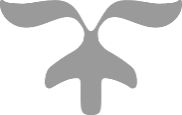




HACKATHON ON AWS



**TEAM MEMBERS**

**🡪A.YASASWINI**

**🡪M.TRIVENI**

**🡪B.SRIYA**

**🡪P.SAKULJI**

**🡪B.ANURAGH**

**🡪N.SHANKAR**

**What Is AWS?**

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

**What is DEVOPS?**

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

**Benefits of DevOps**

* Speed and Recovery
* Rapid Delivery
* Reliability
* Scale
* Security
* Improved collaboration

**AWS TOOLS**

AWS provides a wide range of tools and services However, there are a few tools that are widely used and considered fundamental in many AWS deployments. Here are some of the most commonly used tools of AWS:

* Amazon EC2 (Elastic Compute Cloud)

It provides a virtual server in the cloud, allowing you to run applications and services on a variety of operating systems.it also used to create various instances.

* Amazon RDS (Relational Database Service)

It Provides managed database services for various relational database engines, including Amazon Aurora, MySQL, PostgreSQL, Oracle, and SQL Server.

* Amazon VPC (Virtual Private Cloud)

It Offers a logically isolated virtual network within the AWS cloud. It allows you to launch AWS resources in a defined virtual network, providing control over IP addressing, subnets, and network gateways.

* *Amazon SQS (Simple Queue Service)*

It A fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.

* AWS CloudTrail

It Captures and logs API activity and events within your AWS infrastructure, providing audit trails for compliance, security analysis, and troubleshooting.

**ADVANTAGES:**

Amazon Web Services (AWS) offers several advantages that have contributed to its popularity and widespread adoption among businesses and developers. Some of the key advantages of AWS are:

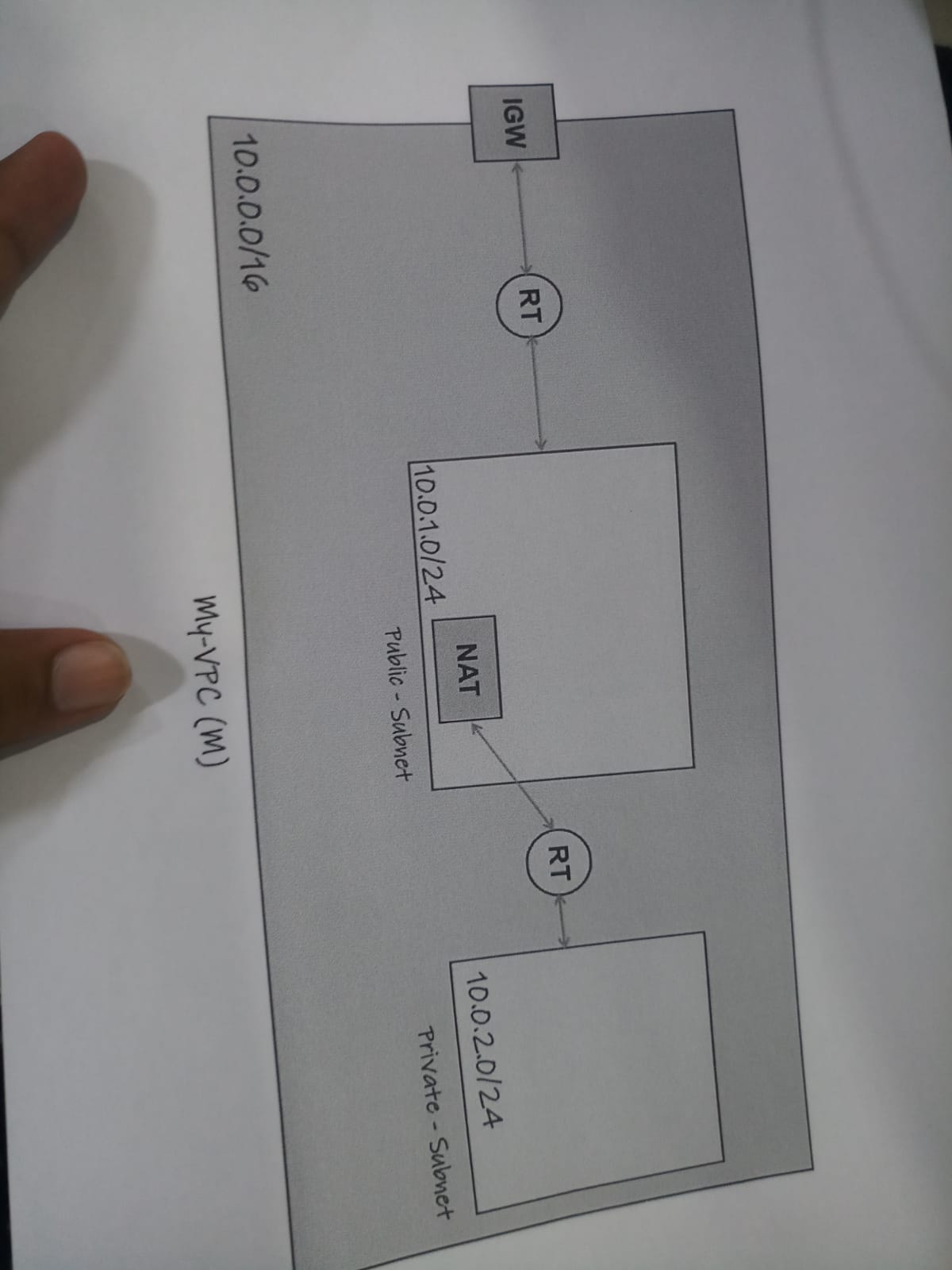
* Scalability : scalability enables businesses to handle sudden traffic spikes, accommodate growth, and optimize costs by paying only for the resources they use.
* Cost-effectiveness: AWS operates on a pay-as-you-go pricing model. It eliminates the need for upfront capital investment in hardware and infrastructure, making it cost-effective for businesses of all sizes.
* Global Infrastructure : AWS has a vast global infrastructure with multiple data centers located across different regions worldwide.
* Security : AWS has implemented a robust and comprehensive security framework to protect customer data and infrastructure. It offers various security features, such as identity and access management (IAM), encryption, network security, and data protection.
* Flexibility : WS provides a wide range of services and tools that cater to different application requirements and use cases. It supports various operating systems, programming languages, databases, and frameworks, allowing businesses to choose the technologies that best suit their needs.

**REAL LIFE APPLICATIONS**

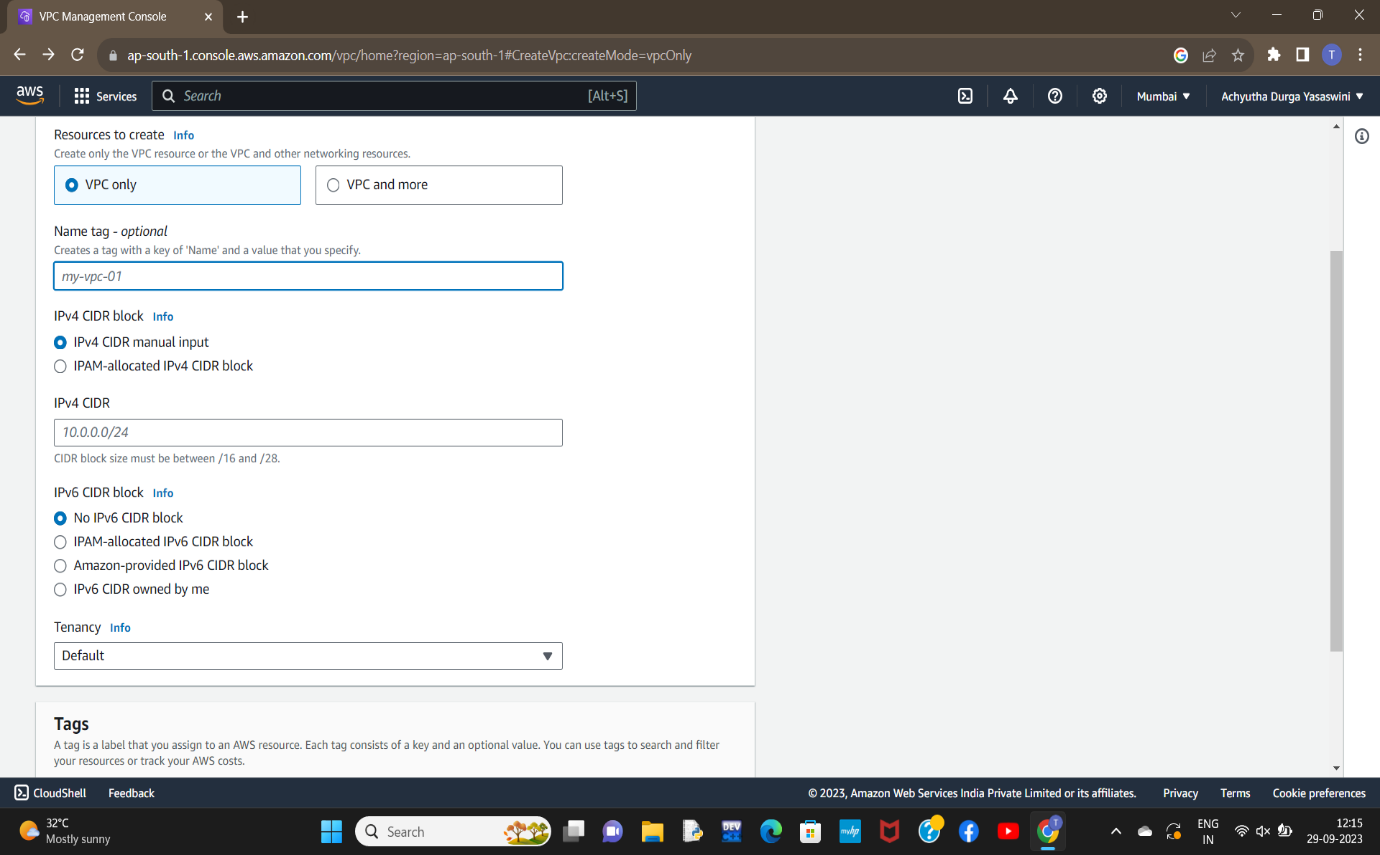
Amazon Web Services (AWS) offers a broad range of services and tools that can be utilized in various applications across different industries. Some of them were :

* NETFLIX : Netflix utilizes AWS's scalability and global infrastructure to stream its content to millions of users worldwide.Its content delivery services enable Netflix to deliver a seamless streaming experience.
* NASA : NASA utilizes AWS for various purposes, including data storage, processing, and analysis.It allows scientists and researchers to access and analyze data efficiently, enabling advancements in space exploration and research.
* PHILIPS : Philips is a global healthcare technology company that utilizes AWS for cloud based computing technology. It is used to store patient data and provide remote monitoring capabilities, and develop AI-driven healthcare applications.
* CAPITAL ONE : capital one is a leading financial institution.it uses AWS for its digital banking services and customer-facing applications. AWS provides the scalability, security, and compliance required to handle financial transactions, store sensitive customer data, and deliver real-time financial services to its customers.

**STAGE-1:**

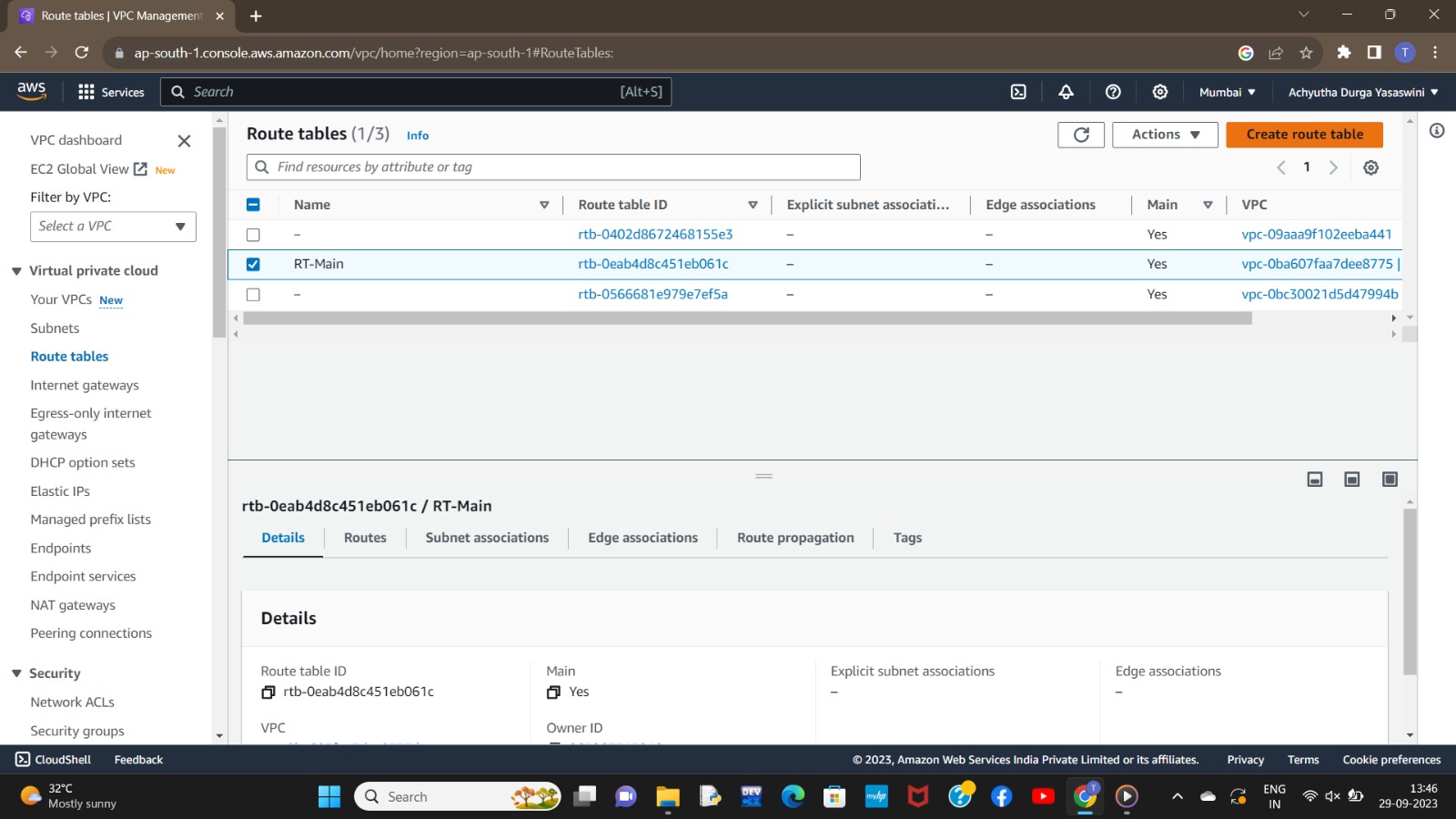
****

**Creating a VPC (Virtual Private Cloud):**

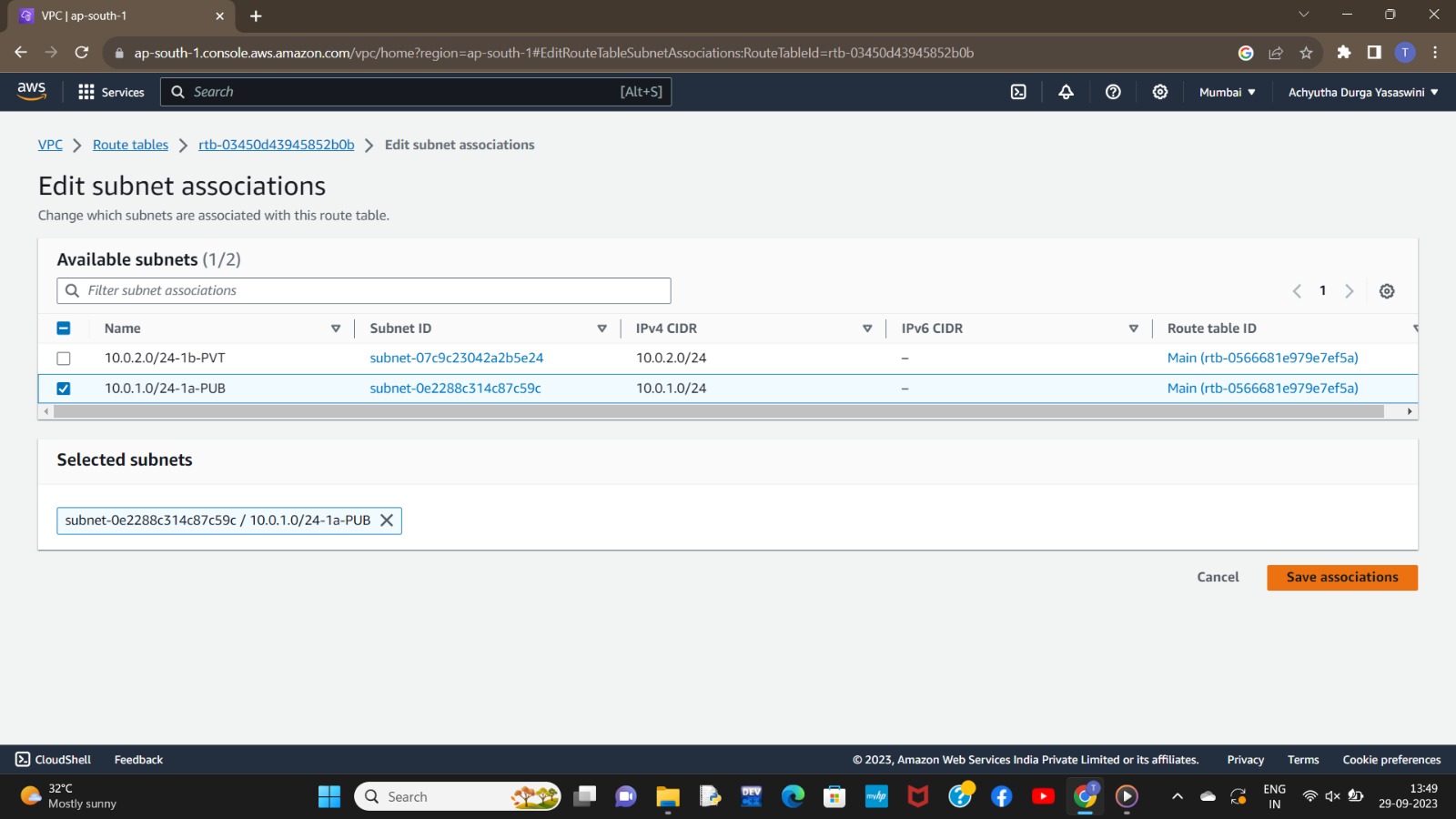
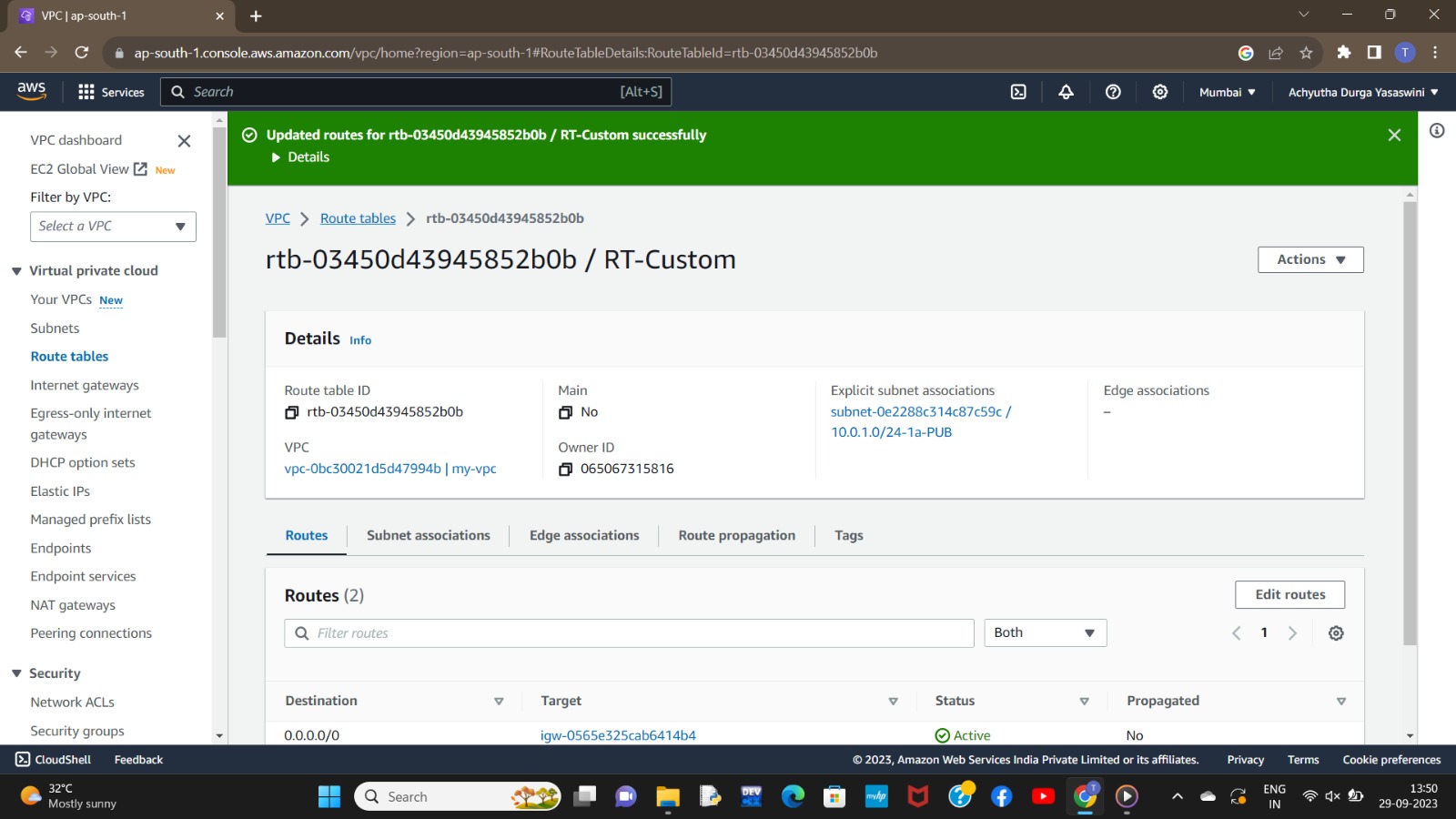
* In the VPC dashboard click on the “create VPC” Button to start the VPC creation wizard.
* Configure the VPC settings:
  + - Provide a name for your VPC.
    - Specify the IPv4 CIDR block for your VPC's IP address range ( 10.0.0.0/16).
    - Optionally, you can assign an IPv6 CIDR block to your VPC.
* Configure the VPC's subnets:
* Specify the IPv4 CIDR block for your first subnet (e.g., 10.0.0.0/24).
* Choose the availability zone where you want to create the subnet.
* Repeat this step to create additional subnets if needed.
* Configure the VPC's route table:
* Create a new route table or select an existing one.
* Associate the subnets created in the previous step with the route table.
* Configure the VPC's internet gateway:
* Create a new internet gateway or select an existing one.
* Attach the internet gateway to your VPC.
* Configure the VPC's security groups:
* Create new security groups or select existing ones.
* Define the inbound and outbound rules for each security group to control network traffic.
* Review all the configuration details and settings for your VPC. If everything looks correct, click on the "Create VPC" button to create your VPC.

**INTERNET GATEWAYS (IGW)**

**ROUTE TABLES (RT):**

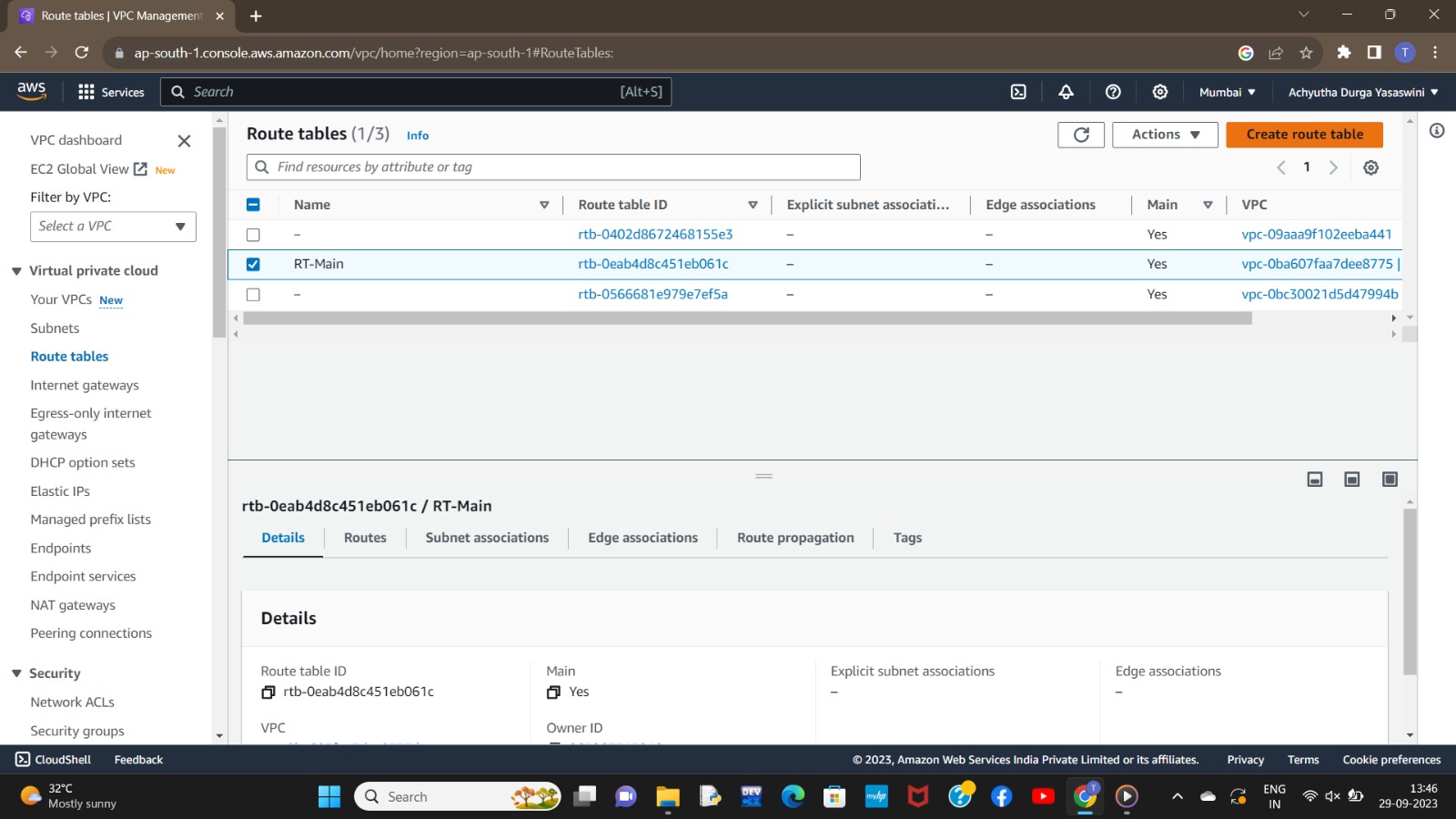
* Go to the "Route Tables" section: Within the selected VPC, click on the "Route Tables" option in the left navigation menu. This will display the list of existing route tables in the selected VPC**.**
* Create a new route table: Click on the "Create Route Table" button to create a new route table within the selected VPC.
* Configure the route table settings:
* Provide a name for the route table to identify it.
* Select the VPC in which you want to create the route table.
* Choose the desired subnet associations for the route table. Subnets can be associated with multiple route tables, and each subnet must be associated with at least one route table.
* Configure the routes:
* Click on the "Edit routes" button to add or edit routes in the route table.
* Add the desired routes by specifying the destination IP range and the target (e.g., an internet gateway, a virtual private gateway, or a NAT gateway)
* Save the route table: Click on the "Save" button to save the configured route table.
* Associate subnets with the route table:
* In the "Associations" tab of the route table, click on the "Edit subnet associations" button.
* Select the subnets you want to associate with the route table and click on the "Save" button.
* Review the route table: Verify the route table settings, associations, and routes in the AWS Management Console.
* Create a new route table: Click on the "Create Route Table" button to create a new route table within the selected VPC.

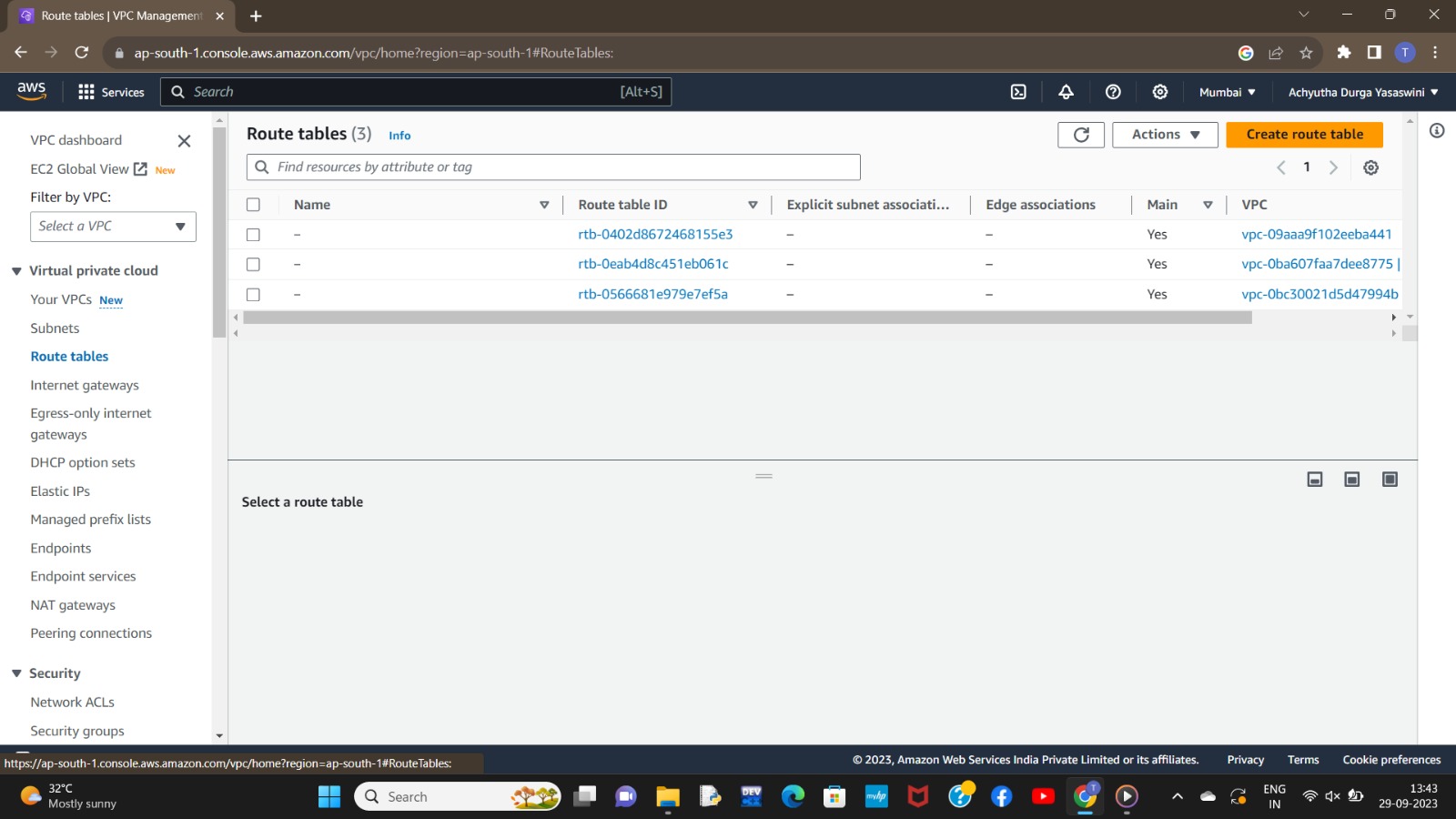
**PUBLIC SUBNET**

* Click on the "Create Subnet" button to create a new subnet.
* Configure the subnet settings:
* Select the VPC in which you want to create the subnet.
* Provide a name and a suitable CIDR block for the subnet. Ensure that the CIDR block falls within the IP address range of the VPC and doesn't overlap with other subnets.
* Select the desired availability zone for the subnet. It's recommended to create subnets in multiple availability zones for high availability and fault tolerance.
* Configure the subnet's route table:
* Choose an existing route table or create a new one for the subnet. To make the subnet public, associate it with a route table that has a route to an internet gateway.
* Verify the details of the subnet, including the VPC, CIDR block, availability zone, route table, and NACL settings
* Once you have reviewed and confirmed the configuration, click on the "Create" button to create the public subnet.
* If you require multiple public subnets across different availability zones, repeat the above steps to create them.

**NAT GATEWAYS**

* Click on the "Create NAT Gateway" button to create a new NAT gateway.
* Configure the NAT gateway settings:
* Select the subnet in which you want to create the NAT gateway. The subnet must be a public subnet, meaning it should have a route to an internet gateway.
* Choose an existing Elastic IP address or allocate a new one to associate with the NAT gateway. The Elastic IP address serves as a public IP address for the NAT gateway.
* Verify the configuration details for the NAT gateway, including the selected subnet and Elastic IP address
* Click on the "Create NAT Gateway" button to create the NAT gateway. The creation process may take a few moments.
* Update route tables: After the NAT gateway is created, you need to update the route tables to direct the outbound traffic from private subnets to the NAT gateway.
* Go to the "Route Tables" section in the VPC Dashboard.
* Select the route table associated with the private subnets that need access to the internet via the NAT gateway.
* Add a new route with a destination of "0.0.0.0/0" (or the desired IP range) and set the target as the newly created NAT gateway.
* Test the connectivity by launching an instance in a private subnet and ensuring it can access the internet through the NAT gateway.

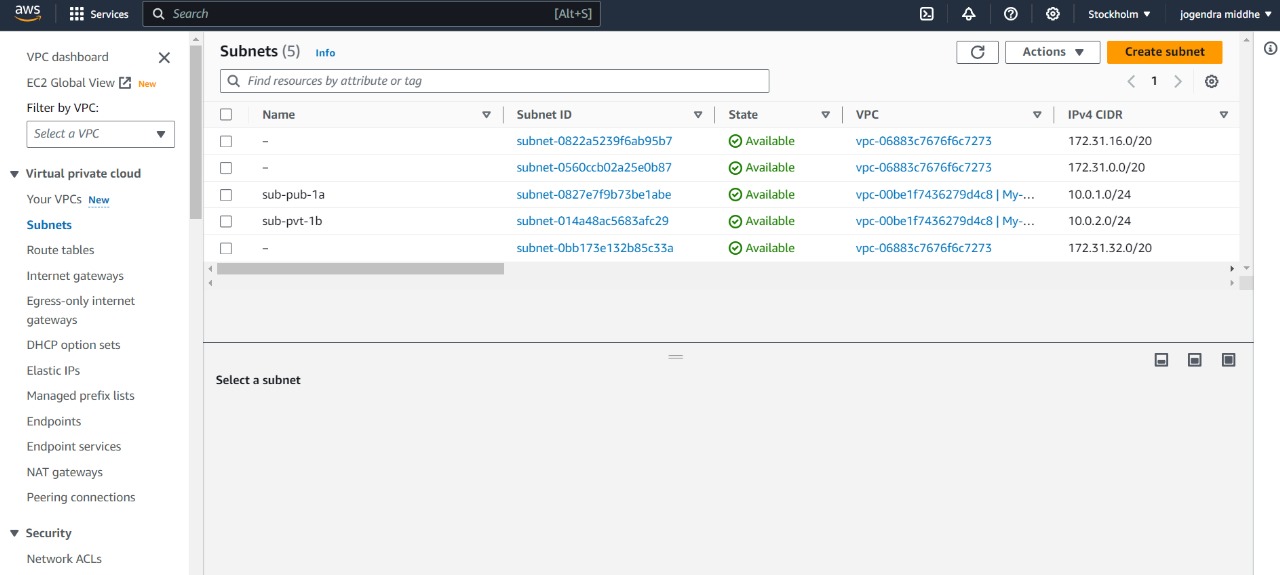
**ROUTE TABLE**

****

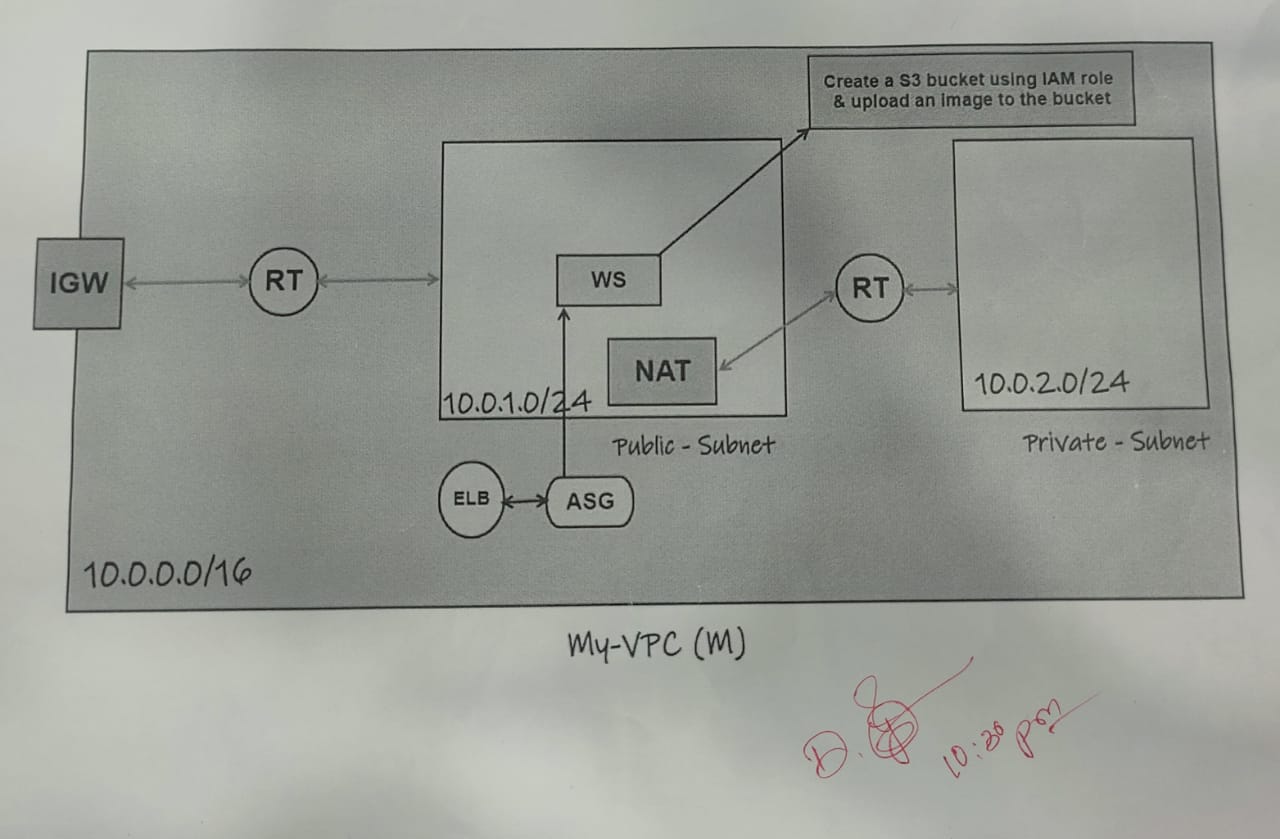
**PRIVATE SUBNET**

* Click on the "Create Subnet" button to create a new subnet.
* Configure the subnet settings:
* Select the VPC in which you want to create the subnet.
* Provide a name and a suitable CIDR block for the subnet. Ensure that the CIDR block falls within the IP address range of the VPC and doesn't overlap with other subnets
* Select the desired availability zone for the subnet. It's recommended to create subnets in multiple availability zones for high availability and fault tolerance.
* Configure the subnet's route table:
* Choose an existing route table or create a new one for the subnet. To make the subnet public, associate it with a route table that has a route to an internet gateway.

Verify the details of the subnet, including the VPC, CIDR block, availability zone, route table, and NACL settings.



**STAGE 2:**



Create an Auto Scaling Group:

An Auto Scaling Group manages a group of EC2 instances and automatically adjusts the number of instances based on scaling policies and health checks. ASG maintains a desired capacity of instances, scales up or down based on defined conditions, and replaces unhealthy instances to ensure the desired state of the application.

Steps to create an Auto Scaling Group:

1. Open the Amazon EC2 service in the AWS Management Console.

2. Click on "Auto Scaling Groups" in the left-hand navigation pane.

3. Click on the "Create Auto Scaling Group" button.

4. Configure the Auto Scaling Group:

- Provide a name for your Auto Scaling Group.

- Select the appropriate launch template or launch configuration.

- Specify the desired capacity, minimum and maximum instances, and the subnet(s) for your instances.

5. Configure scaling policies(optional)

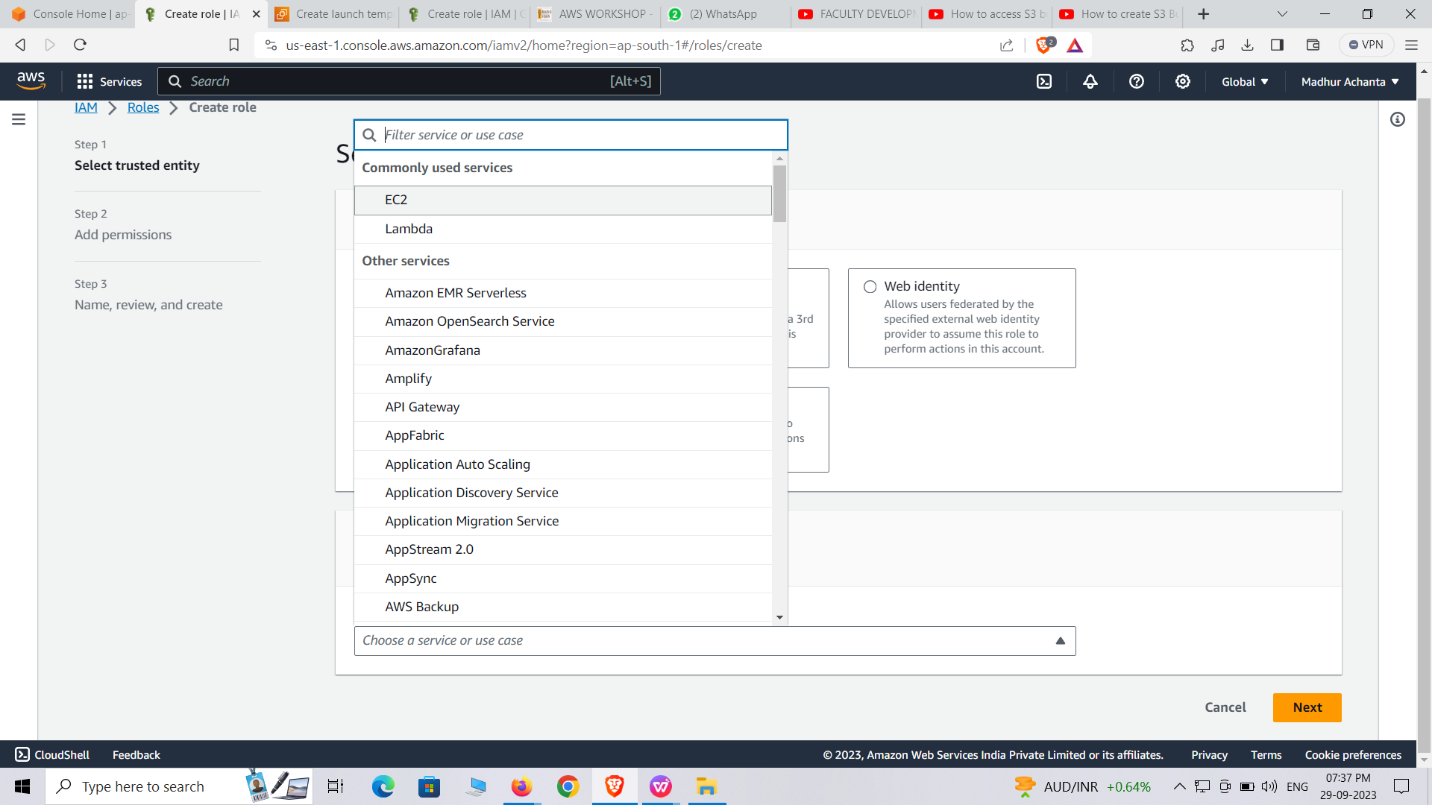
6. Configure notifications (optional)

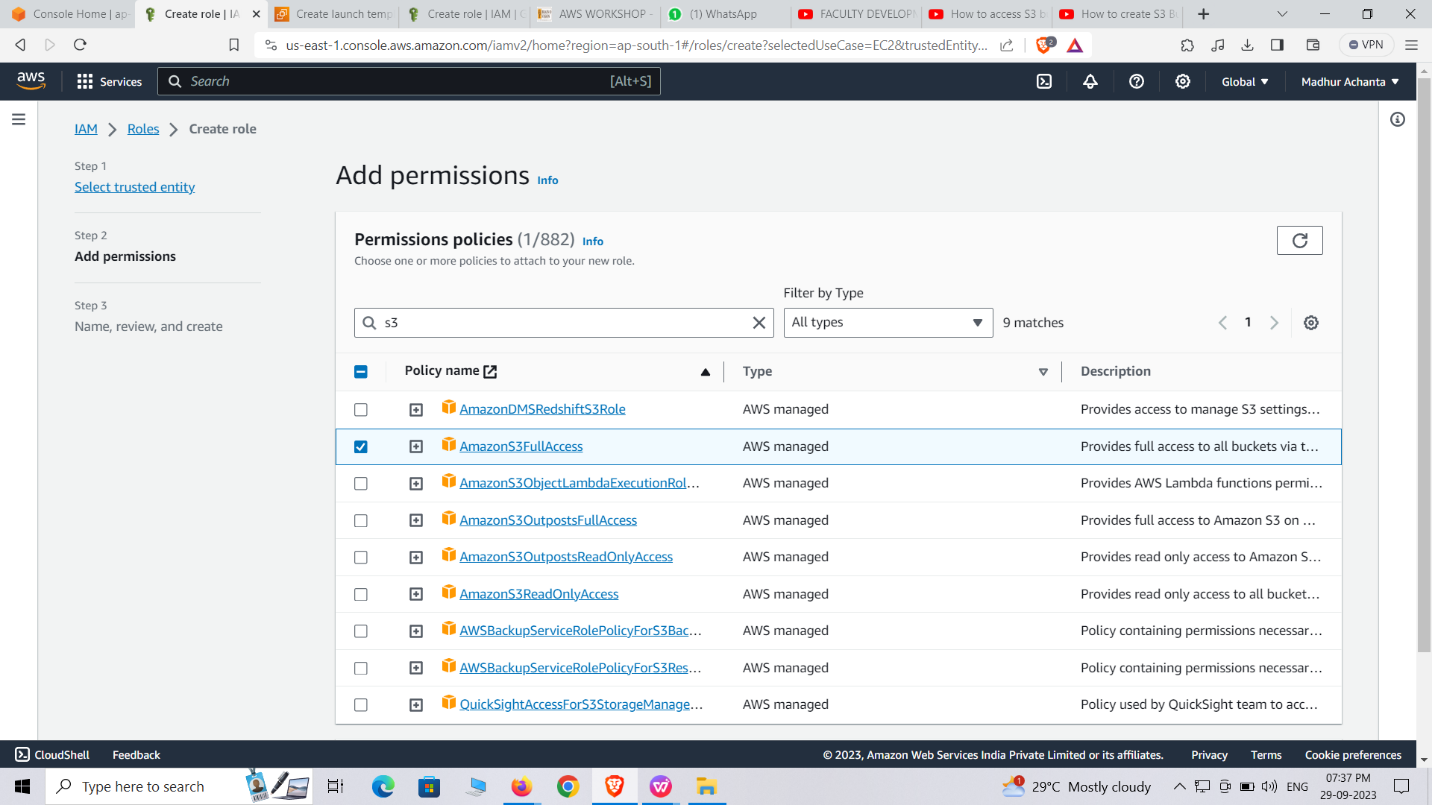
- Set up email notifications for scaling events or policy breaches if desired.

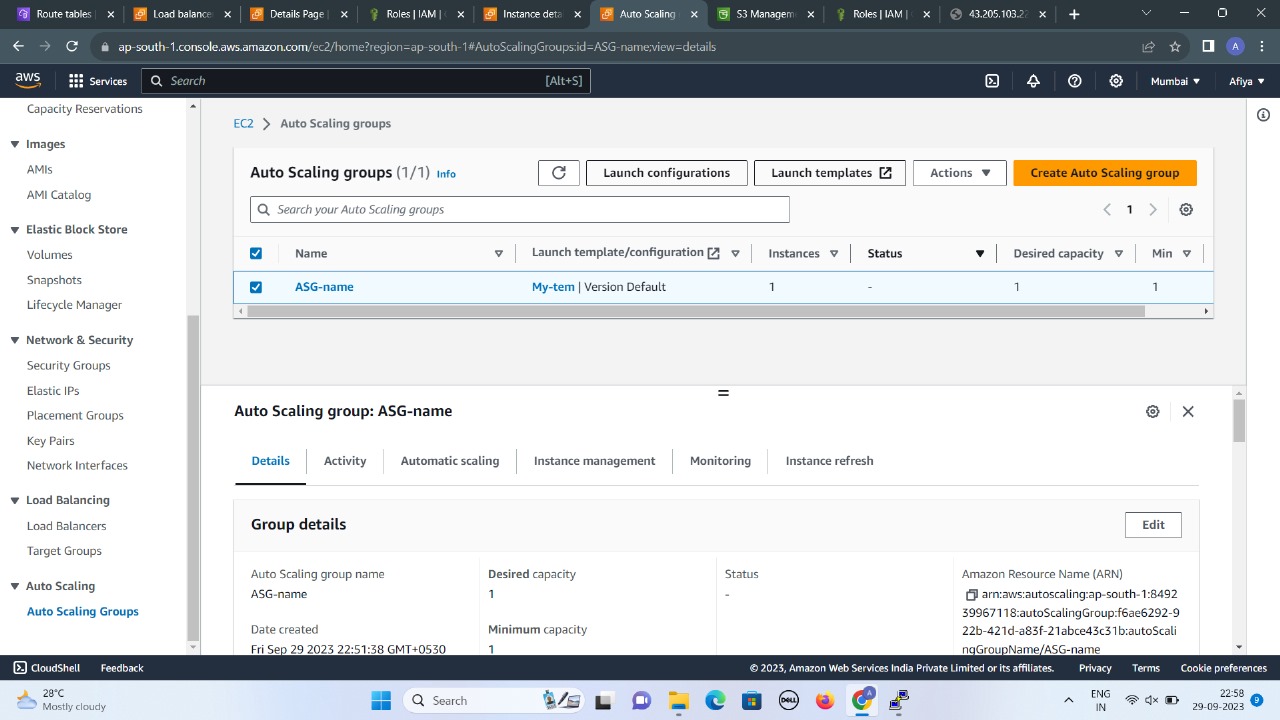
8. Configure tags (optional):

- Add any tags to label and categorize your Auto Scaling Group.

9. Review the configuration and click on the "Create Auto Scaling Group" button.







Create an **Classic Load Balancer** :

A Classic Load Balancer operates at the transport layer (Layer 4) and routes traffic based on network information such as IP addresses and ports.

It performs health checks on registered instances and automatically distributes incoming traffic across those instances to ensure efficient utilization and fault tolerance.

Steps to create a Classic Load Balancer:

Open the Amazon EC2 service in the AWS Management Console.

Click on "Load Balancers" in the left-hand navigation pane.

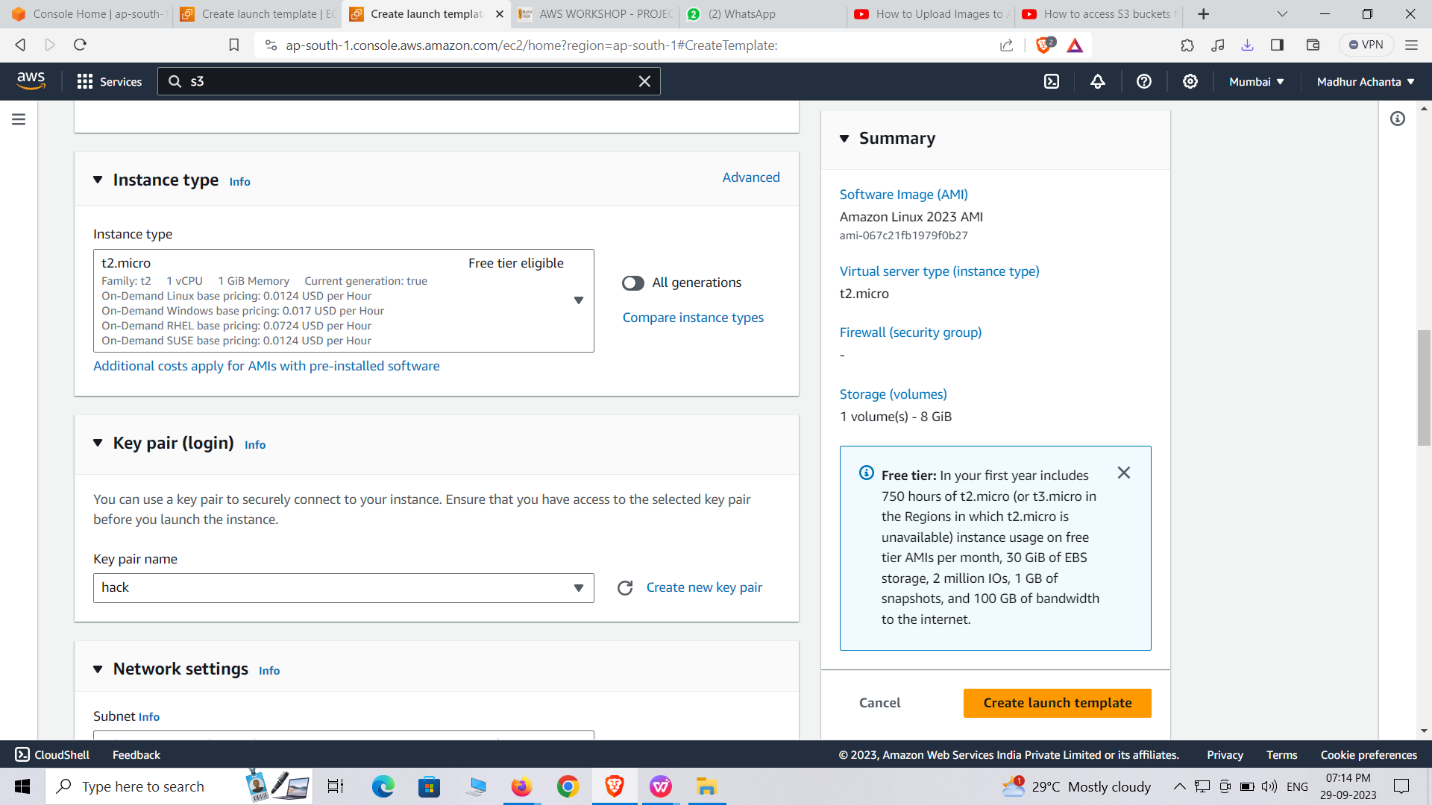
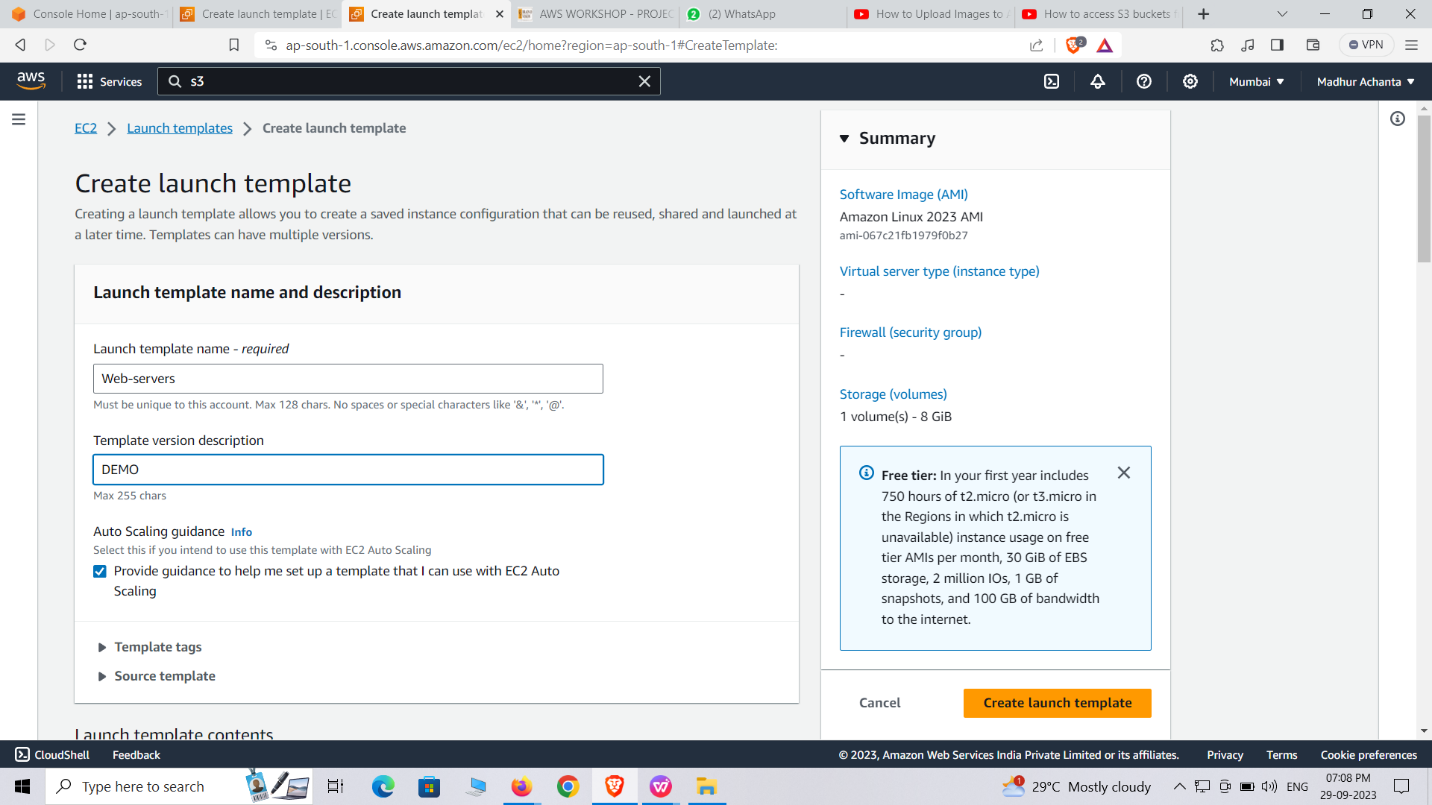
Click on the "Create Load Balancer" button.

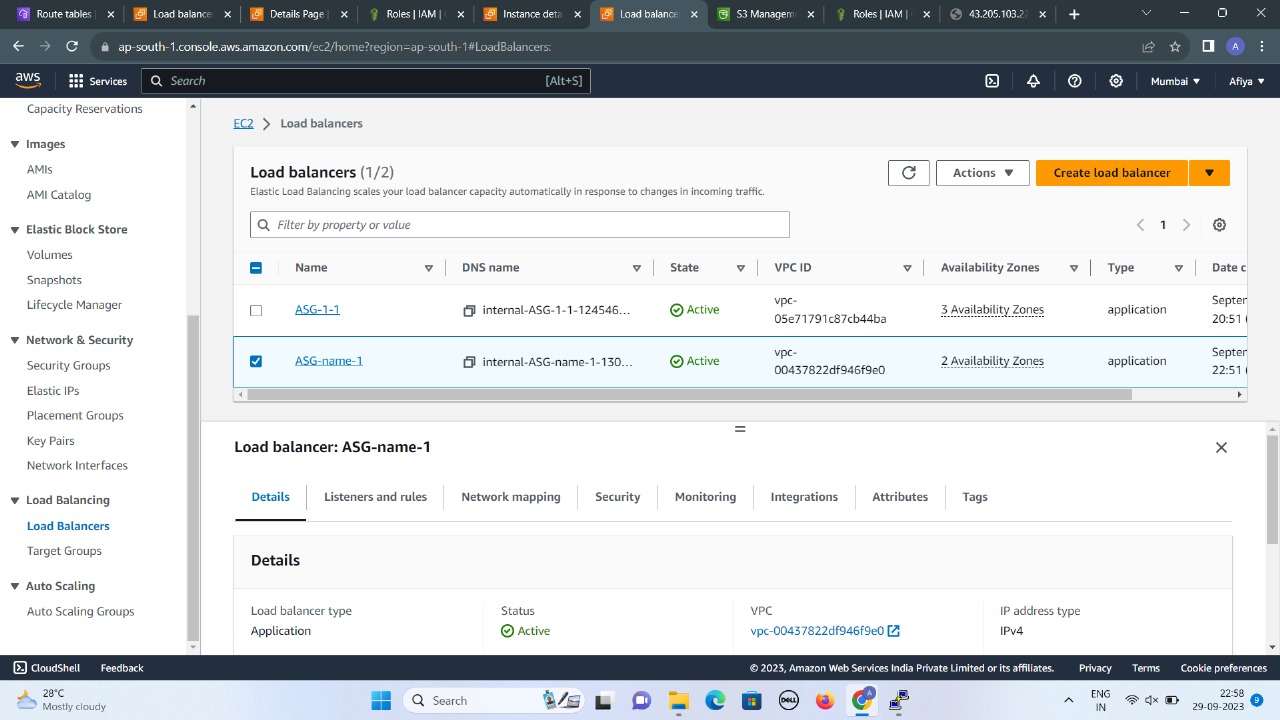
Choose "Classic Load Balancer" as the load balancer type.

Configure the basic settings:

Provide a name for your load balancer.

Select the appropriate VPC or the EC2-Classic network.Define the health check parameters to monitor the health of the instances.Review the configuration and click on the "Create" button.





Configure the Auto Scaling Group with the Load Balancer:

* In the EC2 Auto Scaling console, select your Auto Scaling group.
* Click on the "Edit" button in the "Details" tab.
* In the "Load balancing" section, select "Enable" for "Classic Load Balancer" or "Target groups" for "Application Load Balancer" or "Network Load Balancer."
* Select the load balancer and target groups you created in the previous step.
* Save your changes.

After following the above steps, we can connect to the webserver.

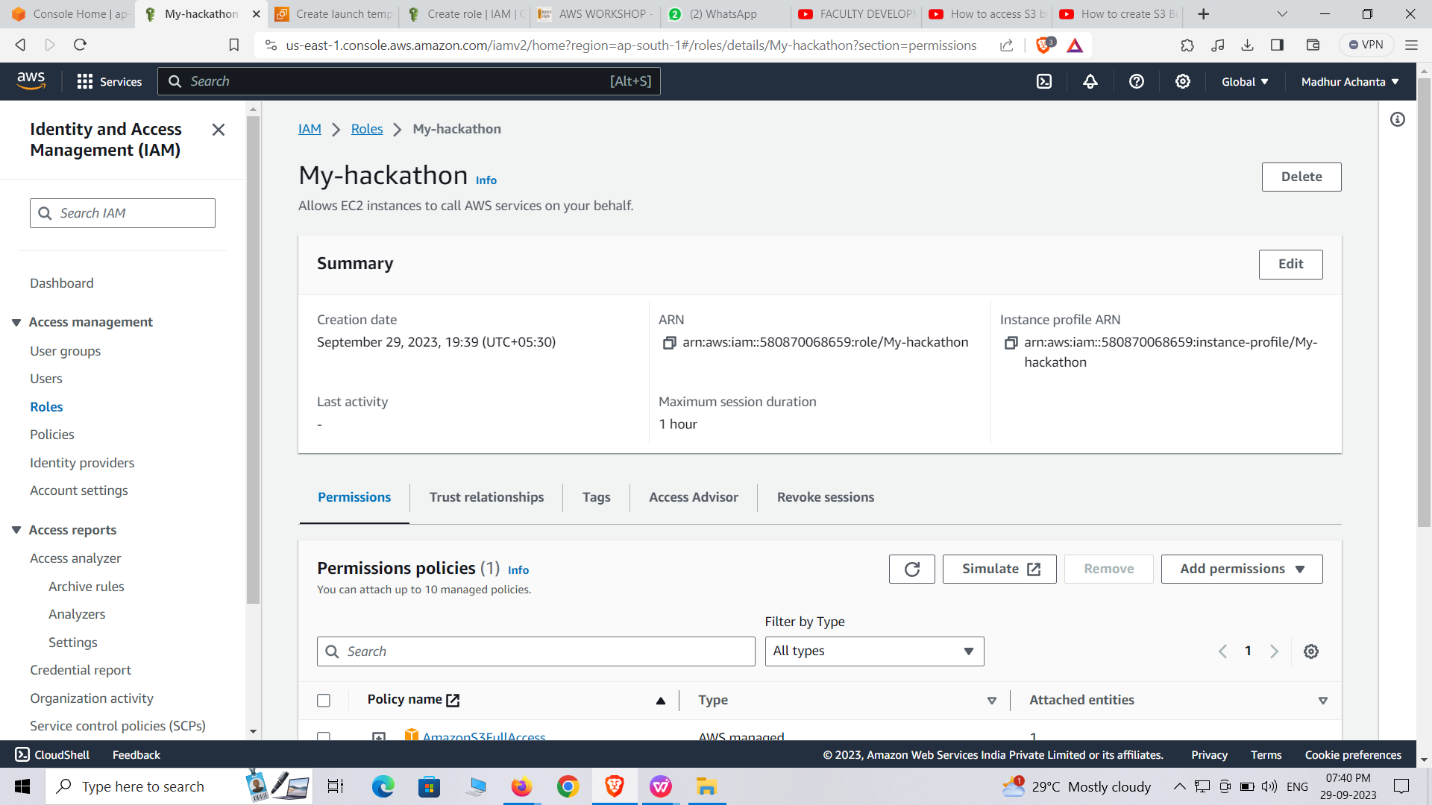
**Webserver**:

A web server in AWS is a virtual machine or container running web server software to host websites and web applications in the cloud.

To set up a web server in AWS:

1. Launch an EC2 instance, choose an appropriate Amazon Machine Image (AMI) with pre-installed web server software, configure security groups, and assign an Elastic IP if needed.

2. Connect to the instance, install any additional dependencies or modules, configure the web server software (e.g., Apache, Nginx), upload website files, and configure DNS or load balancer settings to direct traffic to the web server.



IAM Dashboard

\* To create an IAM user (console)On the Console Home page, select the IAM service.

\*In the navigation pane, select Users and then select Add users. On the Specify user details page, under User details, in User name, enter the name for the new user. This is their sign-in name for AWS.

# Steps to create S3 Bucket

1. In the left navigation pane, choose Buckets.
2. Choose Create bucket. ...
3. For Bucket name, enter a name for your bucket. ...
4. For Region, choose the AWS Region where you want the bucket to reside.

# Step 1: Create your S3 bucket

# After you sign up for AWS, you're ready to create a bucket in Amazon S3 using the AWS Management Console. Every object in Amazon S3 is stored in a bucket. Before you can store data in Amazon S3, you must create a bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.

The **Create bucket** page opens.

1. For **Bucket name**, enter a name for your bucket.

The bucket name must:

* + Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: aws (Standard Regions), aws-cn (China Regions), and aws-us-gov (AWS GovCloud (US) Regions).
  + Be between 3 and 63 characters long.
  + Consist only of lowercase letters, numbers, dots (.), and hyphens (-). For best compatibility, we recommend that you avoid using dots (.) in bucket names, except for buckets that are used only for static website hosting.
  + Begin and end with a letter or number.

After you create the bucket, you cannot change its name. For more information about naming buckets,

1. For **Region**, choose the AWS Region where you want the bucket to reside.

To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see [AWS service endpoints](https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region) in the Amazon Web Services General Reference.

1. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

###### ACLs disabled

* + **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually. For more information, see [Controlling ownership of objects and disabling ACLs for your bucket](https://docs.aws.amazon.com/AmazonS3/latest/userguide/about-object-ownership.html).

###### ACLs enabled

* + **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the bucket-owner-full-control canned ACL.

If you apply the **Bucket owner preferred** setting, to require all Amazon S3 uploads to include the bucket-owner-full-control canned ACL, you can [add a bucket policy](https://docs.aws.amazon.com/AmazonS3/latest/userguide/ensure-object-ownership.html#ensure-object-ownership-bucket-policy) that allows only object uploads that use this ACL.

* + **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

1. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.
2. By default, all four Block Public Access settings are enabled. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see [Blocking public access to your Amazon S3 storage](https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-control-block-public-access.html).
3. (Optional) Under **Bucket Versioning**, you can choose if you wish to keep variants of objects in your bucket. For more information about versioning, see [Using versioning in S3 buckets](https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html).

To disable or enable versioning on your bucket, choose either **Disable** or **Enable**.

1. (Optional) Under **Tags**, you can choose to add tags to your bucket. Tags are key-value pairs used to categorize storage.

To add a bucket tag, enter a **Key** and optionally a **Value** and choose **Add Tag**.

1. Under **Default encryption**, choose **Edit**.
2. To configure default encryption, under **Encryption type**, choose one of the following:
   * **Amazon S3 managed key (SSE-S3)**
   * **AWS Key Management Service key (SSE-KMS)**
3. Buckets and new objects are encrypted with server-side encryption with an **Amazon S3 managed key** as the base level of encryption configuration. For more information about default encryption, see [Setting default server-side encryption behavior for Amazon S3 buckets](https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-encryption.html).

For more information about using Amazon S3 server-side encryption to encrypt your data, see [Using server-side encryption with Amazon S3 managed keys (SSE-S3)](https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingServerSideEncryption.html).

1. If you chose **AWS Key Management Service key (SSE-KMS)**, do the following:
   * Under **AWS KMS key**, specify your KMS key in one of the following ways:
     1. To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

Both the AWS managed key (aws/s3) and your customer managed keys appear in this list. For more information about customer managed keys, see [Customer keys and AWS keys](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#key-mgmt) in the AWS Key Management Service Developer Guide.

* + 1. To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
    2. To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

For more information about creating an AWS KMS key, see [Creating keys](https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html) in the AWS Key Management Service Developer Guide.

* 1. For more information about creating an AWS KMS key, see [Creating keys](https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html) in the *AWS Key Management Service Developer Guide*. For more information about using AWS KMS with Amazon S3, see [Using server-side encryption with AWS KMS keys (SSE-KMS)](https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingKMSEncryption.html).
  2. When you configure your bucket to use default encryption with SSE-KMS, you can also enable S3 Bucket Keys. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see [Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys](https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-key.html).

To use S3 Bucket Keys, under **Bucket Key**, choose **Enable**.

1. (Optional) If you want to enable S3 Object Lock, do the following:
   1. Choose **Advanced settings**.
2. If you want to enable Object Lock, choose **Enable**, read the warning that appears, and acknowledge it.

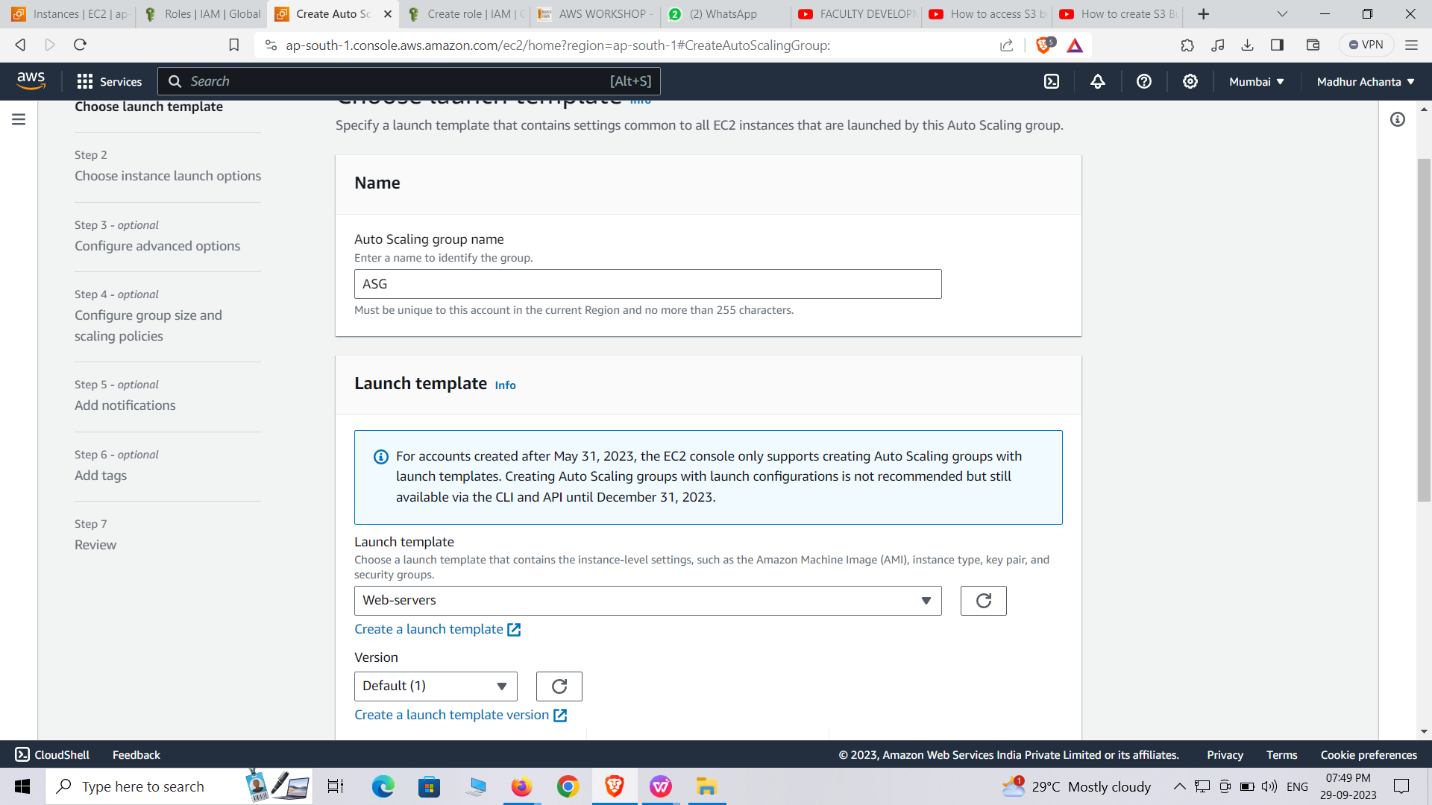
For more information, see [Using S3 Object Lock](https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html).

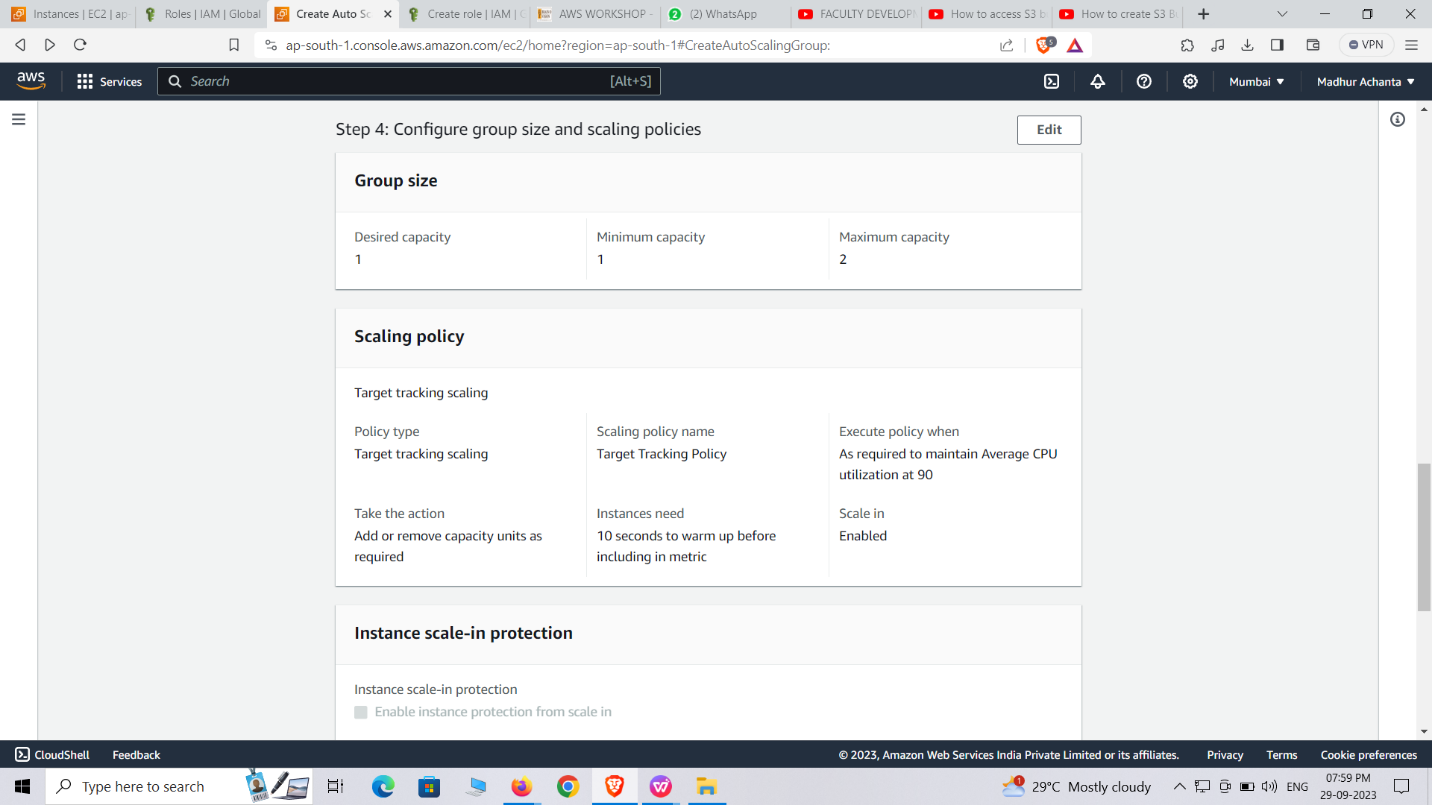
1. Choose **Create bucket**.

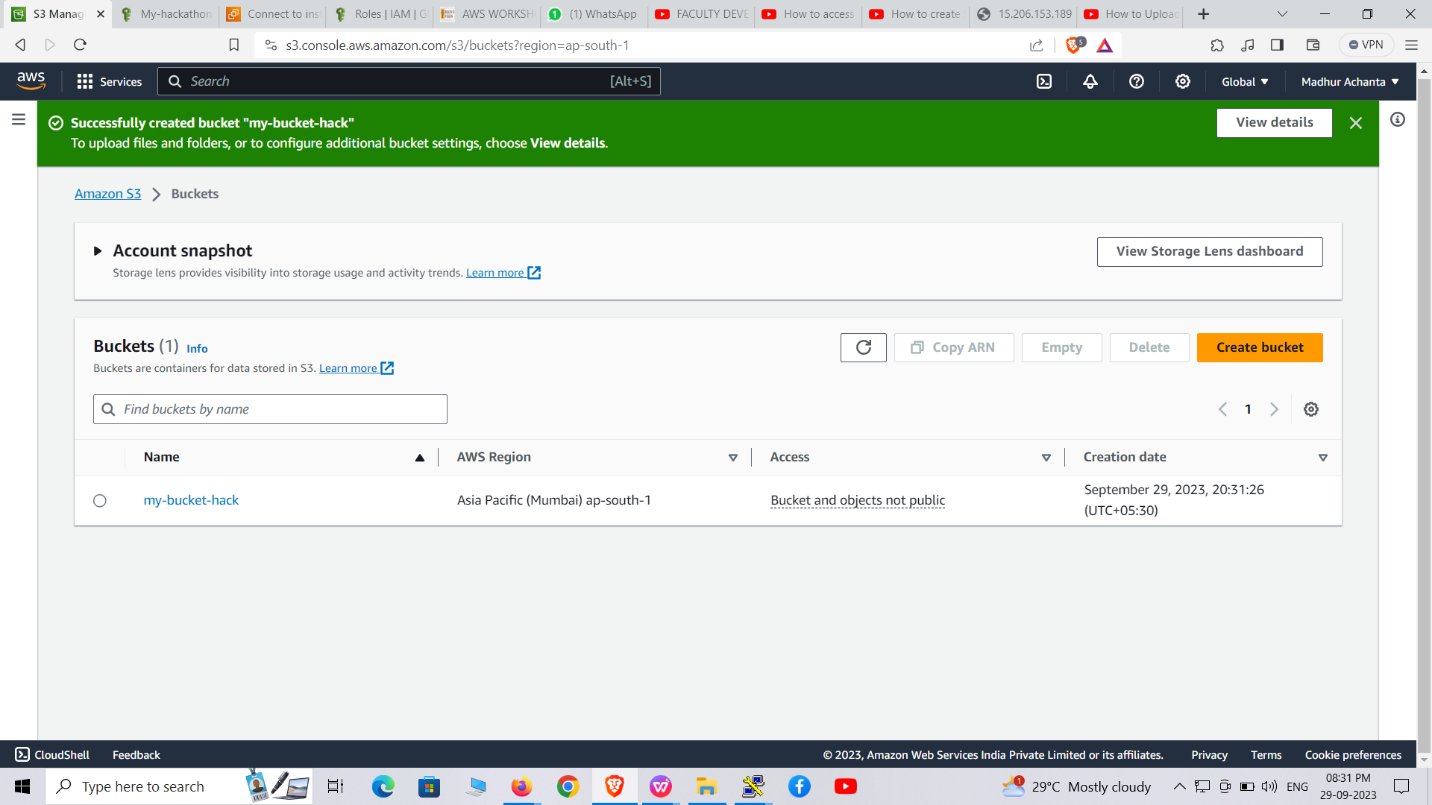
You've created a bucket in Amazon S3.

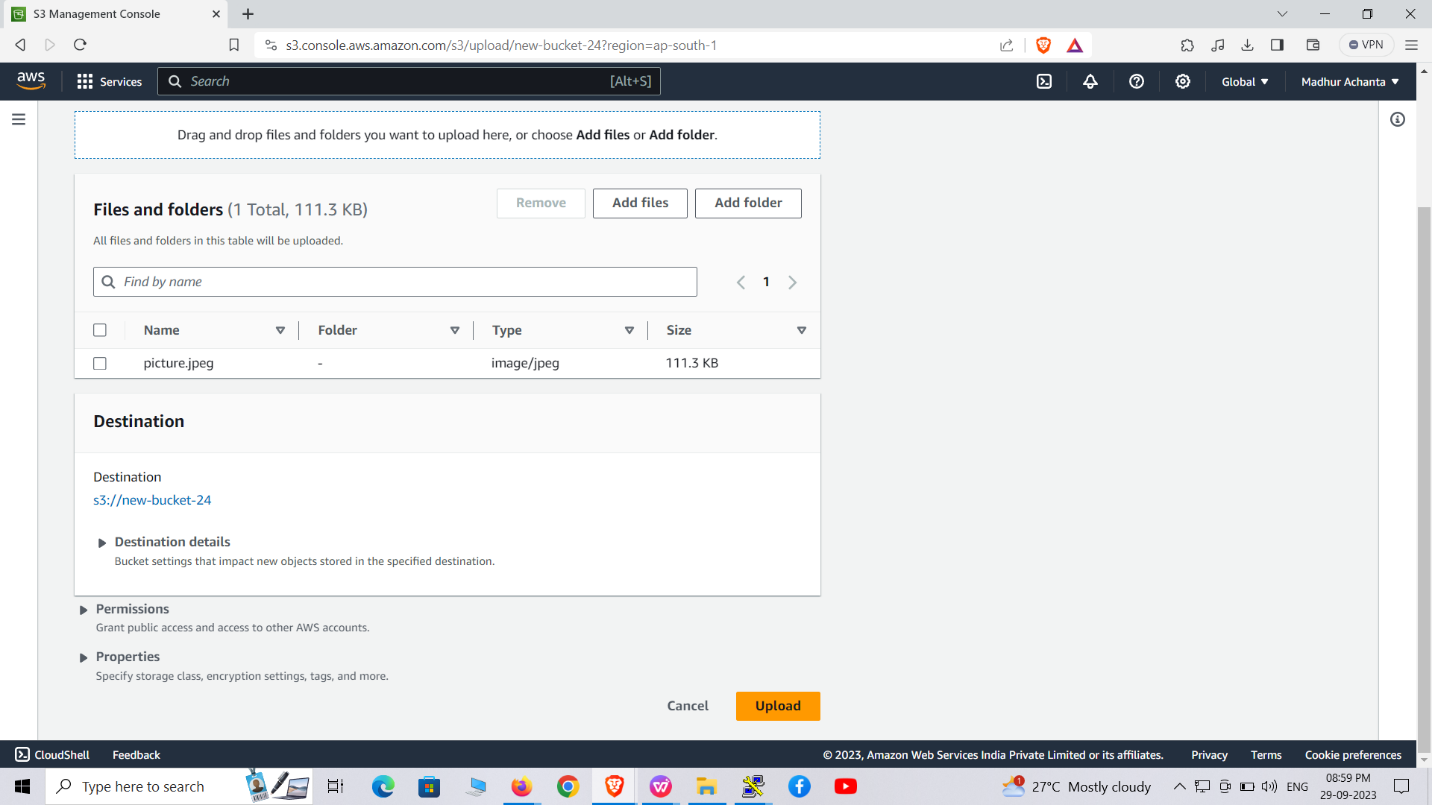
###### Next step

To add an object to your bucket, see [Step 2: Upload an object to your bucket](https://docs.aws.amazon.com/AmazonS3/latest/userguide/uploading-an-object-bucket.html).

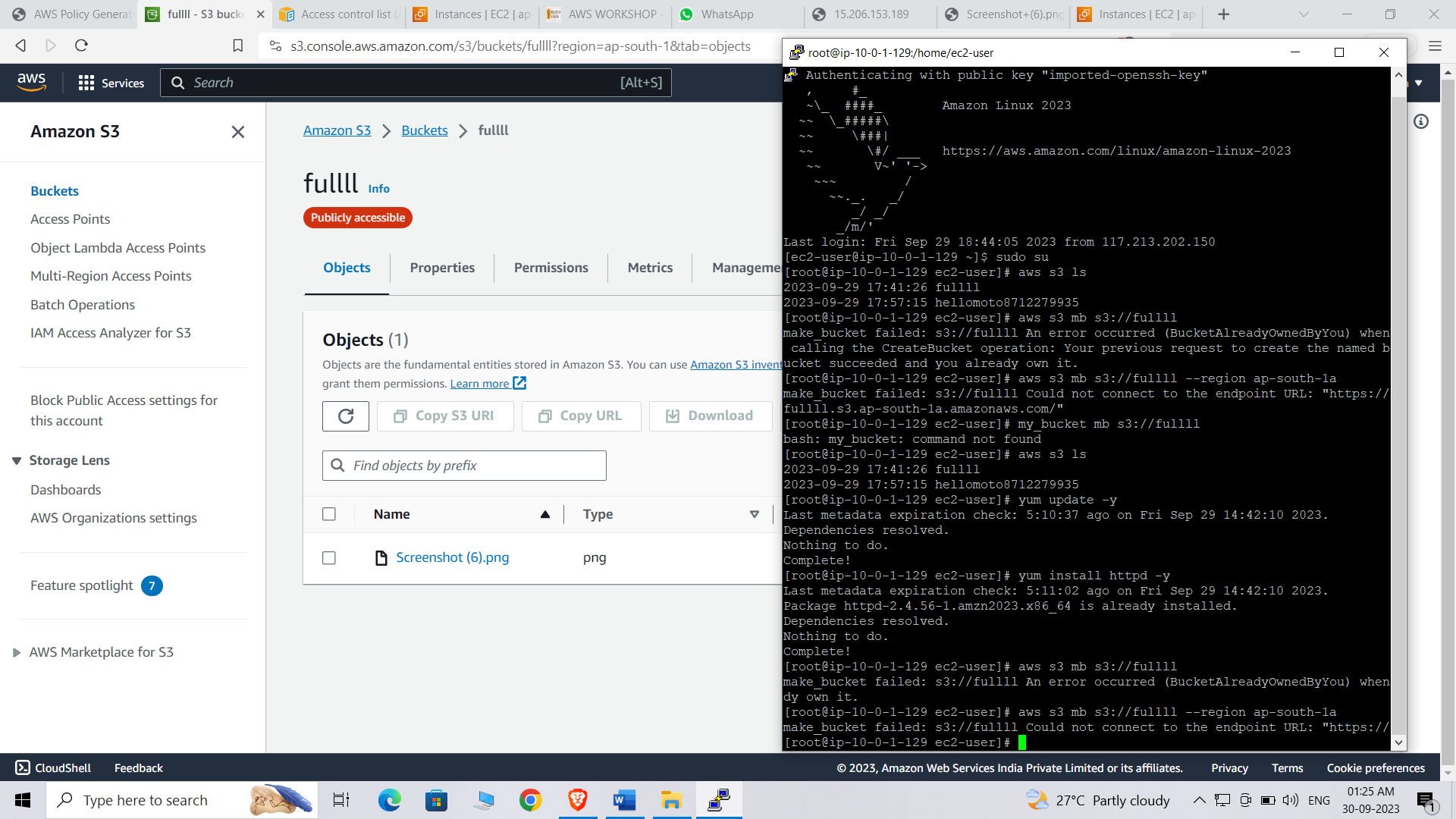
****

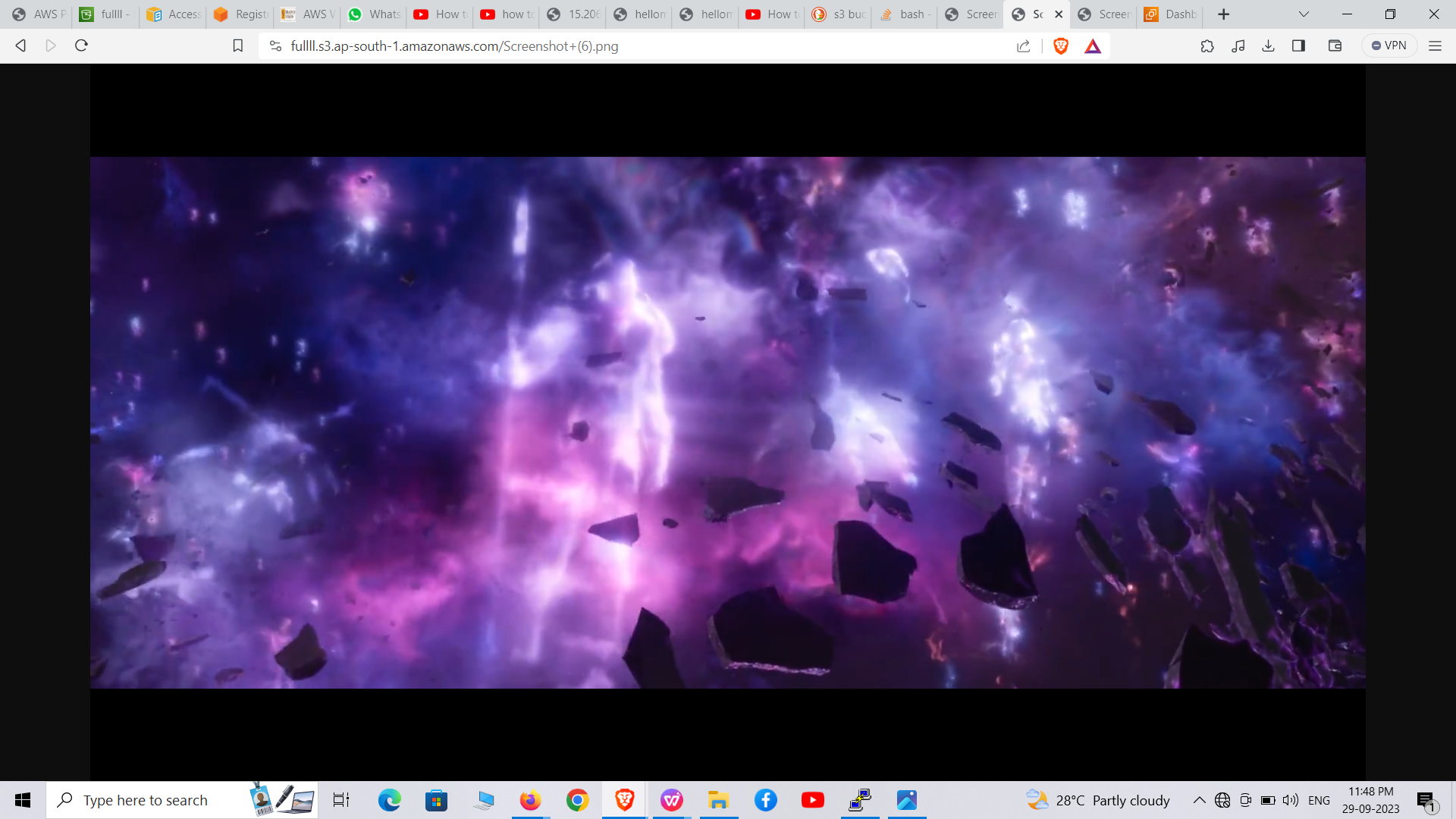
****



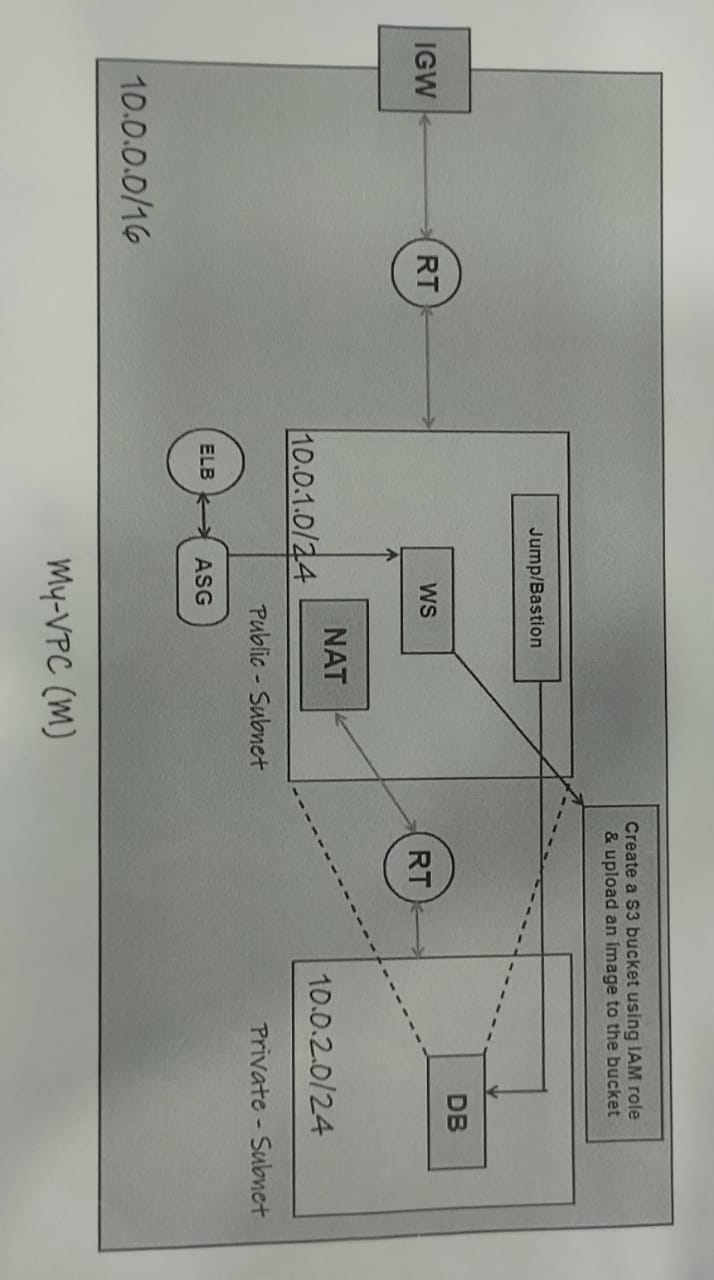


**DISPLAYING IMAGE:**

****

****

**STAGE 3:**

****

To connect a jump bastion instance to a database instance in AWS, you can follow these steps:

1. Launch a Jump Bastion Instance:

- Go to the EC2 console in the AWS Management Console.

- Click on "Launch Instances" to start the instance creation wizard.

- Choose an appropriate Amazon Machine Image (AMI) for the jump bastion instance, such as an Amazon Linux or Ubuntu Server.

- Select an instance type that suits your needs.

- Configure the instance details, including network settings, security groups, and storage options.

- Review your configuration and launch the instance.

2. Configure Security Groups:

- While launching the instance or after the launch, configure the security groups for both the jump bastion instance and the database instance.

- Create a new security group or choose an existing one for each instance.

- Define inbound and outbound rules to allow the necessary network traffic.

- For the jump bastion instance, allow SSH access (port 22) from your IP address or a specific range of IP addresses.

- For the database instance, allow the required ports for database connectivity (e.g., port 3306 for MySQL, port 5432 for PostgreSQL).

3. Connect to the Jump Bastion Instance:

- Retrieve the public IP address or DNS name of the jump bastion instance from the EC2 console.

- Use an SSH client (e.g., Terminal on macOS/Linux, PuTTY on Windows) to connect to the jump bastion instance.

- Provide the necessary credentials (SSH key pair or username/password) to establish the connection.

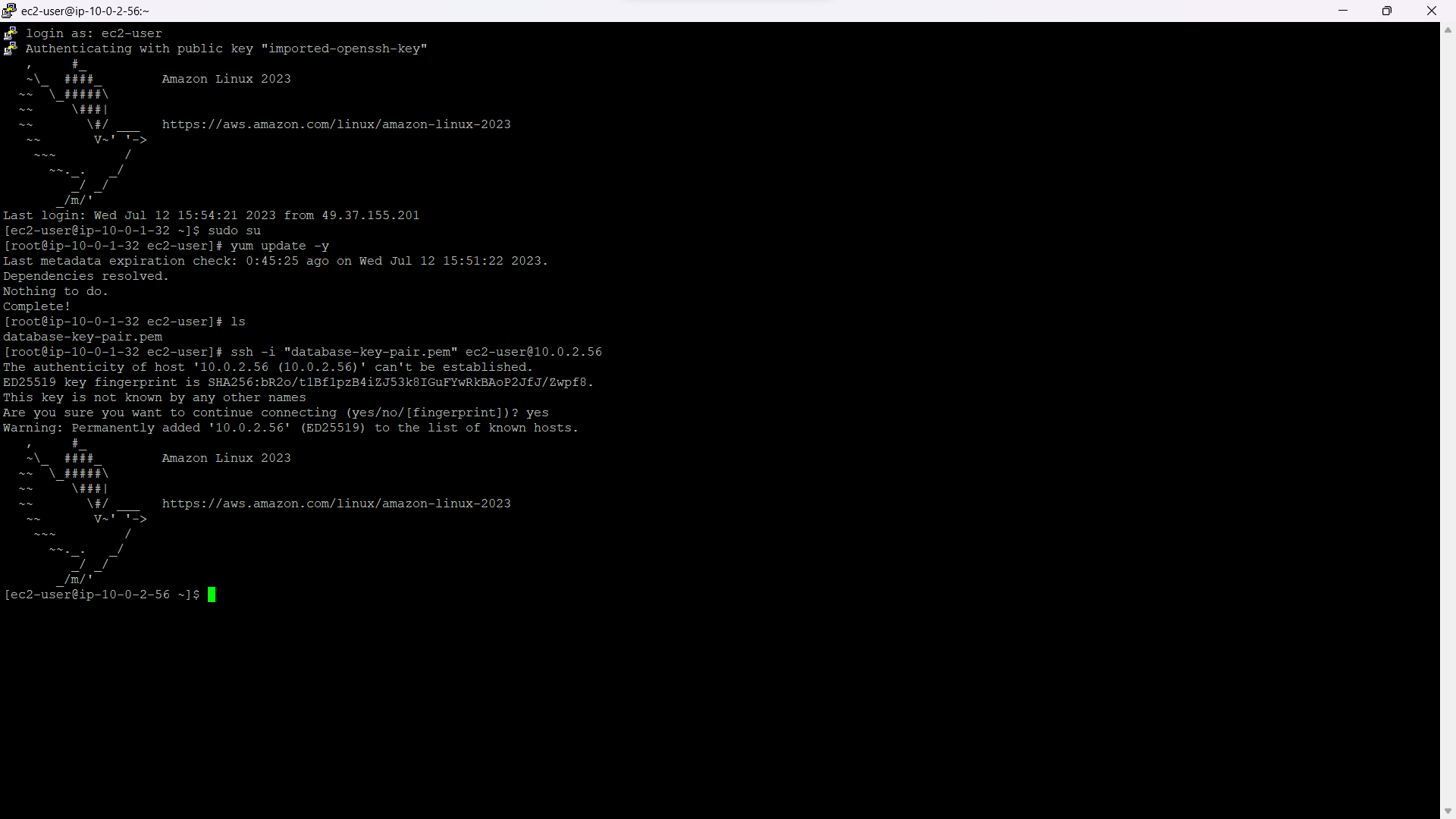
4. Connect to the Database Instance:

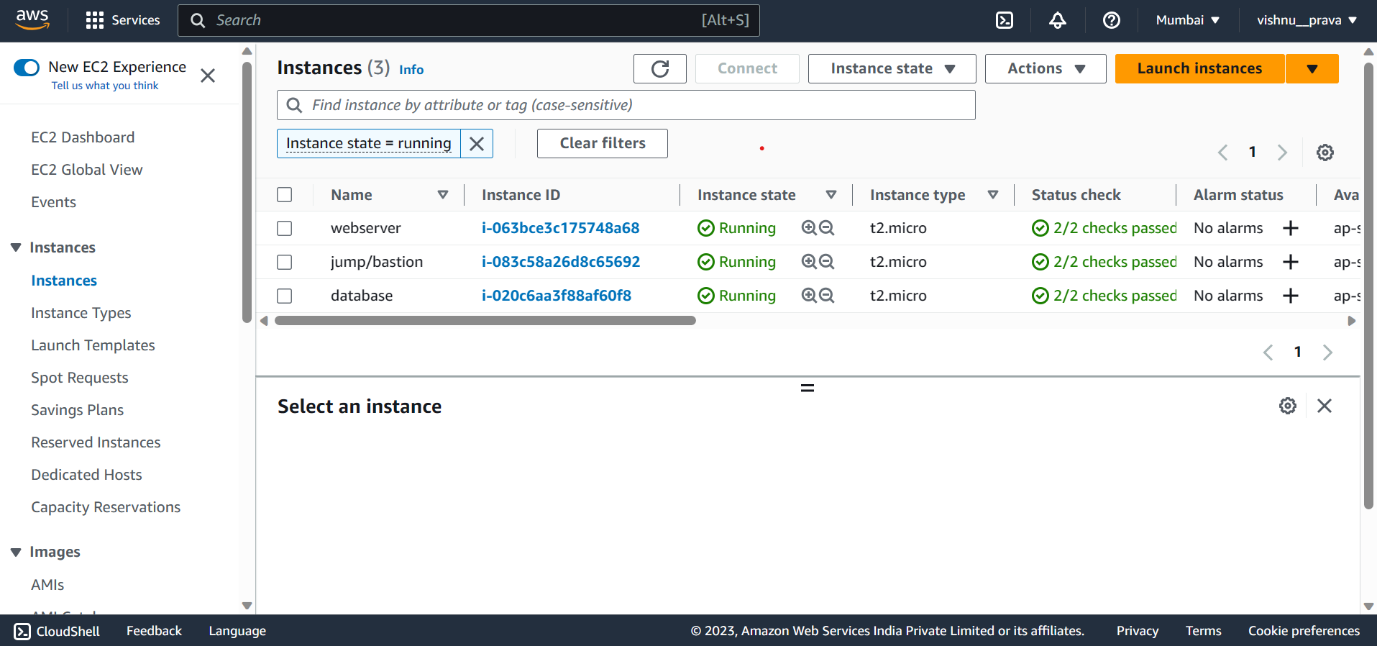
- From the jump bastion instance, you can establish a secure connection to the database instance.

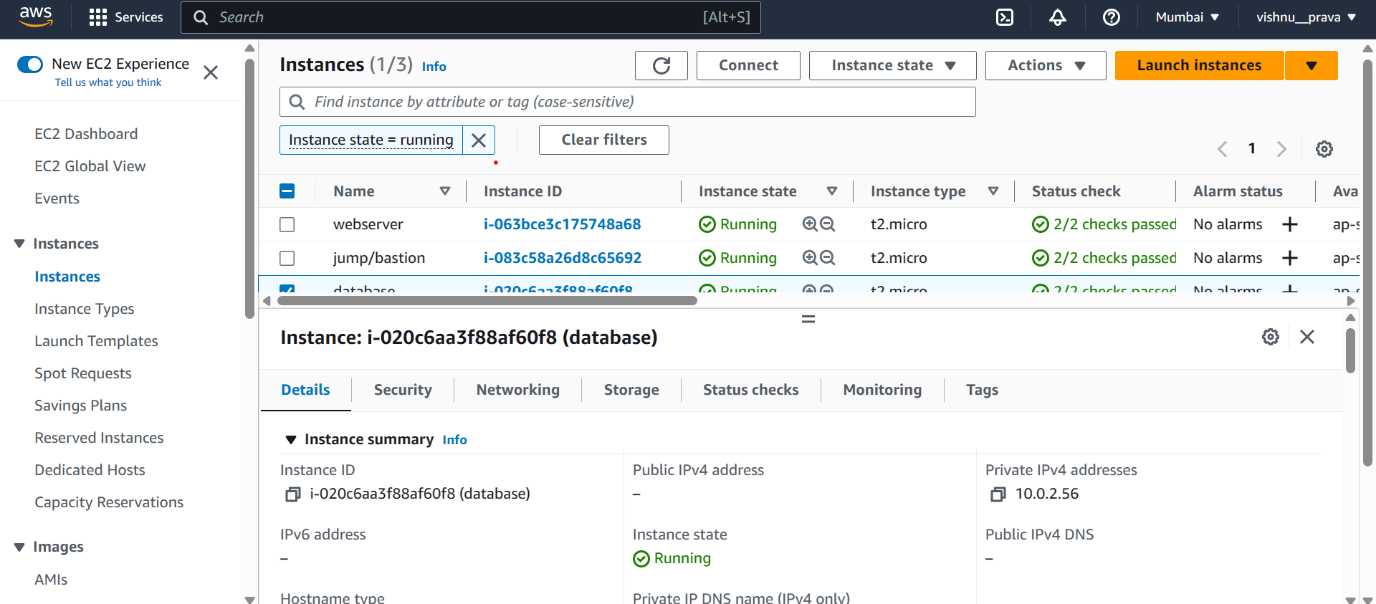
- Use the appropriate client or command-line tool for your database (e.g., MySQL Workbench for MySQL, psql for PostgreSQL).

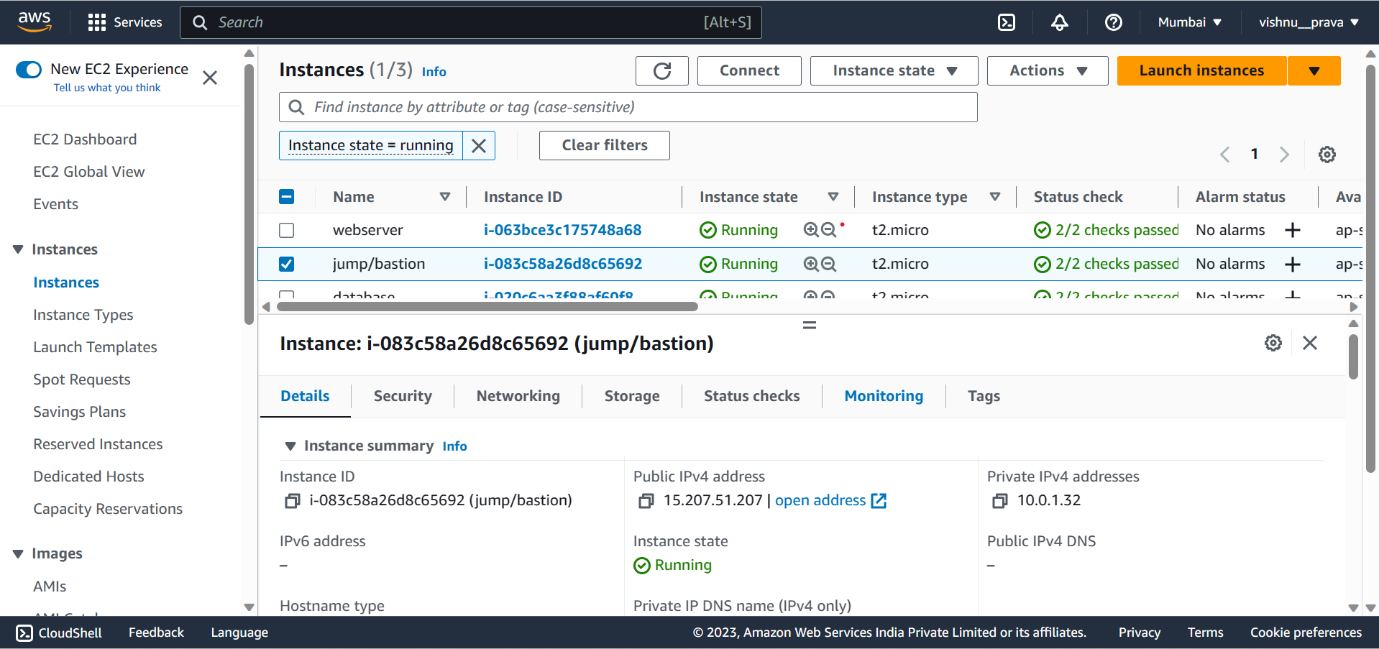
- Provide the necessary connection details such as the database host, port, username, and password.

- Connect to the database instance.

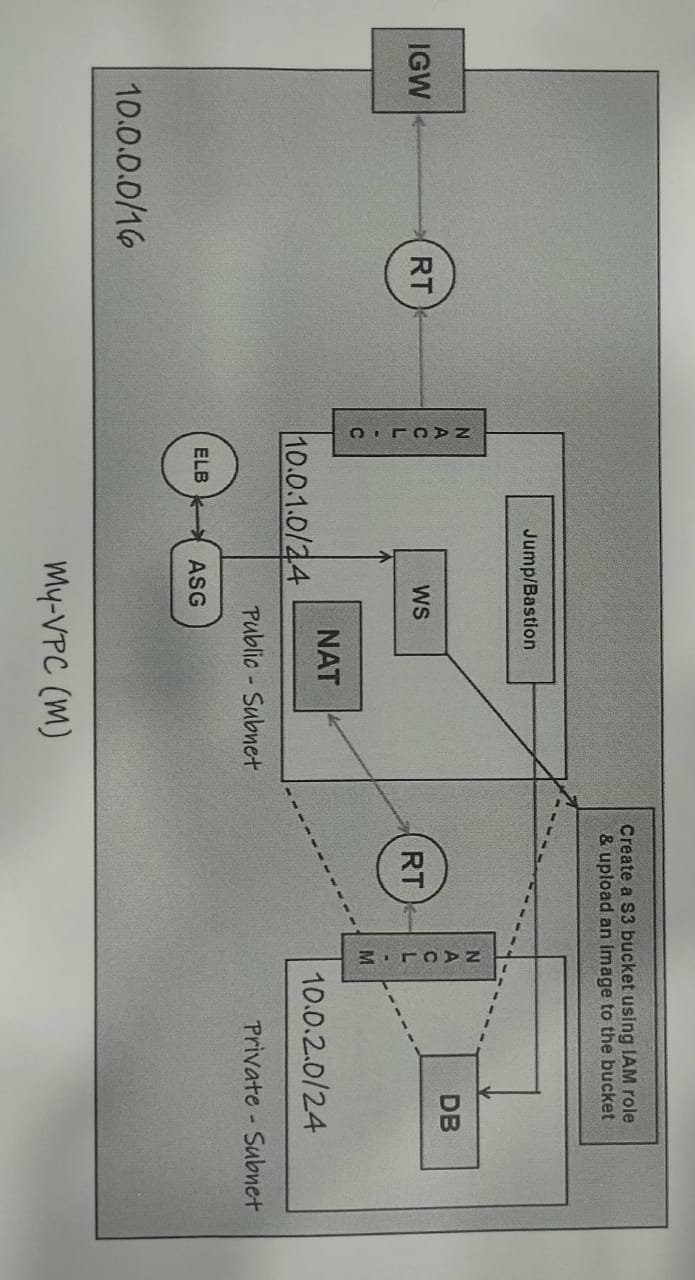








**Stage-4:**

****

In this stage we created nacl for subnets.

In AWS, Network Access Control Lists (NACLs) are an optional layer of security that act as stateless firewalls for controlling inbound and outbound traffic at the subnet level. NACLs operate at the subnet level and evaluate rules in a sequential order to determine whether to allow or deny traffic. Each subnet in AWS can be associated with one NACL, and by default, a subnet is associated with the default NACL.

Here are the steps to create and attach a Network Access Control List (NACL) to subnets in AWS:

1. Open the Amazon VPC Console:

- Go to the Amazon VPC console in the AWS Management Console.

2. Create a Network Access Control List (NACL):

- In the VPC dashboard, click on "Network ACLs" in the sidebar.

- Click on "Create Network ACL" and specify a name and the VPC for the NACL.

- Once created, the NACL will have default rules that allow all inbound and outbound traffic.

3. Configure Inbound and Outbound Rules:

- Select the newly created NACL and click on the "Inbound Rules" or "Outbound Rules" tab.

- Click on "Edit" and add or modify the rules according to your requirements.

- Rules can be based on IP addresses, port ranges, protocols, and whether to allow or deny traffic.

- Rules are evaluated in the order they appear, so ensure they are ordered correctly.

4. Associate the NACL with Subnets:

- Click on the "Subnet Associations" tab in the NACL configuration.

- Click on "Edit" and select the subnets you want to associate with the NACL.

- You can choose to associate multiple subnets with the same NACL.

- Make sure to confirm your changes to associate the NACL with the selected subnets.

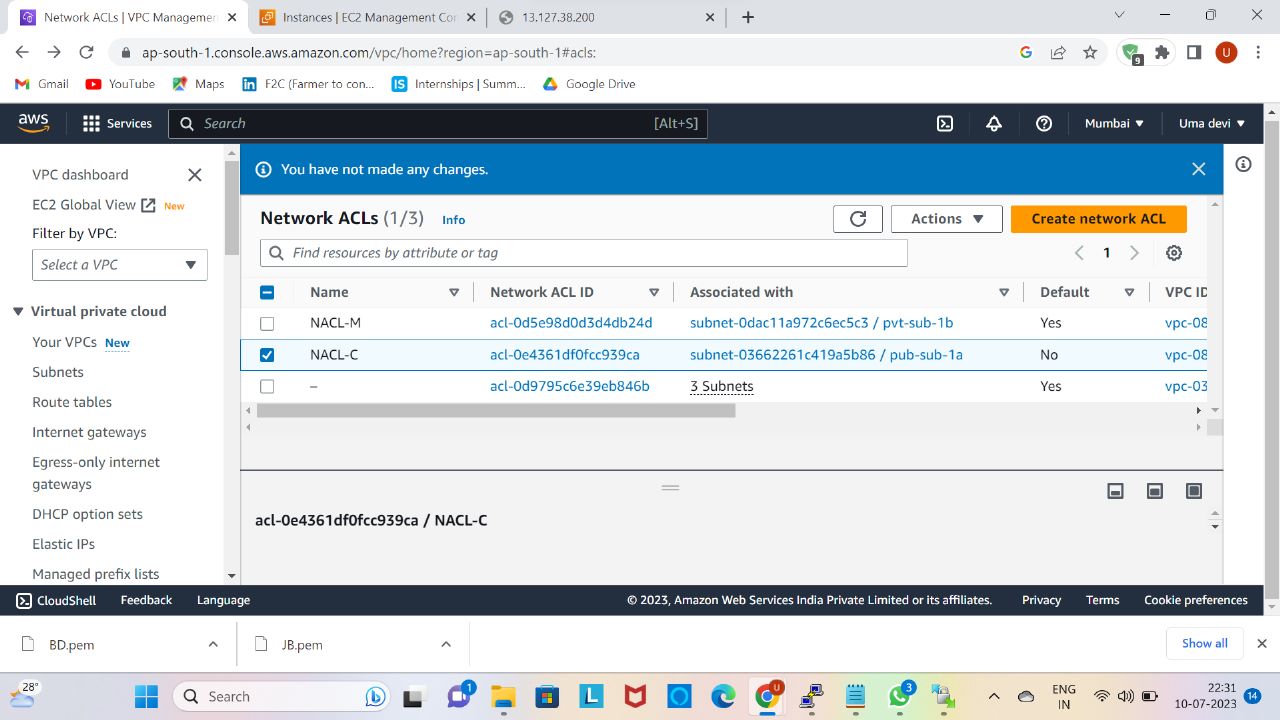
5. Verify and Test:

- Once the NACL is associated with the subnets, it will start governing the traffic.

- Verify that the NACL rules are correctly configured and are allowing or denying traffic as intended.

- Test the connectivity to ensure that the traffic is behaving as expected.

By following these steps, you can create and attach a Network Access Control List (NACL) to subnets in AWS. NACLs provide an additional layer of security by controlling inbound and outbound traffic at the subnet level, allowing you to define fine-grained rules for network traffic in your VPC.



### CONCLUSION:

Participating in the hackathon was a valuable learning experience. We faced several challenges, such as time constraints, technical hurdles, and coordination among team members. However, these challenges helped us develop problem-solving skills, collaboration, and adaptability.

Through the hackathon, we learned the importance of effective planning.Clear delegation of tasks, regular updates, and utilizing collaborative tools improved our productivity and efficiency.

We also recognized the significance of leveraging cloud services, specifically AWS, for rapid prototyping and deployment. Using AWS services like VPC,EC2, ELB, ASG, and CodePipeline streamlined our development process and enabled us to create scalable and resilient solutions.

In future hackathons, we would aim to allocate more time for planning and ideation, allowing for better execution. Additionally, conducting thorough research on AWS services and best practices beforehand would enhance our ability to leverage the platform effectively.

Overall, the hackathon experience provided us with practical insights into solving real-world challenges, sharpened our technical skills, and emphasized the importance of teamwork and innovation.

-----Thank You-----