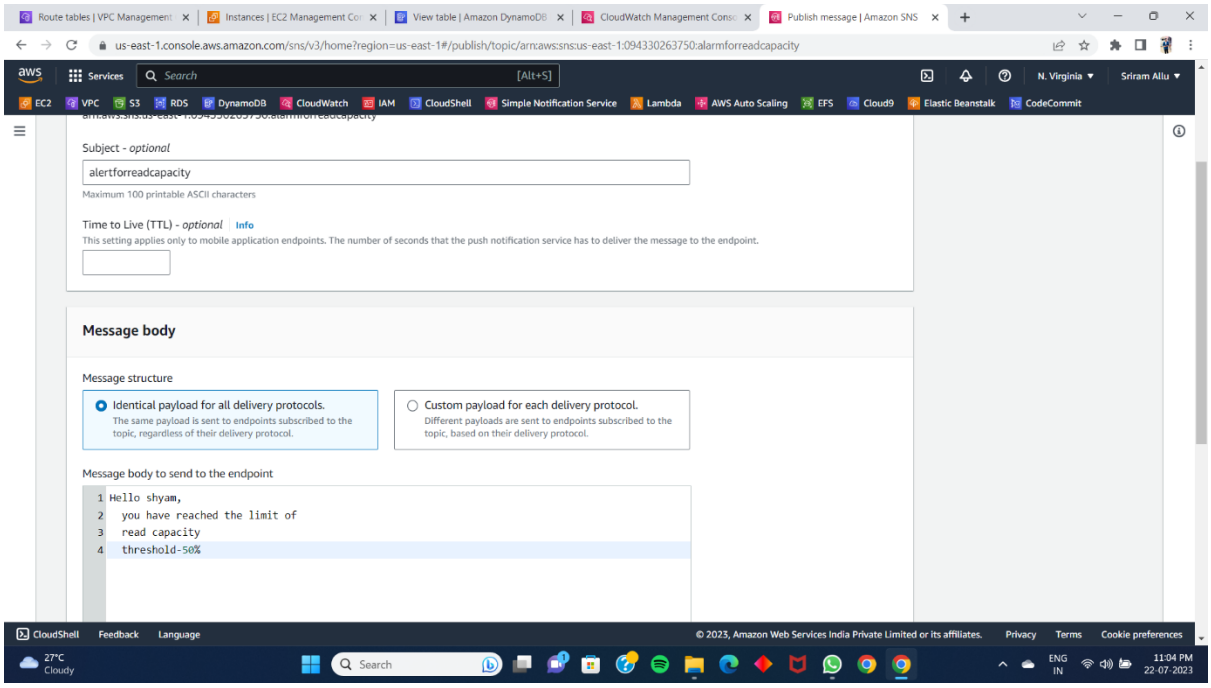
• Alarm created for readcapacity.



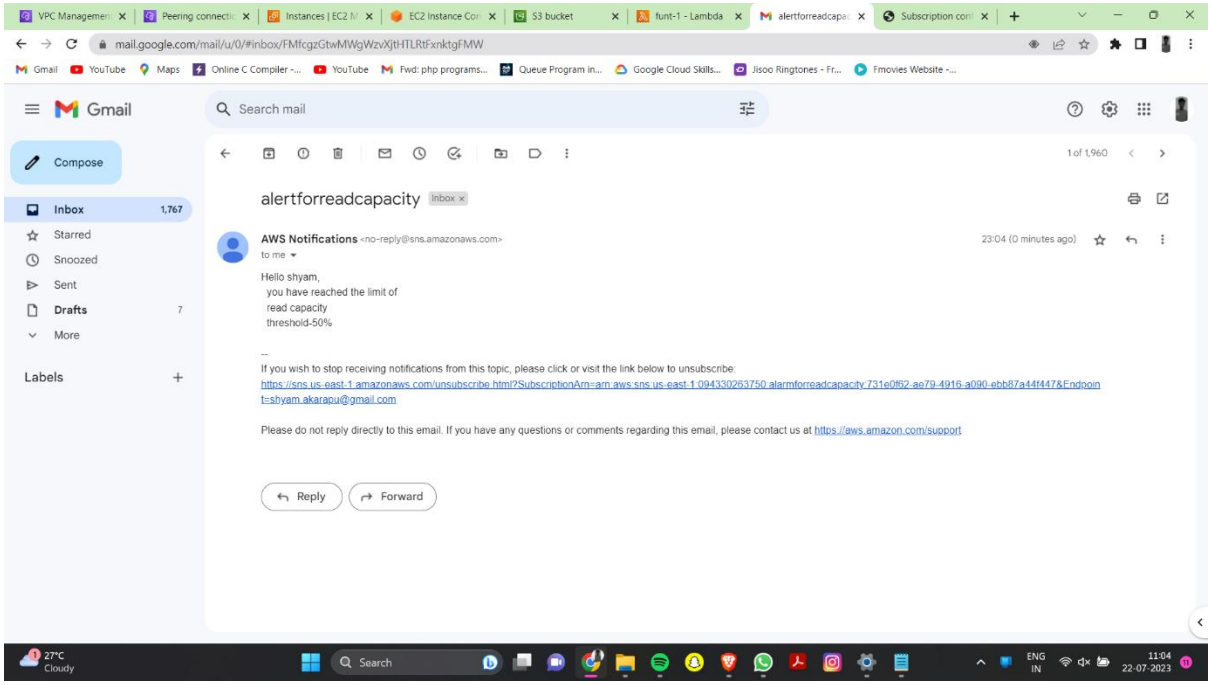
SIMPLE NOTIFICATION SERVICE (SNS):

• Publishing a message using simple notification service(SNS).



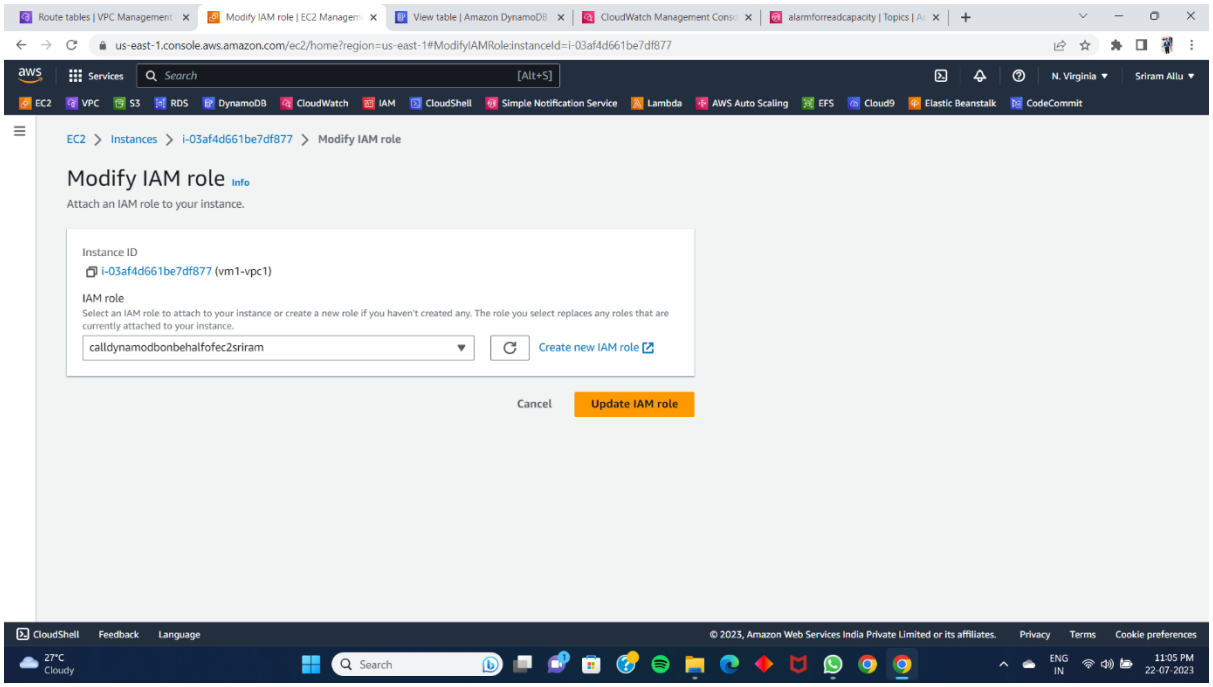


• Message Received by Subscriber regarding Readcapacity.

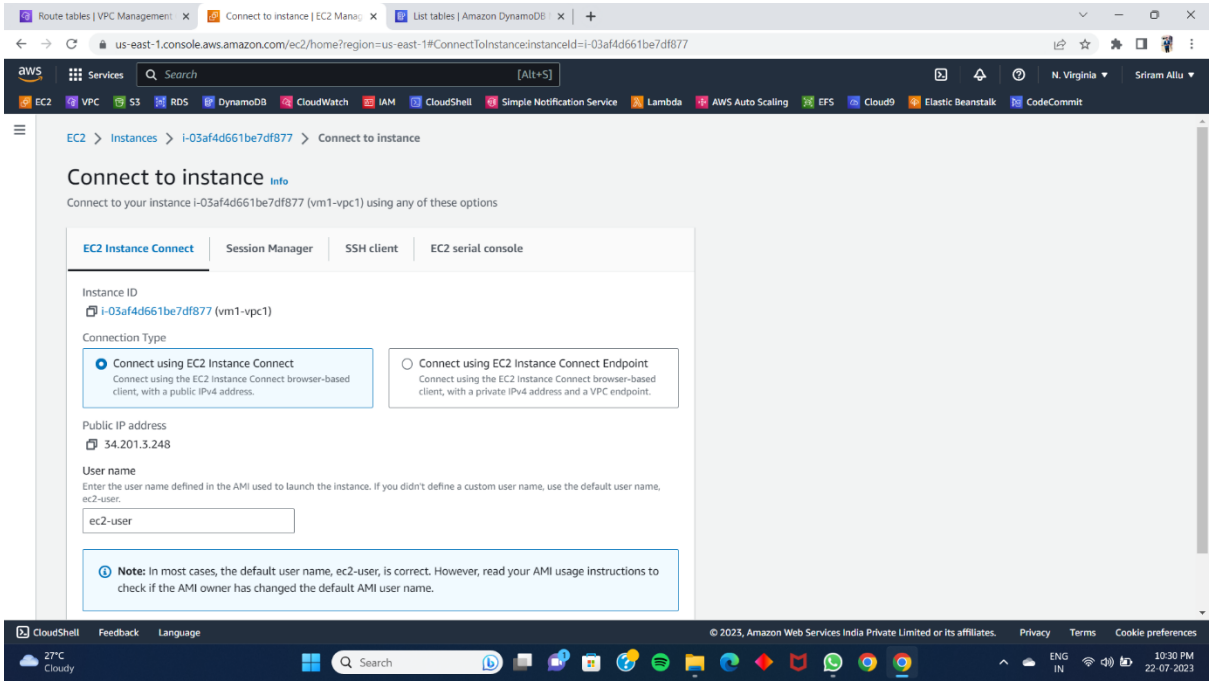


IAM ROLE:

• Creating a Iam Role for accessing dynamodb through Ec2.



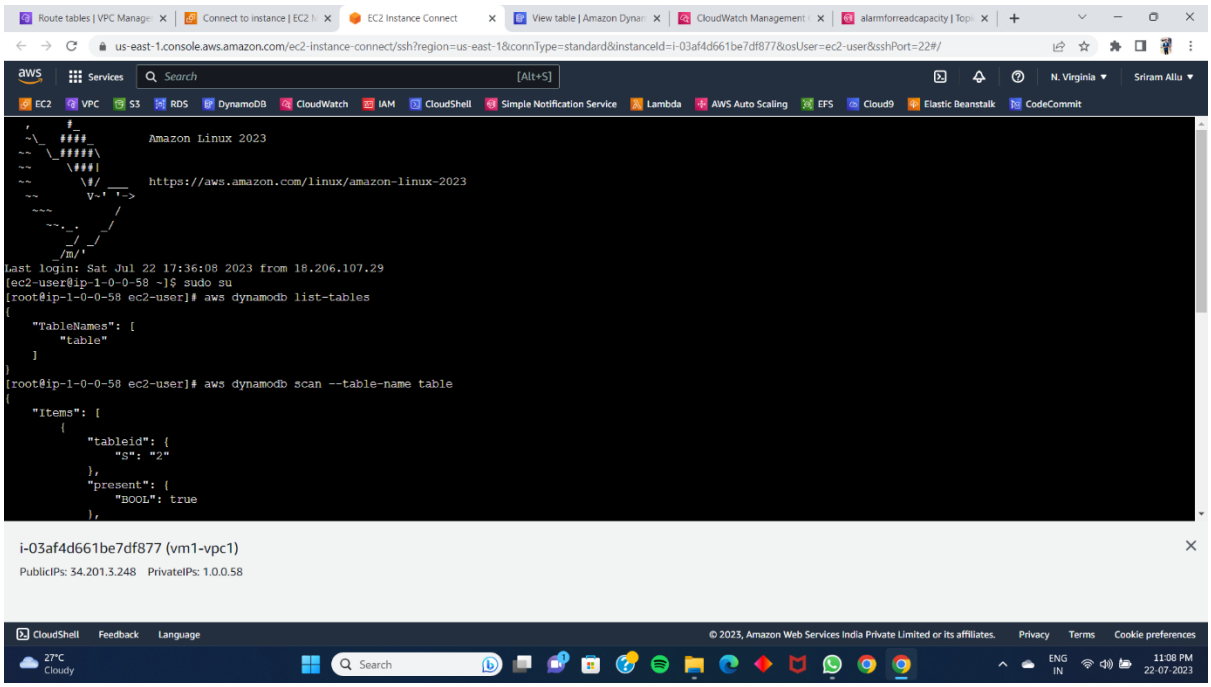
• Connecting a instance.



• Accessing Dynamodb contents through Ec2 instance.

• aws dynamodb list-tables.

• aws dyanodb scan –table-name table are the commands used.



• We created a custom VPC-1.

• Created Custom subnets,internet gateways,edited routes.

• Created Dynamodb Table and and its items.

• Monitored Its readcapacity using Cloudwatch.

• Created a Alarm in cloudwatch for readcapacity.

• Monitored and published a message to the subscriber when readcapacity exceeds 50%.

• Created Iam Role and connected through custom instance.

• Accessed the contents of dynamodb through the EC2 instance.

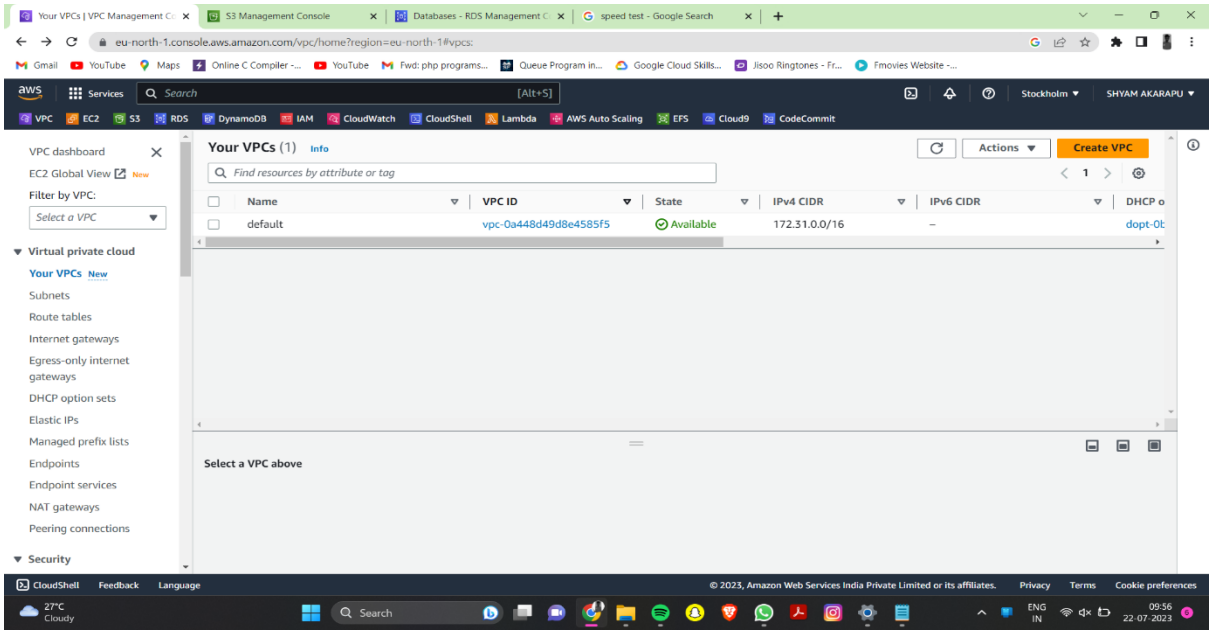
• **the above screeshots are taken by Sriram**.

• Now the remaining architect is continued in another account….

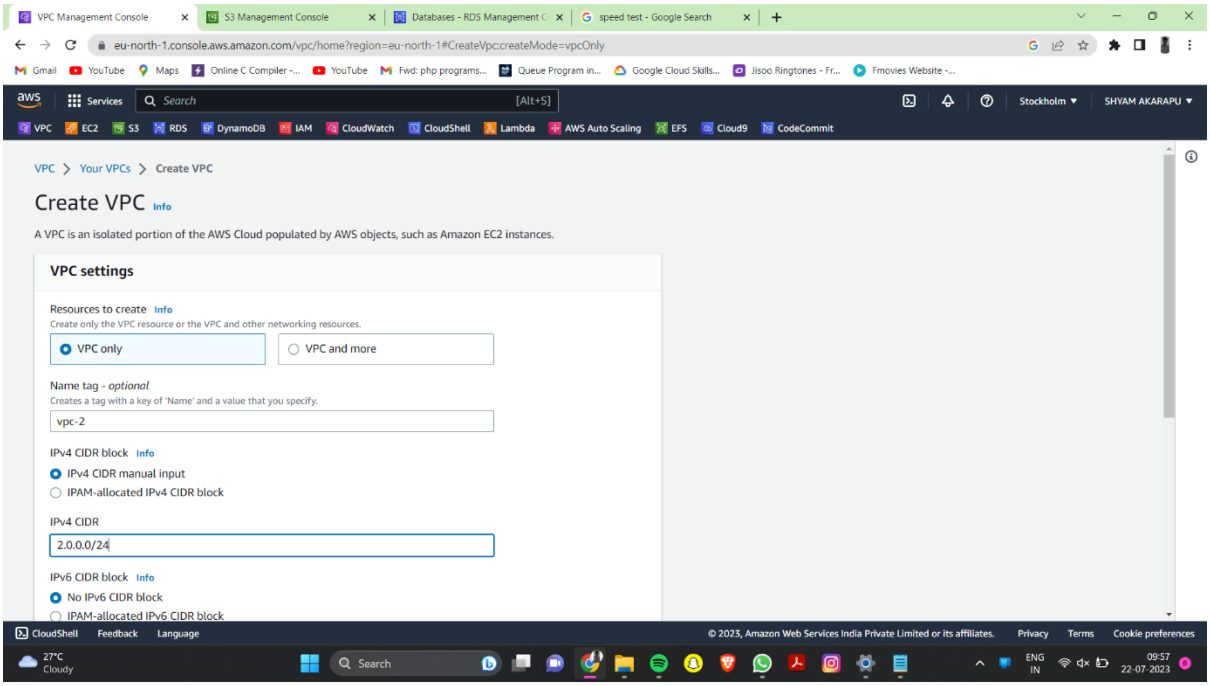
In eu-north-1 Acc id: 6367-2738-5025:-

CUSTOM VPC:

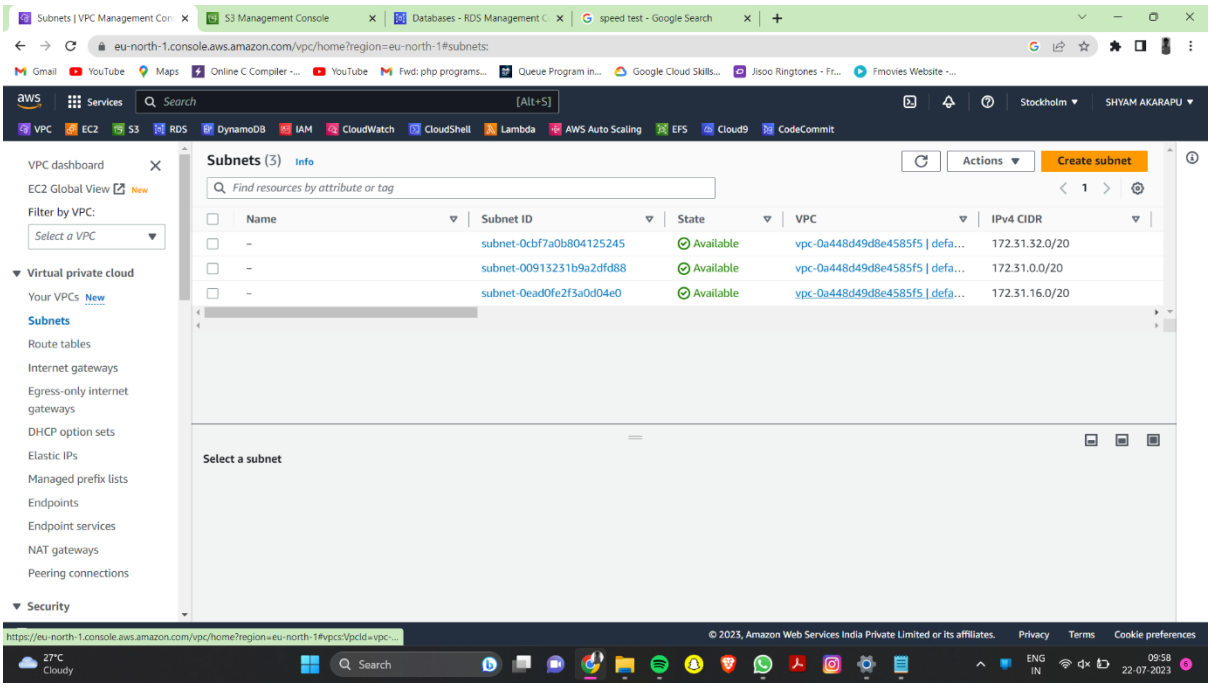
- First Create an another Custom VPC in the region Stockholm.
- To create a VPC with the name "VPC2," follow these steps:
- Sign into the AWS Management Console.
- Open the Virtual Private Cloud Service.
- Click on the "Create VPC" option.
- Give the VPC name as vpc2 and CIDR value as 2.0.0.0/24.
- First usally there a default vpc is present in every region.



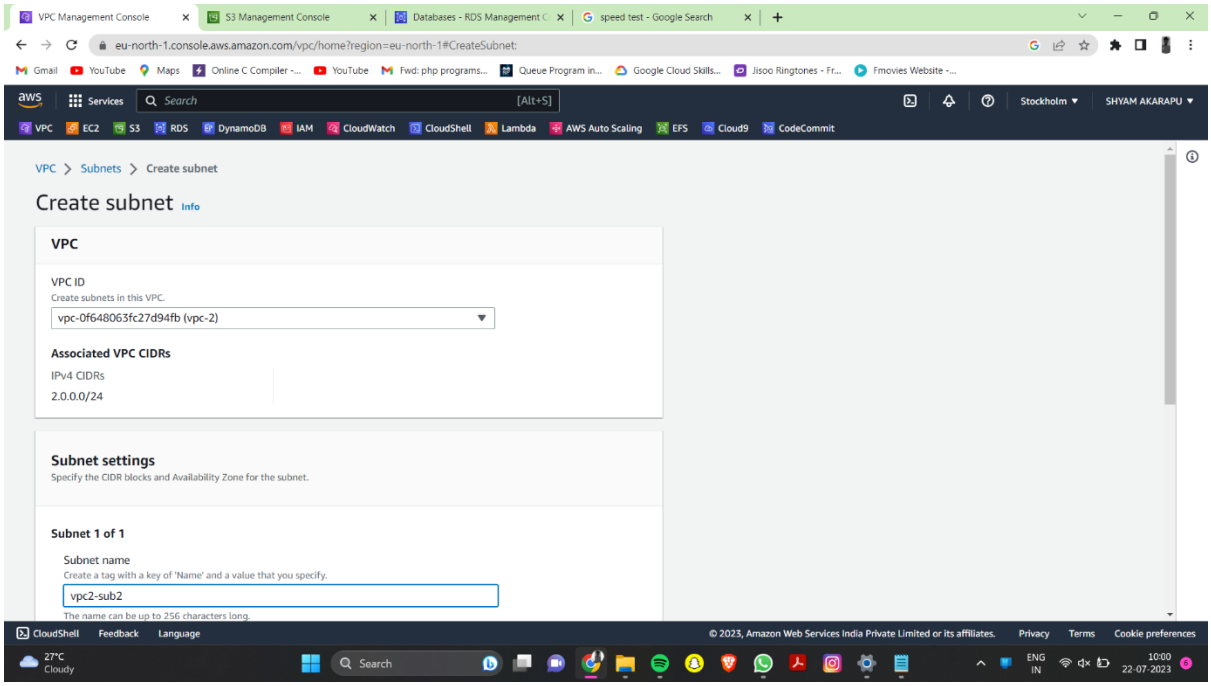
- It usally shows like above snap.
- After we have to click create a vpc
- Then a new custom vpc



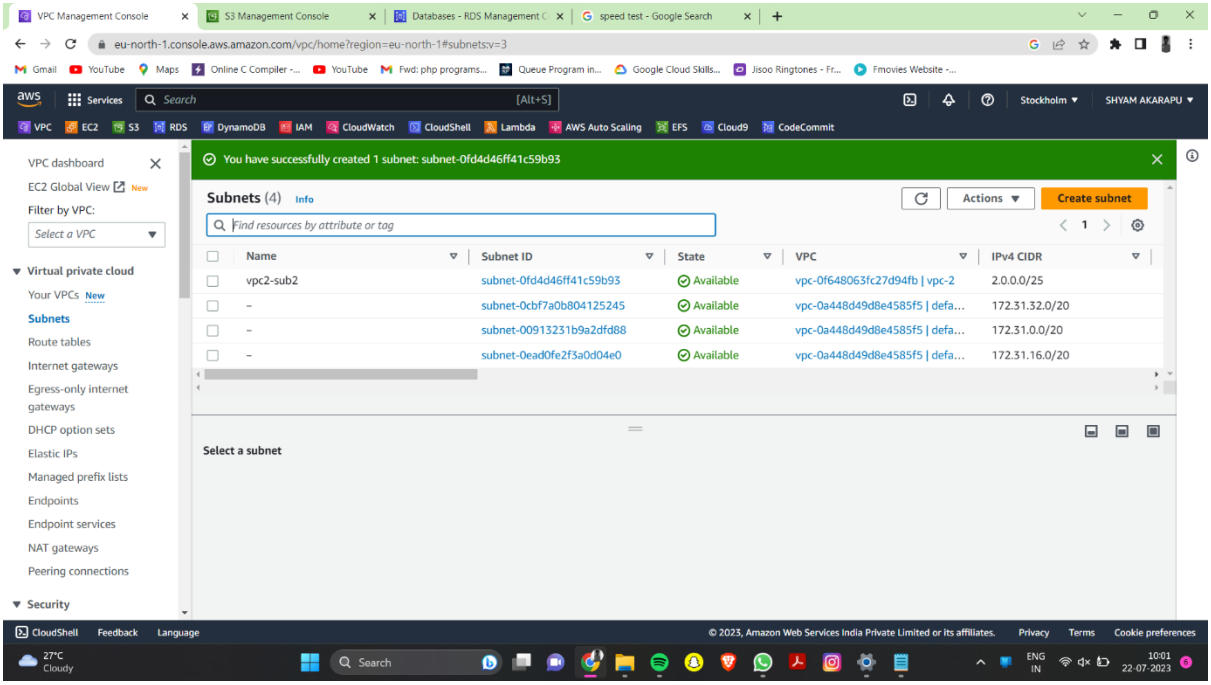
- fill the all the matter for the vpc
- And finally create the vpc in these region.
- **CREATING A SUBNET:**



- There are three defaults subnets are present in the vpc2.
- We have to create a new subnet in region .



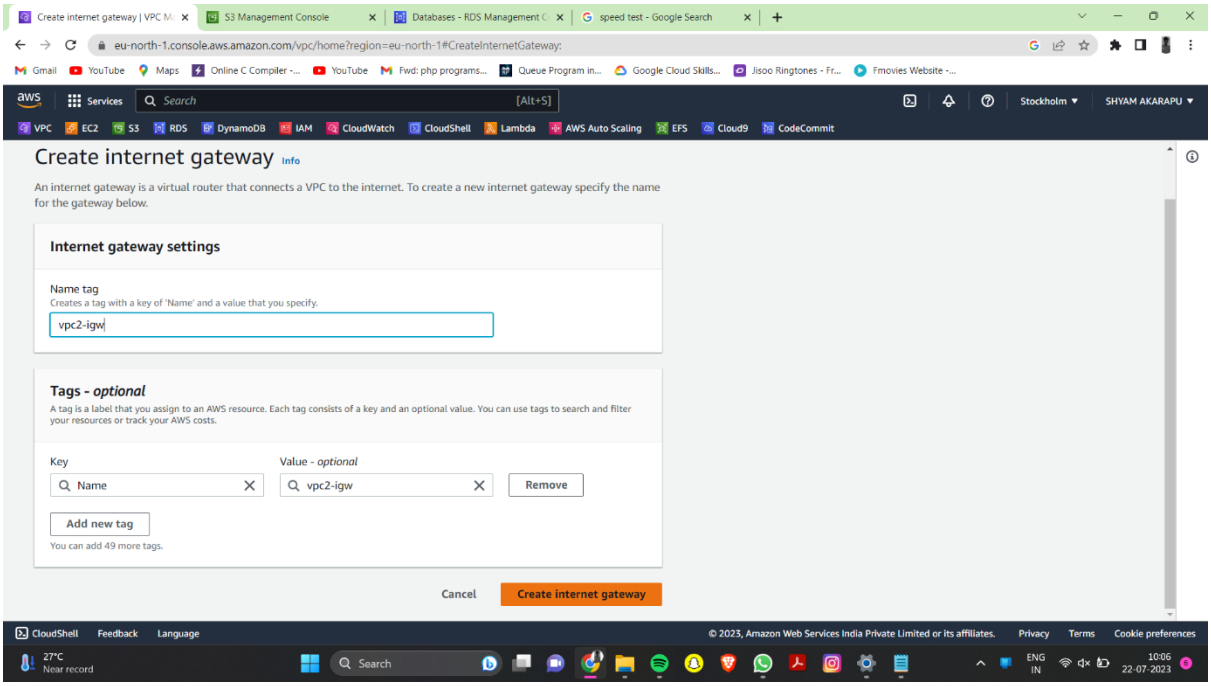
- creating the subnet under vpc2.
- With name of “vpc2-sub-2”.
- AnWith the CIDR 2.0.0.0/25.
- d click on the create the subnet.



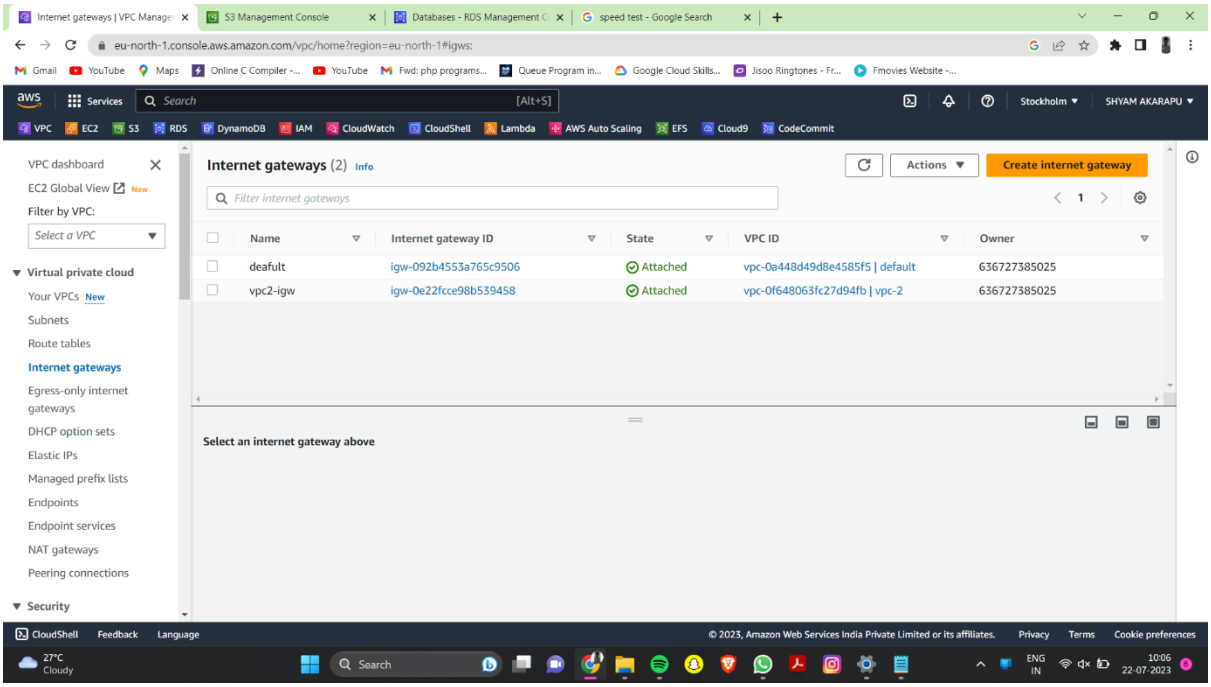
- Now go the route table and edit the subnet association.
- Attach the subnet and save the changes.

CREATING INTERNET GATEWAY:

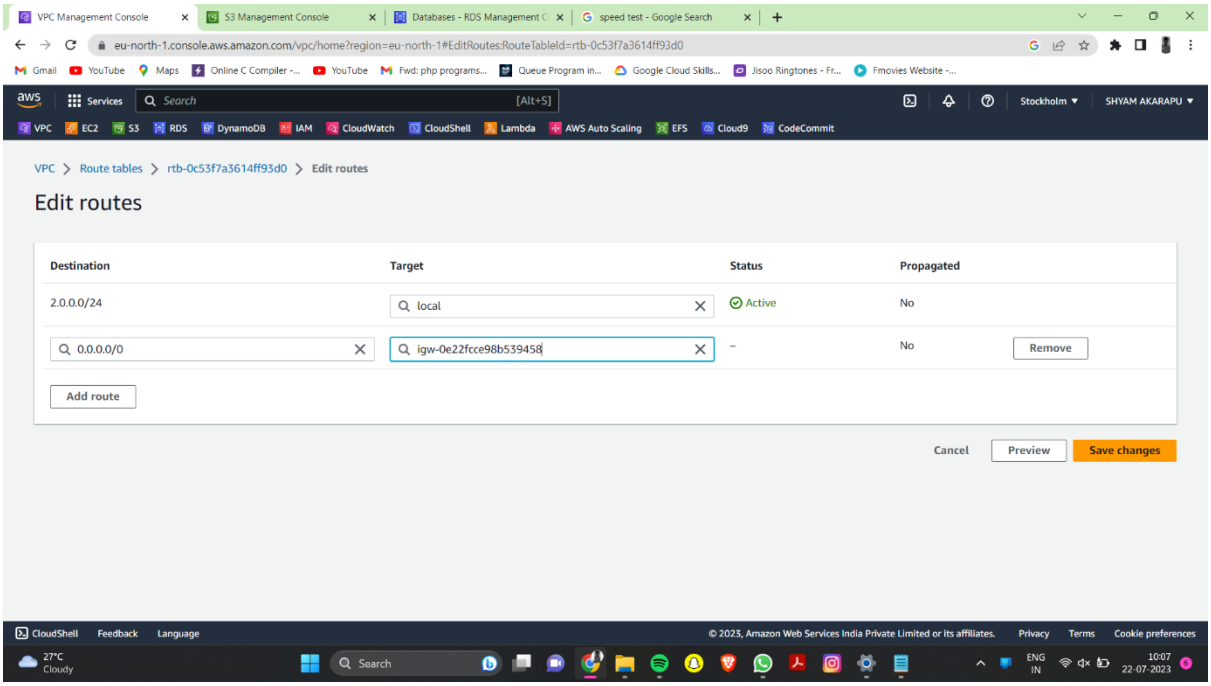
- We have to create internet gateway for to connect to the internet.



- Name the internet gateway by “vpc2-igw”.
- Attach to the vpc 2.



- There is a also default internet gateway for the default vpc to connect to internet.
- After creating the igw in the route table edit the routes.

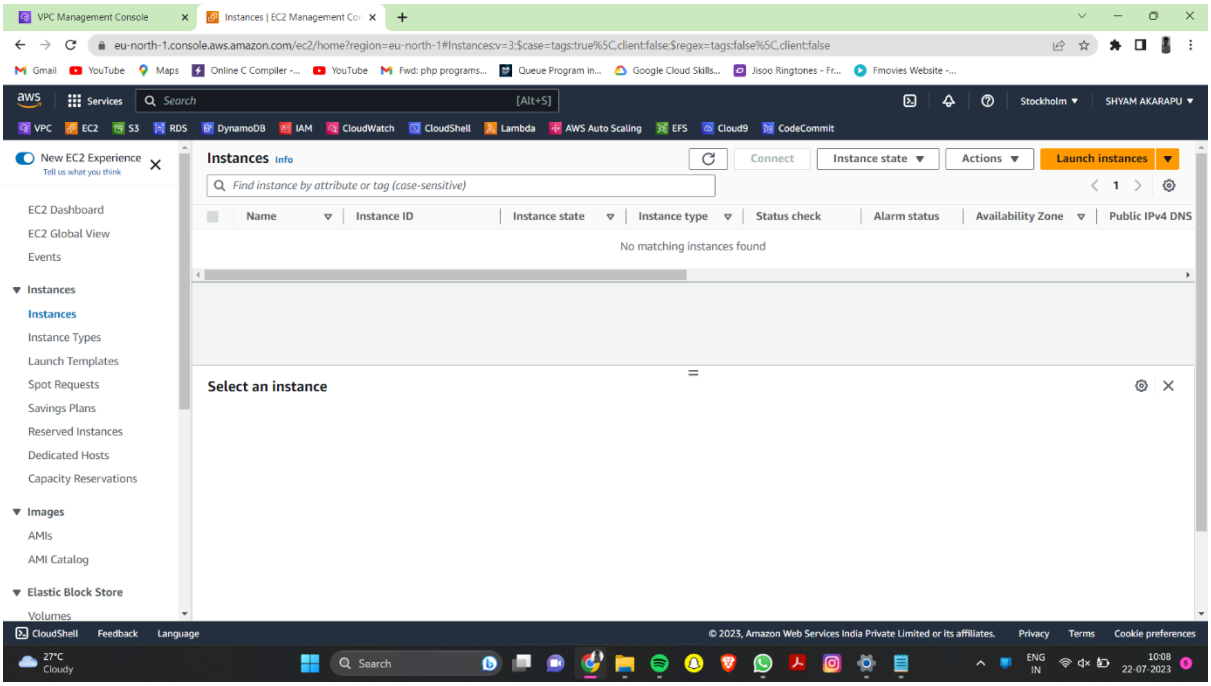


- save the changes.

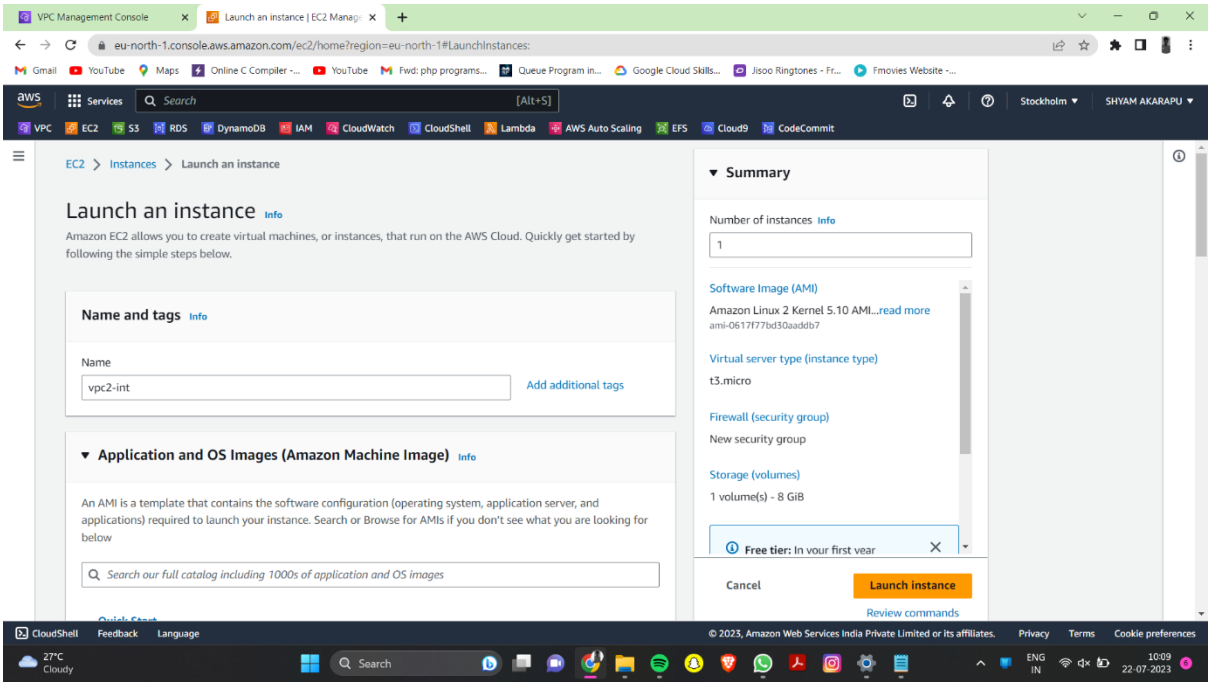
EC2:

CREATING INSTANCE :

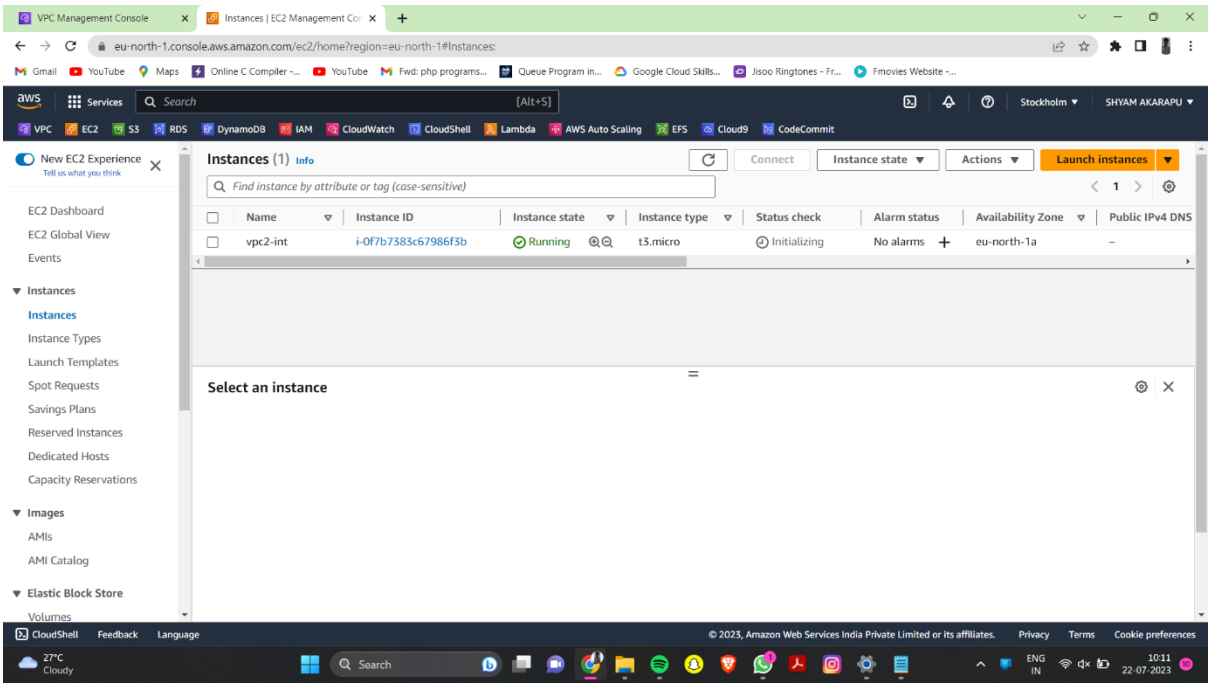
- IN ec2 we have to launch the instances.



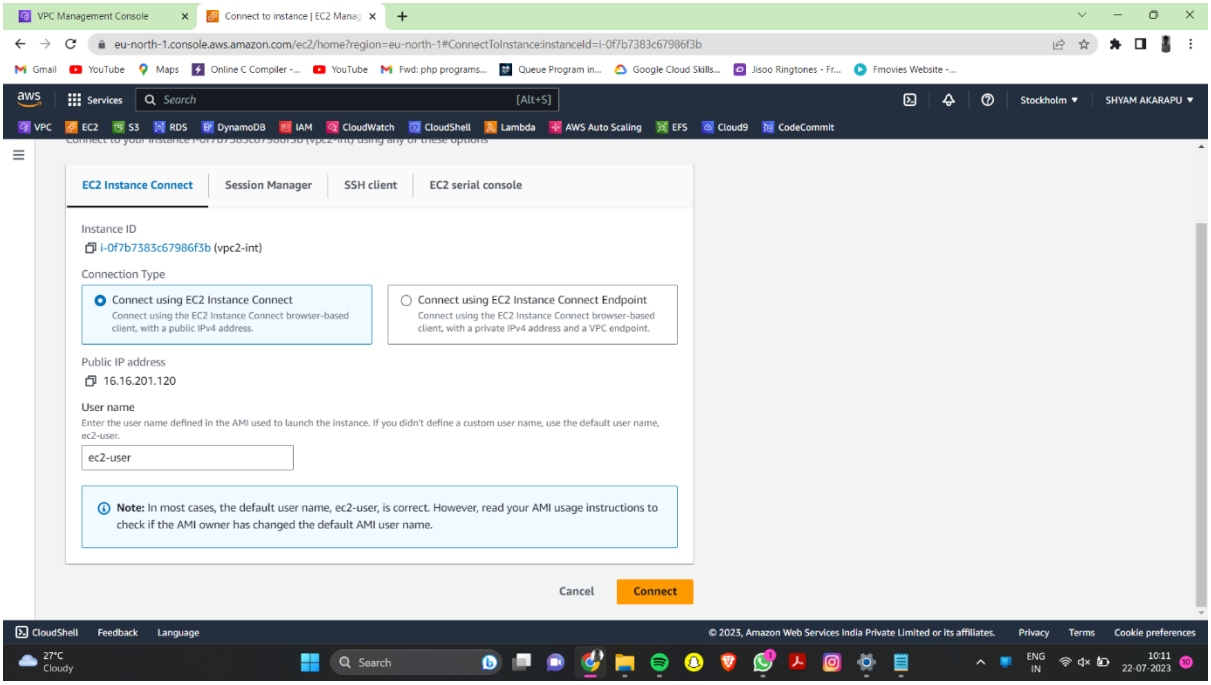
- At first there is no instances present in the ec2
- We have to create a instance for our customs vpc2
- For that we have to launch a new instances.



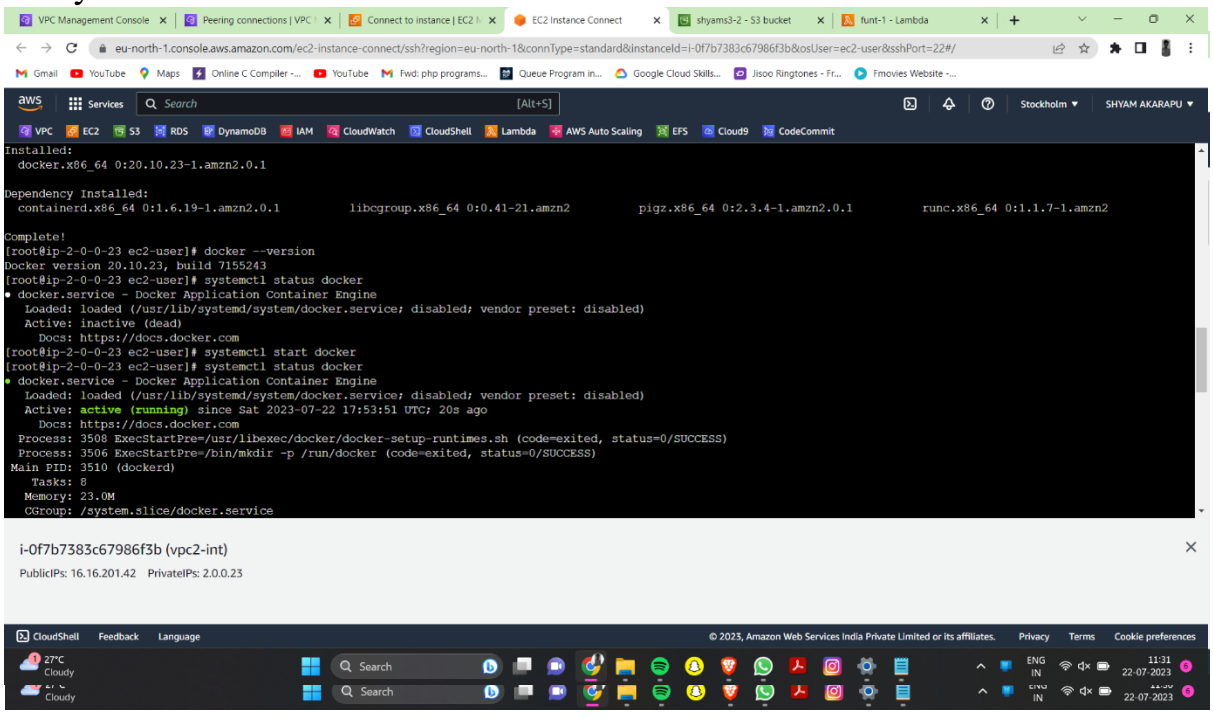
- Name the instance by vpc2-int1>
- With the amazon unix.
- And it consists of HVM.
- And it was created by the t3.micro.
- The instances is creating under the vpc2,subnet2.
- Create a new security group called sg1.
- And launch the instance.

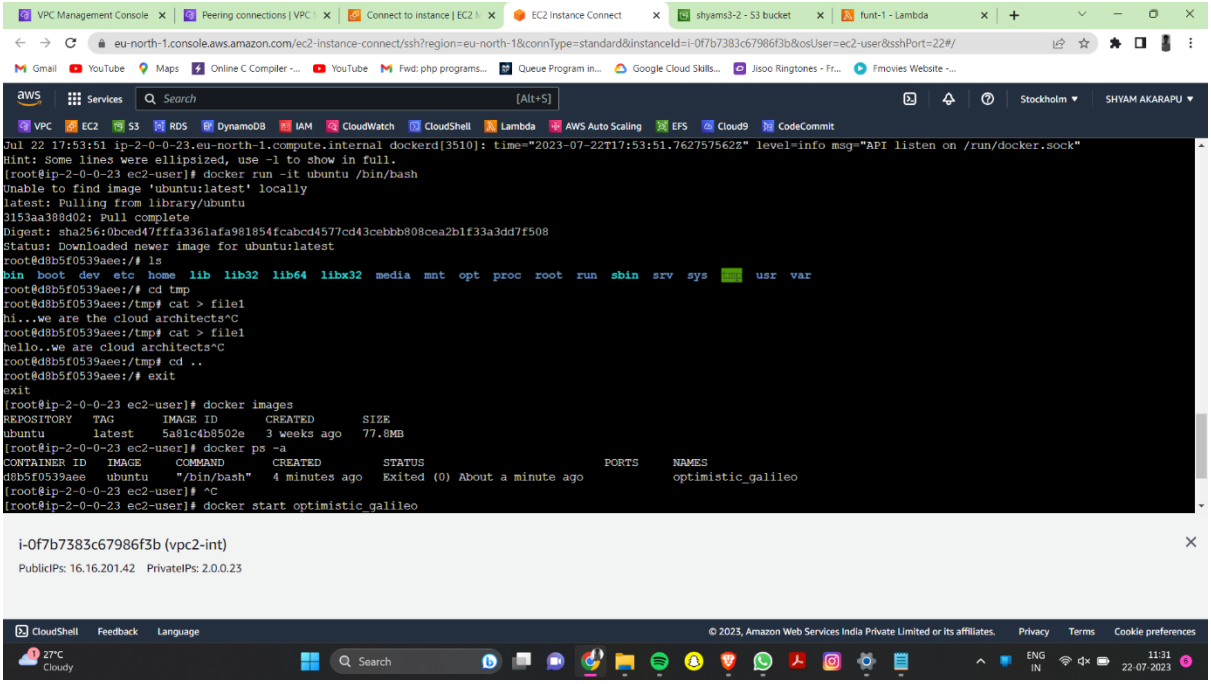


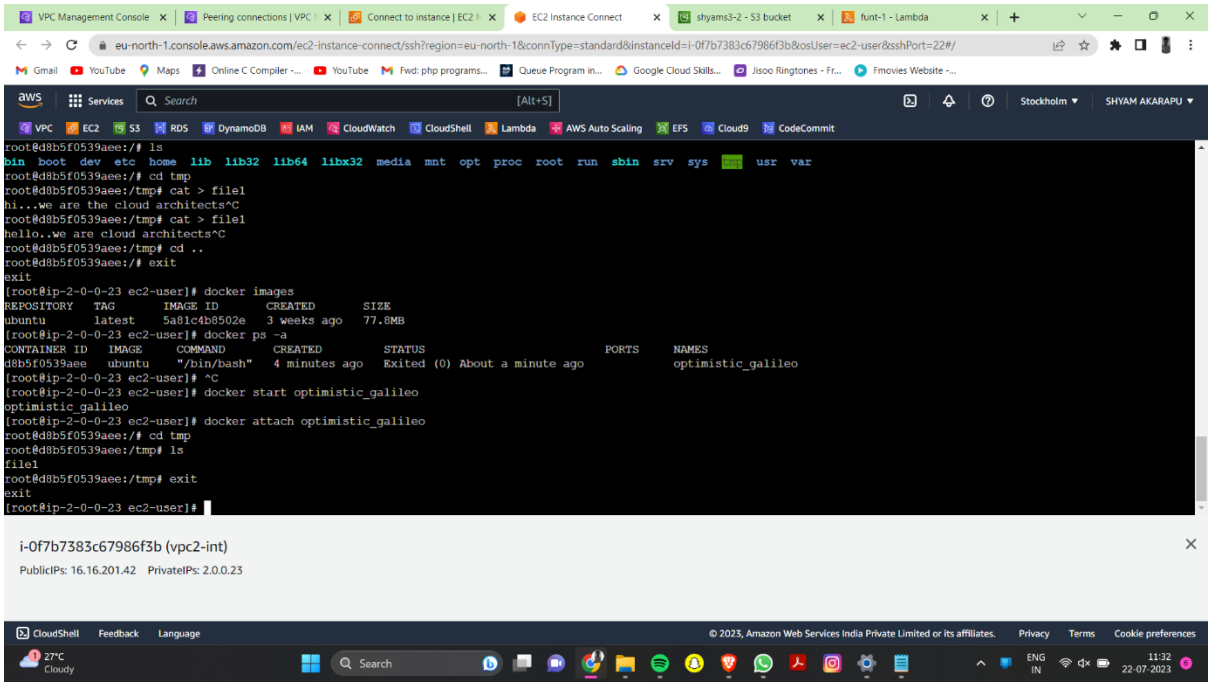
- After that connect the instance to the console.



- The public ip is 16.16.201.120
- The private ip is 2.0.0.23
- After that install the docker engine to the instance.
- By the following commands
- Sudo su
- Yum install docker -y
- Docker –version
- Docker -v
- Systemctl status docker
- System start docker
- System status docker




VPC Management Console
Peering connections | VPC
Connect to instance | EC2
EC2 Instance Connect
shyams3-2 - S3 bucket
funt-1 - Lambda
eu-north-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=eu-north-1&connType=standard&instanceId=i-0f7b7383c67986f3b&osUser=ec2-user&sshPort=22#/
Gmail
YouTube
Maps
Online C Compiler -...
YouTube
Fwd: php programs...
Queue Program in...
Google Cloud Skills...
Jisoo Ringtones - Fr...
Fmovies Website -...
aws
Services
Search
[Alt+S]
Stockholm
SHYAM AKARAPU
VPC
EC2
S3
RDS
DynamoDB
IAM
CloudWatch
CloudShell
Lambda
AWS Auto Scaling
EFS
Cloud9
CodeCommit
Jul 22 17:53:51 ip-2-0-0-23.eu-north-1.compute.internal dockerd[3510]: time="2023-07-22T17:53:51.762757562Z" level=info msg="API listen on /run/docker.sock"
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-2-0-0-23 ec2-user]# docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
3153aa388d02: Pull complete
Digest: sha256:0bced47fffa3361afa981854fcabcd4577cd43cebbb808cea2b1f33a3dd7f508
Status: Downloaded newer image for ubuntu:latest
root@d8b5f0539aee:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@d8b5f0539aee:/# cd tmp
root@d8b5f0539aee:/tmp# cat > file1
hi...we are the cloud architects^C
root@d8b5f0539aee:/tmp# cat > file1
hello..we are cloud architects^C
root@d8b5f0539aee:/tmp# cd ..
root@d8b5f0539aee:/# exit
exit
[root@ip-2-0-0-23 ec2-user]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu latest 5a81c4b8502e 3 weeks ago 77.8MB
[root@ip-2-0-0-23 ec2-user]# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d8b5f0539aee ubuntu "/bin/bash" 4 minutes ago Exited (0) About a minute ago optimistic_galileo
[root@ip-2-0-0-23 ec2-user]# ^C
[root@ip-2-0-0-23 ec2-user]# docker start optimistic_galileo
i-0f7b7383c67986f3b (vpc2-int)
PublicIPs: 16.16.201.42 PrivateIPs: 2.0.0.23
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
27°C
Cloudy
Search
ENG
IN
11:31
22-07-2023

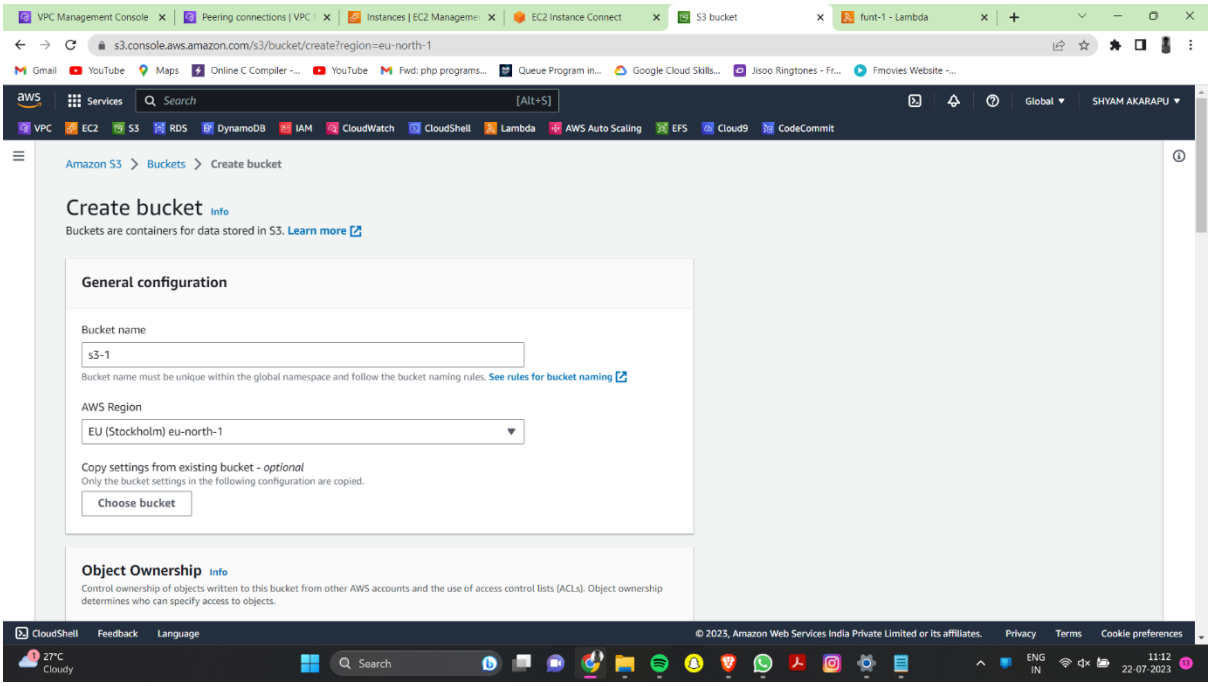


- we are exit from the docker.

S3 SERVICE:

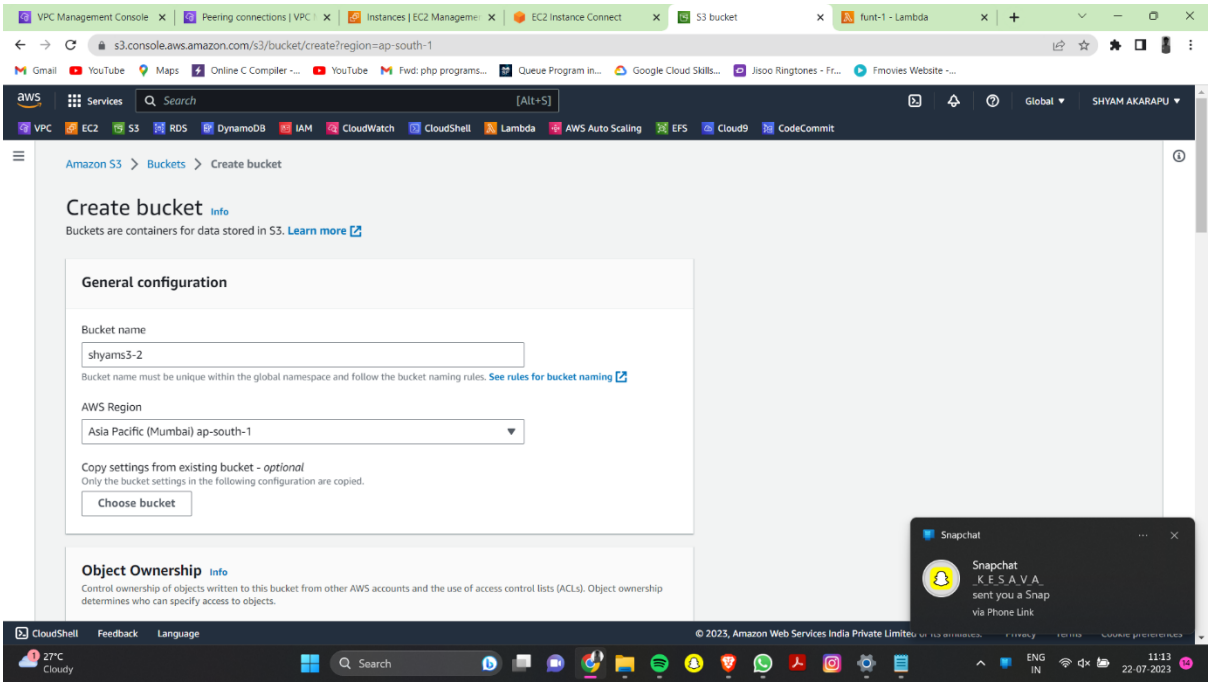
CREATING A S3 BUCKET:

- S3 bucket is the storage systems .
- The buckets are created by the unqie names.
- At first there is no buckets are present .
- S3 is a global service but buckets are created under the regions.
- In these we have to create two buckets.

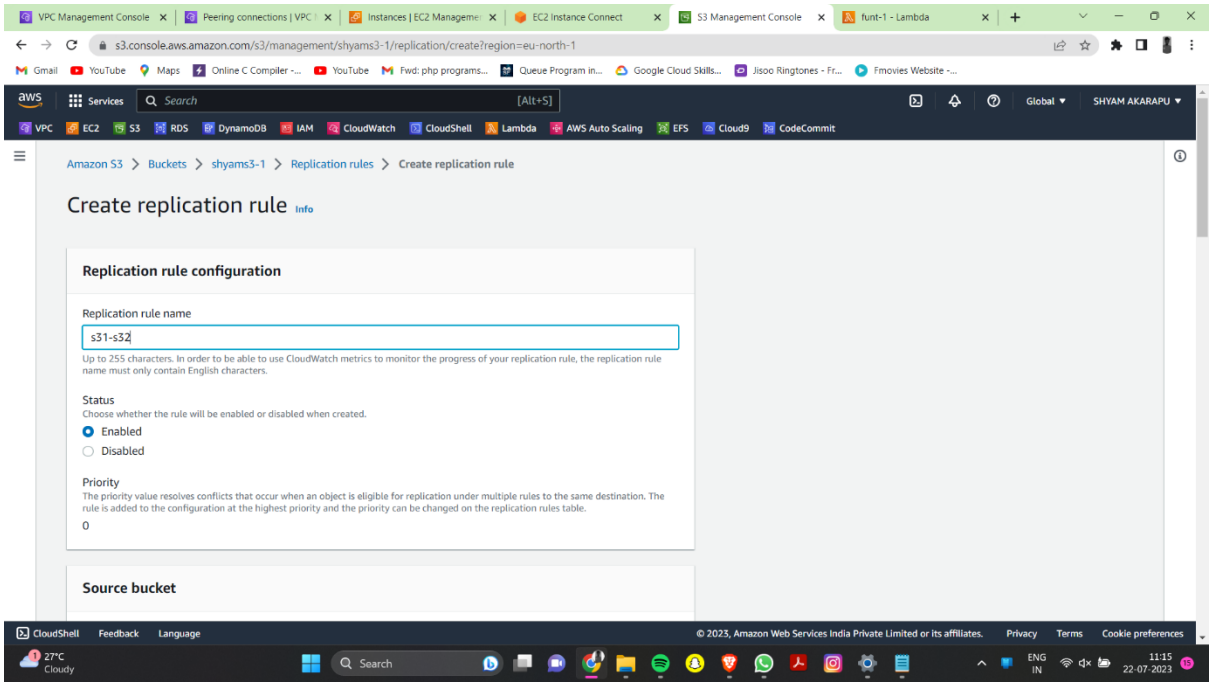


- With name of “shyams3-1”.
- In the eu(Stockholm) eu-north-1.

- Enable the bucket versioning
- And enable the alc.
- And create the bucket.
- And we have to create the another bucket with name of “shyams3-2”.



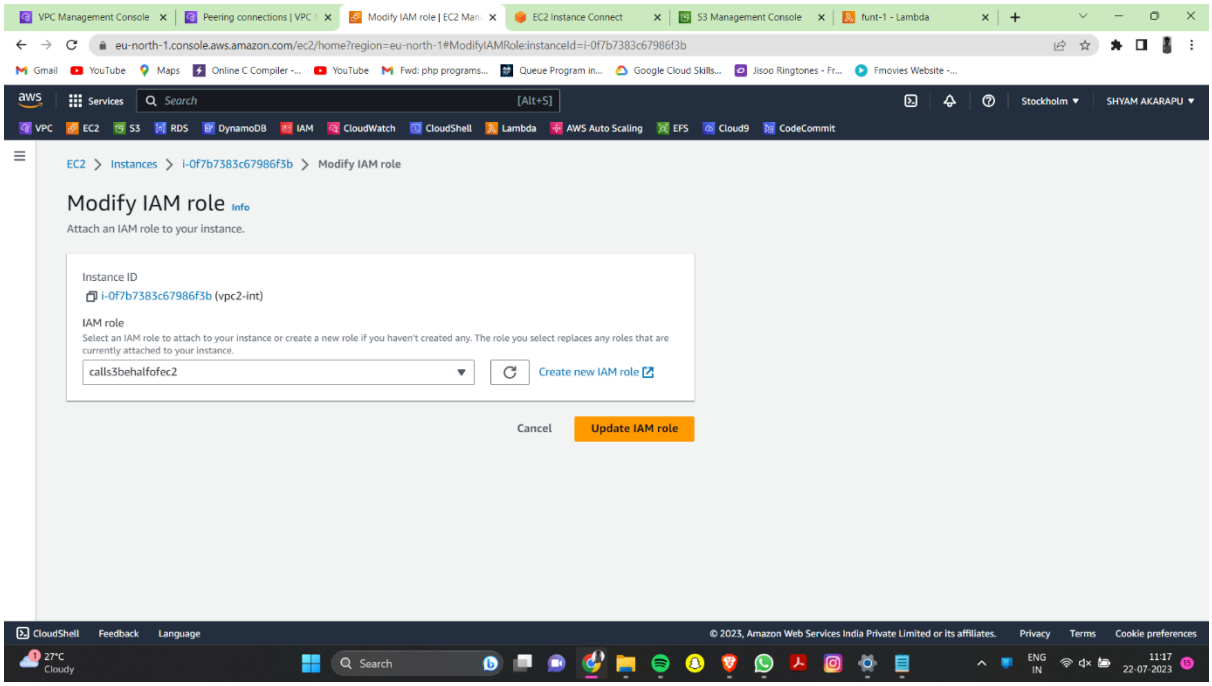
- It is creates in the asia pacific (Mumbai) ap-south-1 region.
- Enable the bucket versioning.
- And also enable the alc.
- Initially there are no objects are present in the both the buckets.
- After that we have to create cross region replication for the buckets .
- So copying the information from one bucket to another bucket is possible.
- Create the cross region replication for the shyams3-1 bucket.
- So that objects present the shyams3-1 bucket is copied to the shyams3-2.



- Name the cross region replication s31-s32.
- Create the crr from shyams3-1 to shyams3-2.

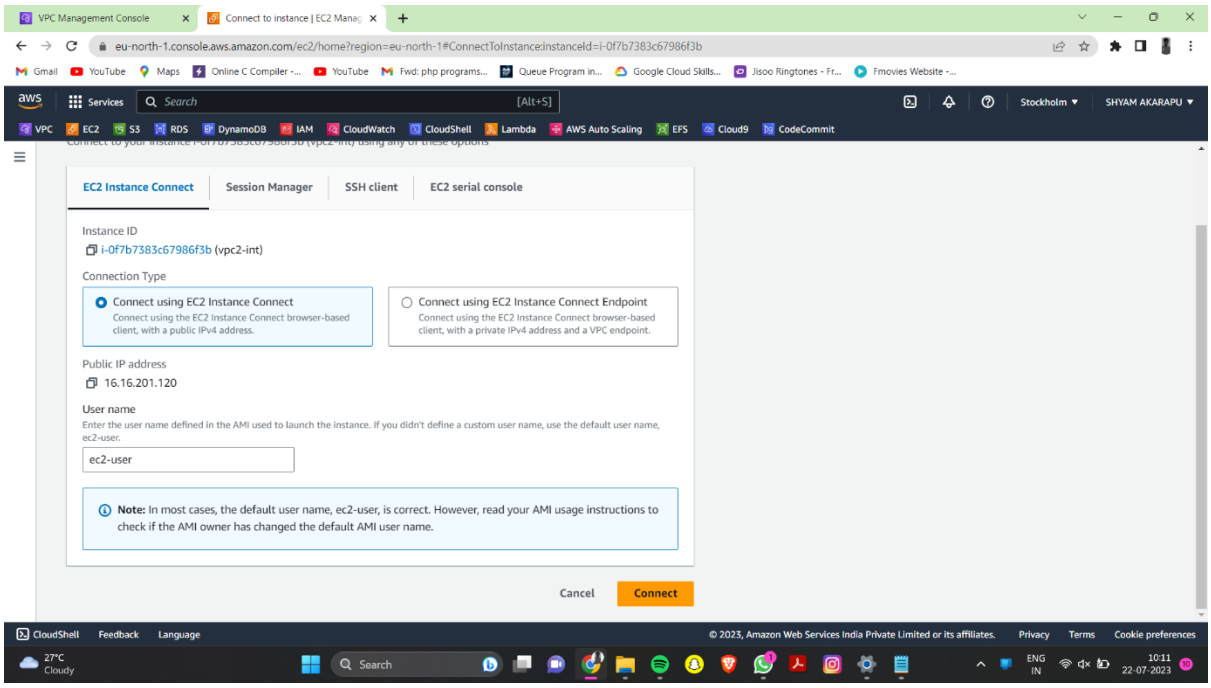
Iam role:

Modifying the iam role to instance:

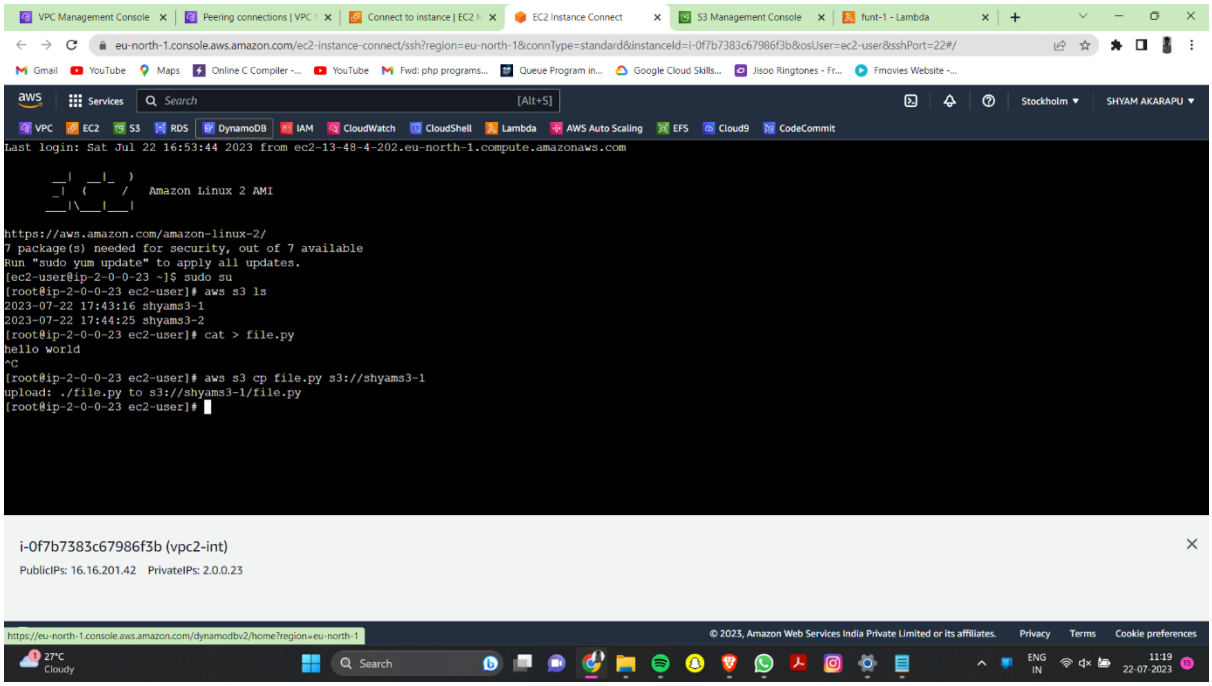


- We have to modify the iam role of the instance.
- Because for accesing the s3 buckets by the instances we have modify.
- So the we have to connect the s3 buckets though the instance terminal.

- So that we have to connect the instance

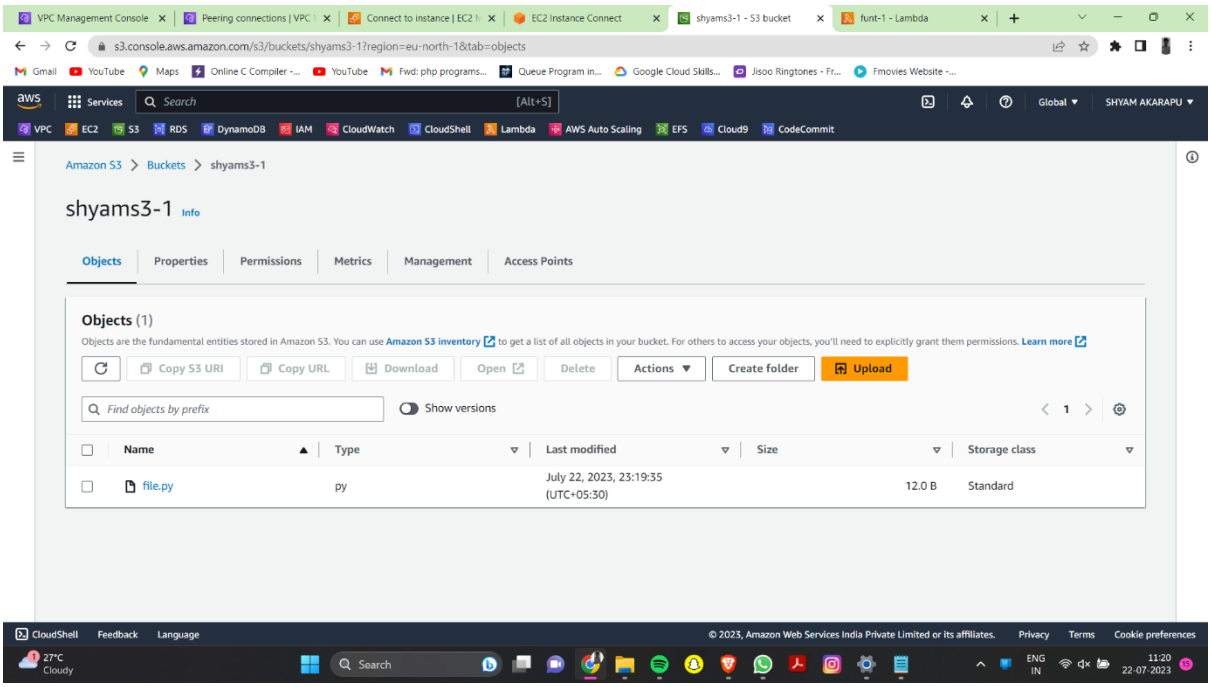


- After connecting we have acces the s3 bucket

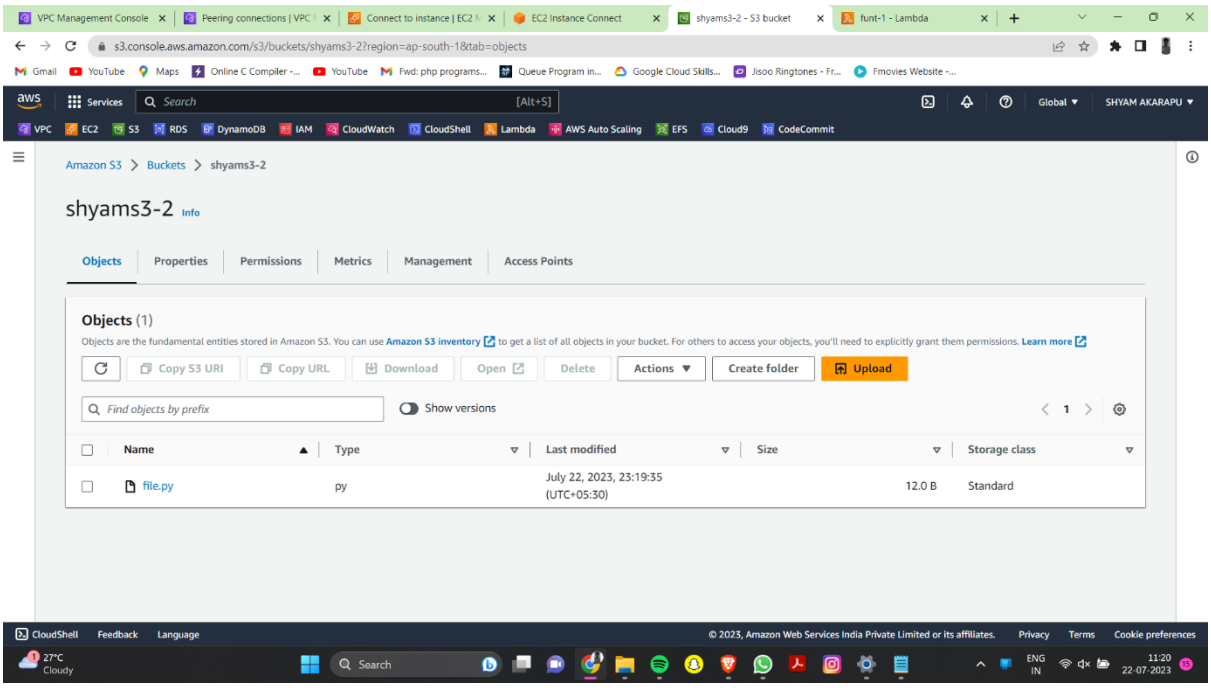


- Following commands are used below.
- Sudo su
- Aws s3 ls
- Cat
- Aws s3 cp fiel s3://bucket name

- First we have create a file .
- In that file put some data.
- And check the buckets present in s3.
- After that cp that file into bucket which we have created the cross replication .
- Check the buckets for that file .

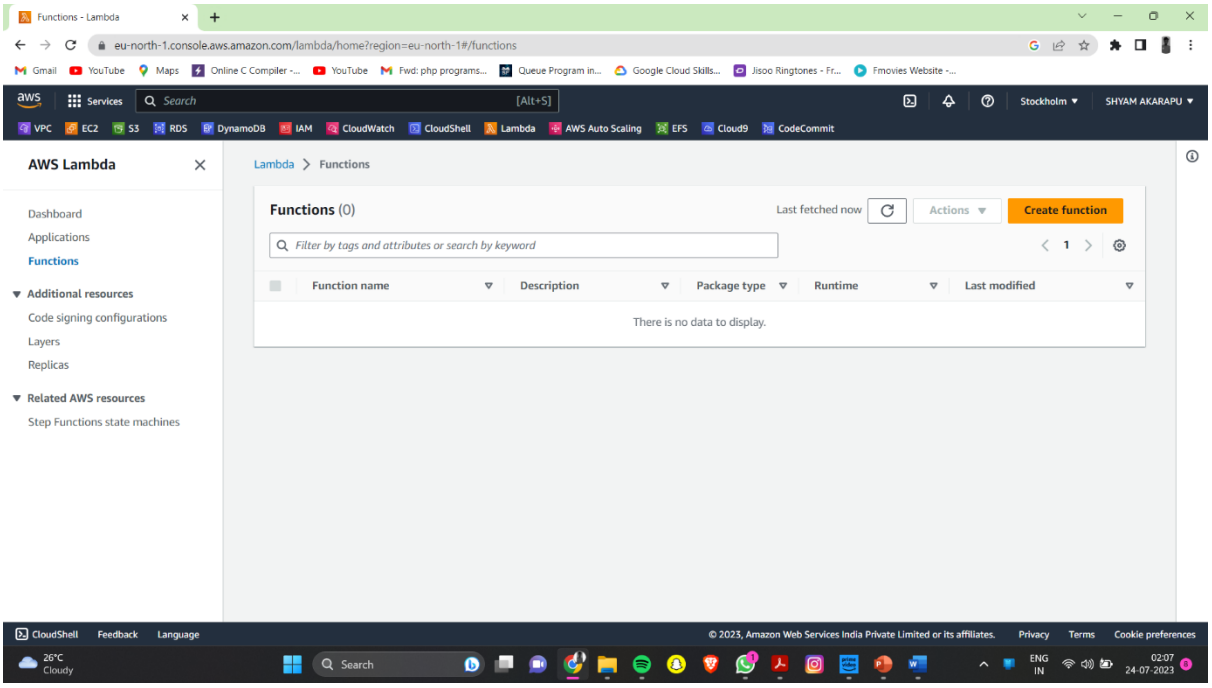


- There is a object is present in that is nothing but our copied file

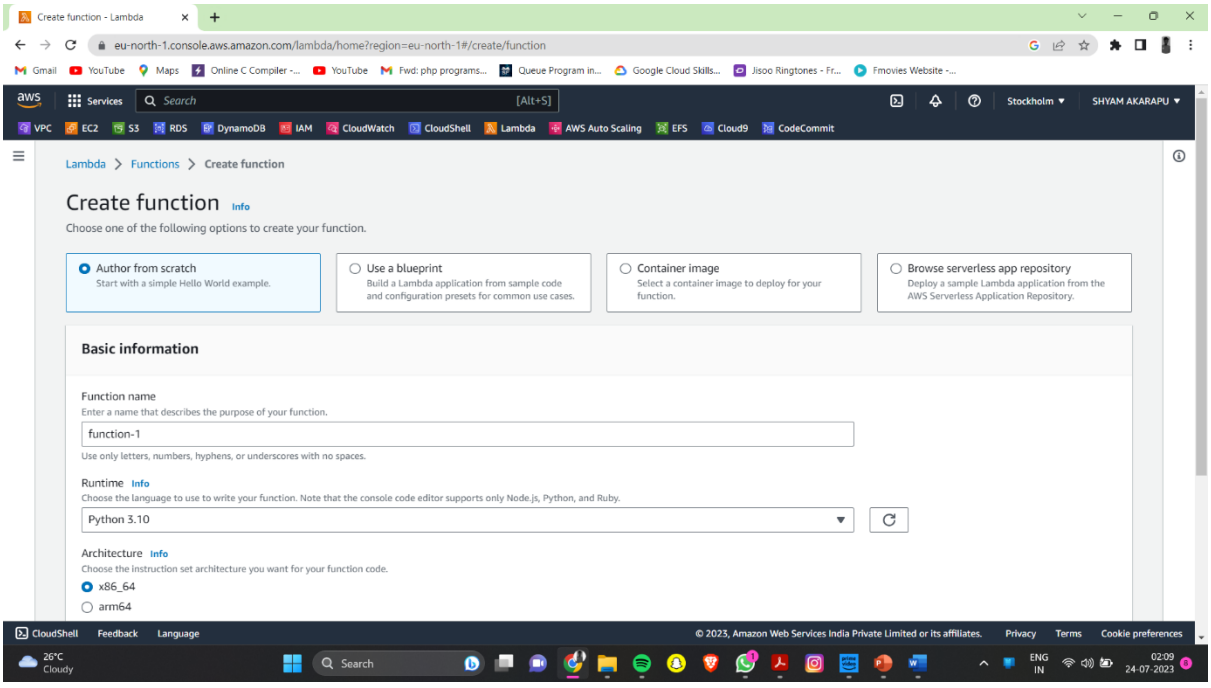


- That object is also present in the shyams3-2.

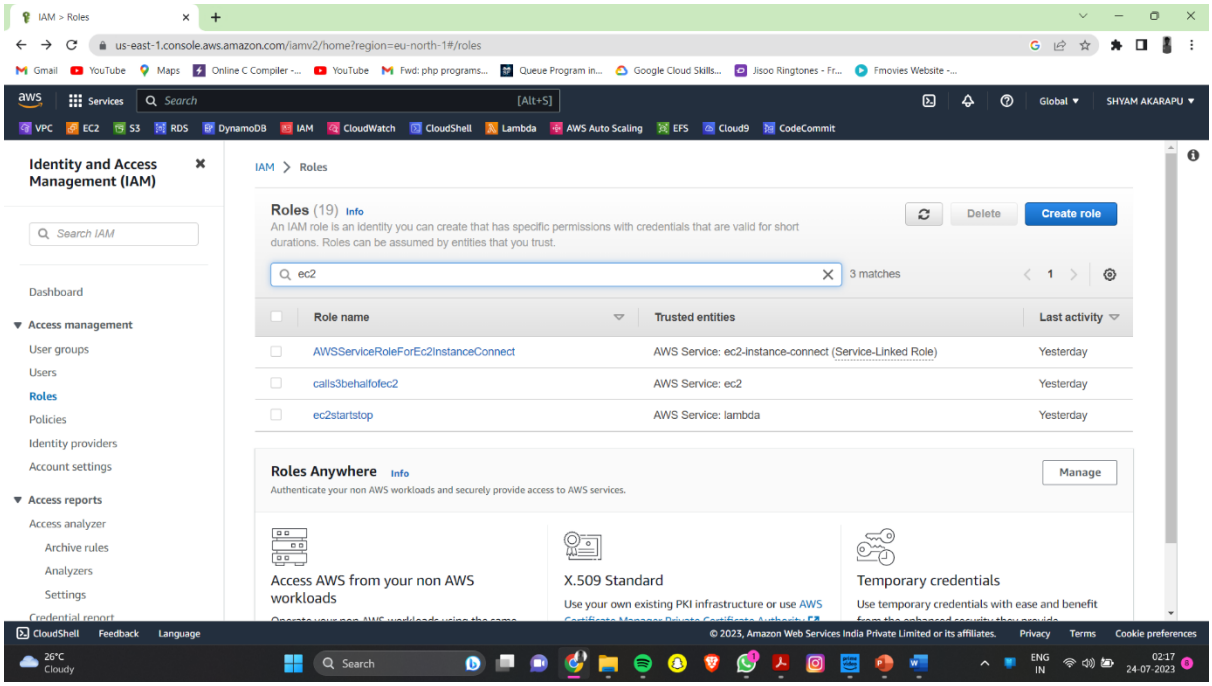
- Due to croos replication that file is copied into these bucket also.
- **LAMBDA :**
- IT'S a service present in the aws.
- First we have create a function in the lambda.



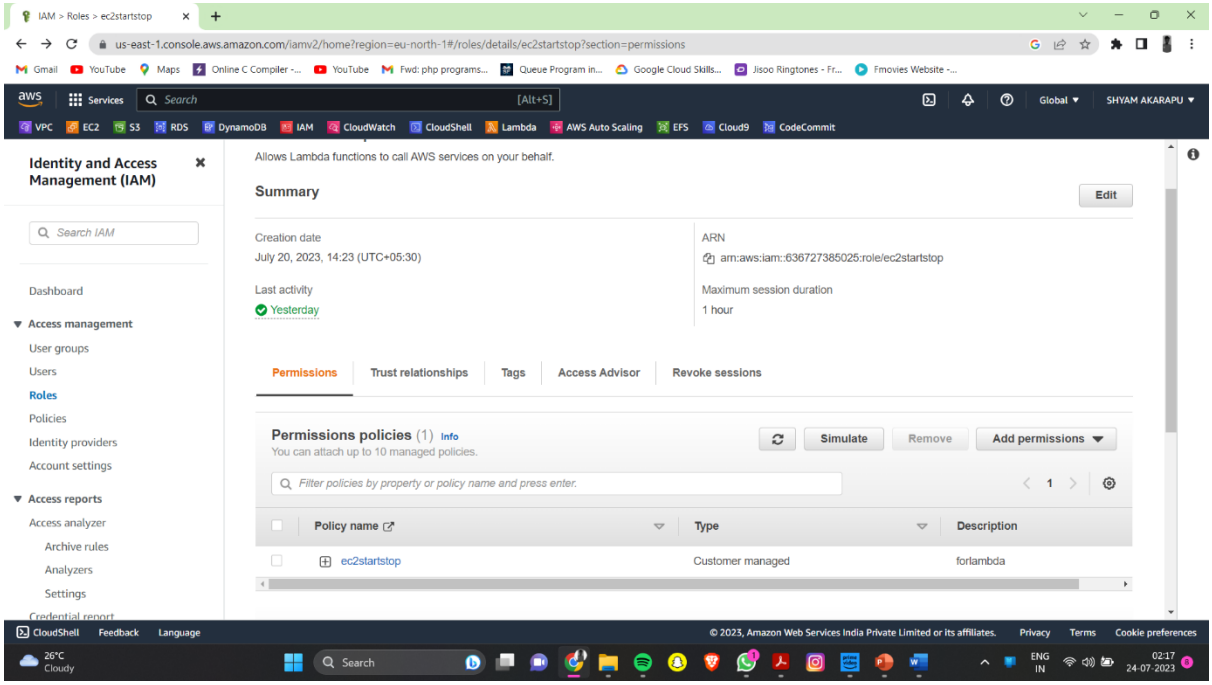
- **There** is no function is present at first



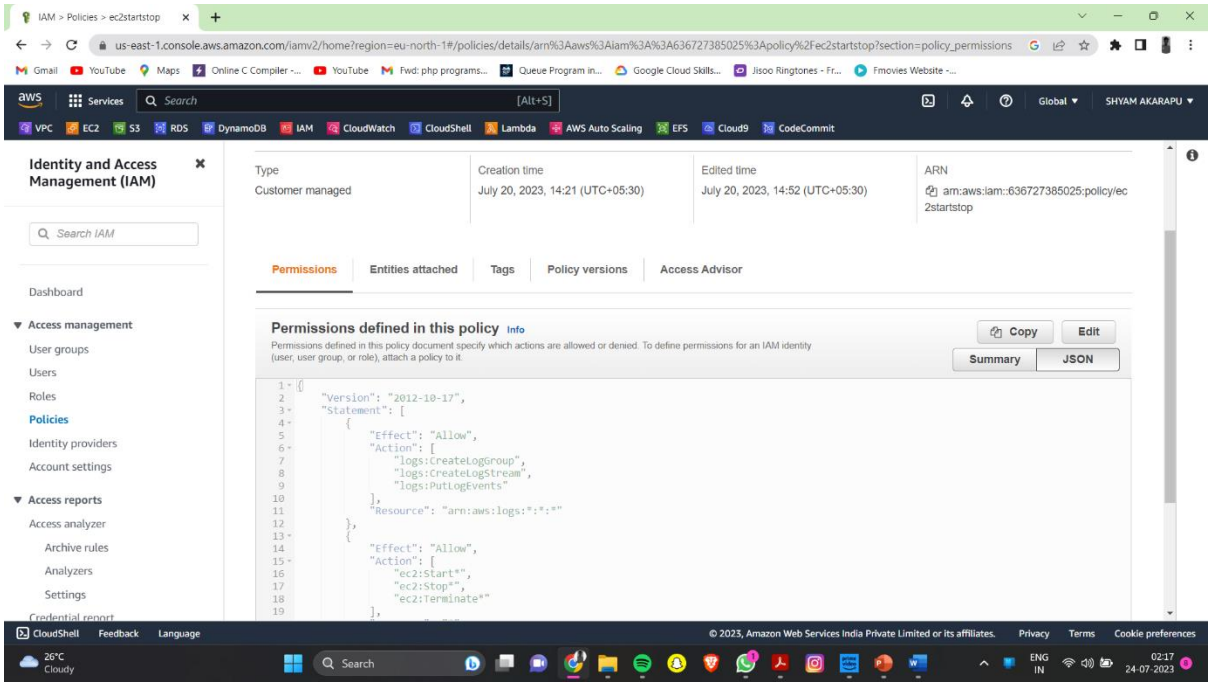
- function 1 is the name of the function a
- after that we have create iam role



- create iam role name the role with the ec2startstop



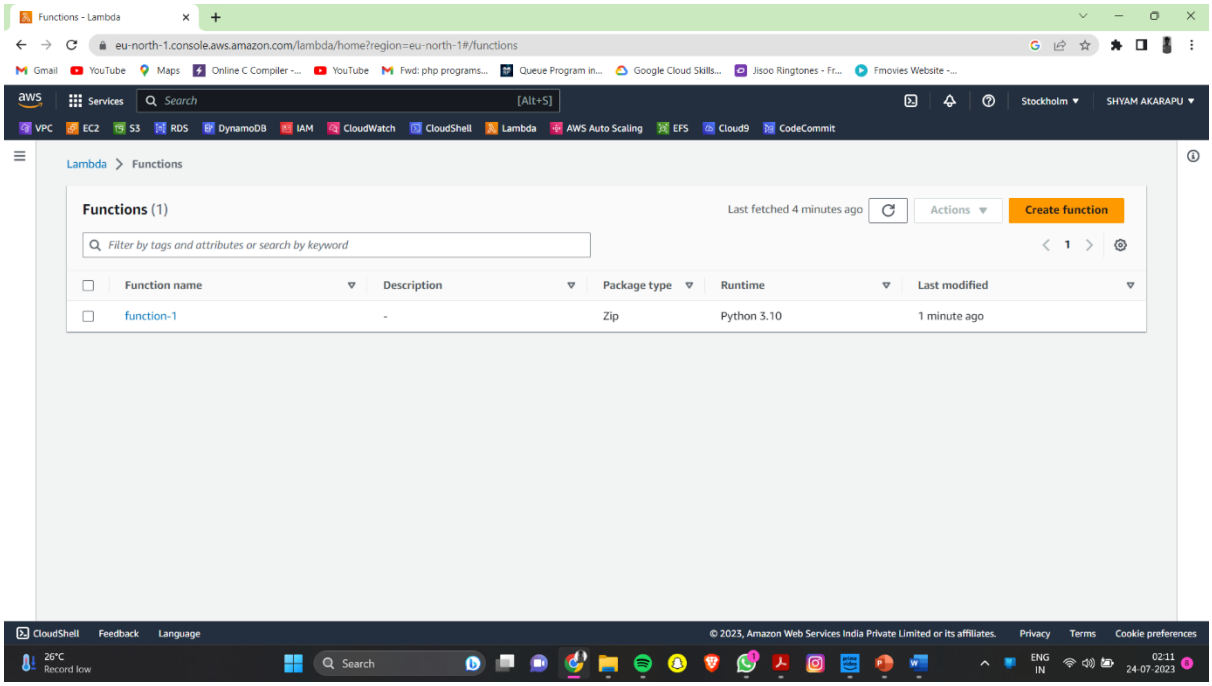
- After that create a policy
- In that policy we have give the permission to the lambda function which we are created so that it can start stop and terminate your instance with that lambda function.



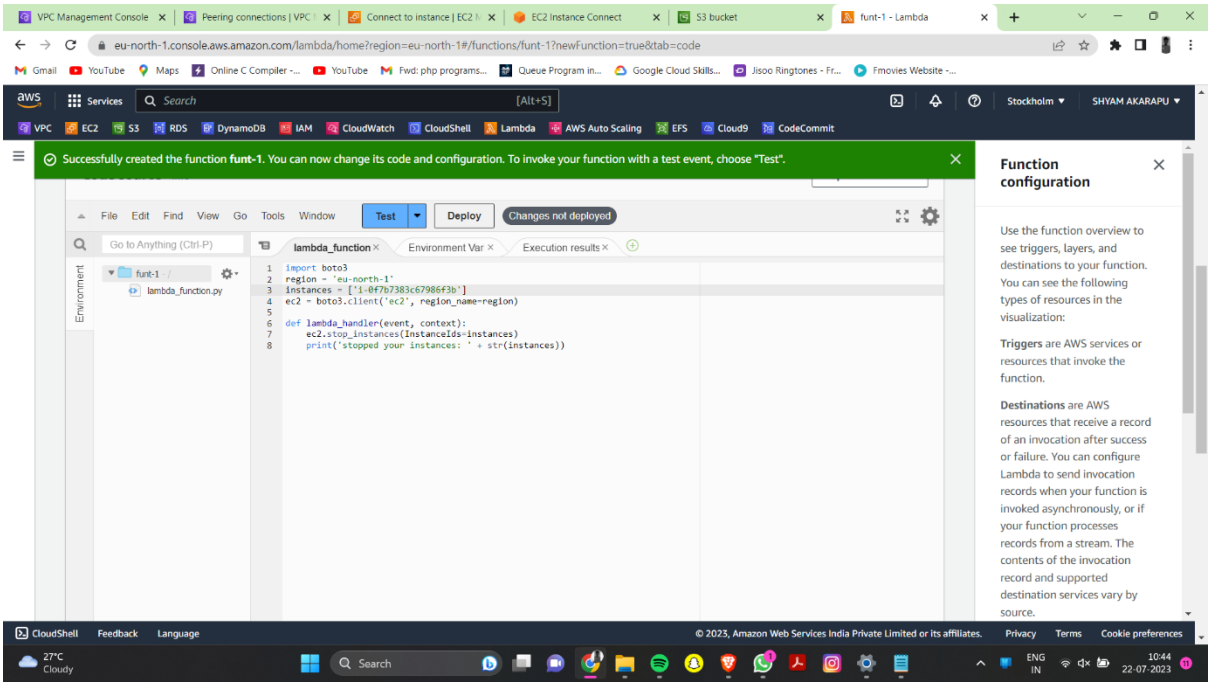
- we have write a code in the policy in the json view .

```
{
 "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": "*"
  }
  ]
}
```



- after creating a function in lambda .

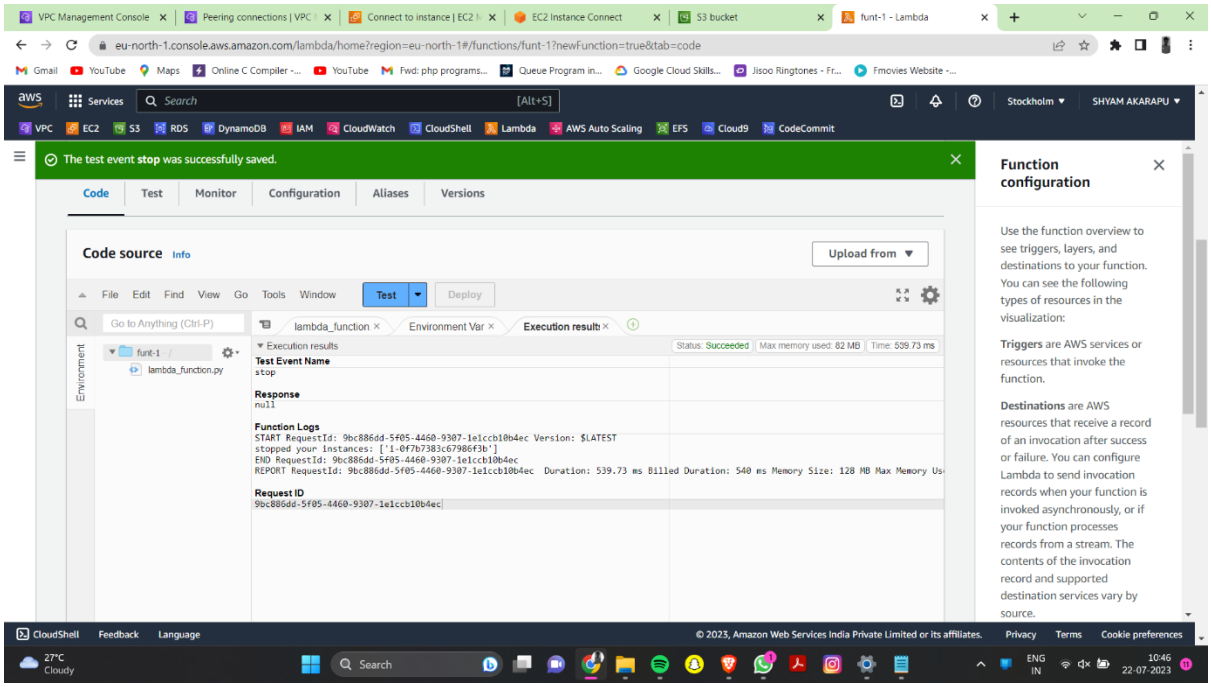


- We have to test the code
- These is the code for the stop

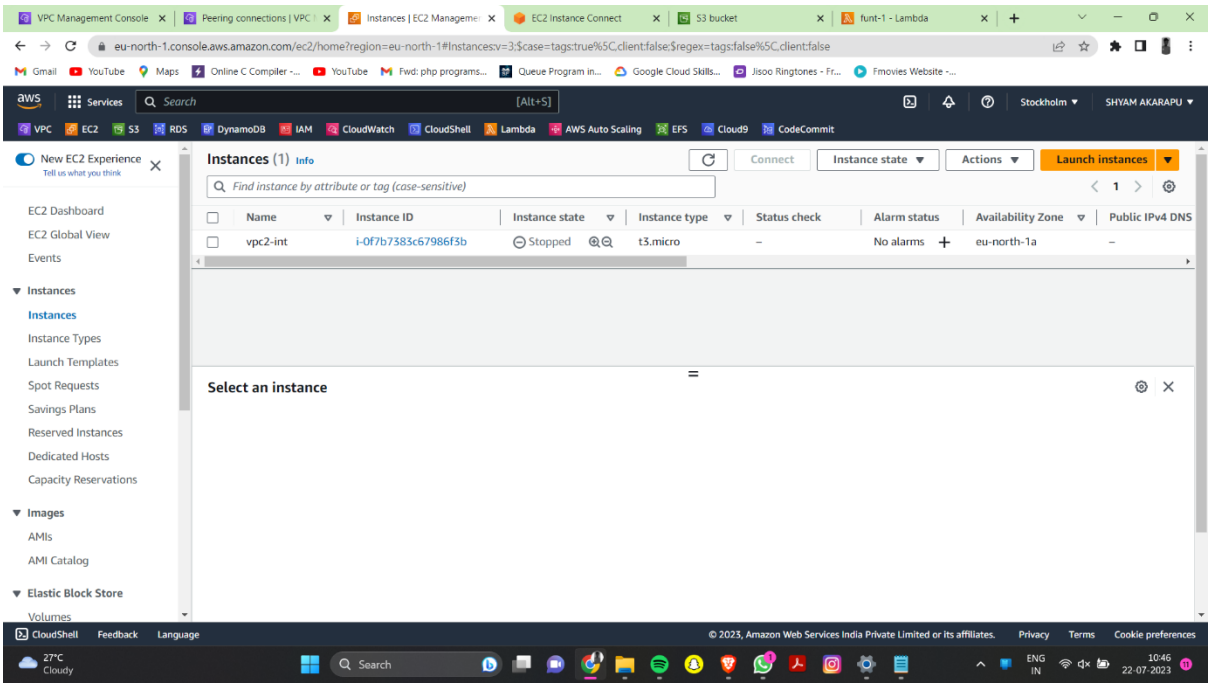
```
import boto3
region = 'ap-south-1'
instances = ['i-12345cb6de4f78g9h']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.stopt_instances(InstanceIds=instances)
    print('started your instances: ' + str(instances))
```

- These is used to stop the instances.

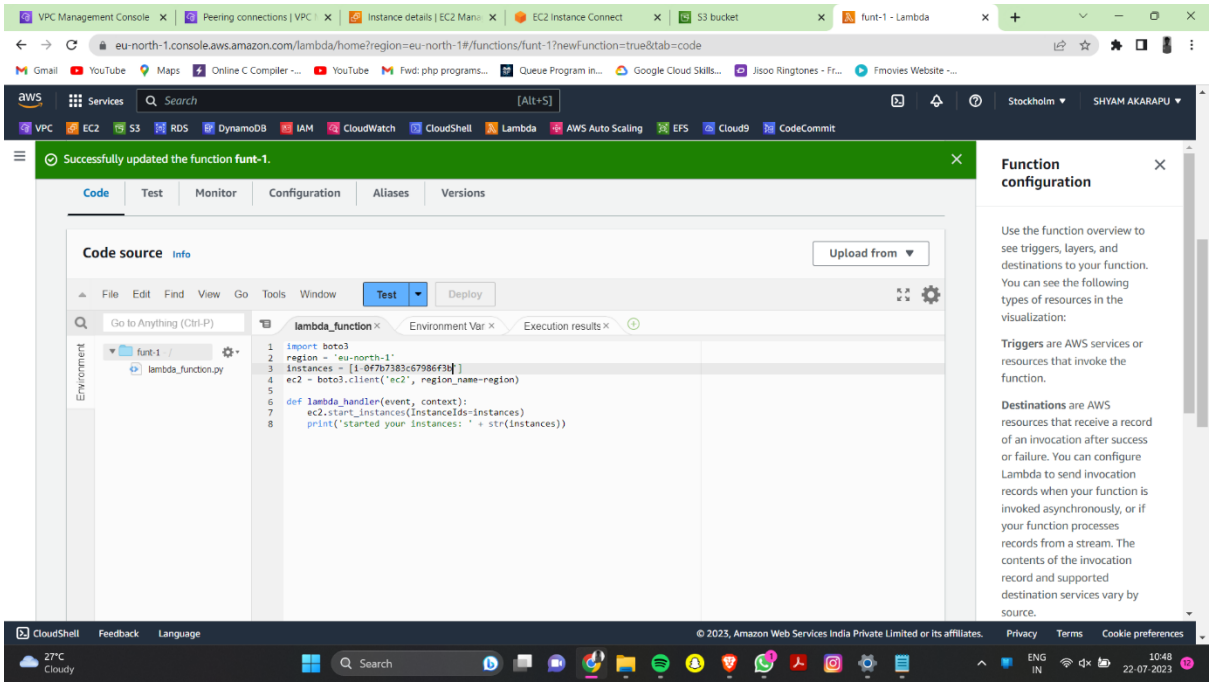


- We have to deploy the code after that we have to test the code.
- If the response is null then only the instances is stop.

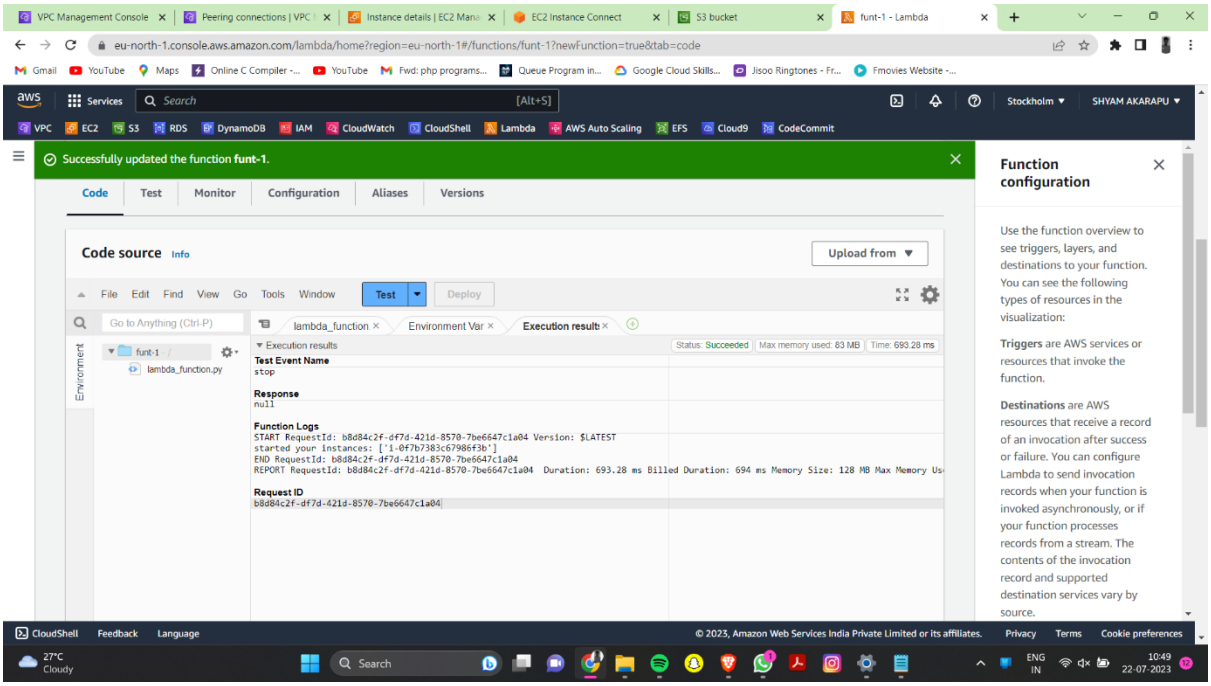


- In the case the instance is stop

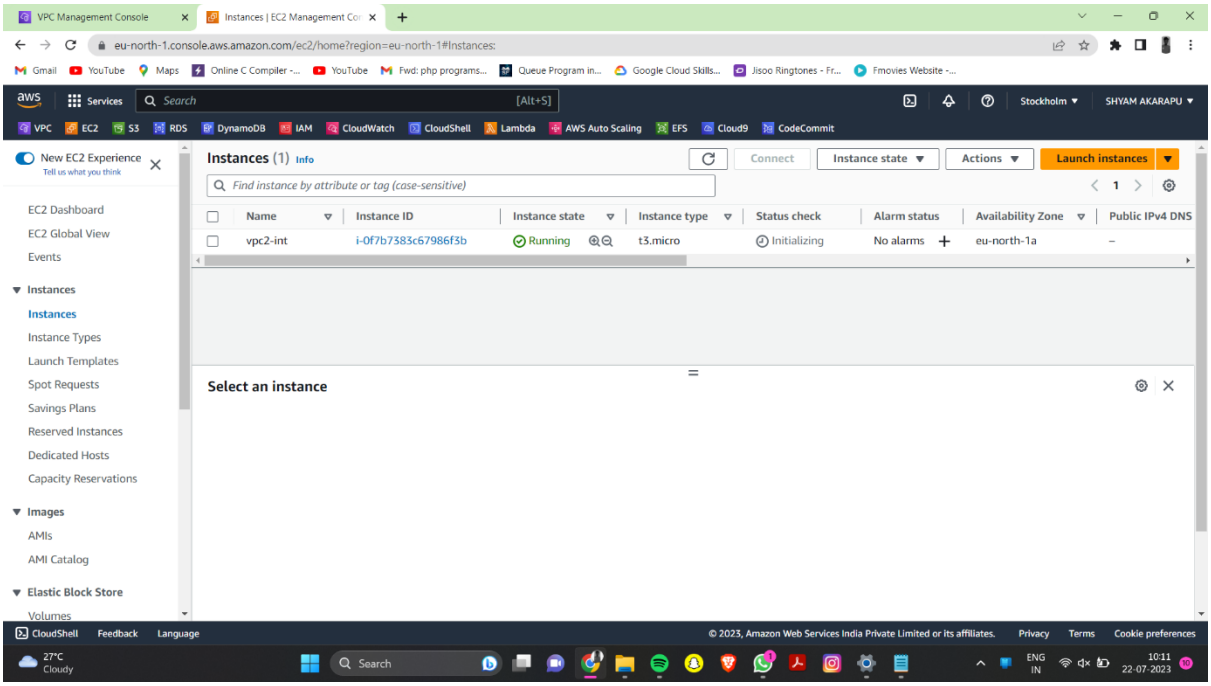
- After that we have have to start the instance so that also we have to change the code for the start the instances.



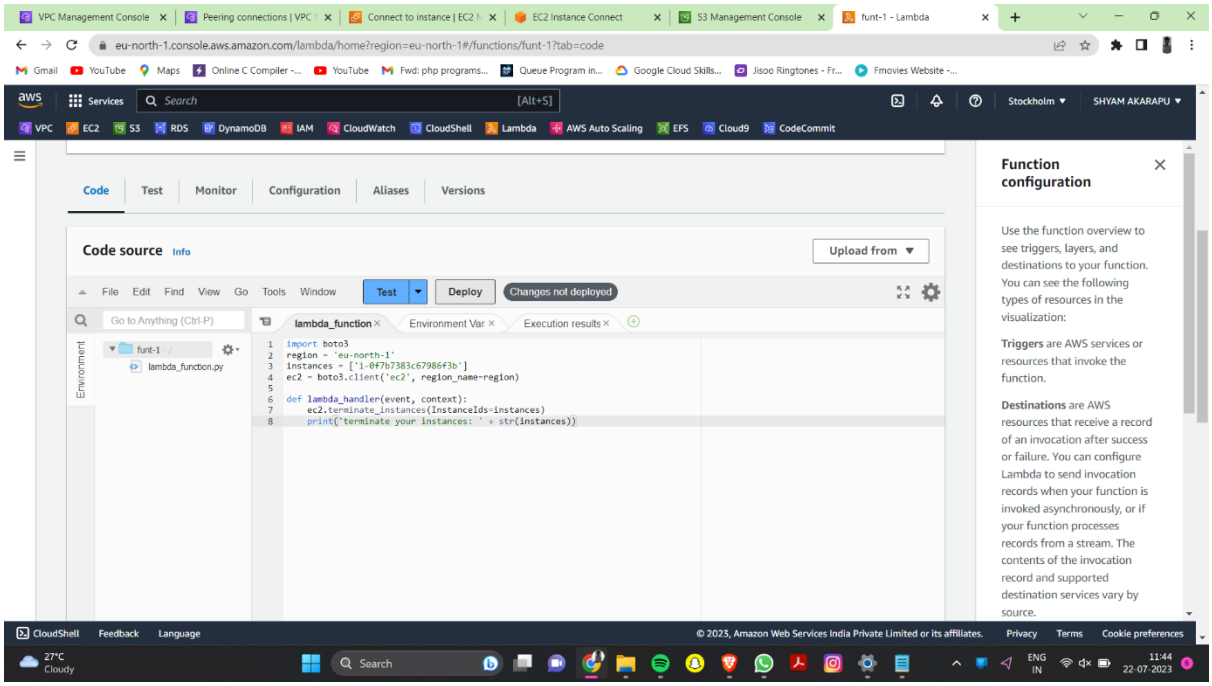
- The code we have written in the python .
- At the first while creating the func we select the python language to run these the following codes
- When the change the code in the function we have deloy first
- Then we have have test the code
- If code is good then then there is no error it shows response null and status of the code is running.
- these the code for the start the instances.
  - import boto3
  - region = 'ap-south-1'
  - instances = ['i-12345cb6de4f78g9h']
  - ec2 = boto3.client('ec2', region_name=region)
  -
  - def lambda_handler(event, context):
  - ec2.start_instances(InstanceIds=instances)
  - print('started your instances: ' + str(instances))



- in these response are null
- And the status of the code is success.



- In above the case the instances is stoppes
- Now the instance is running.

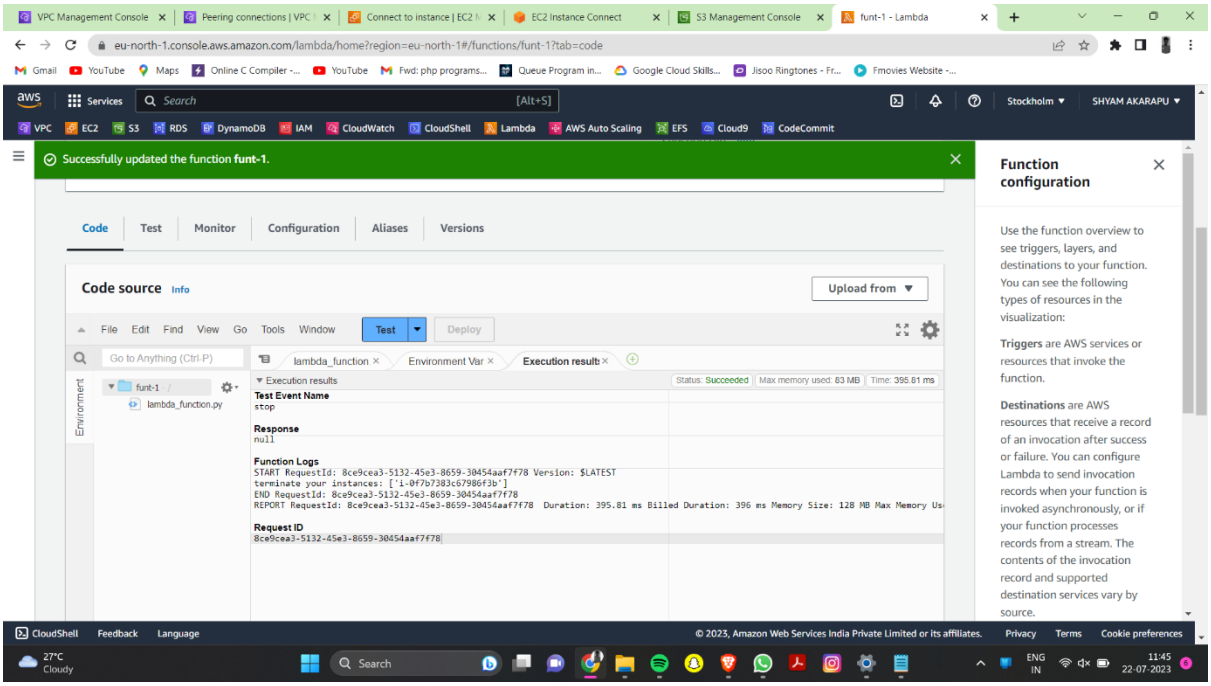


- these the following code for the termination of the instances

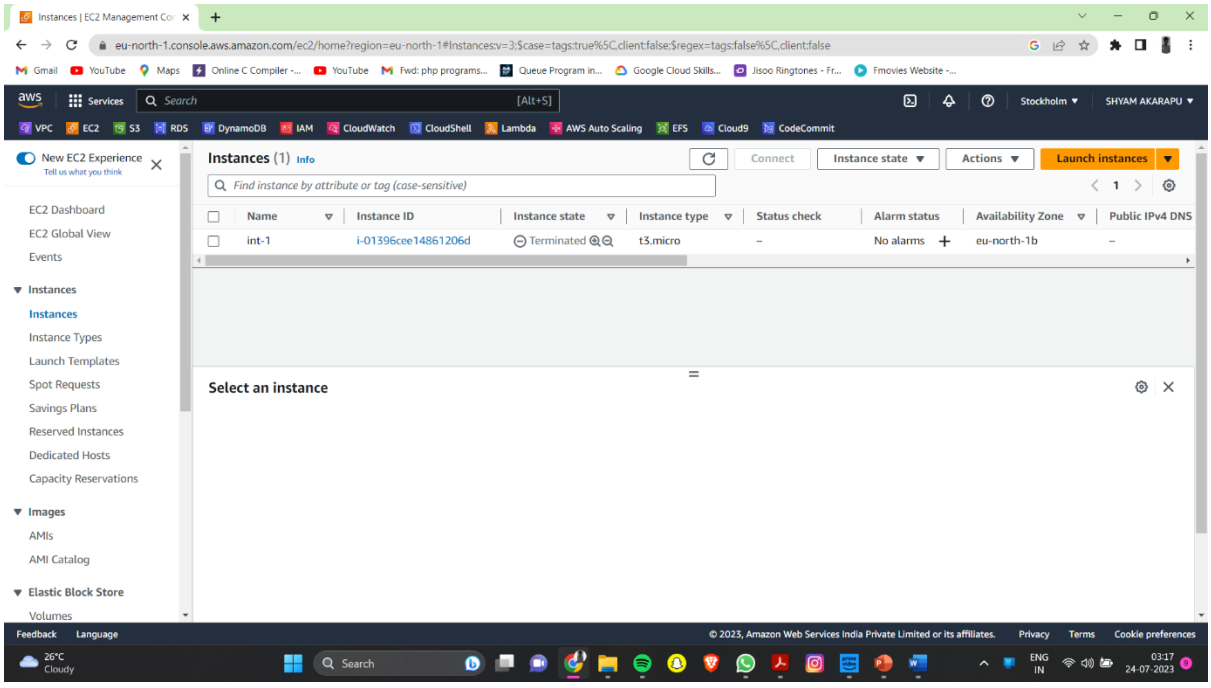
```
import boto3
region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.terminate_instances(InstanceIds=instances)
    print('terminate your instances: ' + str(instances))
```

- These code is present in the python
- It used to teriminate the instance
- The varaiable type is used is list
- So that at time no of instances can be terminated
- Exactly one or more instance can be terminated by these code



- after test the code there is no errors and the responses are null
- And the status of the code success



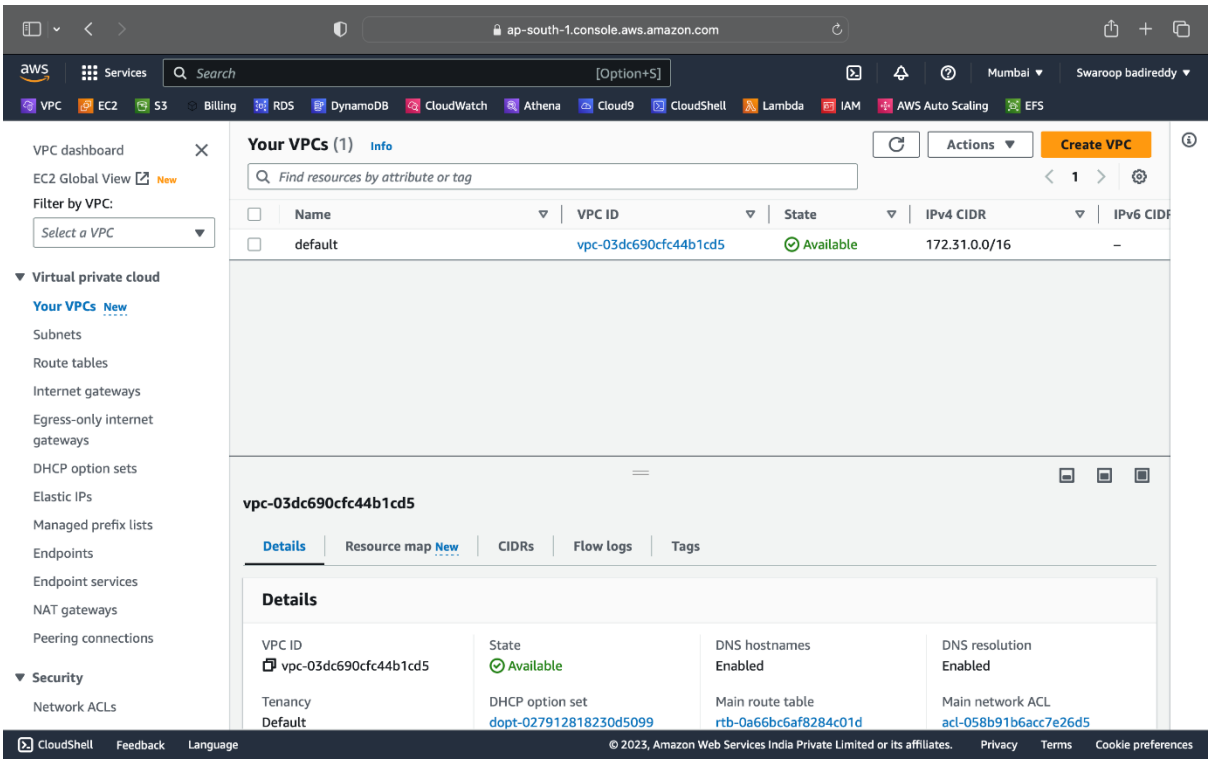
- The instance get terminated by the ternmination code in function by lambda.

• Now the remaining architect is continued in another account….

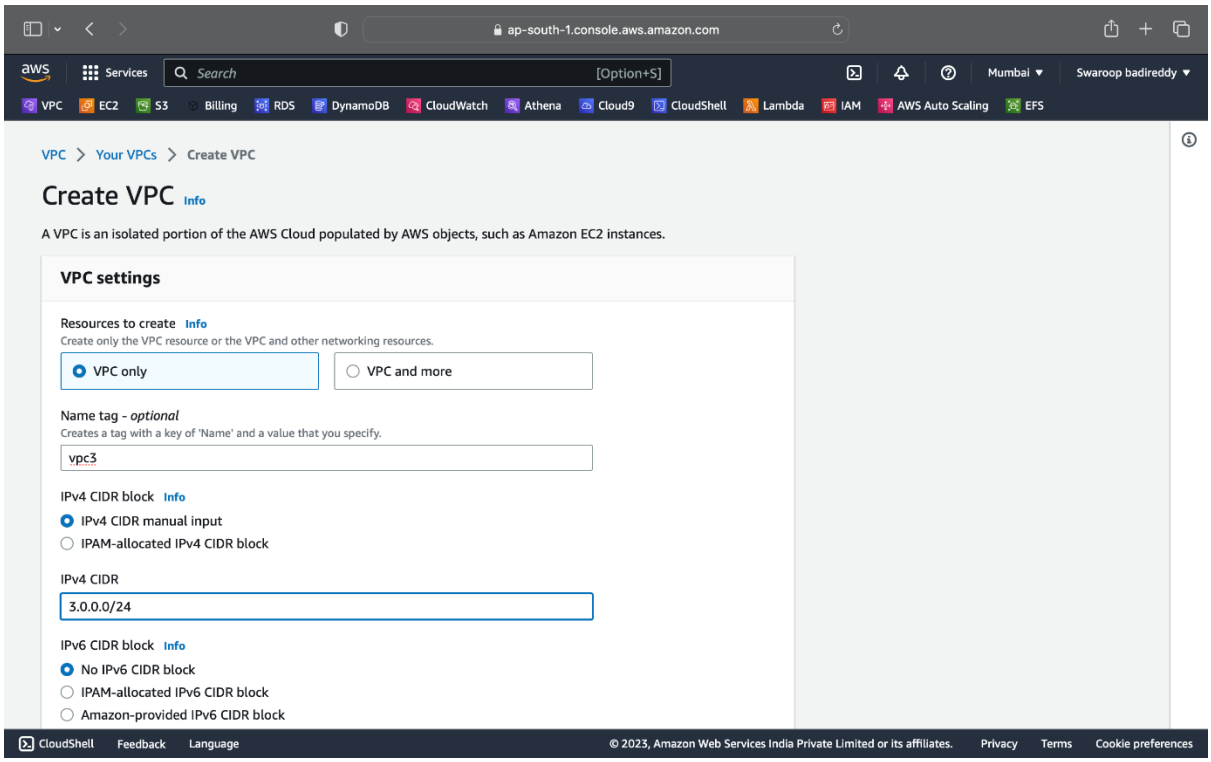
In ap-south-1 Acc id: 4839-6869-6965:-

CUSTOM VPC:

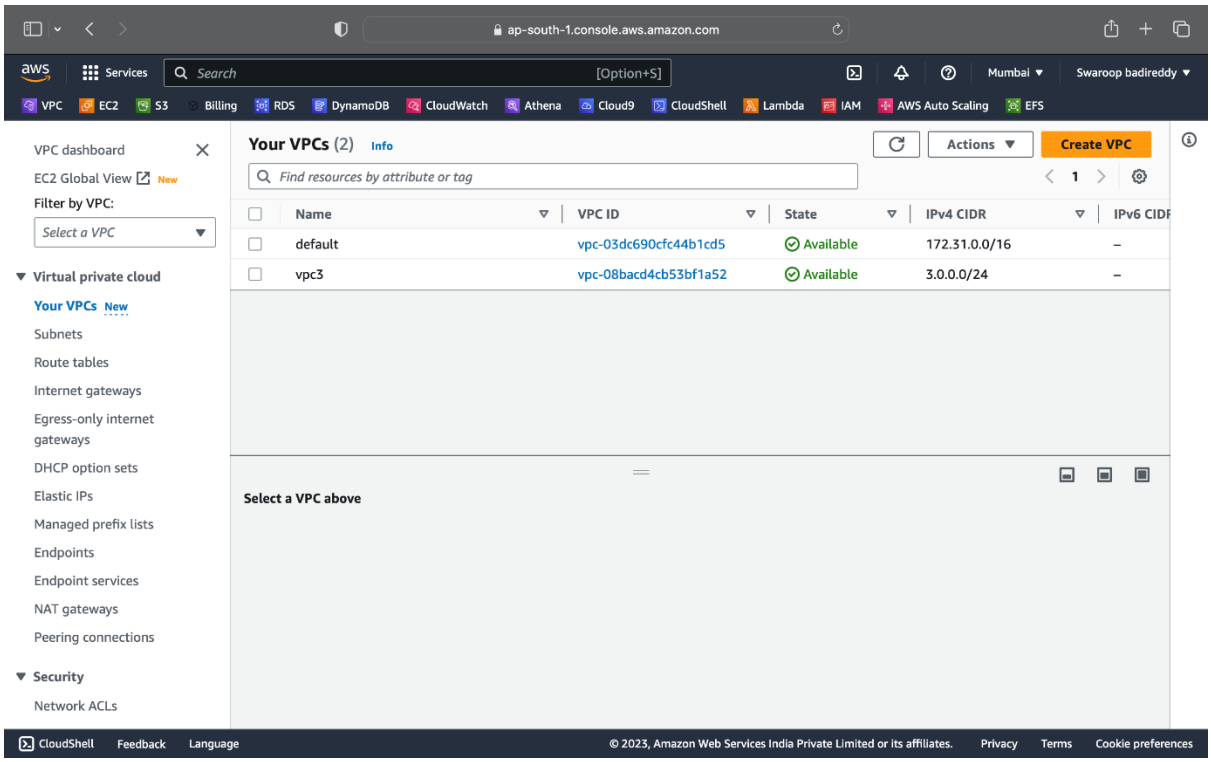
- First Create a Custom VPC in the region Mumbai.
- To create a VPC with the name "VPC3," follow these steps:
- Sign into the AWS Management Console.
- Open the Virtual Private Cloud Service.
- Click on the "Create VPC" option.
- Give the VPC name as vpc3 and CIDR value as 3.0.0.0/24.
- Click on create VPC.
- Before creation of vpc,there is a default vpc present.
- After the vpc is created it shows 2 vpc’s
    - First one is default.
    - second one is vpc3.



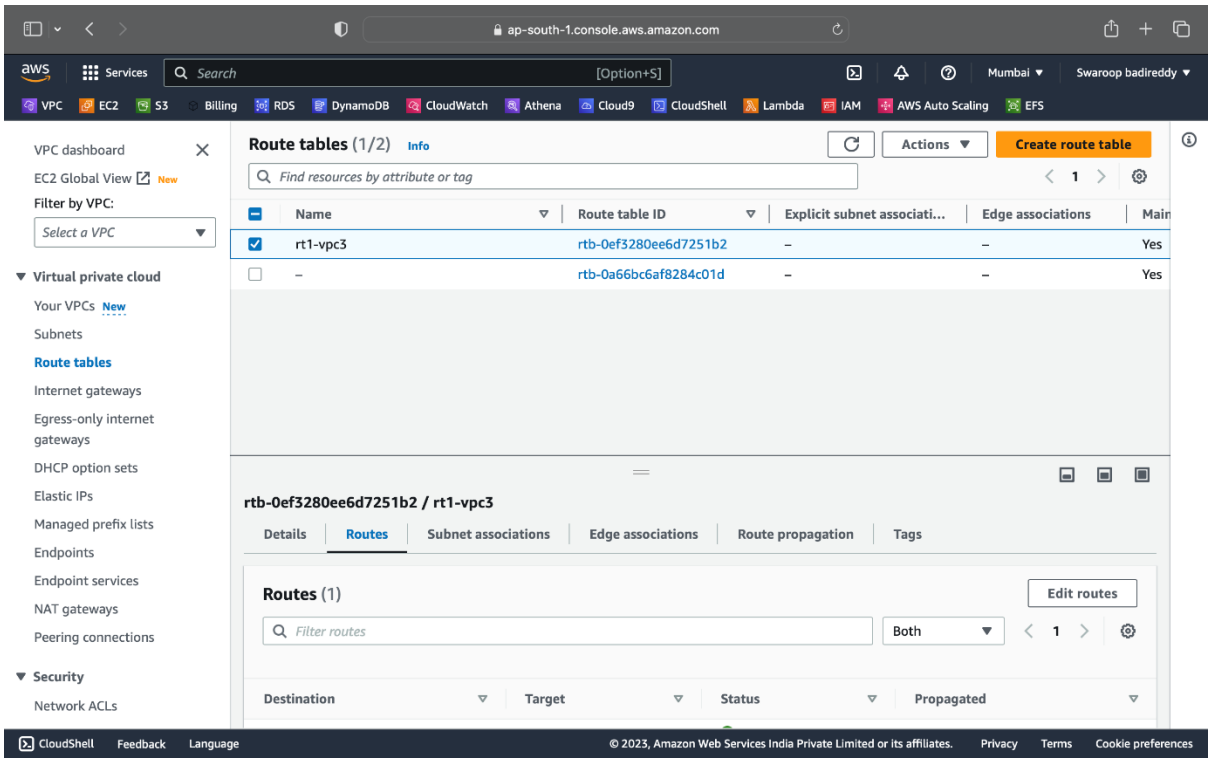
Creating vpc:



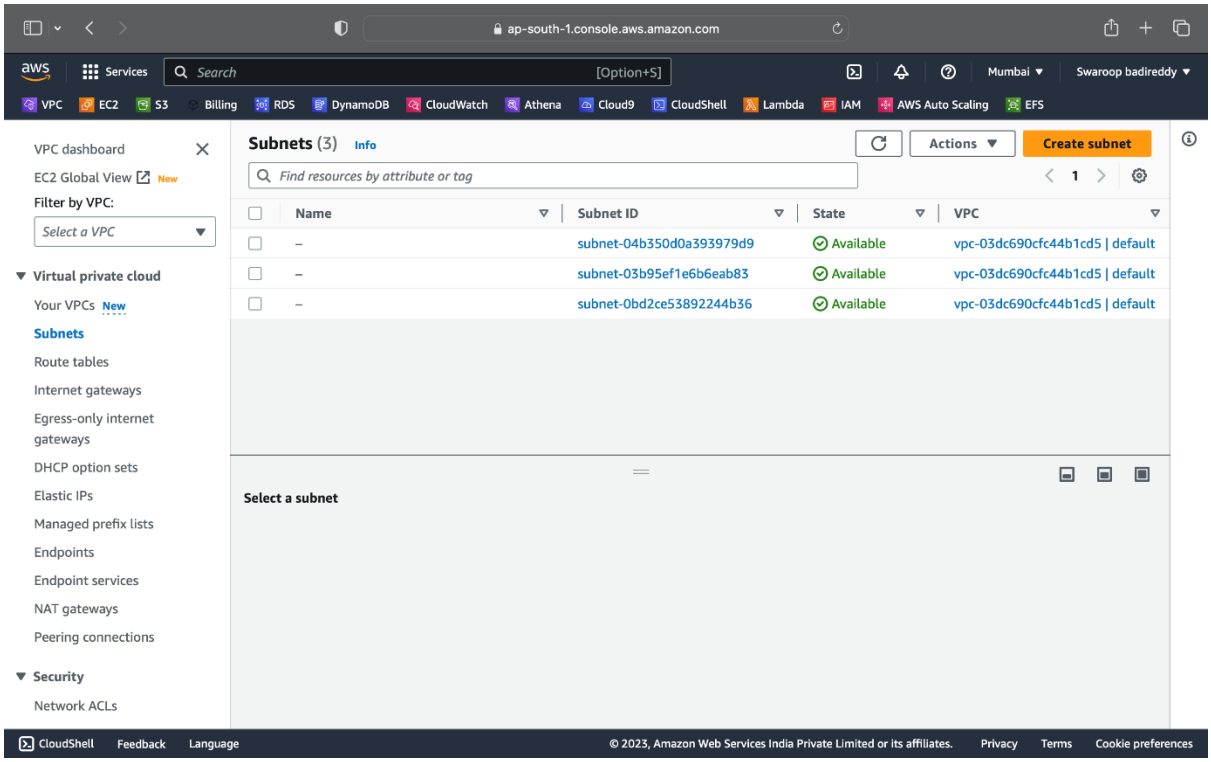
After creation:



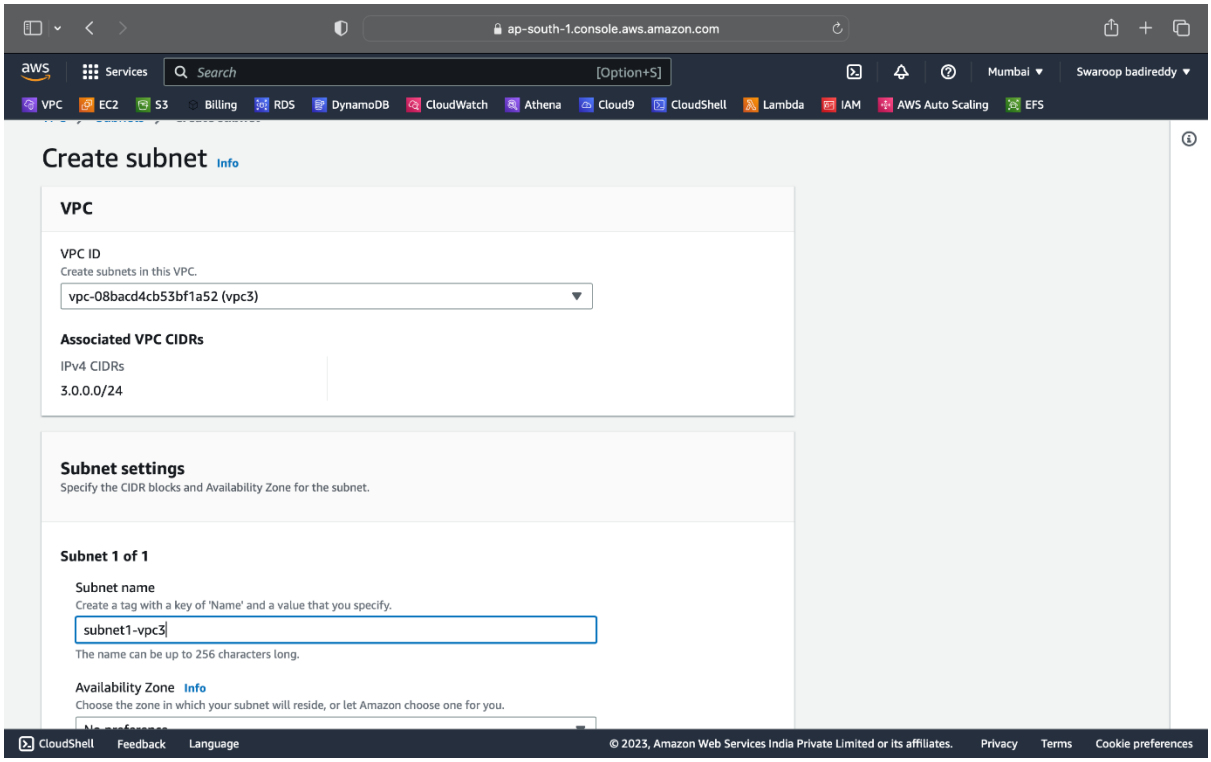
- Go to route table and edit the name of route table as rt1-vpc3 that is created along with vpc3.



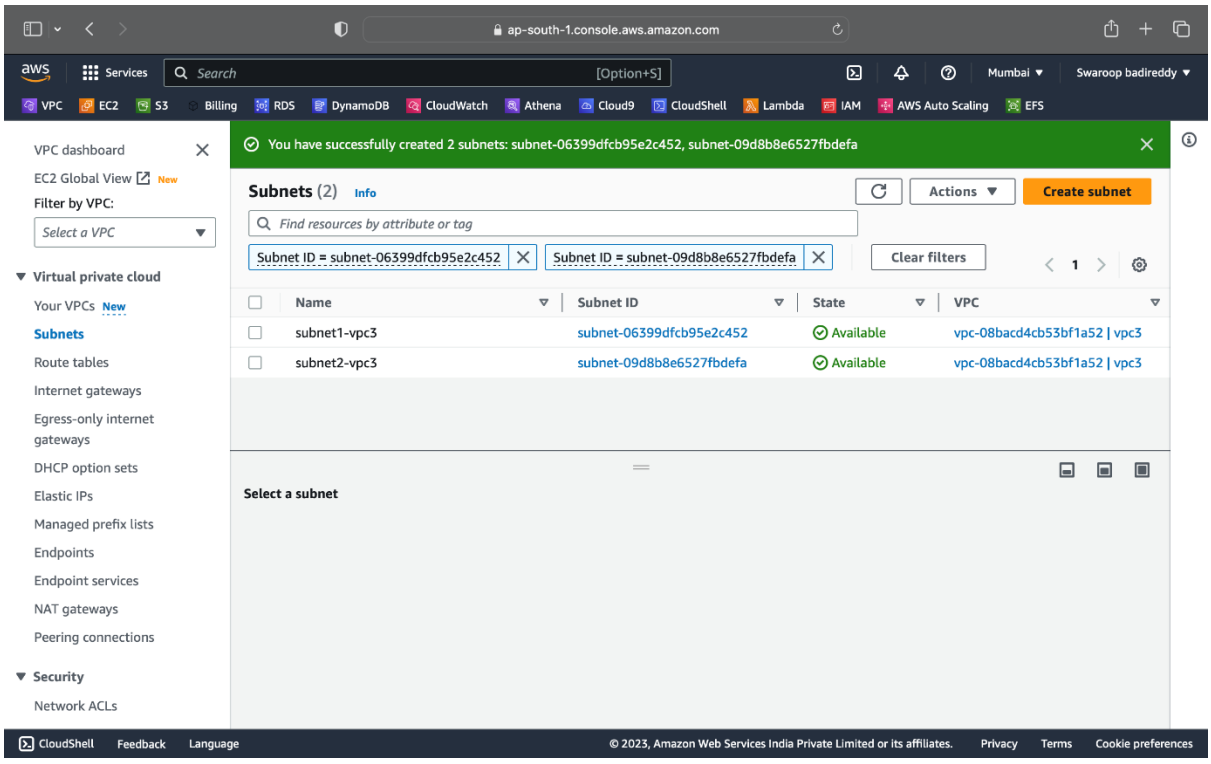
- After editing the route table name then go to subnets.
- In subnets,first there will be three default subnets present.



- Then we will create 2 subnets ap-south-1a and ap-south-1b with the names subnet1-vpc3 and subnet2-vpc3.

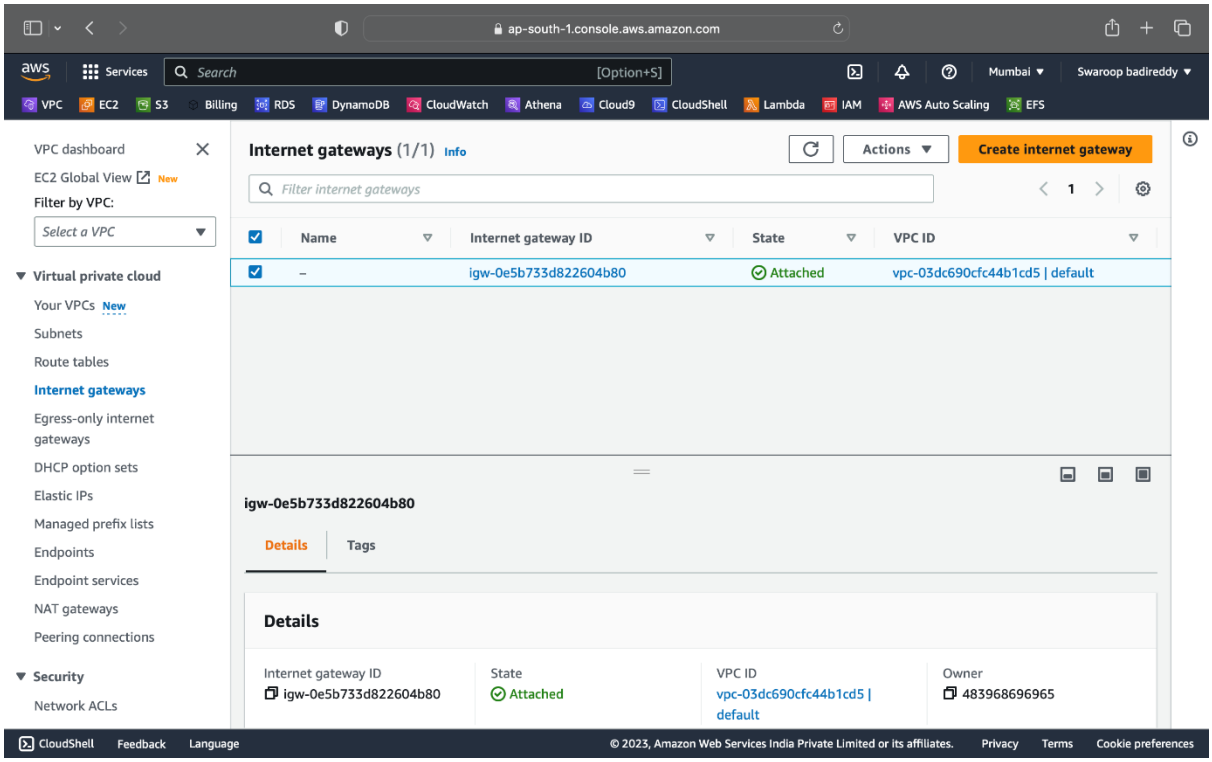


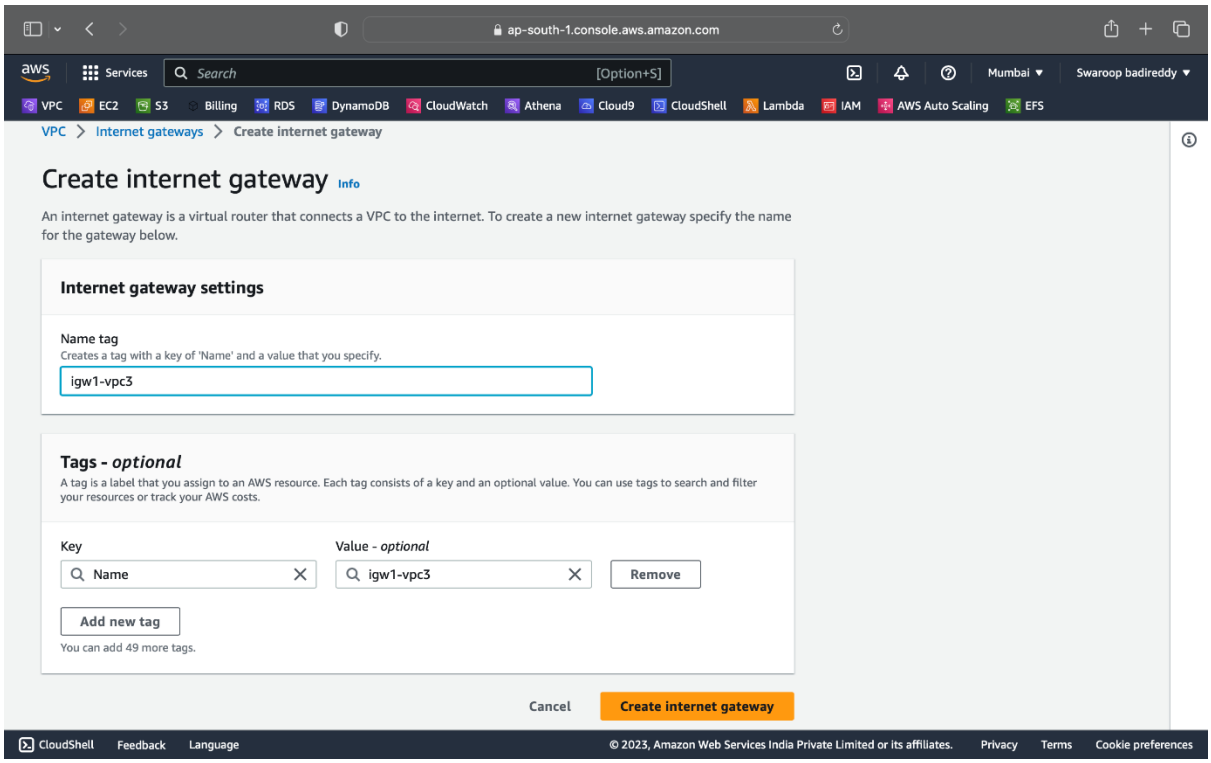
- After creation there will be 2 custom subnets.



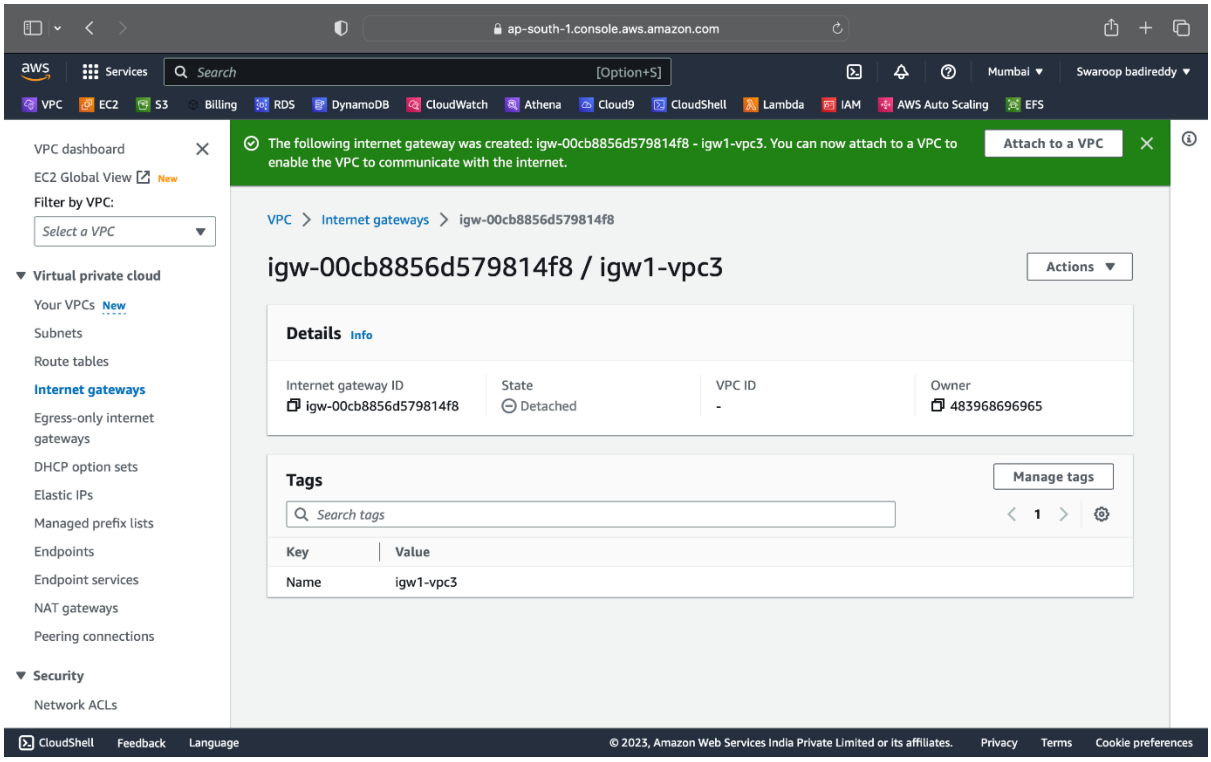
Creation of internet gate way:

- First there is a default igw.
- Then we create an internet gate way which is used to ping google.com and ip's.
- Then an internet gate way igw1-vpc3 is created.

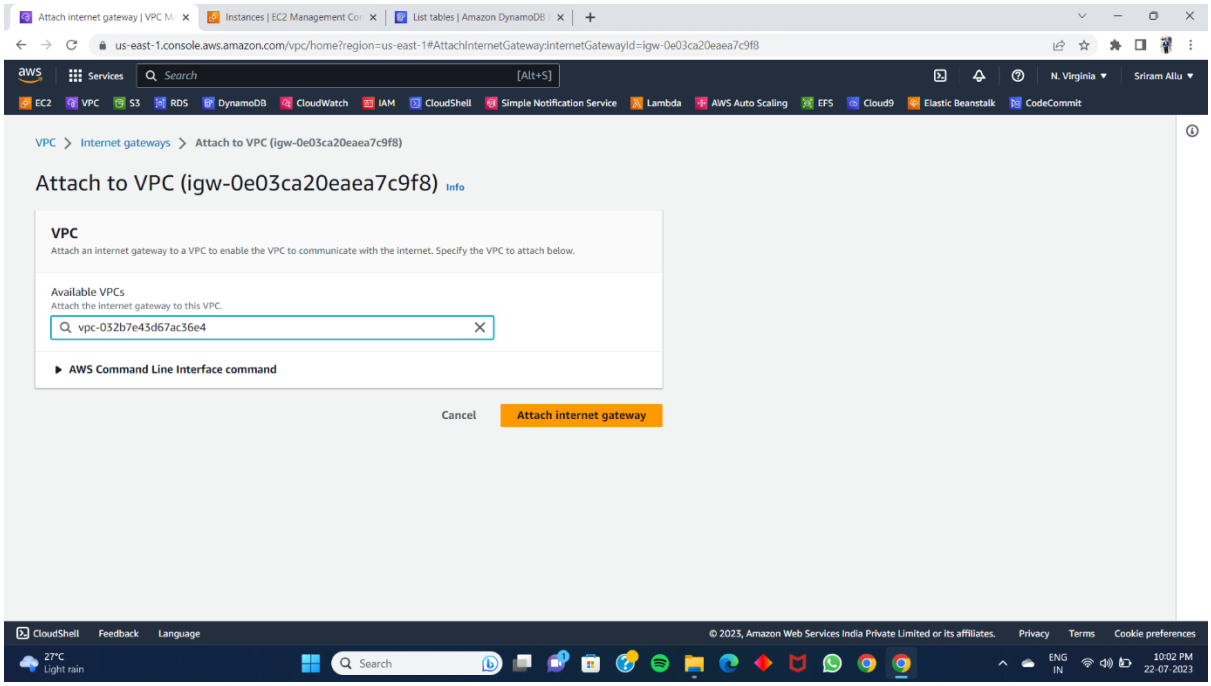




After igw1-vpc3 creation:

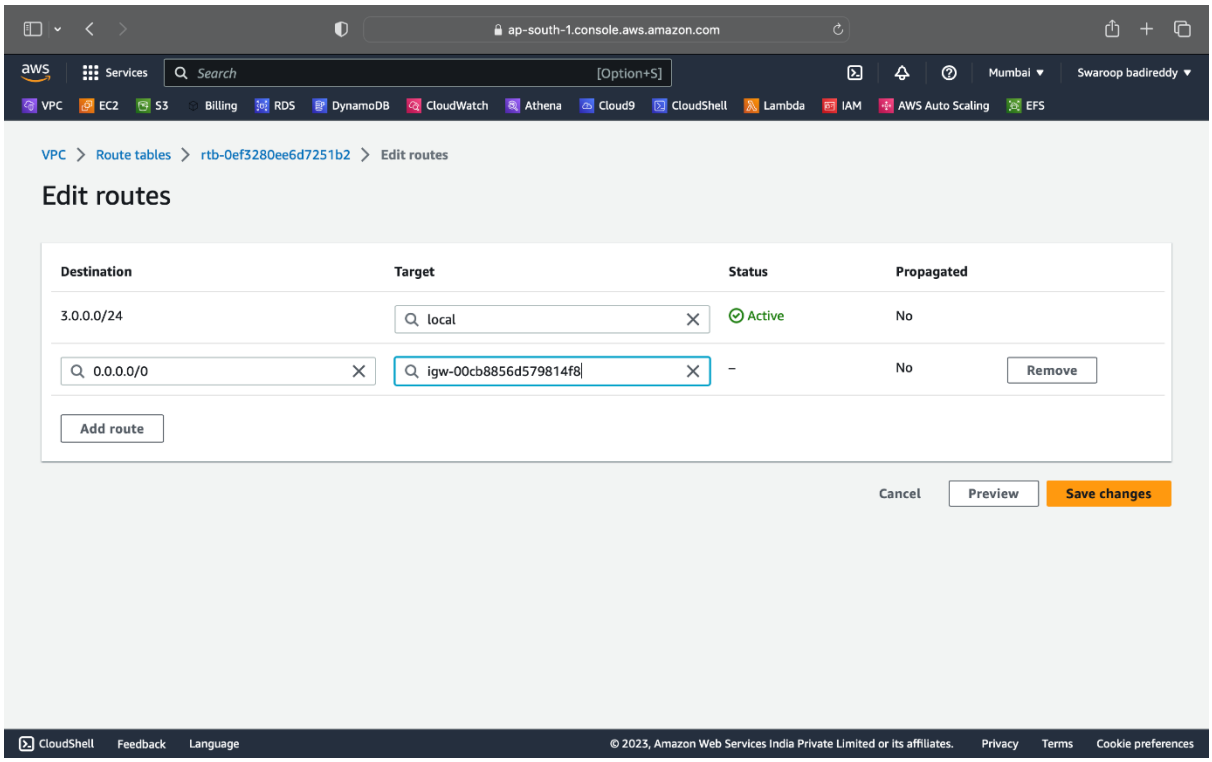


• Attach custom IGW to custom VPC



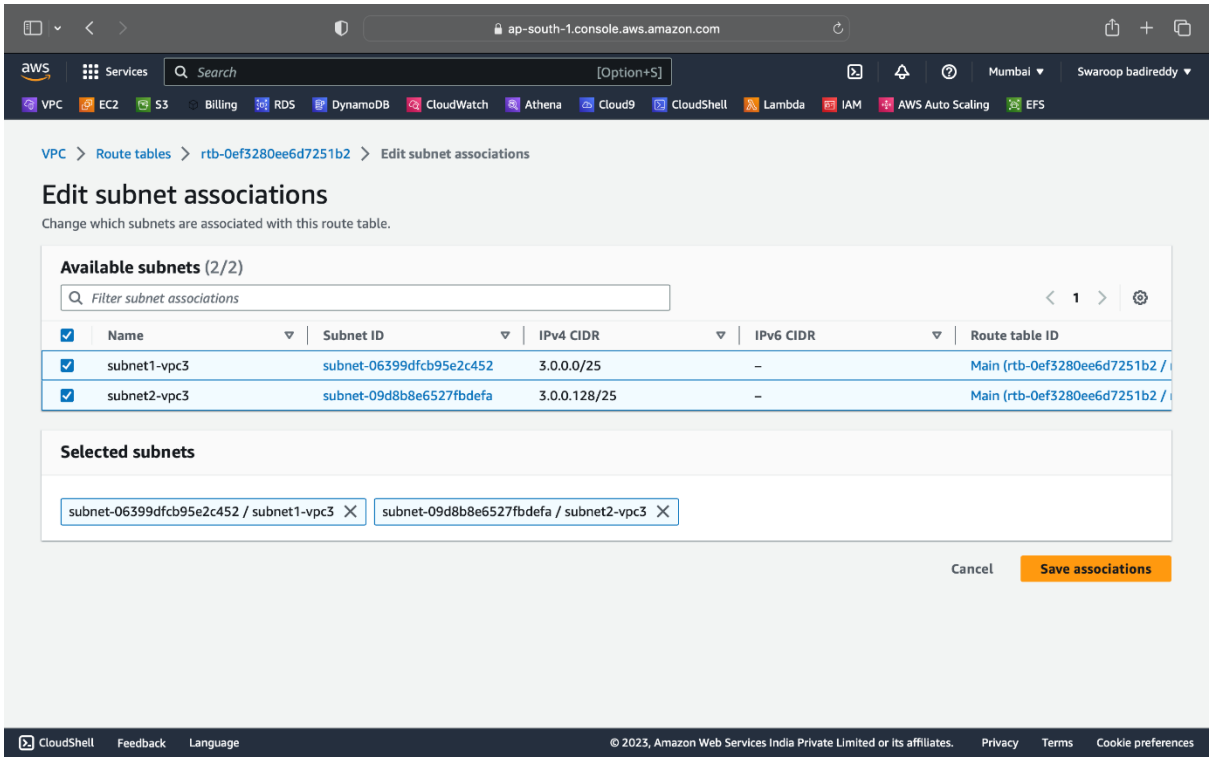
Editing routes:

- After attaching igw to custom vpc then go to route tables.
- Click on rt1-vpc1 and click on routes and edit routes.
- Attach internet gateway to 0.0.0.0/24.



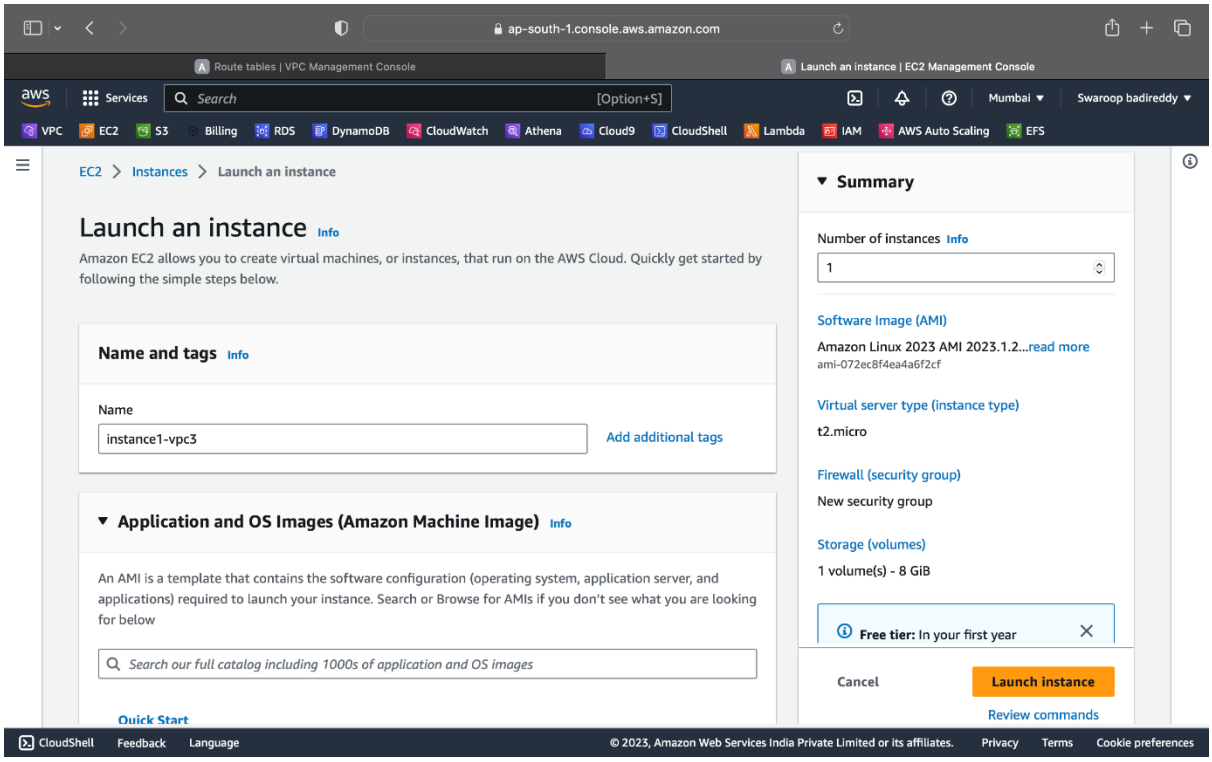
Editing subnet associations:

- After editing routes then go to route tables again.
- Click on rt1-vpc1 and then click on subnet associations.
- Then edit the subnet associations.

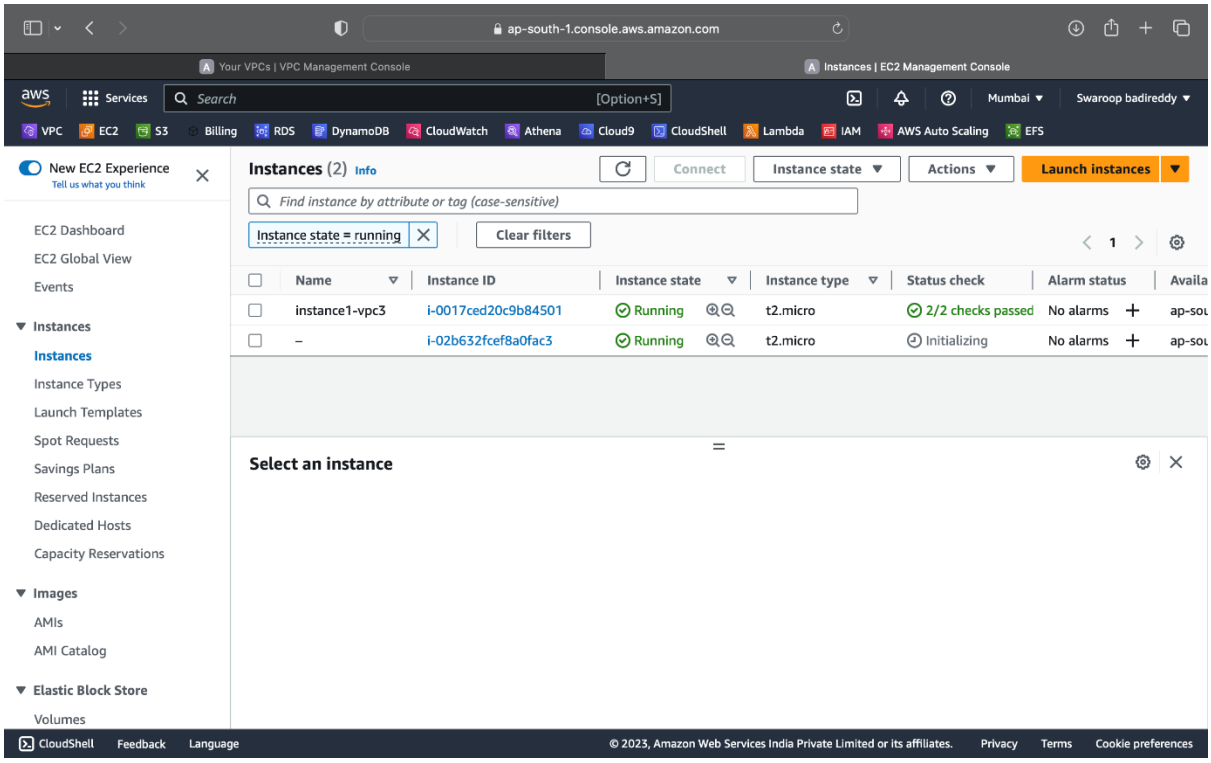


Creating an ec2 instance:

- Go to e2 and select launch instance.
- Then give the name instance1-vpc3 and select the ami version,key pair,vpc3,subnet-1 and security group.
- Then click on launch instance.

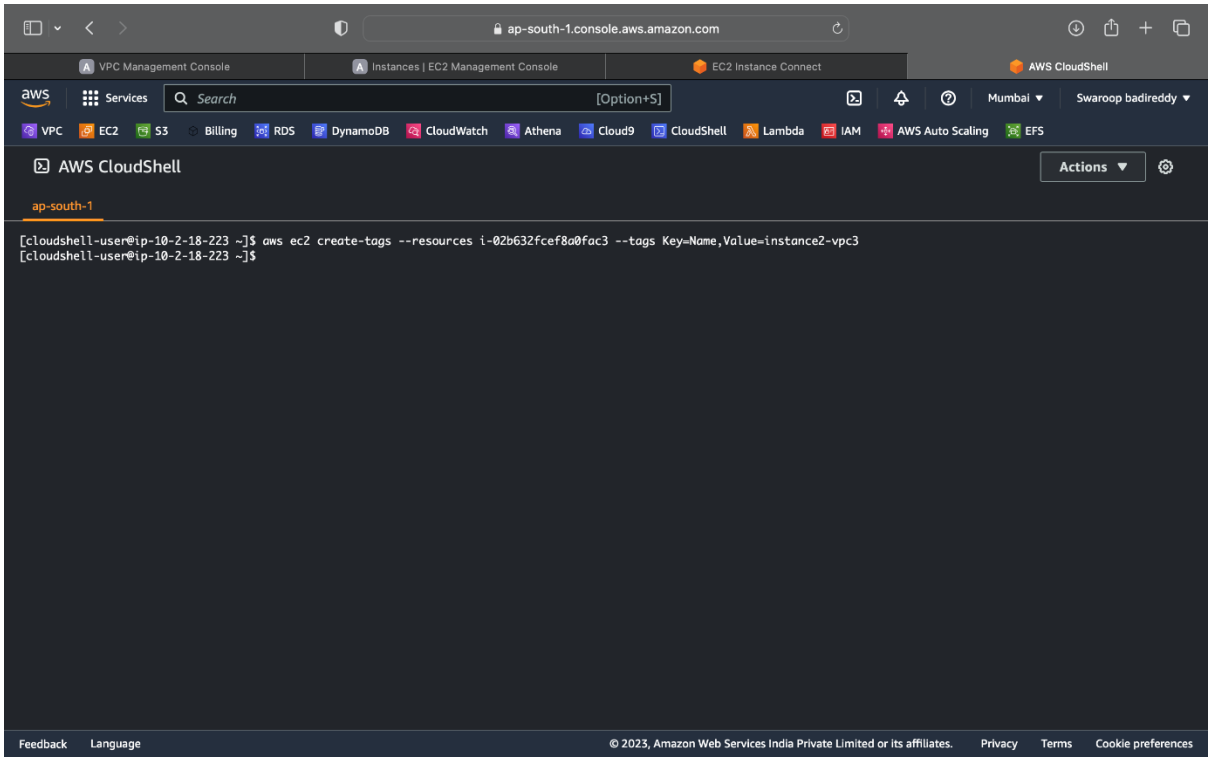


- Similarly,we create another instance in subnet-2 without any name for which we assign a name or tag using cloudshell.
- So,we have 2 running instances.

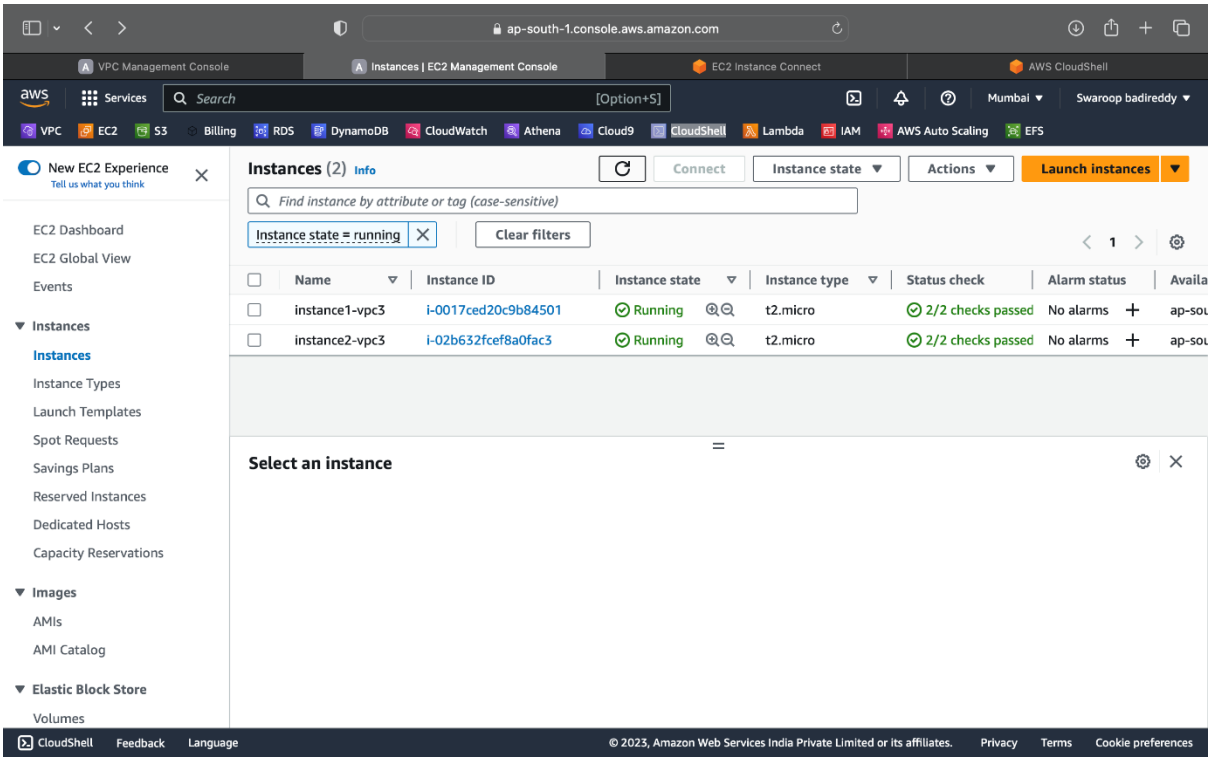


Adding tag using cloudshell:

- Click on the cloudshell service in the amazon all services field.
- Then a terminal will be open with the necessary cloudshell root id.
- Now type the required command to change the tag for the instance,
- Now go to the instance tab and copy the instance id for which we want to change the tag,
- And also give the value ie,the tag what we want to assign to the instance and press enter.
- Now go to the instance page and refresh,
- Check whether the name is changed or not.

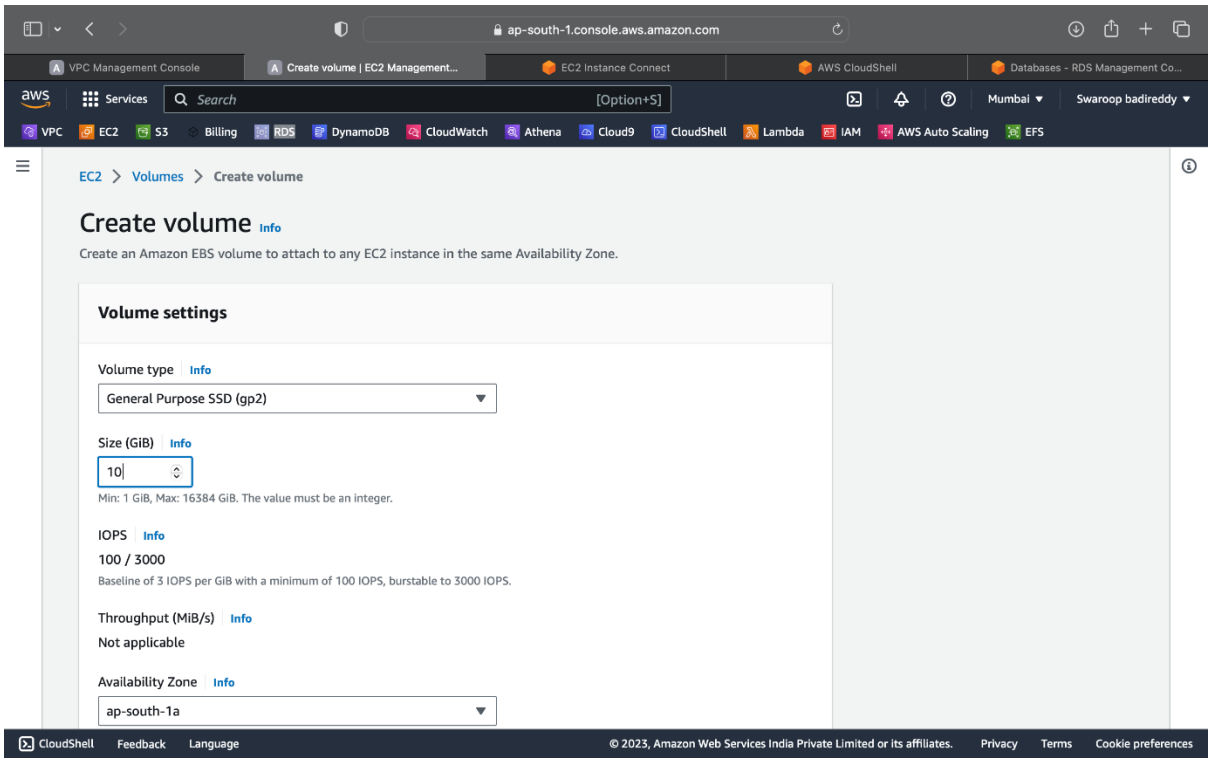


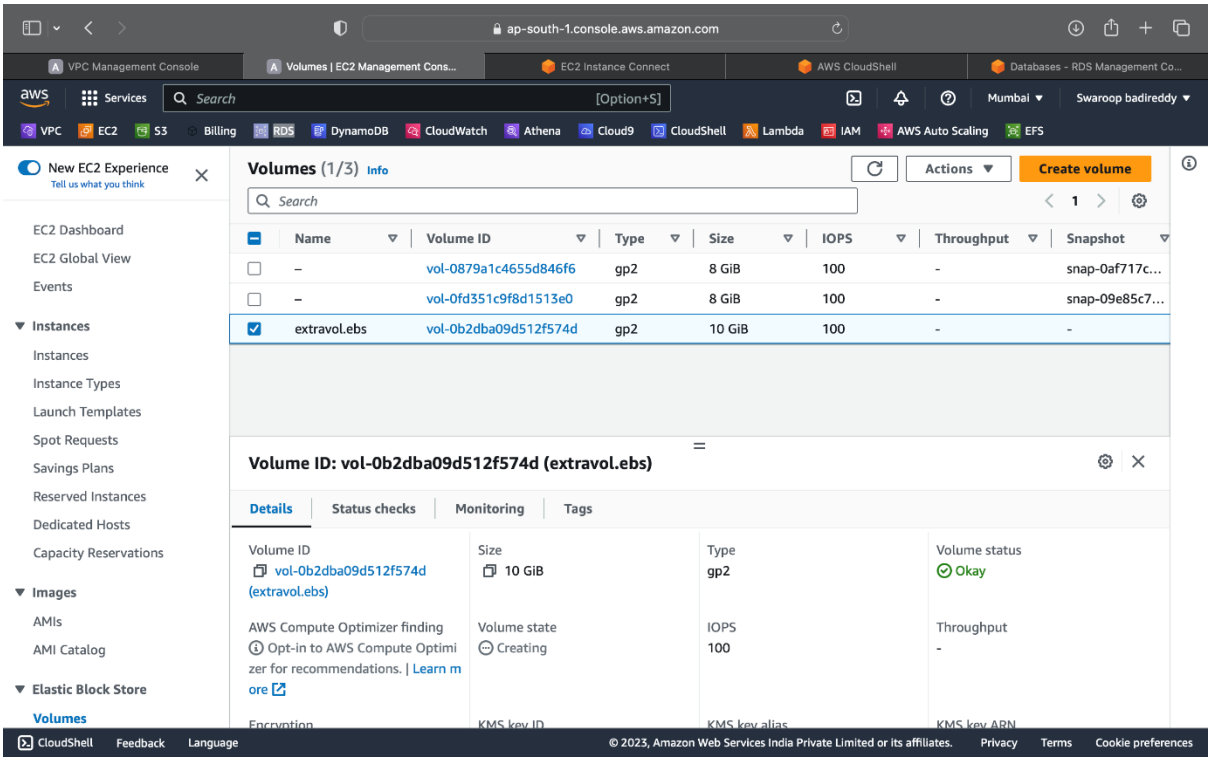
- **Instance changed (instance2-vpc3):**



Creating an extra volume for snapshot:

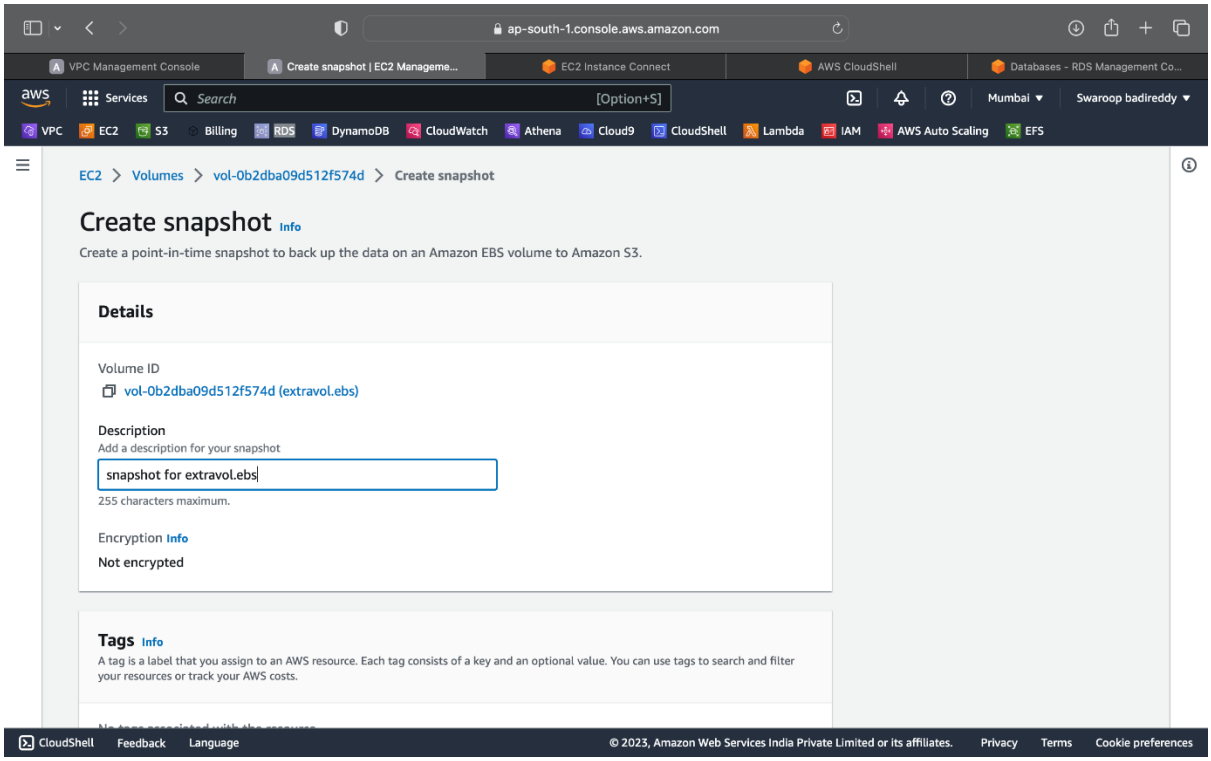
- Go to volumes and click on create volume,
- Then select the size of the extra volume required for snapshot.
- Select the availability zone ap-south-1a.
- Then click on create volume.
- An extra volume is created.

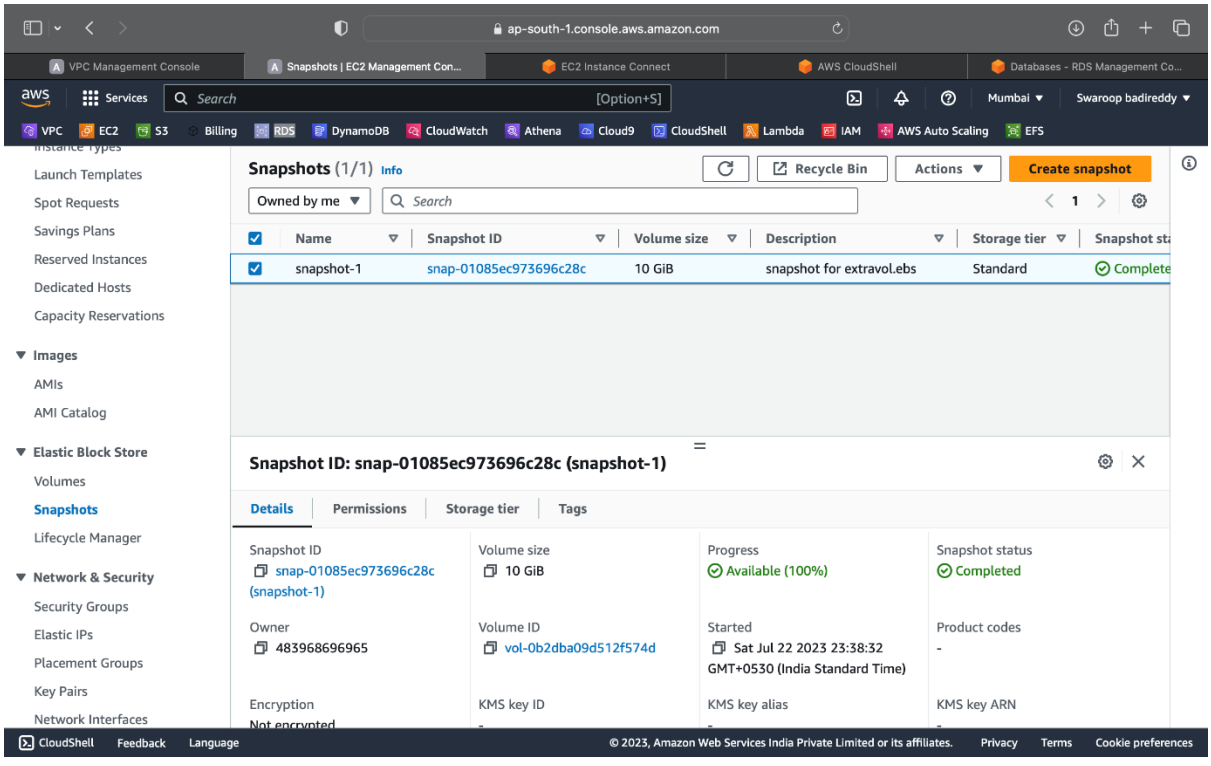




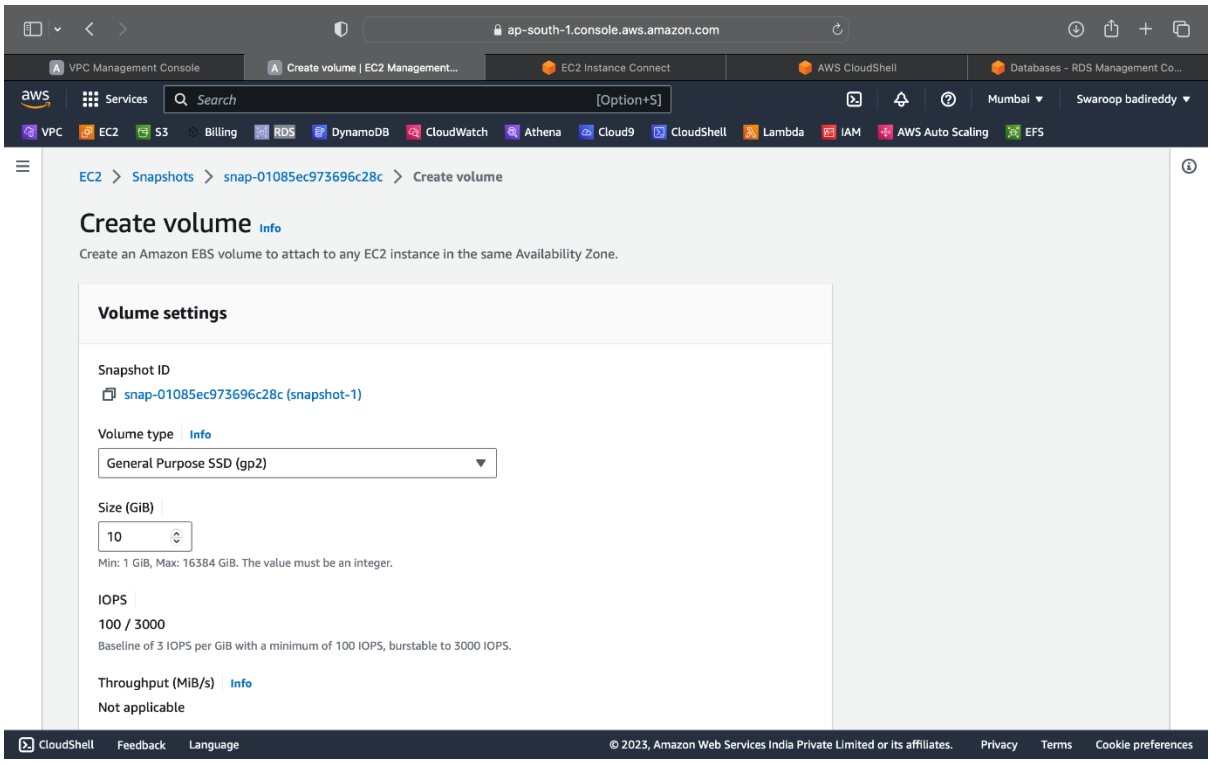
Creating a snapshot:

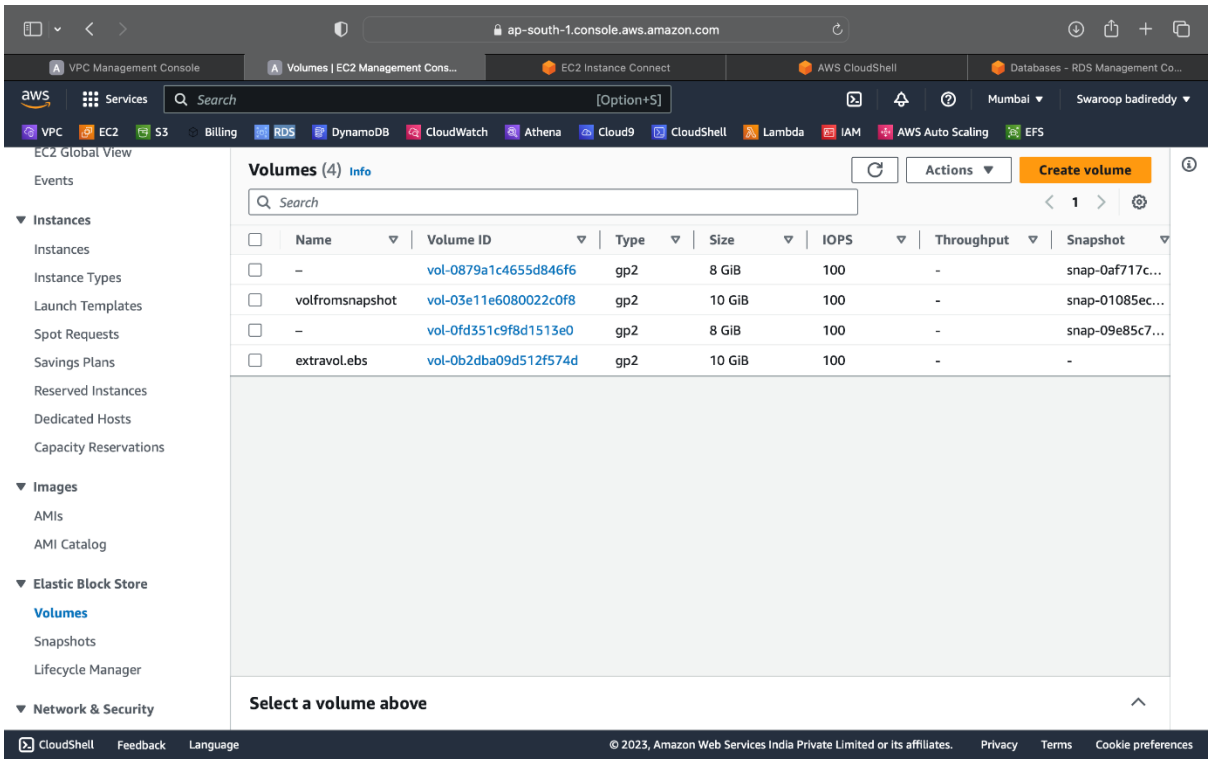
- Go to volumes and select the extra volume created and go to actions and click on create snapshot.
- Then give the required fields and create snapshot.
- A snapshot is created for the volume.





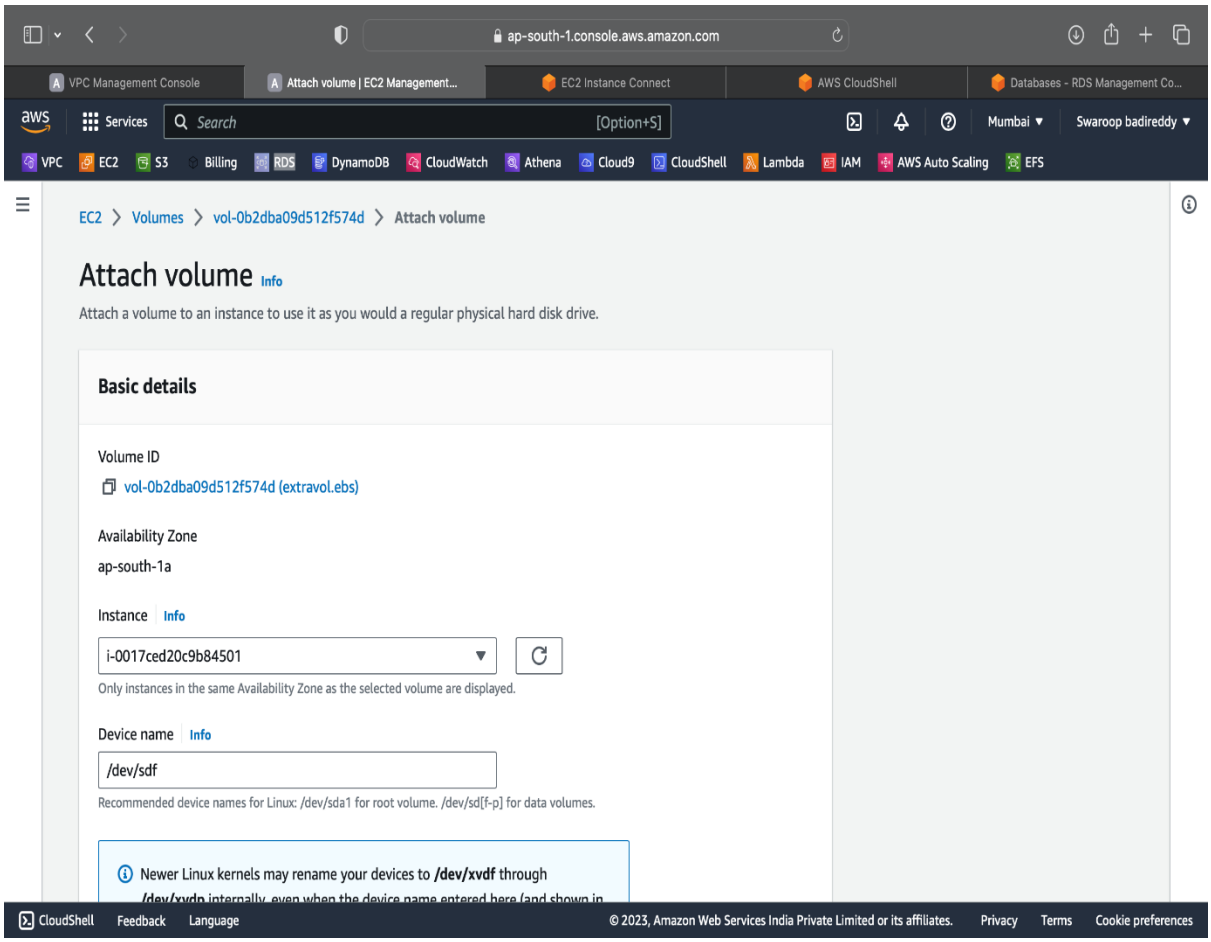
- Now creating another volume in ap-south-1b ie,subnet-2 for snapshot.
- Select the snapshot-1 and go to actions and create volume.
- Then give the required size and tags for the volume.
- A volume for snapshot is created.

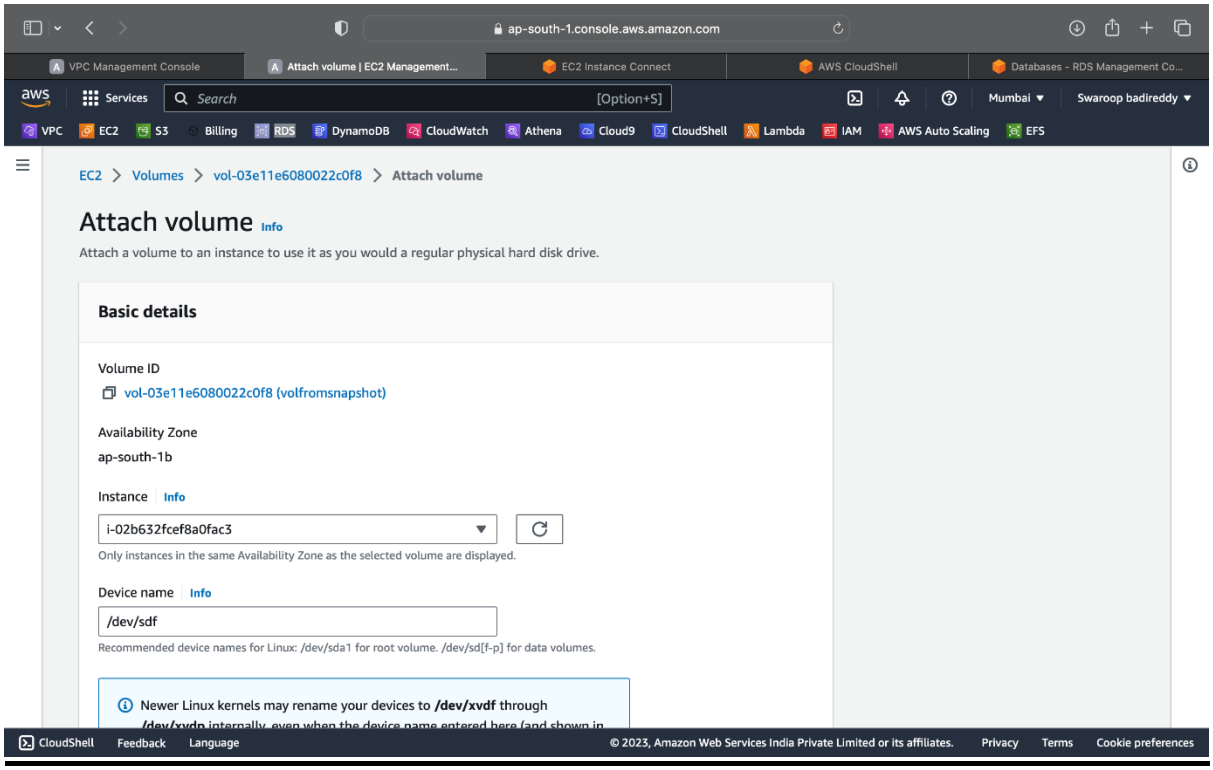




Attaching volumes:

- First go to volumes.
- Then click on the required volume say extravol.ebs
- Then go to actions and click on attach volume.
- A field will open where you should select the preferred zone and click on attach volume.
- The volume will be attached to the snapshot.
- Similarly for volfromsnapshot.



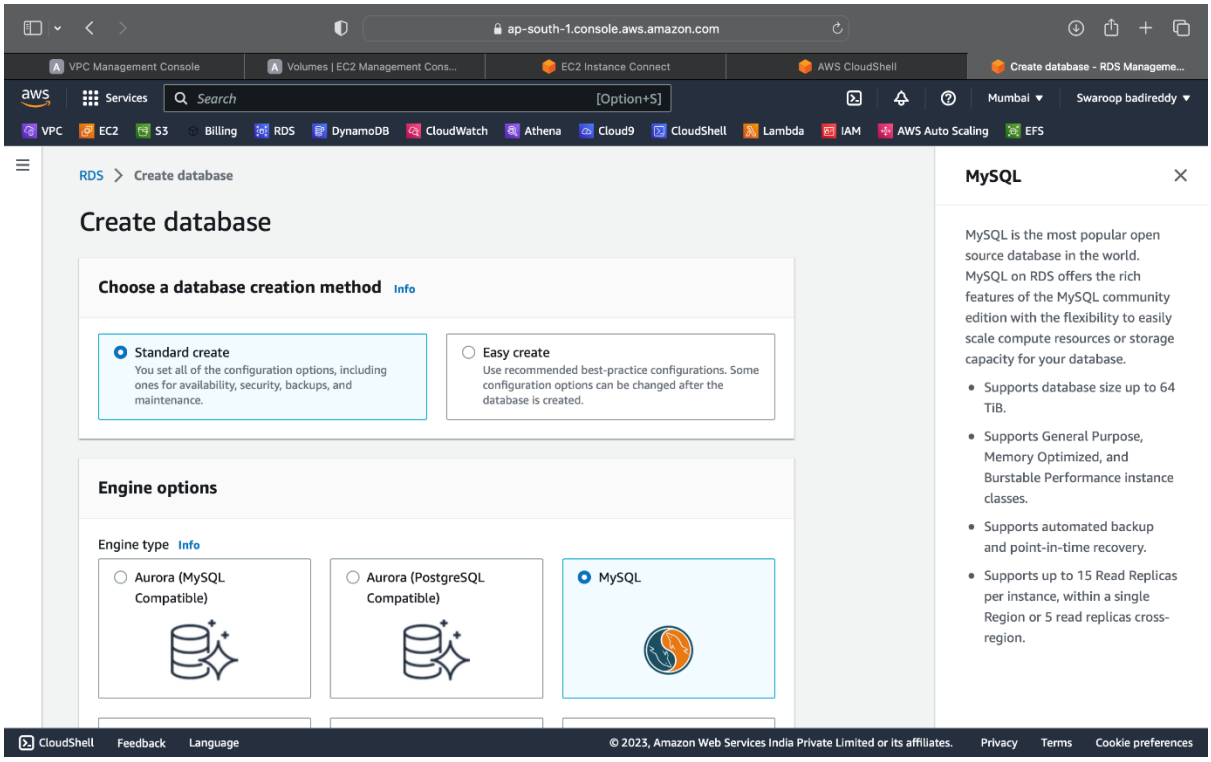


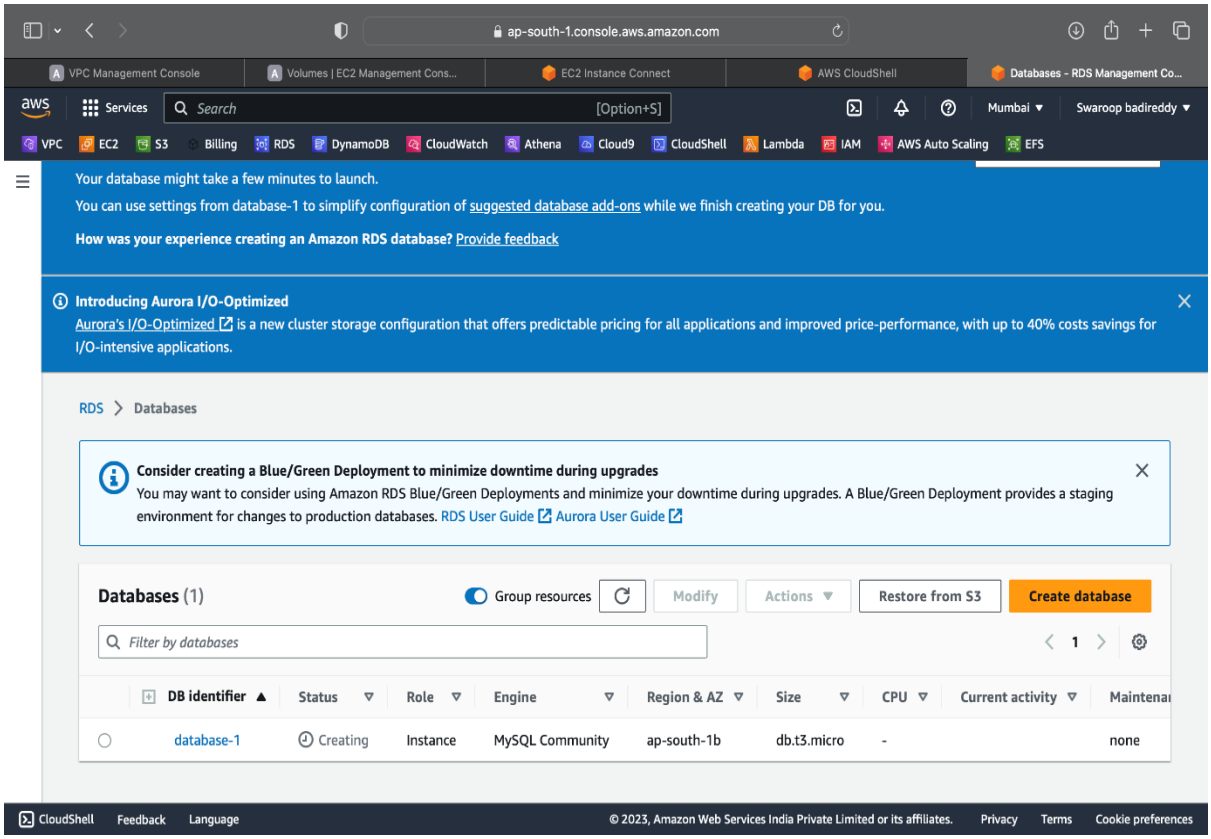
So,this is all about snapshot service.

Creating an rds:

- Open rds service in amazon services.
- Then click on create database.
- A tab will open where you need to select the required fields.
- Select standard create.
- Then click on my sql-free tier
- Then it will ask for password
- Enter the password.
- Then modify the limit from 1000 to 22.

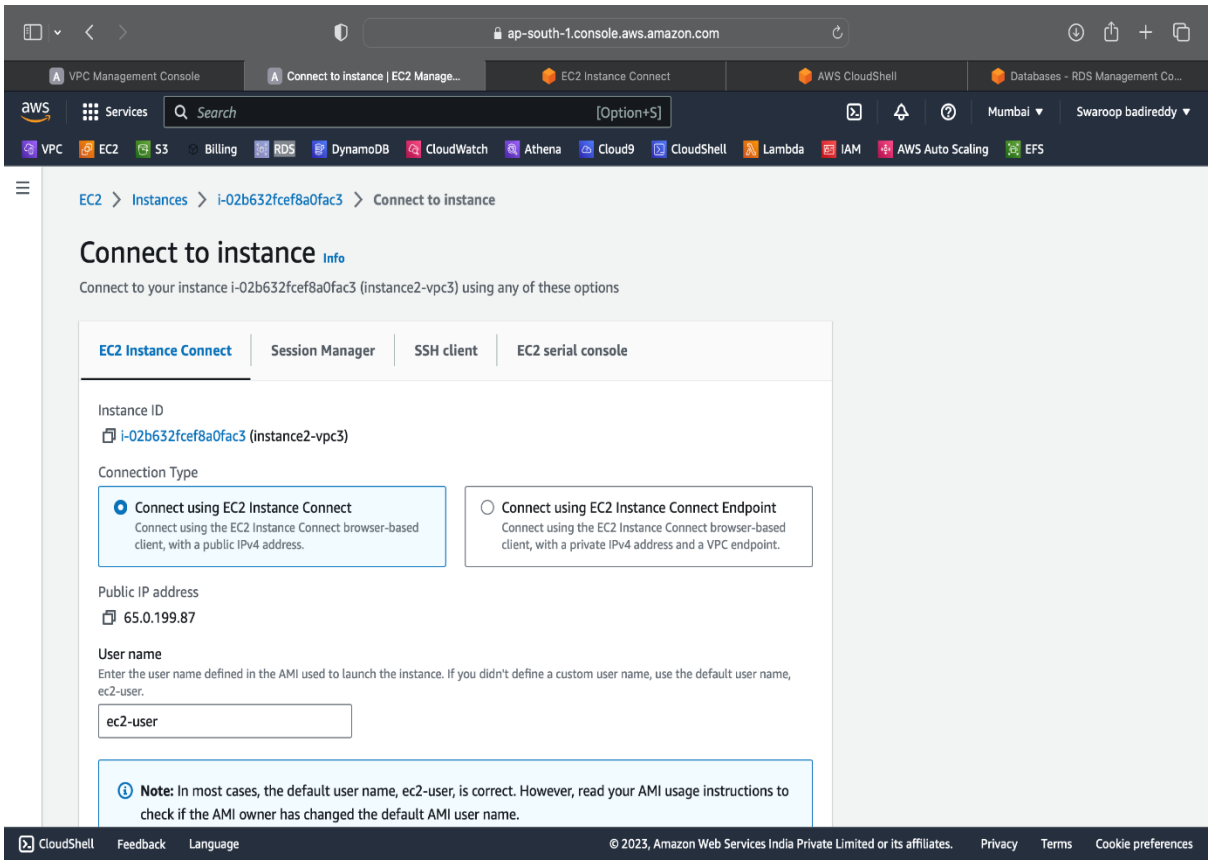
- Then all the remaining fields are as it is and then click on create database.





Connecting to ec2 instance:

- After the database gets backed up,then go to the ec2 instance tab.
- Select instance2-vpc3 and click on connect.
- Then in the next page click on connect to ec2-user connect.
- Then a new tab will open where a connection is established between the rds and the ec2 instance as they both are present in the same subnet.

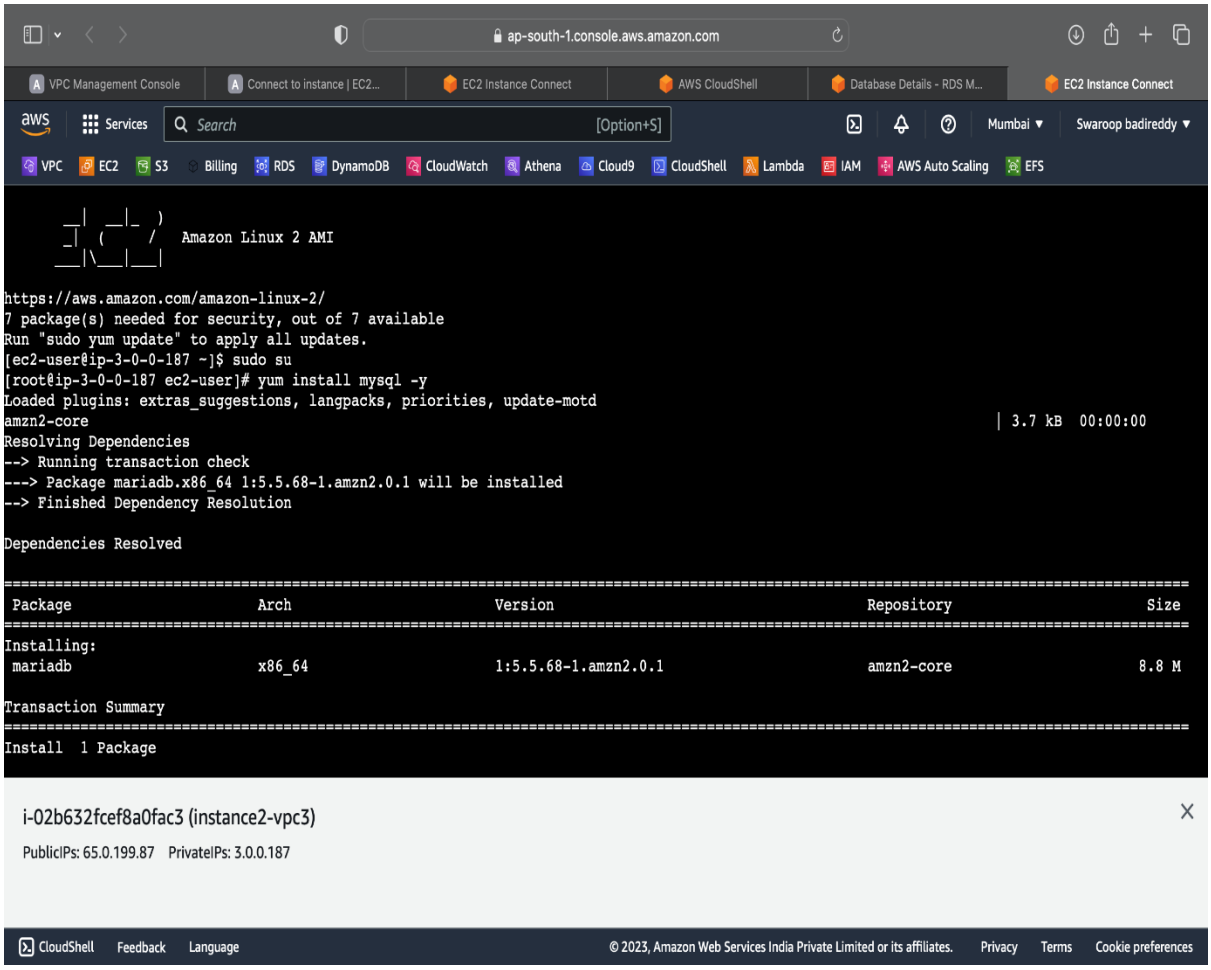


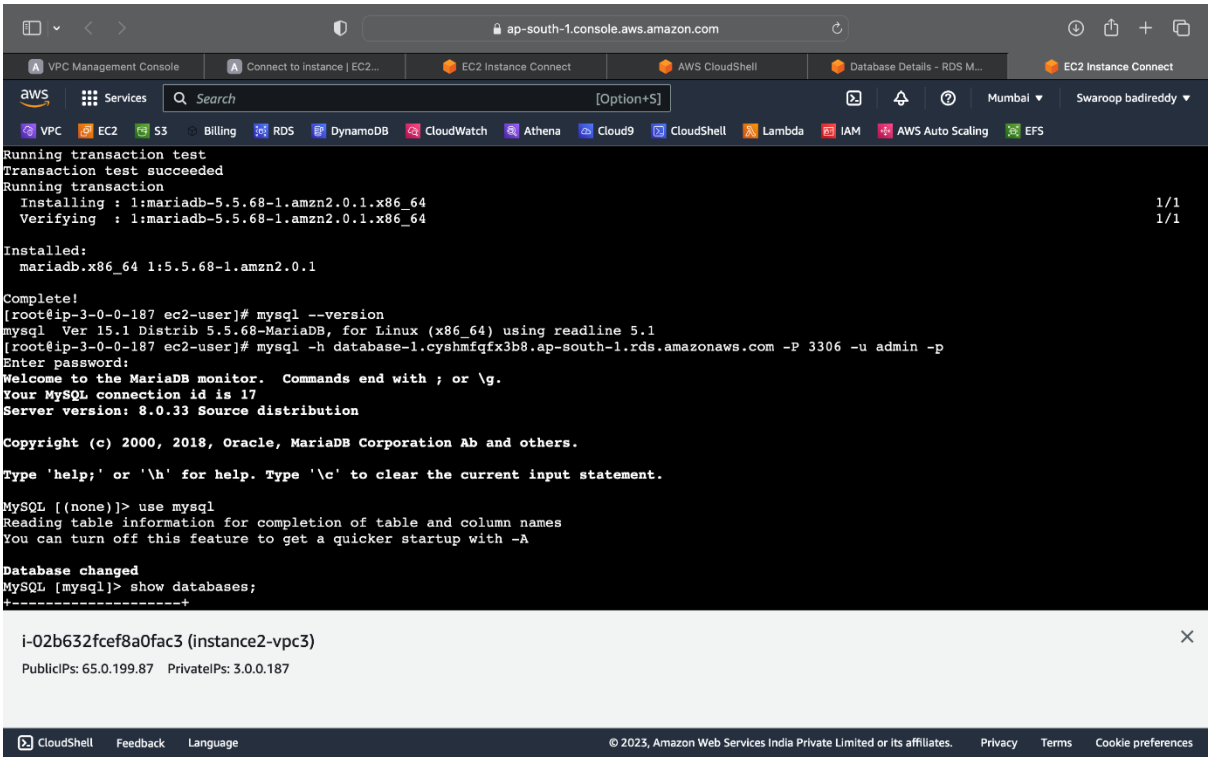
- Here we should use some command to check whether the ec2 instance and the database are connected are not.

The following are the commands used in this console.

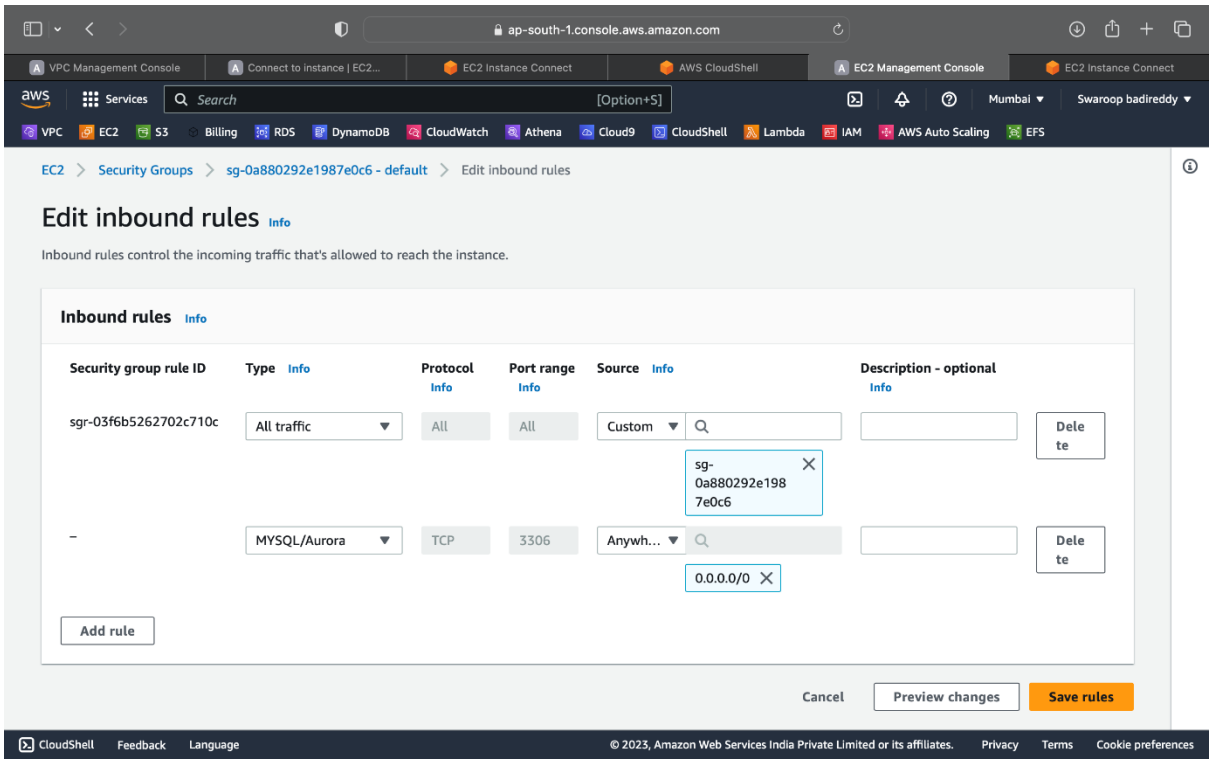
- sudo su
- yum install mysql -y

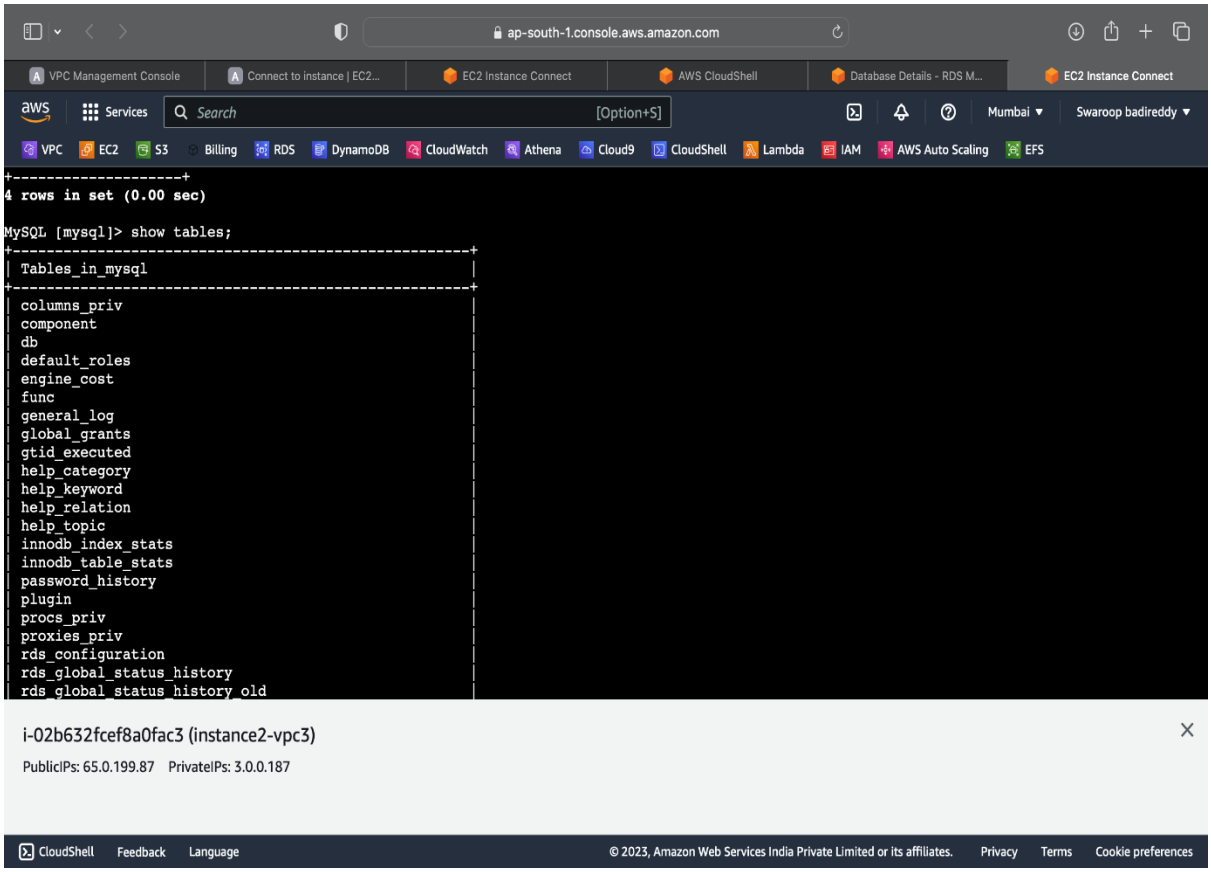
- mysql –version
- mysql -h database endpoint.rds.com -P 3306 -u admin -p
- Enter password:
- show databases;
- use mysql;
- show tables;
- create database db;
- use db;
- CREATE TABLE Tb;
- Select * from;
- exit

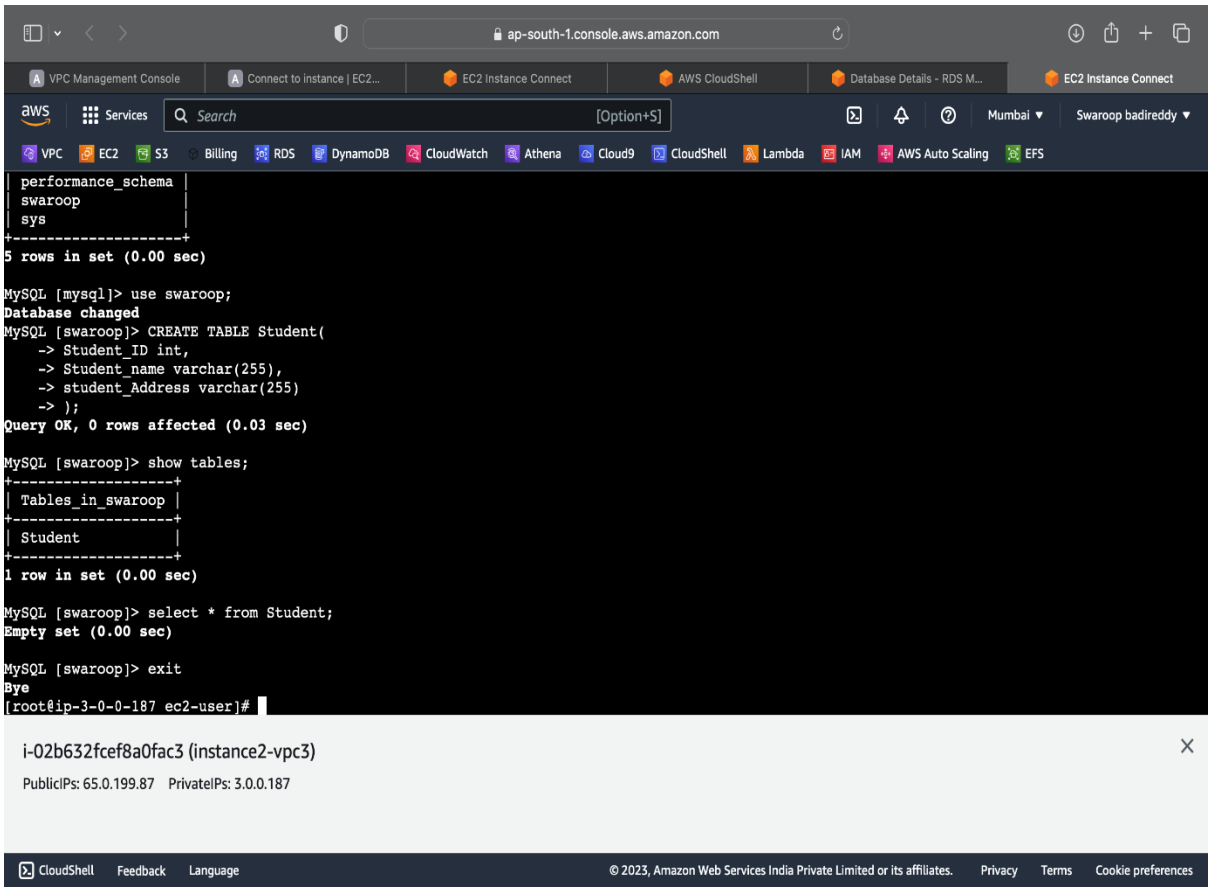




- While entering the password if it get stucked at the password then go to database.
- Click on the database and go to security groups.
- Then click on edit inbound rules in security groups.
- There click on mysql/aurora and anywhere.
- And save changes.
- Return to ec2 console and execute the commands.

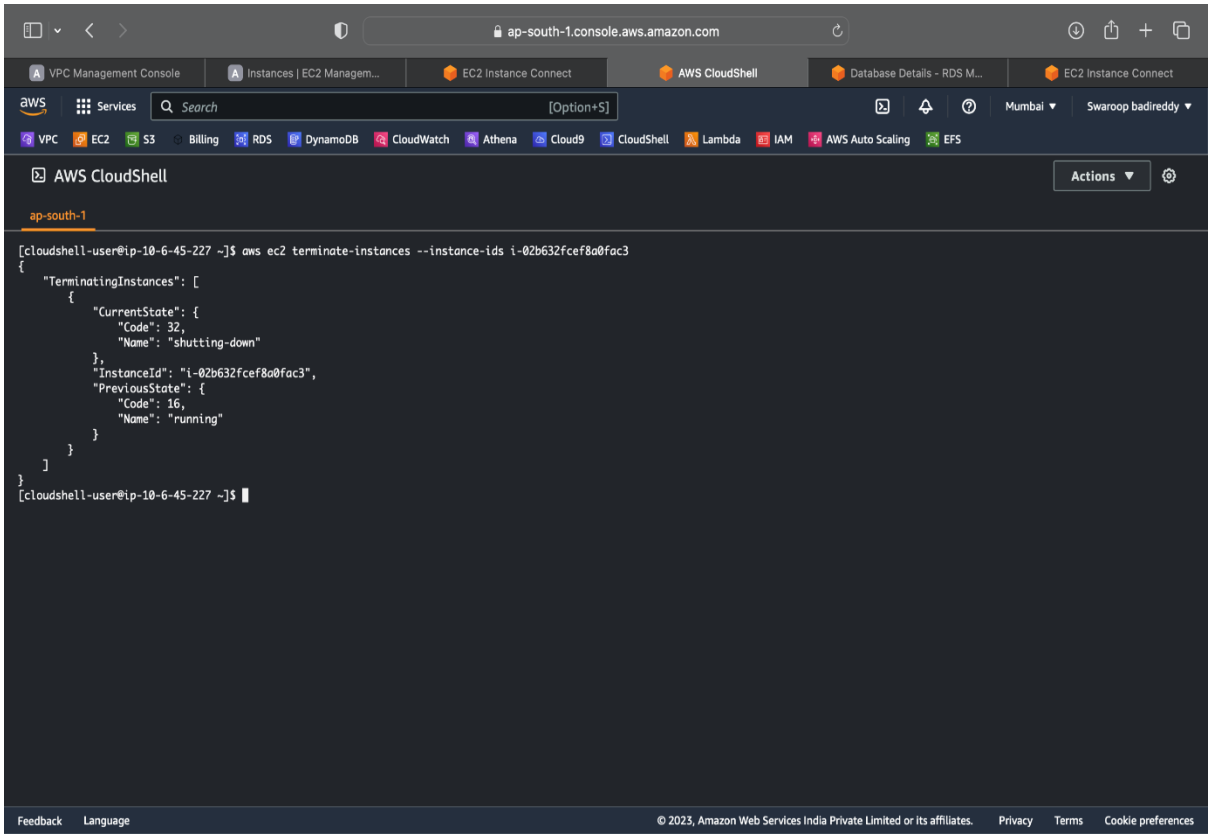



ap-south-1.console.aws.amazon.com
VPC Management Console
Connect to instance | EC2...
EC2 Instance Connect
AWS CloudShell
Database Details - RDS M...
EC2 Instance Connect
Services
Search
[Option+S]
Mumbai
Swaroop badireddy
VPC EC2 S3 Billing RDS DynamoDB CloudWatch Athena Cloud9 CloudShell Lambda IAM AWS Auto Scaling EFS
+--------------------+
4 rows in set (0.00 sec)
MySQL [mysql]> show tables;
+--+
| Tables_in_mysql |
+--+
| columns_priv |
| component |
| db |
| default_roles |
| engine_cost |
| func |
| general_log |
| global_grants |
| gtid_executed |
| help_category |
| help_keyword |
| help_relation |
| help_topic |
| innodb_index_stats |
| innodb_table_stats |
| password_history |
| plugin |
| procs_priv |
| proxies_priv |
| rds_configuration |
| rds_global_status_history |
| rds_global_status_history_old |
i-02b632fcef8a0fac3 (instance2-vpc3)
PublicIPs: 65.0.199.87 PrivateIPs: 3.0.0.187
CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

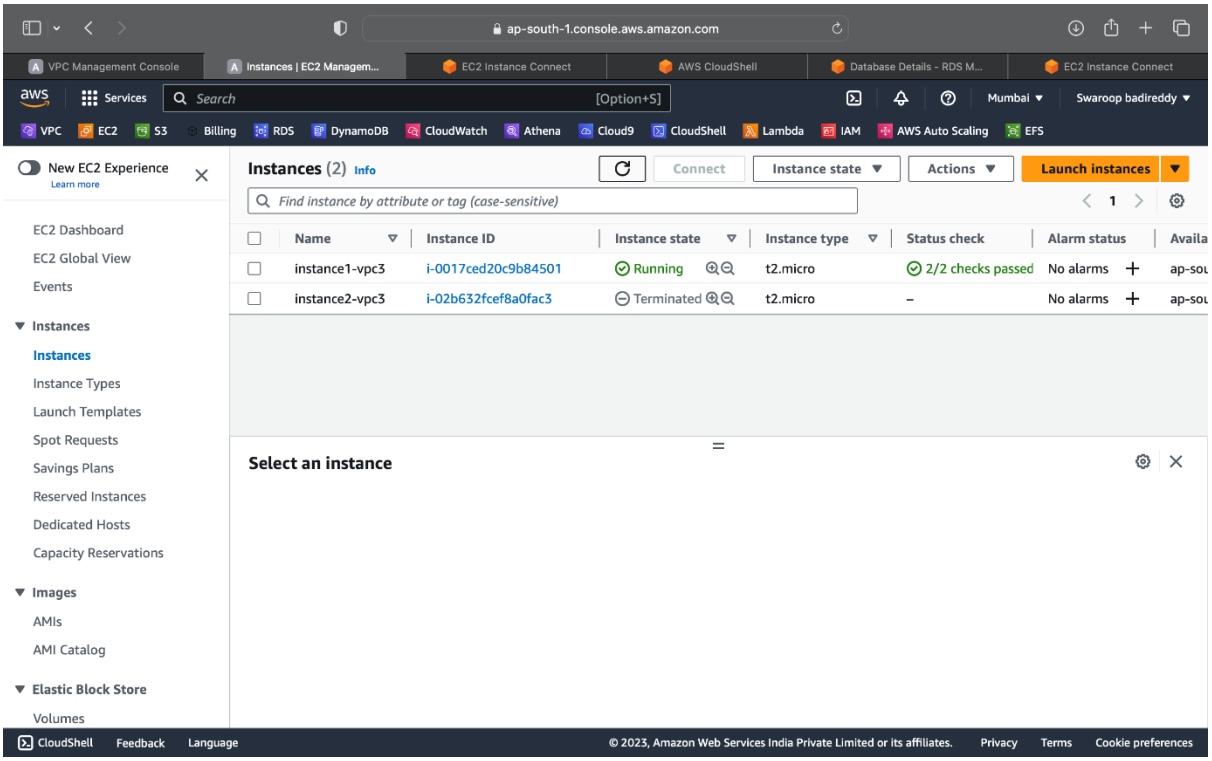

ap-south-1.console.aws.amazon.com
VPC Management Console
Connect to instance | EC2...
EC2 Instance Connect
AWS CloudShell
Database Details - RDS M...
EC2 Instance Connect
Services
Search
[Option+S]
Mumbai
Swaroop badireddy
VPC EC2 S3 Billing RDS DynamoDB CloudWatch Athena Cloud9 CloudShell Lambda IAM AWS Auto Scaling EFS
| performance_schema |
| swaroop |
| sys |
+--------------------+
5 rows in set (0.00 sec)
MySQL [mysql]> use swaroop;
Database changed
MySQL [swaroop]> CREATE TABLE Student(
-> Student_ID int,
-> Student_name varchar(255),
-> student_Address varchar(255)
->);
Query OK, 0 rows affected (0.03 sec)
MySQL [swaroop]> show tables;
+--------------------+
| Tables_in_swaroop |
+--------------------+
| Student |
+--------------------+
1 row in set (0.00 sec)
MySQL [swaroop]> select * from Student;
Empty set (0.00 sec)
MySQL [swaroop]> exit
Bye
[root@ip-3-0-0-187 ec2-user]#
i-02b632fcef8a0fac3 (instance2-vpc3)
PublicIPs: 65.0.199.87 PrivateIPs: 3.0.0.187
CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Terminating instance using cloud shell:

- Now disconnect the ec2 instance.
- Go to cloudshell terminal.
- Copy the command used for termination of instance,
- Place the instance id of the instance2-vpc3 which has to be terminated,
- And press enter.
- The command gets executed and the instance termination begins.



- After pressing enter go to ec2 instance tab and refresh the page.
- And the instance2-gets terminated,



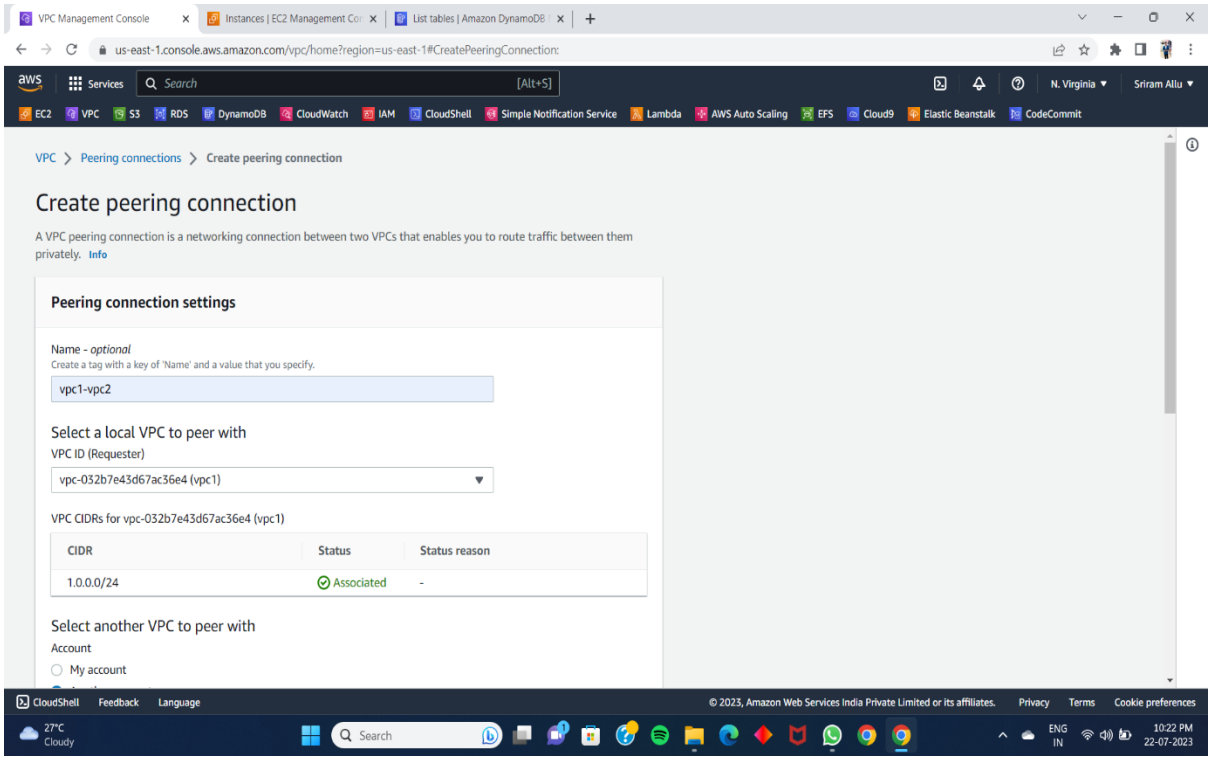
Establishing peering connections:

- To establish a peering connection,first go to vpc services.
- Then click on peering connection and create peering a connection.
- In the next page,fill all the necessary fields to establish a peering connection like,
  - Vpc id requestor
  - Vpc id acceptor
  - Same account or different account
  - Same region or different region etc

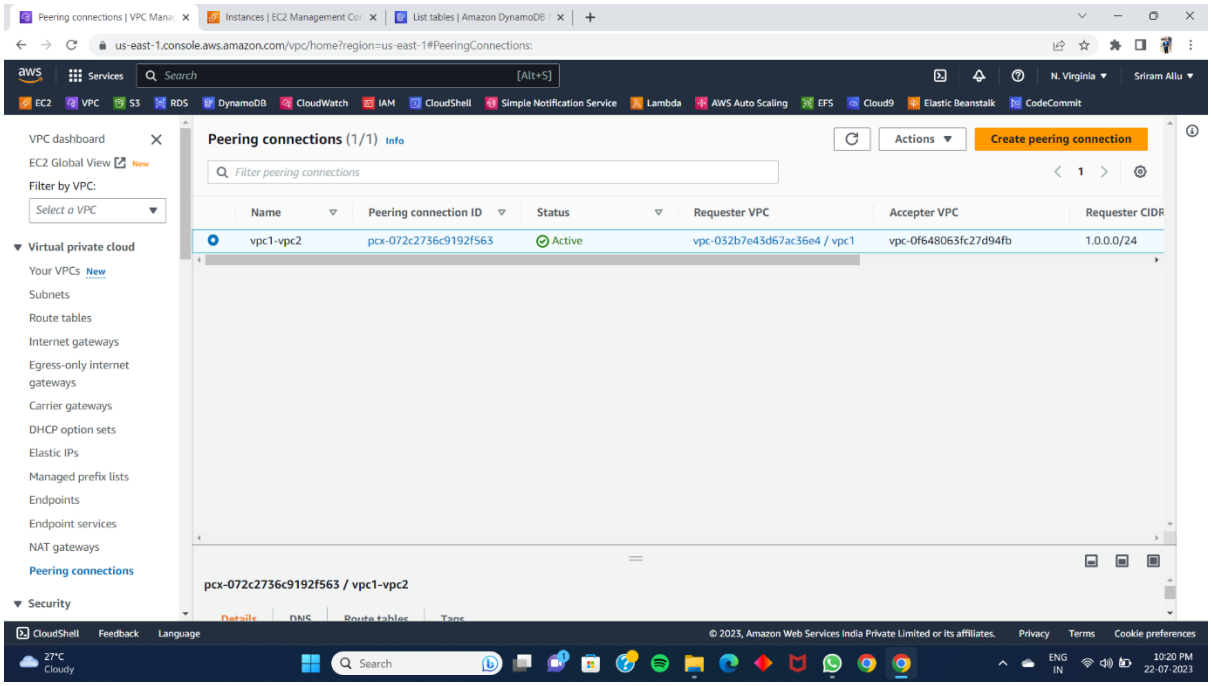
- Then click on create peering connection.
- A peering connection between two vpc's will be established.
- For example,in us-east-1 a peering connection vpc1-vpc2 is created for vpc2 in eu-north-1.
- Similarly the case for the remaining two accounts in different regions.
- Now,these all accounts are connected using these peering connections by their private ip's.
  - 1.0.0.58
  - 2.0.0.23

- 3.0.0.63

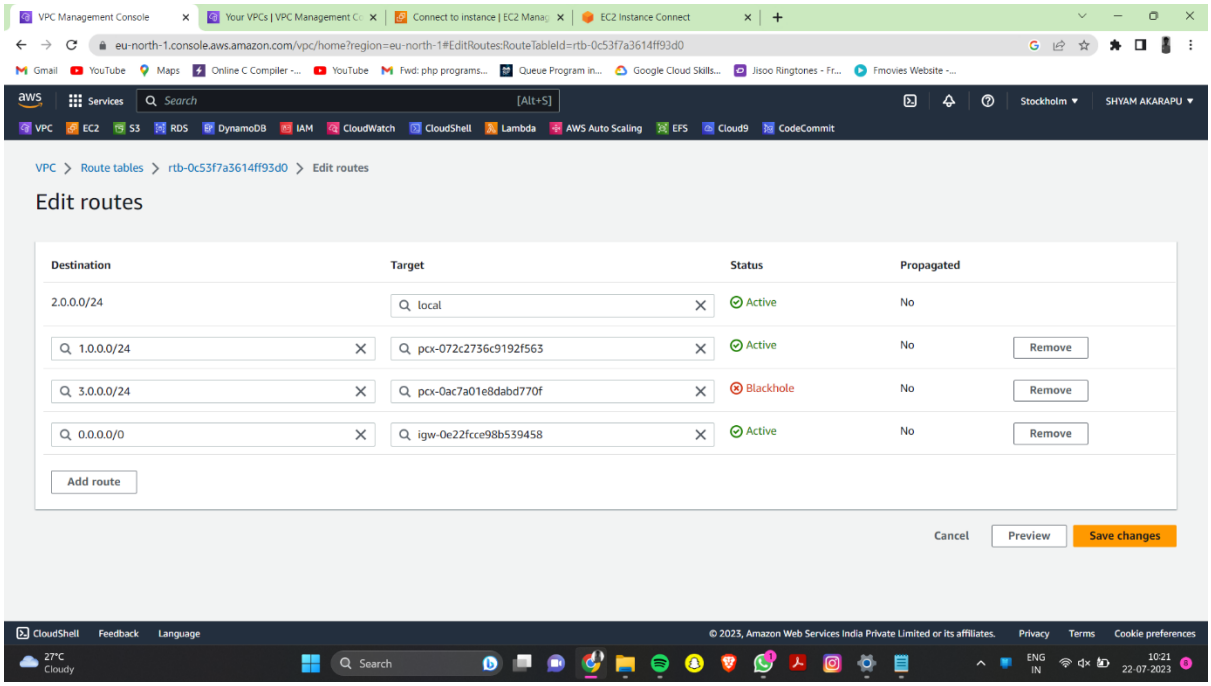
Creating peering connection:

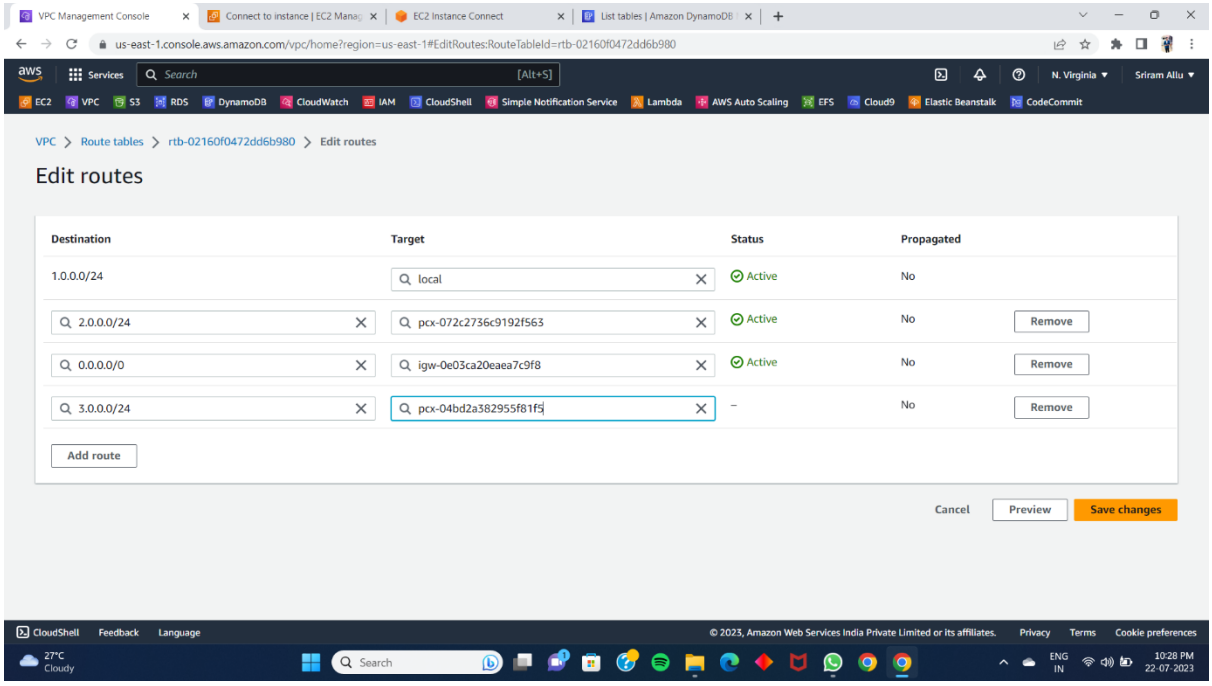


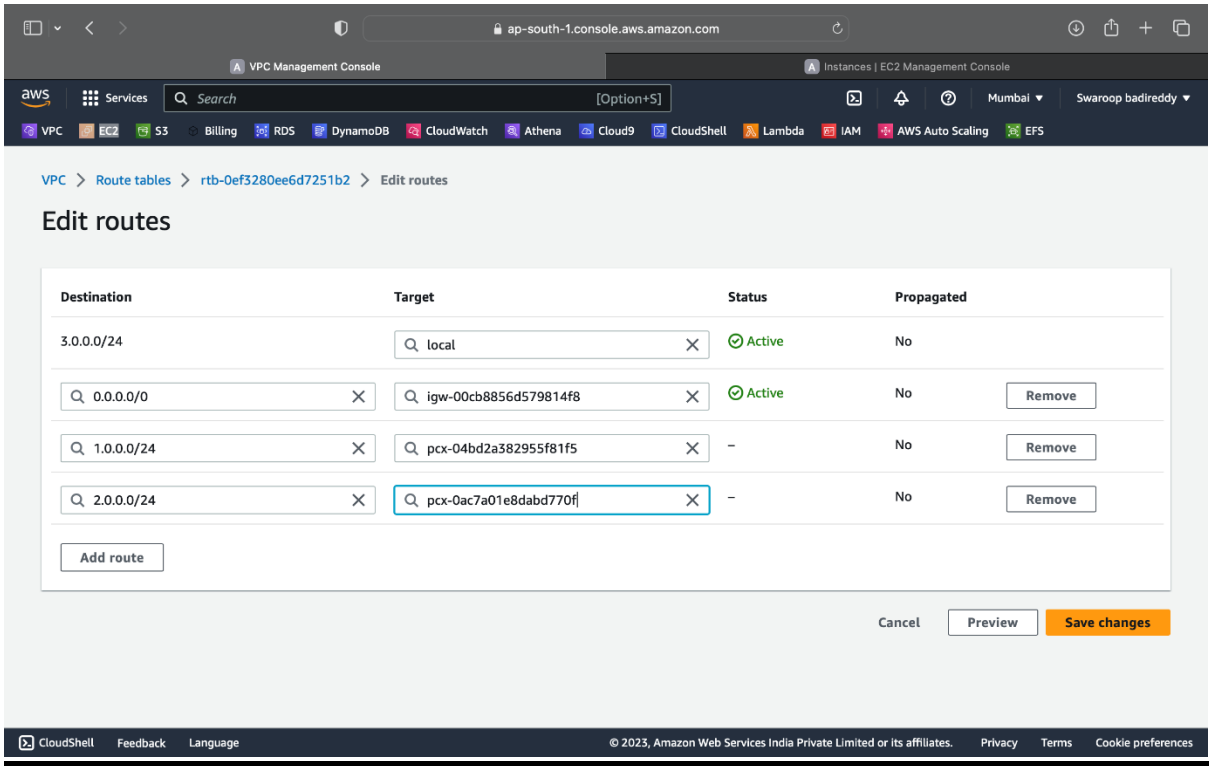
=



Editing routes for peering 3 vpc's:

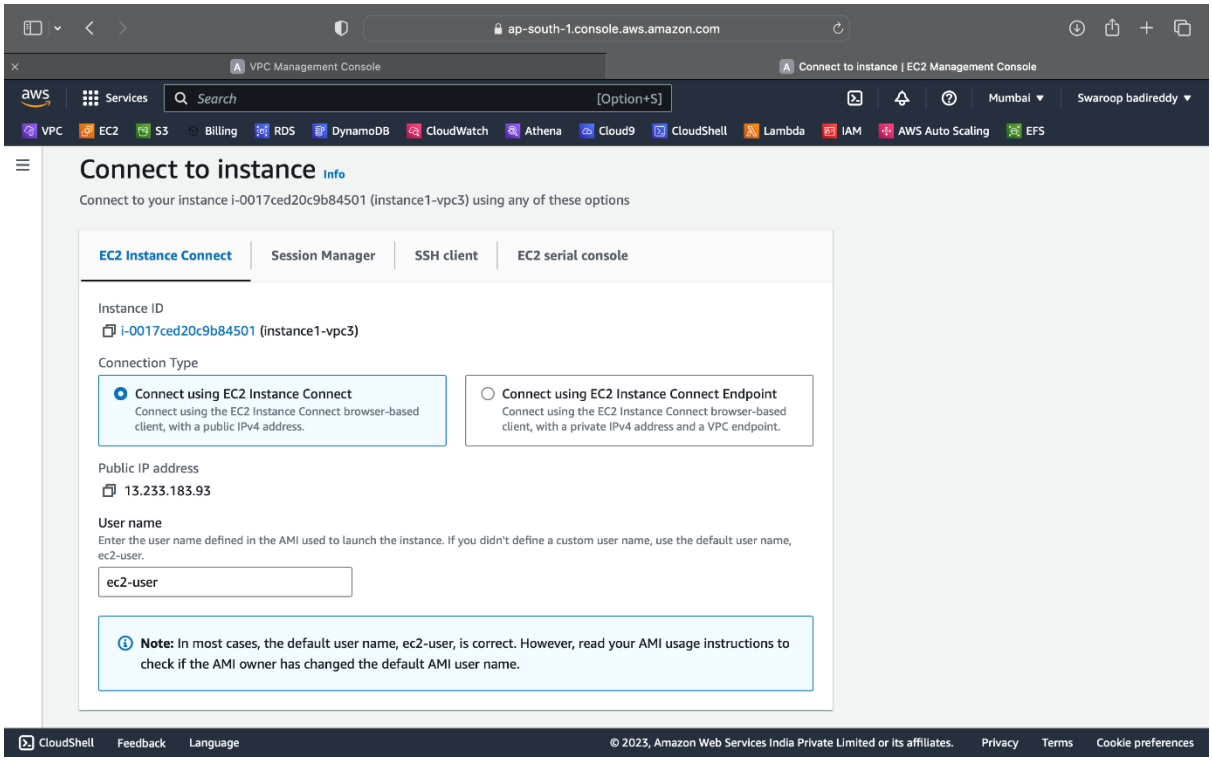




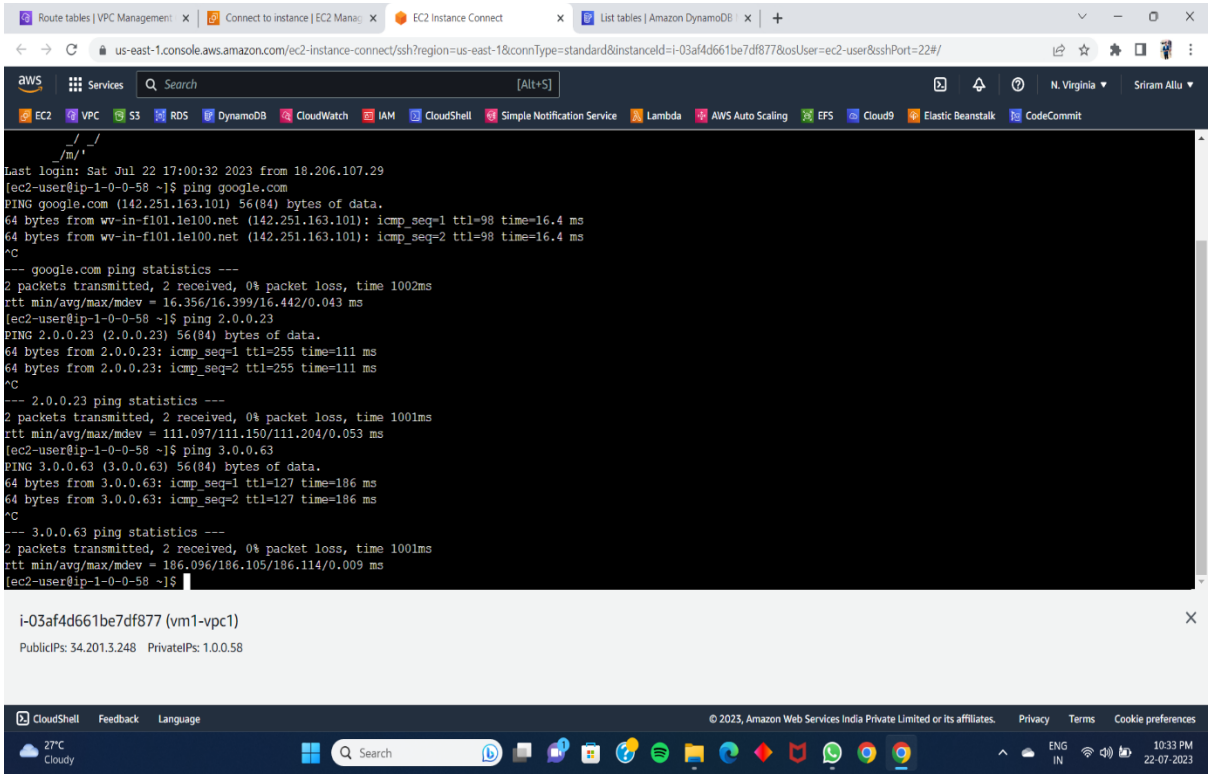


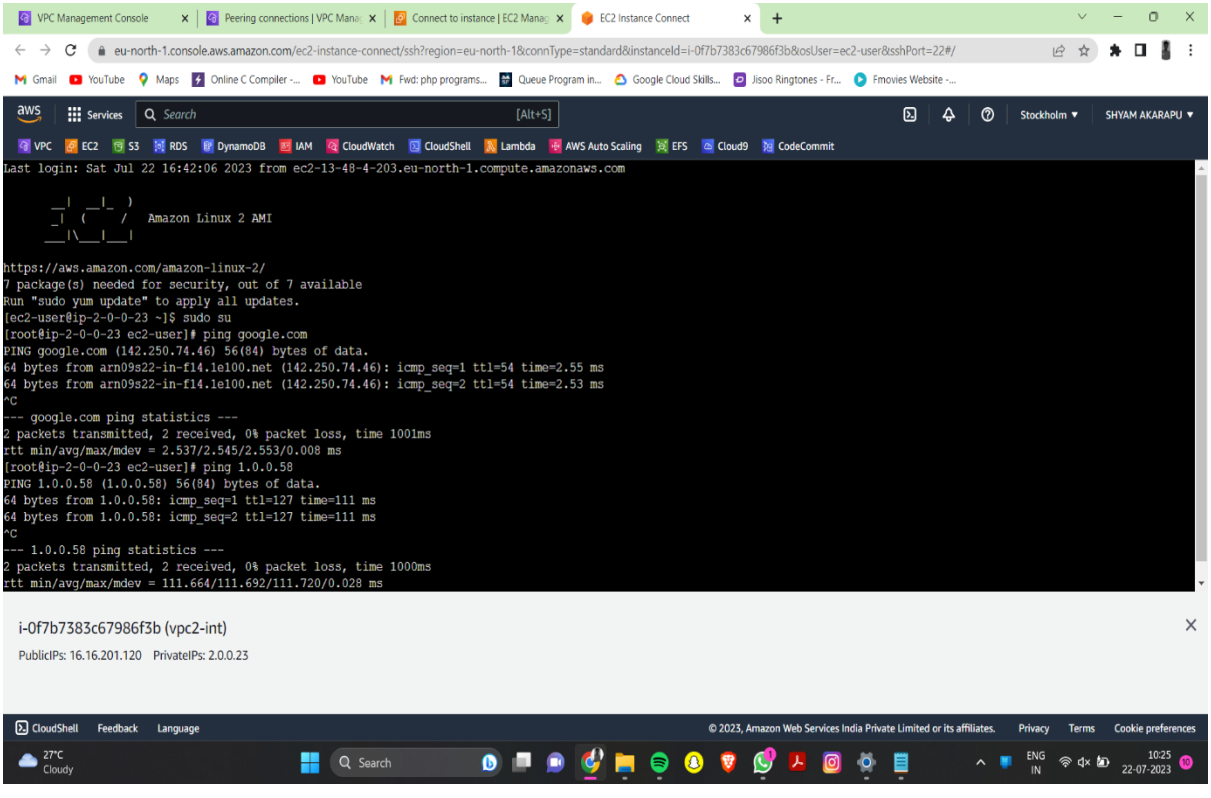
Connecting to ec2 instance:

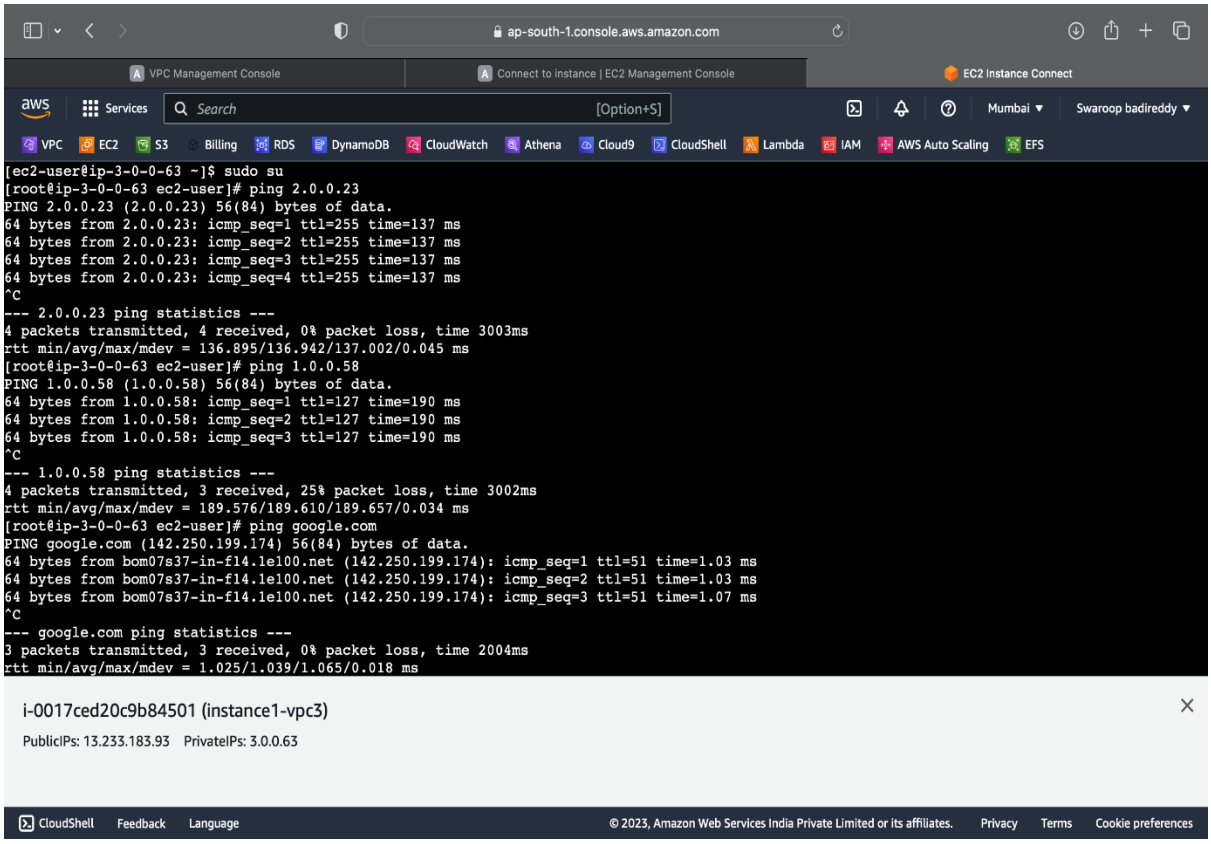
- Select the instance and click on connect.
- Then connect the ec2 instance using ec2-user connect.
- Then a new tab will open like a terminal.



Pinging the 3 vpc's:







Summary:

This architecture contains three different architects containing three different accounts which are connected by peering connections.

Each architect consists of multiple services like,

- Vpc,Ec2,Dynamo db,Cloudwatch,SNS,Iam role.
- Docker hub,S3,Lambda.
- Snapshot,Rds,Cloudshell.

Using the above services this combination of architects results in the Formation of a **12-tier architect.**