Experiments for the Course: AWS Cloud Practitioner

Index

9.	Create a new volume from your snapshot.	05/08/25	15/08/25	24
10.	Use the Amazon Data Lifecycle Manager (DLM) to create a scheduled backup policy.	05/08/25	15/08/25	26
11.	Upload another file and select S3 One Zone-IA or S3 Glacier as the storage class.	06/08/25	15/08/25	29
12.	Change the storage class of an existing object to S3 Intelligent-Tiering.	06/08/25	15/08/25	33
13.	Create a new IAM user with programmatic access and S3 read-only permissions.	07/08/25	15/08/25	35
14.	Generate Access Keys for this user. How would they use the AWS CLI to list the contents of your bucket?	07/08/25	15/08/25	41
	Create a bucket policy to allow public	08/08/25	15/08/25	47

15.	read access to all objects in the bucket.			
16.	Enable server access logging for your S3 bucket. Where are the logs stored?	10/08/25	15/08/25	49
17.	Write a simple handler function that returns: "Welcome to VIT BHOPAL!"	11/08/25	15/08/25	52
18.	Launch a Linux EC2 instance into the PublicSubnet.	12/08/25	15/08/25	56
19.	Modify the security group of the public EC2 instance to allow only your IP on port 22.	13/08/25	15/08/25	59
20.	Create and attach an Internet Gateway (IGW) to your VPC.	14/08/25	15/08/25	62

1.Create a new IAM role for EC2 with the policy AmazonS3ReadOnlyAccess.

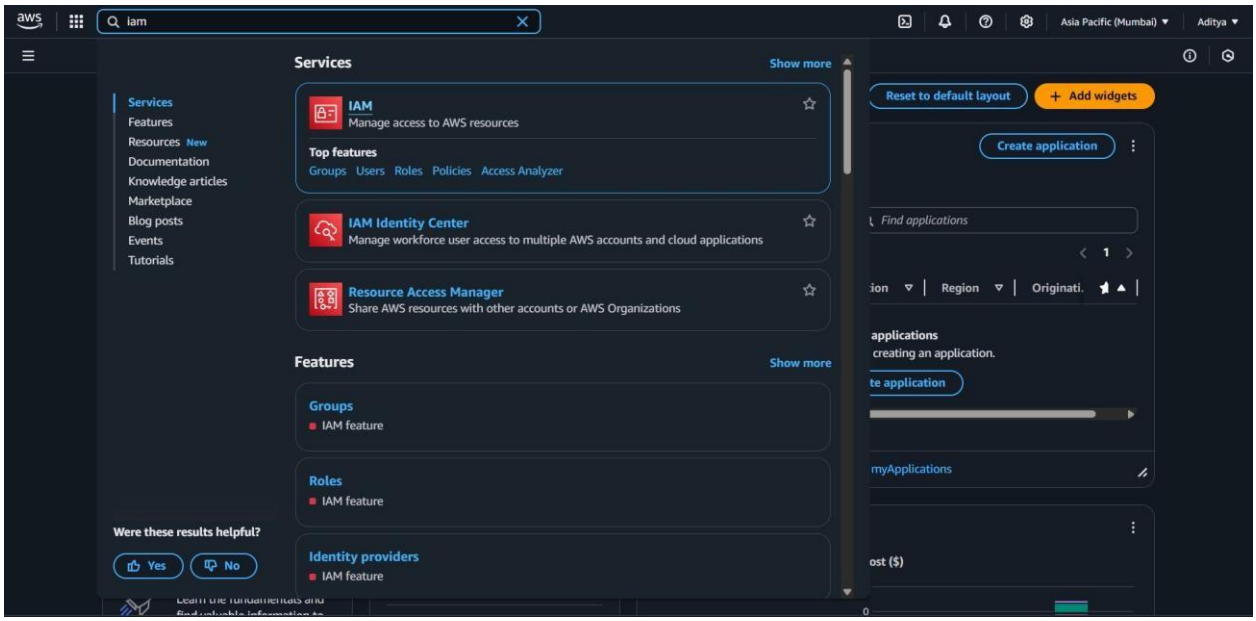
objective:

To create an IAM role that grants an EC2 instance read-only access to Amazon S3 by attaching the built-in AmazonS3ReadOnlyAccess policy.

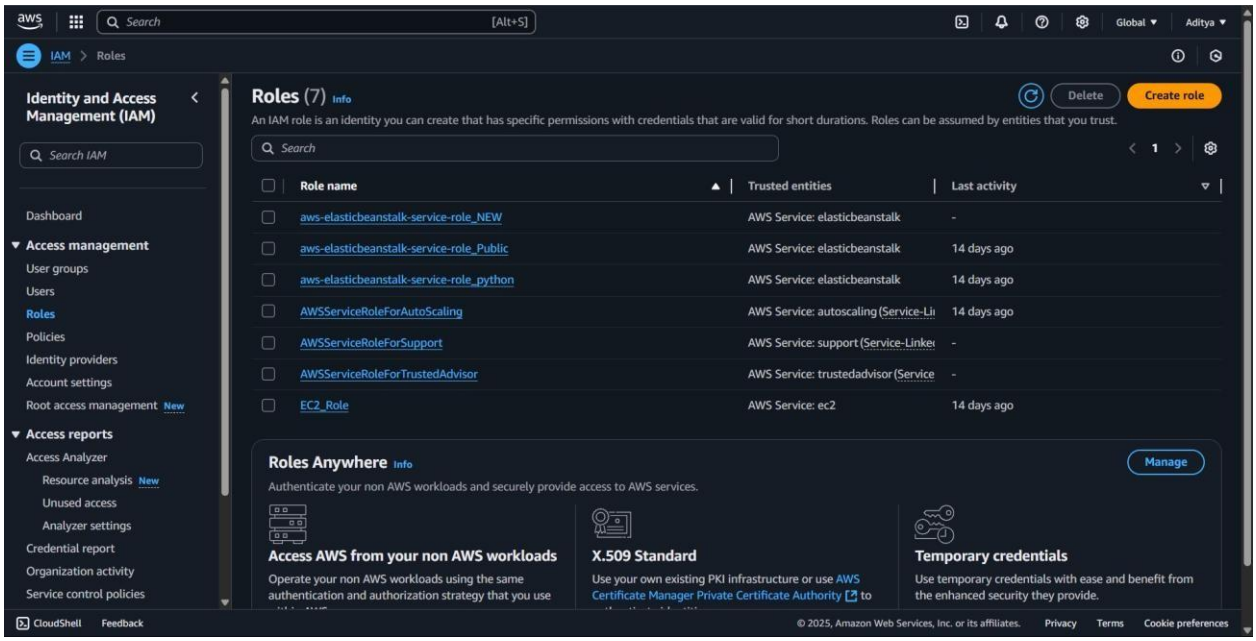
prerequisite:

- An active AWS account.
- Basic understanding of AWS IAM and EC2 services.
- Access permissions to create IAM roles and attach policies.
- AWS Management Console or AWS CLI access.

Step1 : Search for IAM and click on **IAM (Identity and Access Management)**.



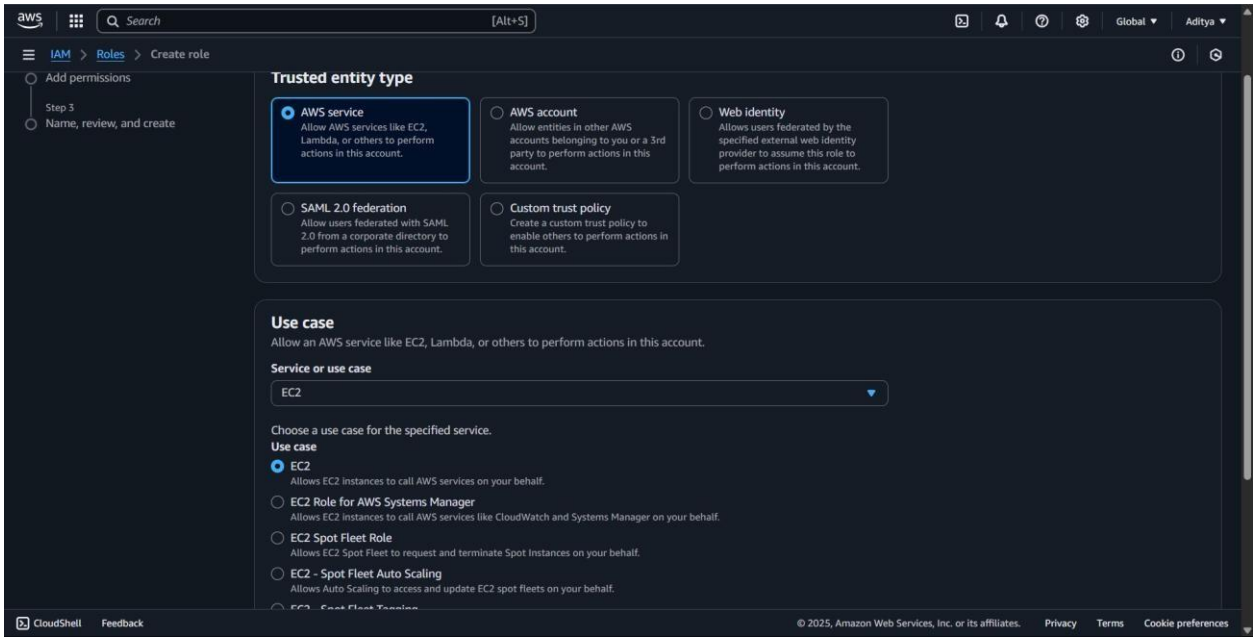
Step2 : In the IAM Dashboard, select **Roles** from the left-hand sidebar.



Step 3 : Click the **“Create role”** button.

Step 4 :Under **"Trusted entity type"**, choose **AWS service**

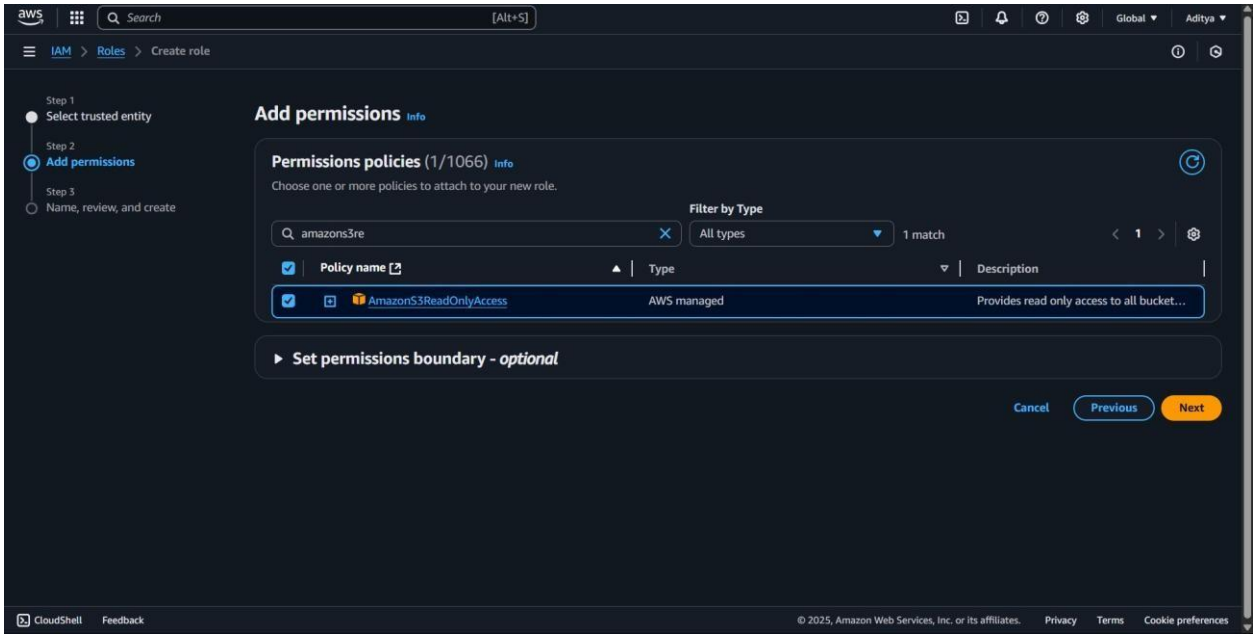
Step5 Under **“Use case”**, select **EC2**.



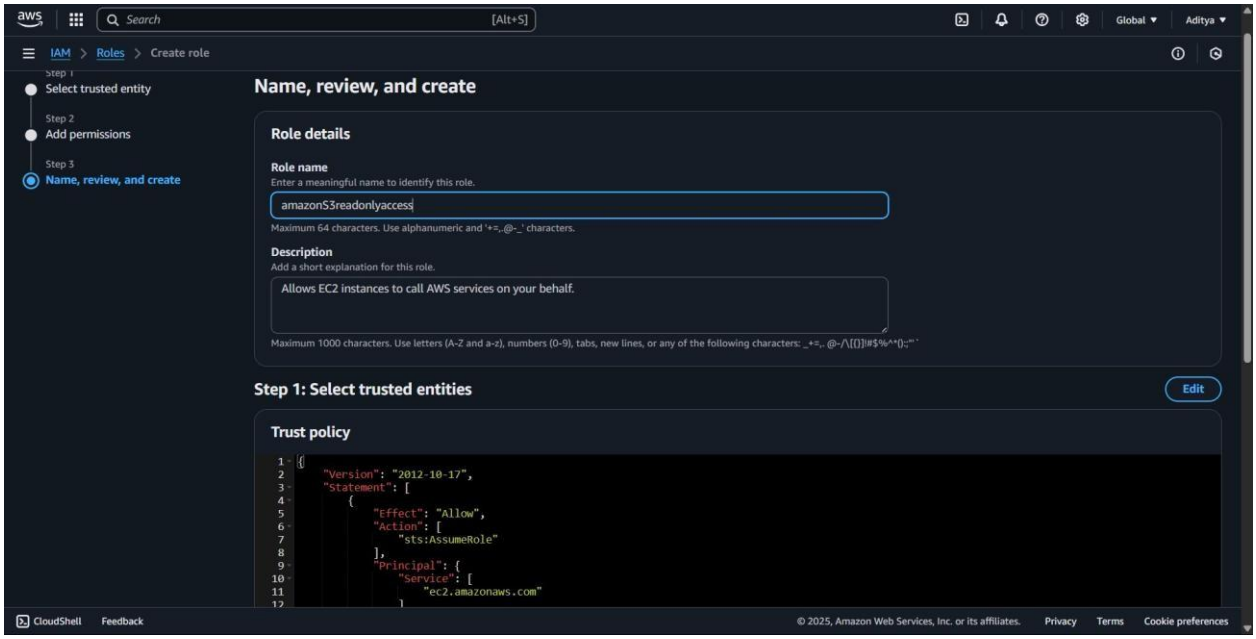
Step 6 : click Next ,

In the **permissions** page, search for the policy: **AmazonS3ReadOnlyAccess**

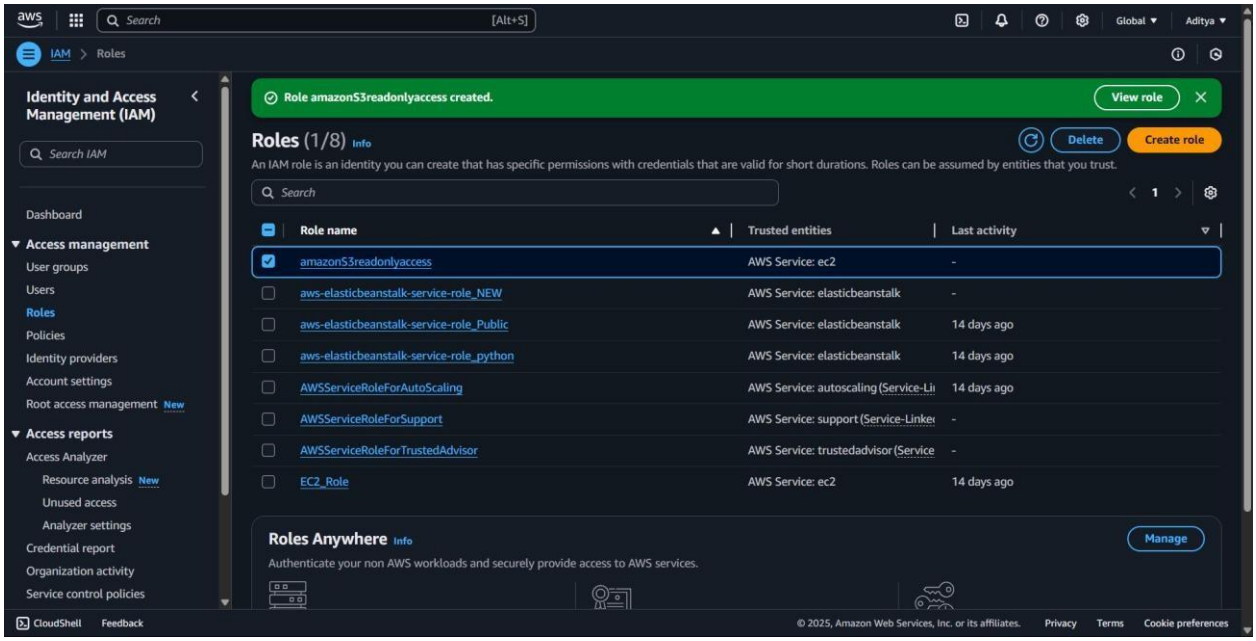
Select the checkbox next to it and Click **Next**.



Step 7: : Name and Create the Role.



Step 8 : click on the create role.



2.Create a custom policy that allows listing all S3 buckets but denies deleting them.

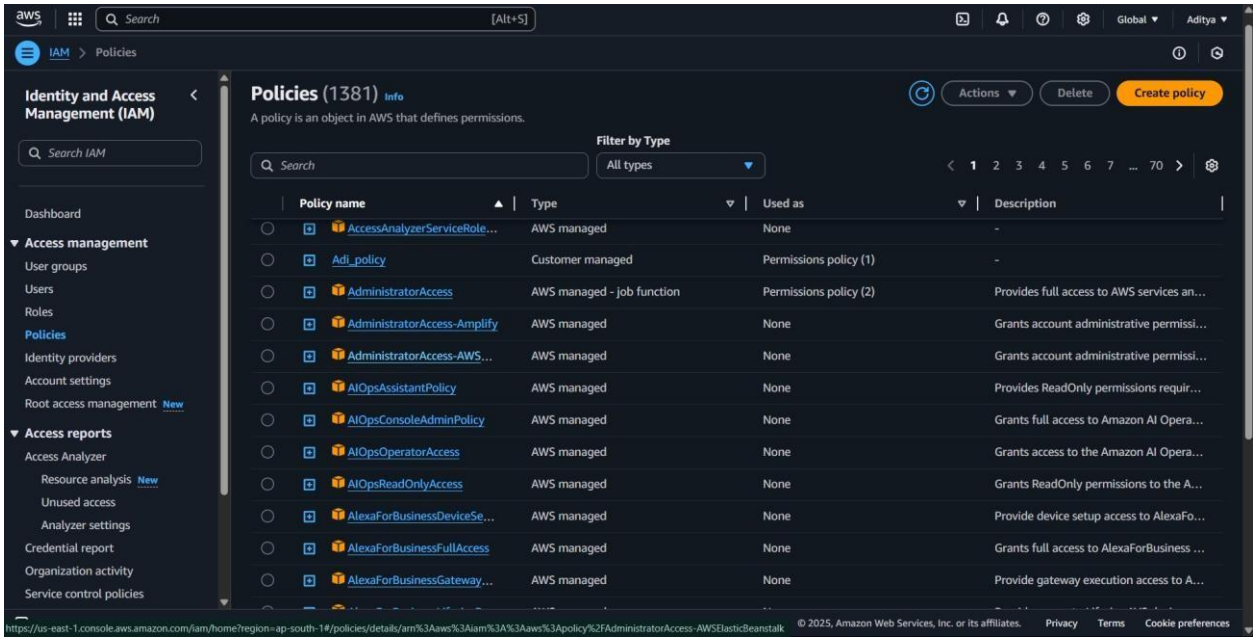
objective :

To create a custom IAM policy that grants permissions to list all Amazon S3 buckets while explicitly denying any delete operations on them.

prerequisite :

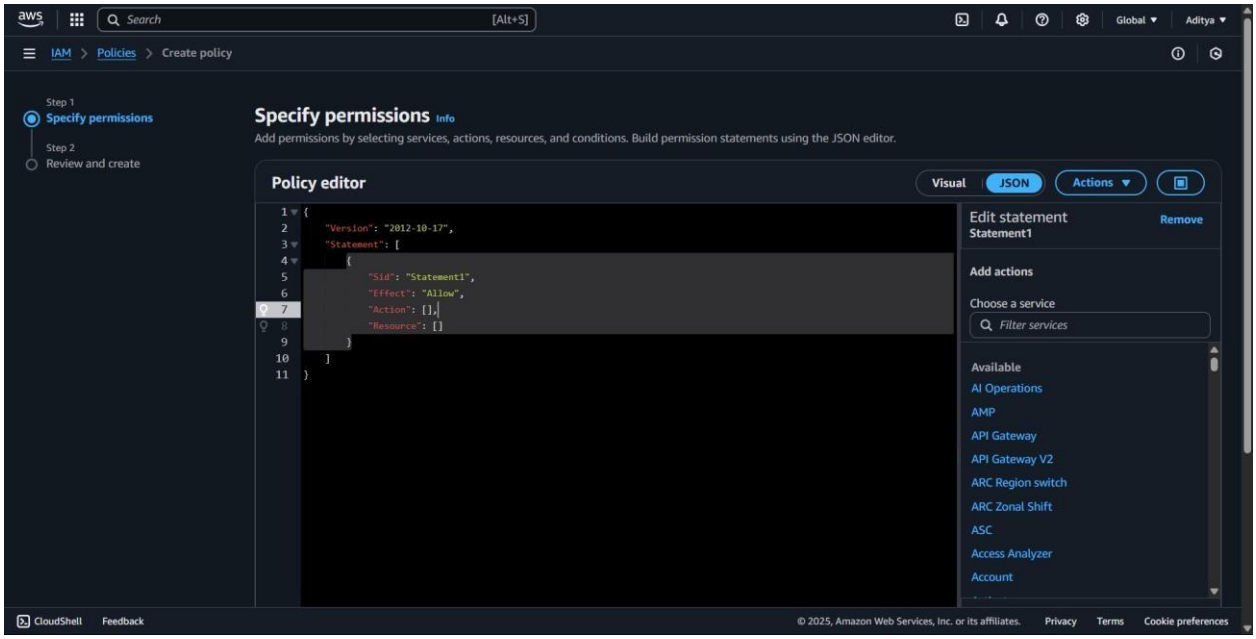
- An active AWS account.
- Knowledge of AWS IAM policies and JSON policy structure.
- Permissions to create and manage IAM policies.
- AWS Management Console or AWS CLI access**.**

Step 1 : In the IAM Dashboard, click on **Policies** from the left sidebar.

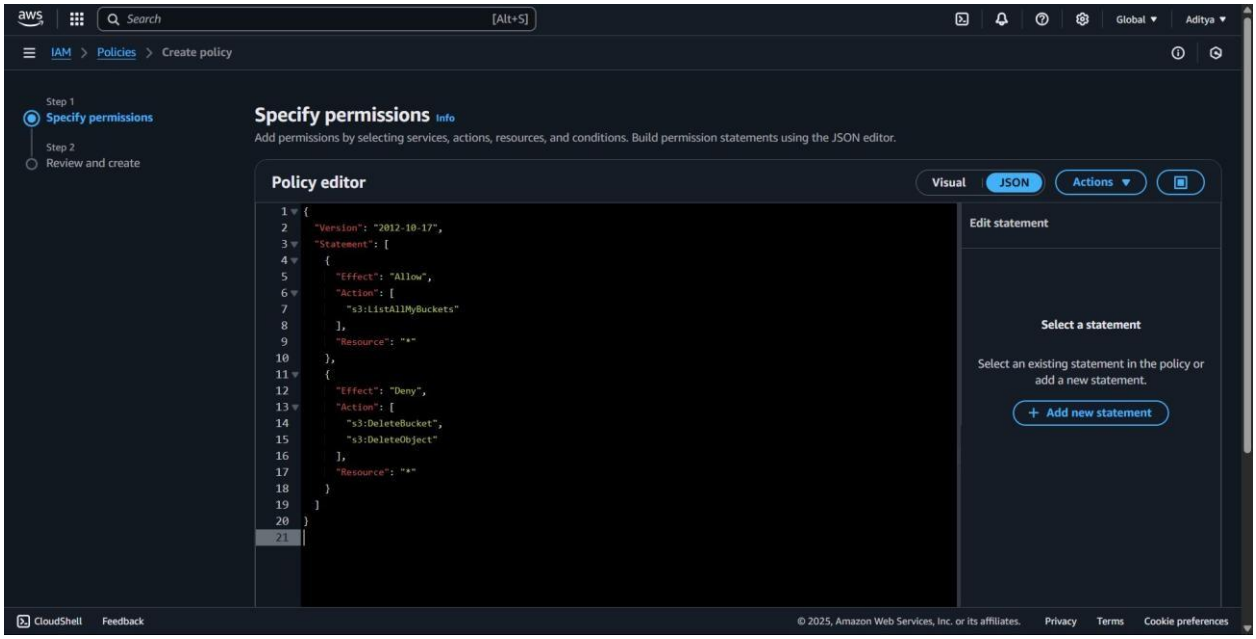


Step 2 : Click the **“Create policy”** button

Step 3 : Switch to JSON Tab.

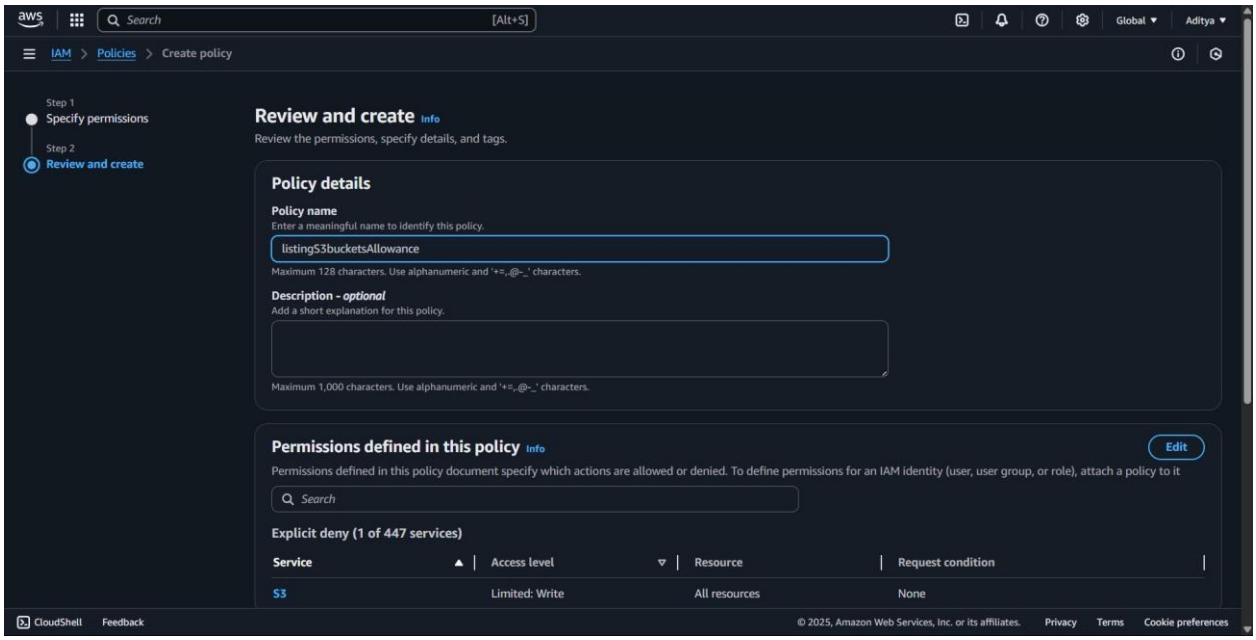


Step 4: Replace the default content with the following custom policy:

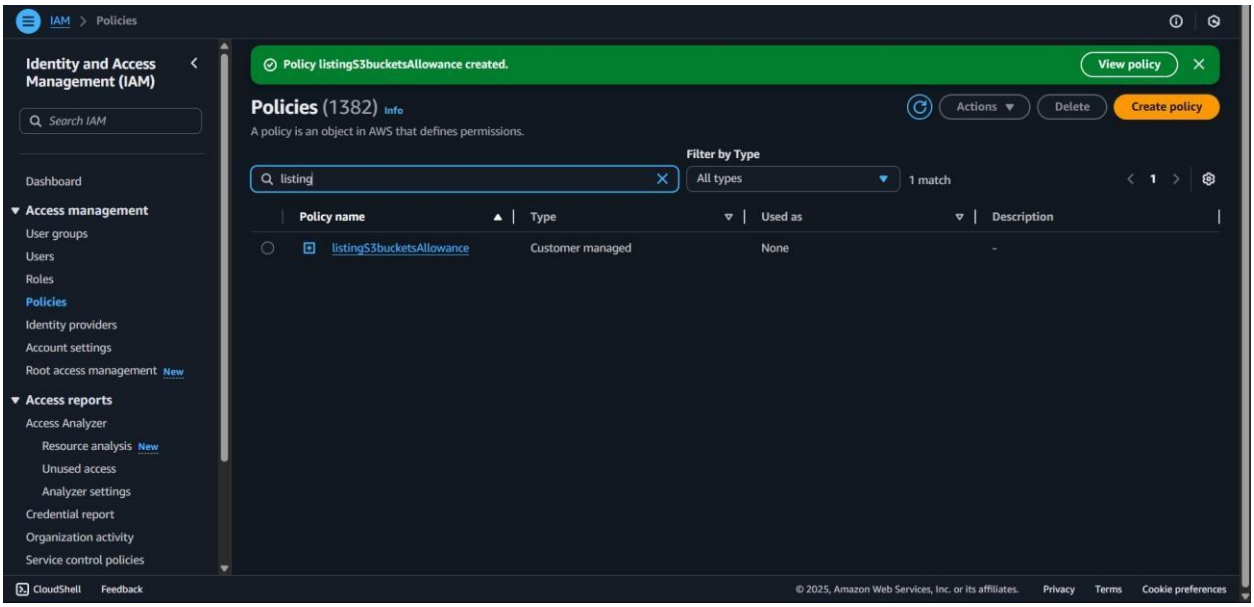


Step 5: click next.

Step 6: Enter the **Name** of the policy and Click **Create policy**.



Step 7: Search for your policy name.


IAM > Policies
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies
Policy listingS3bucketsAllowance created.
View policy
Policies (1382) Info
A policy is an object in AWS that defines permissions.
Actions
Delete
Create policy
Filter by Type
listing
All types
1 match
1
Policy name
Type
Used as
Description
listingS3bucketsAllowance
Customer managed
None
-
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

3. View the IAM Credential Report.

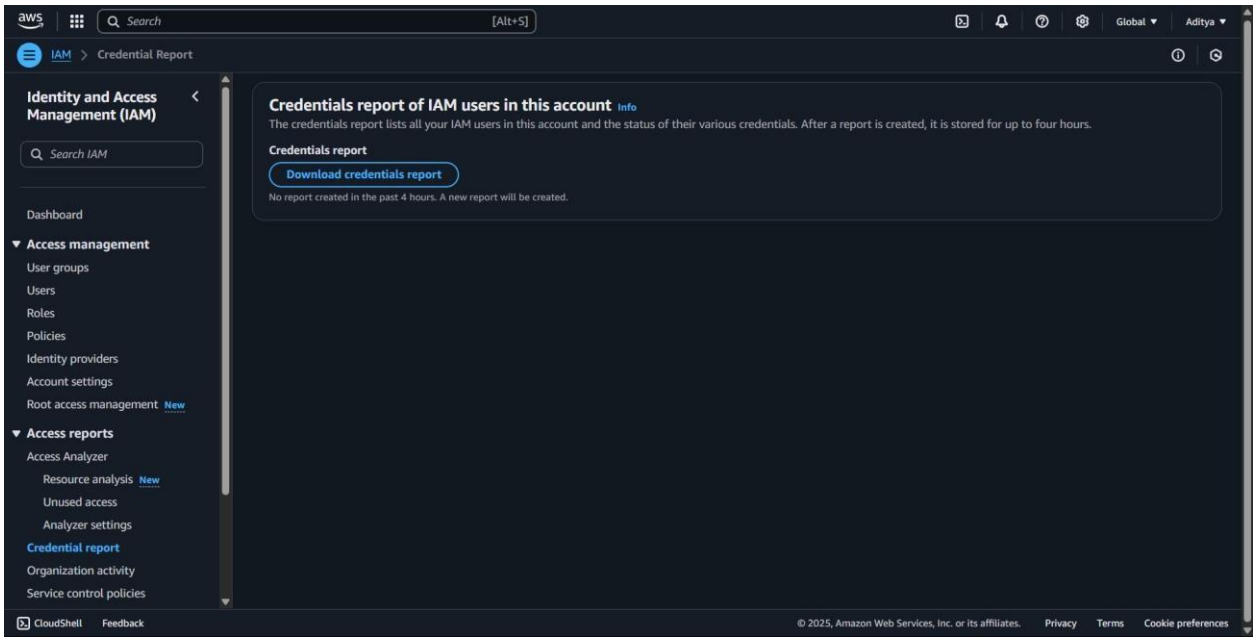
objective:

To generate and view the IAM Credential Report, which provides security-related information about all IAM users in the AWS account.

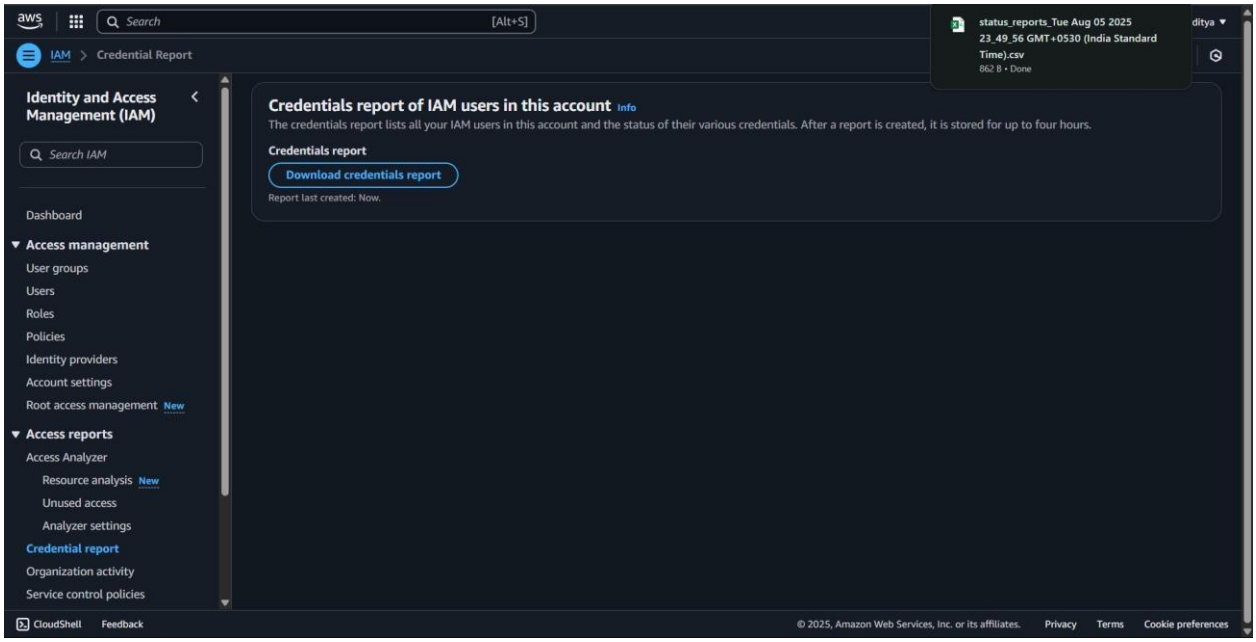
prerequisite:

- An active AWS account.
- Permissions to access and download the IAM Credential Report.
- AWS Management Console or AWS CLI access.

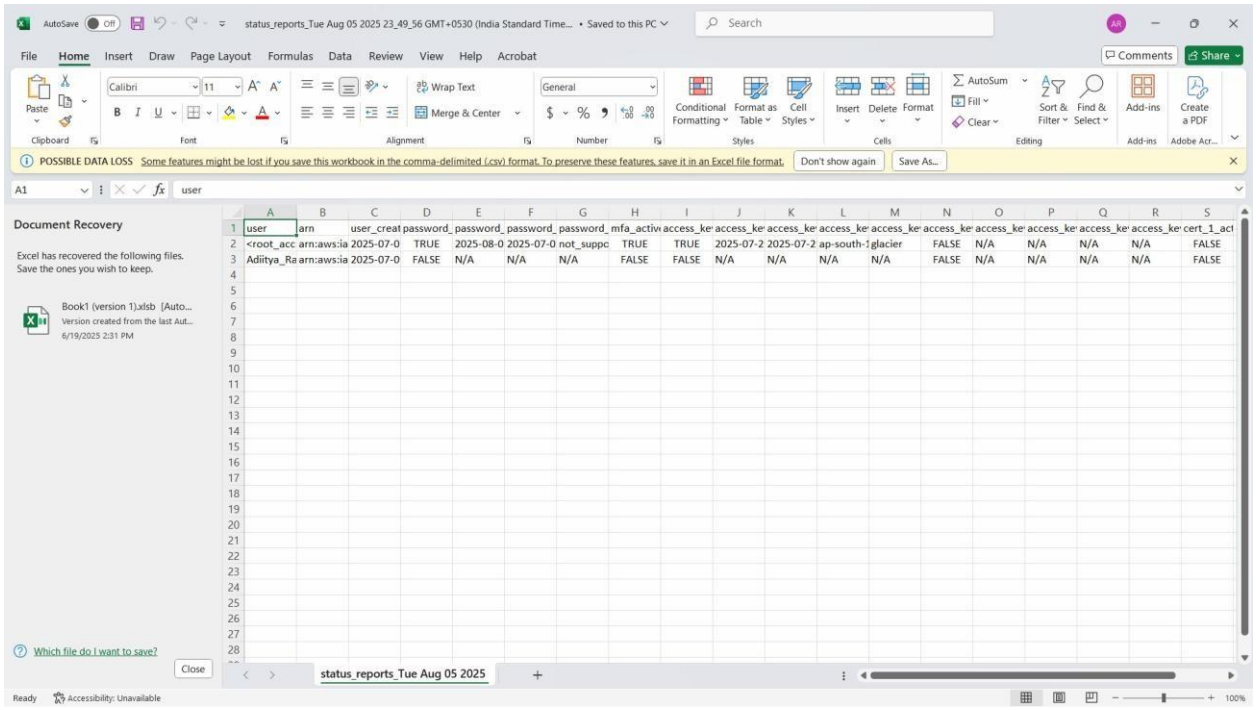
Step 1: In the IAM Dashboard, select **Credential Report** from the left sidebar.



Step 2 : Click on the **“Download Report”** button.



Step 3: Open the downloaded file.



4. Create a Service Control Policy (SCP) using AWS Organizations that blocks deletion of EC2 instances.

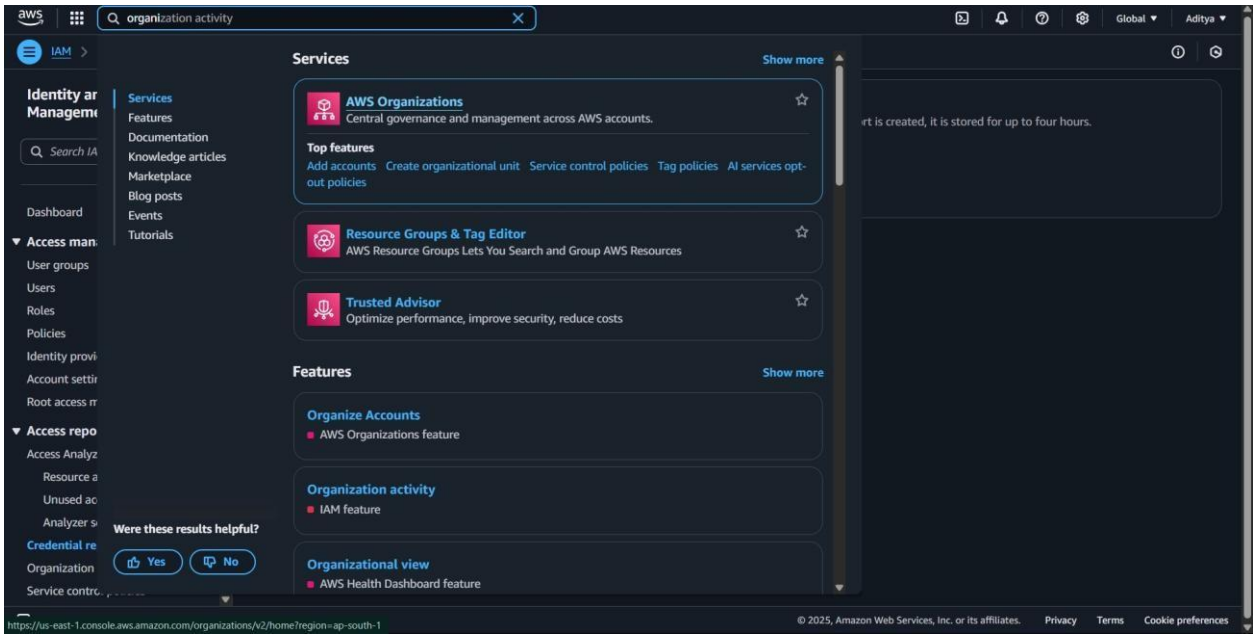
objective :

To create a Service Control Policy (SCP) in AWS Organizations that prevents the deletion of EC2 instances across specified AWS accounts or organizational units.

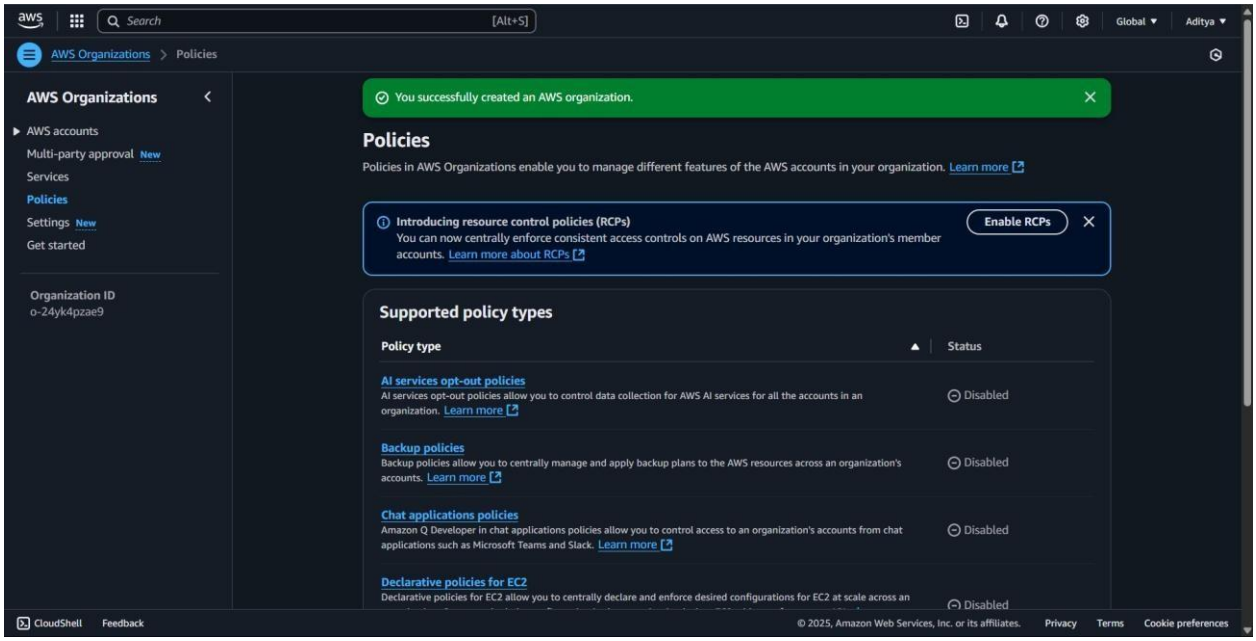
prerequisite :

- An active AWS account with AWS Organizations enabled.
- Permissions to create and manage Service Control Policies.
- Basic knowledge of SCP JSON structure and EC2 actions.
- AWS Management Console or AWS CLI access.

Step 1 : In the AWS Console, go to **Organizations** from the **Services** menu.

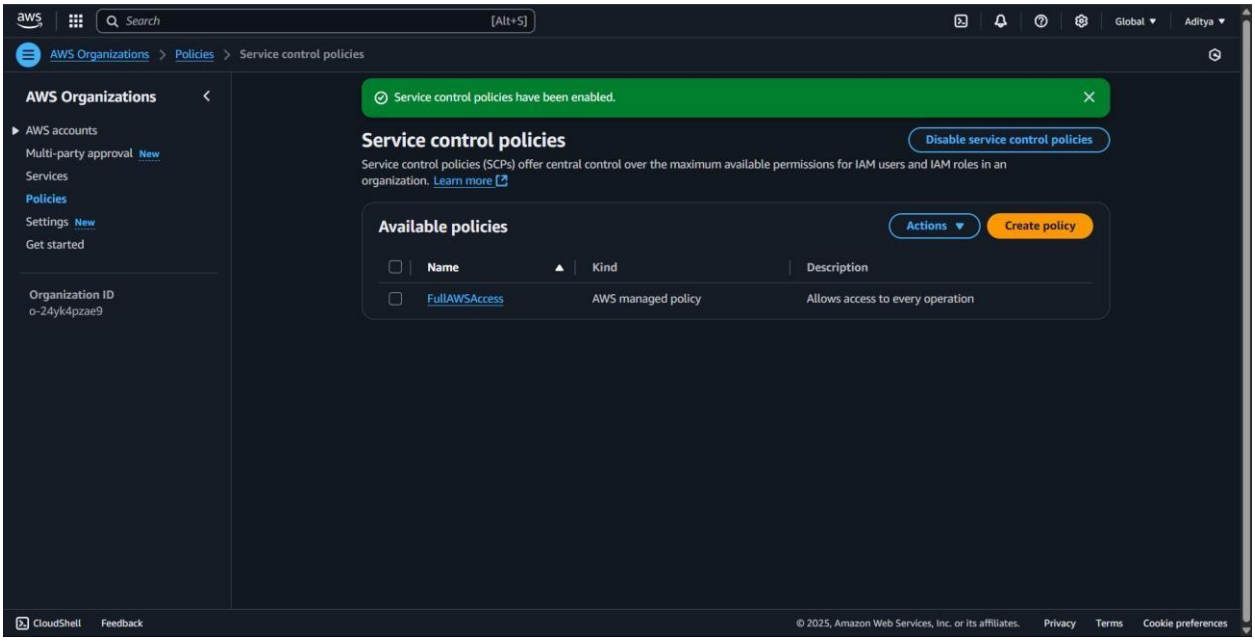


Step 2: In the left-hand menu, click on **Policies**.



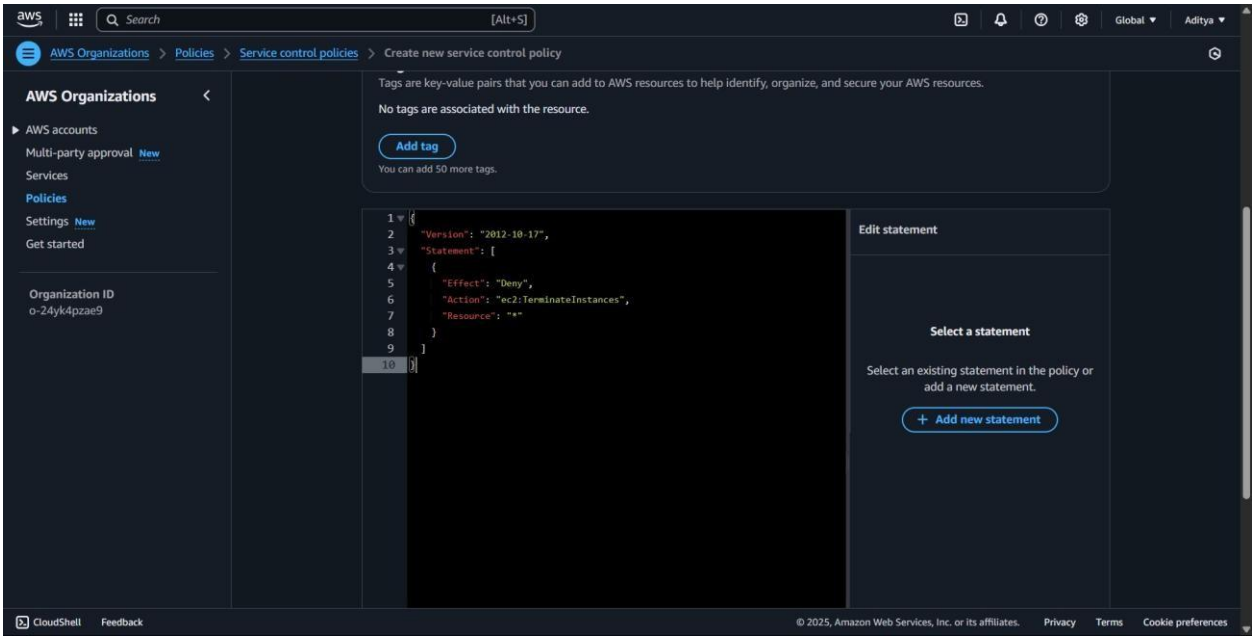
Step 3: Ensure that **Service Control Policies (SCPs)** are enabled for your organization.

If not enabled, click on **Enable SCPs**.

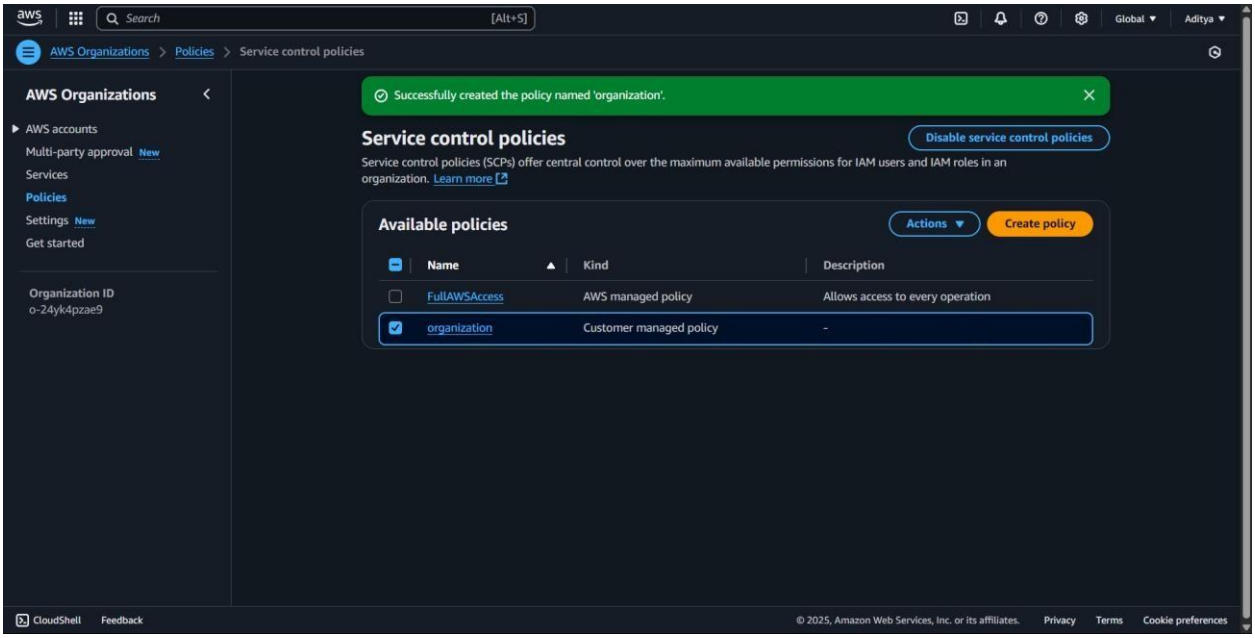


Step 4: **Create a New SCP by** Clicking on the **“Create policy”** button.

Step 5 : Under the **JSON** tab, enter the following SCP:



Step 6: click next.



5. Create a new security group that allows SSH (port 22) access only from your public IP.

objective:

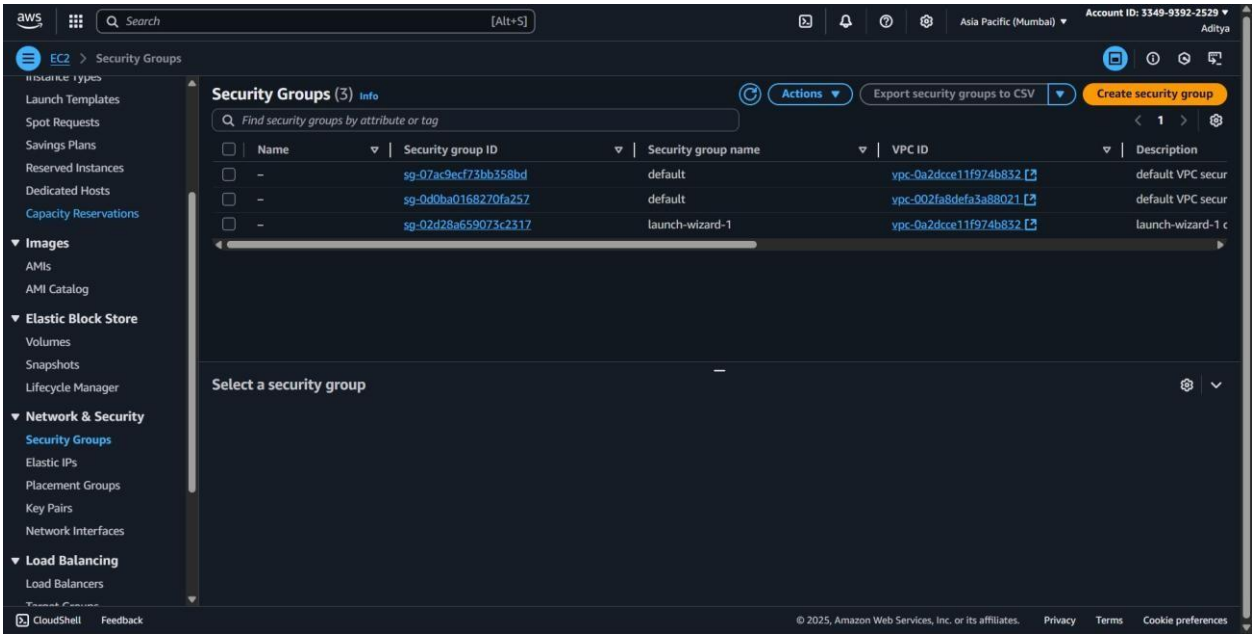
To enhance EC2 instance security by configuring a security group that restricts SSH (port 22) access to only the trusted public IP of the user.

prerequisite:

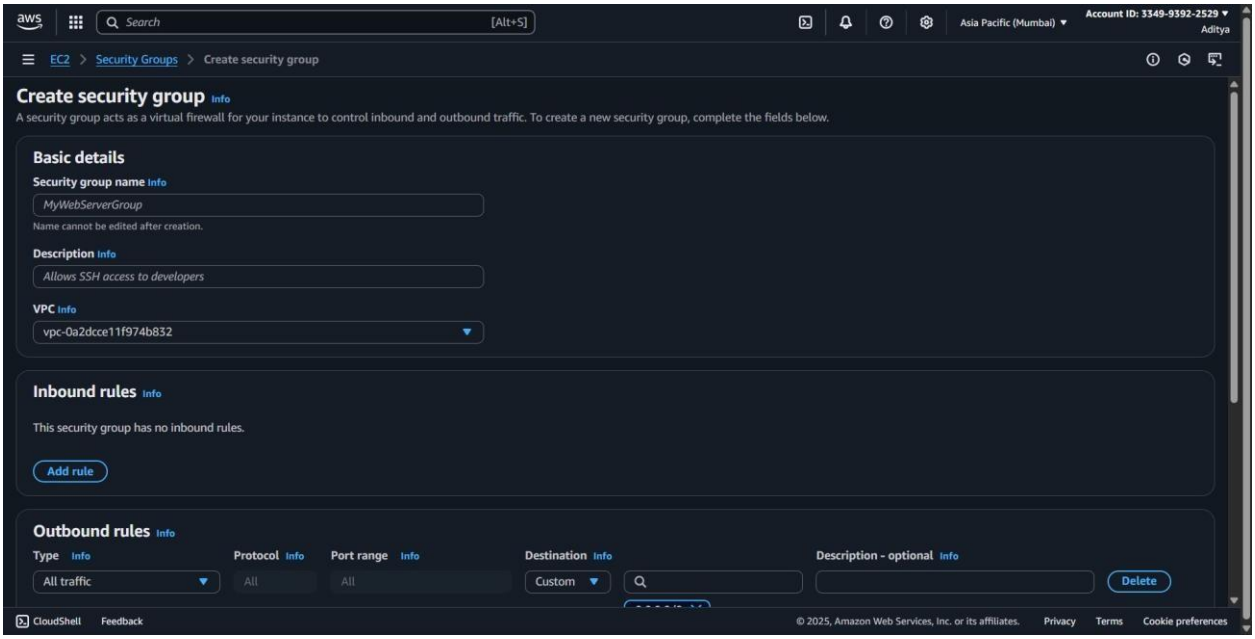
- An active AWS account.
- Knowledge of the current public IP address.
- Permissions to create and manage security groups.
- AWS Management Console or AWS CLI access.

Step 1: Go to the EC2 Dashboard

Step2: Open the Security Groups Page.



Step 3: Create a Security Group.



Step 4: **Enter Basic Details**

Security group name: SSH-Access-From-My-IP (you can choose your own name).

VPC: Select the VPC you want to use (usually the default VPC).

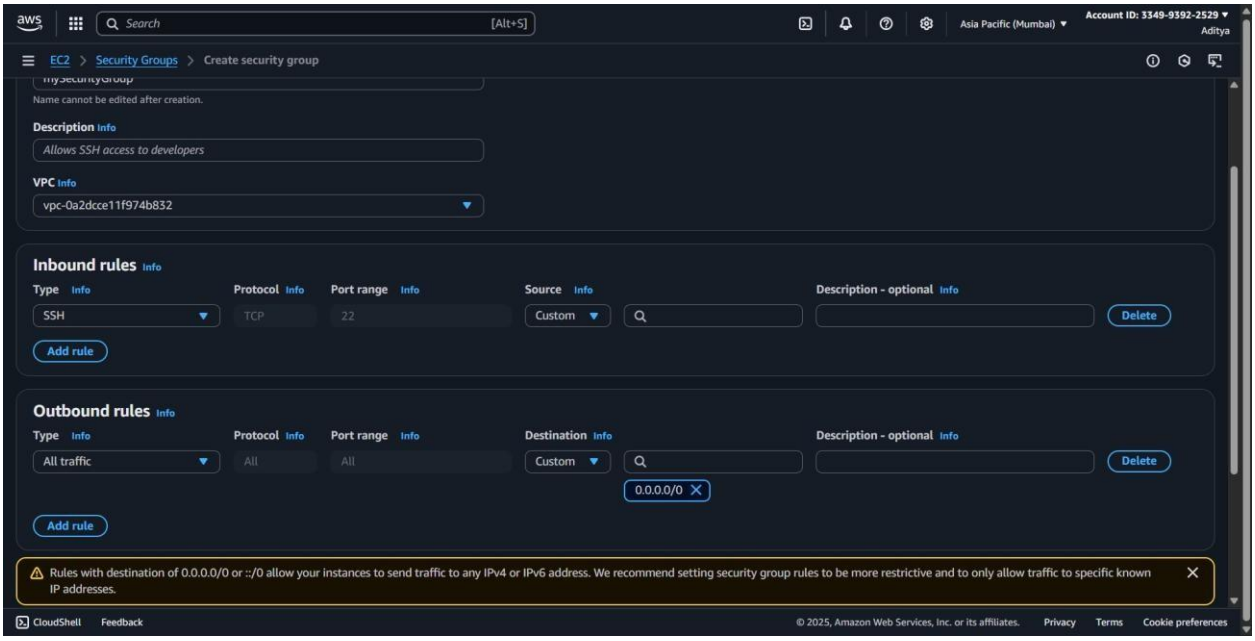
Step 5: Add an Inbound Rule for SSH

In Inbound rules, click Add rule.

Type: Select SSH.

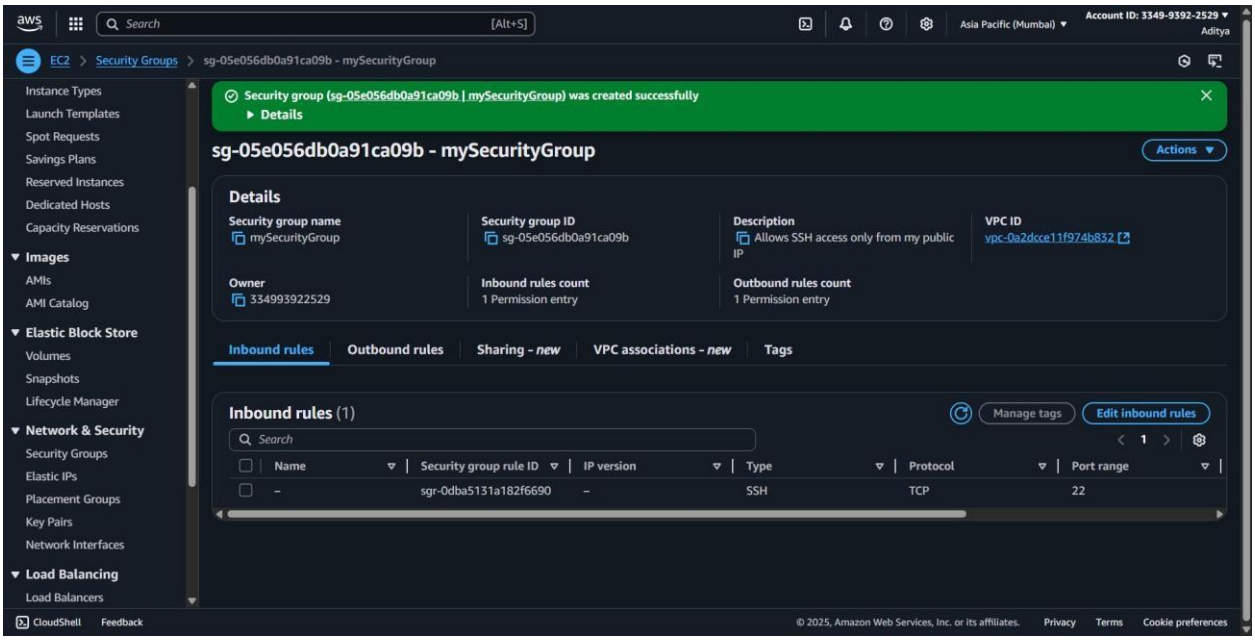
Protocol: Will be auto-filled as TCP.

Port range: Will be auto-filled as 22.



Step 6: Leave Outbound Rules as Default.

Step 7: Create the Security Group.



6. Open PuTTYgen. Load your .pem key file and convert it to .ppk.

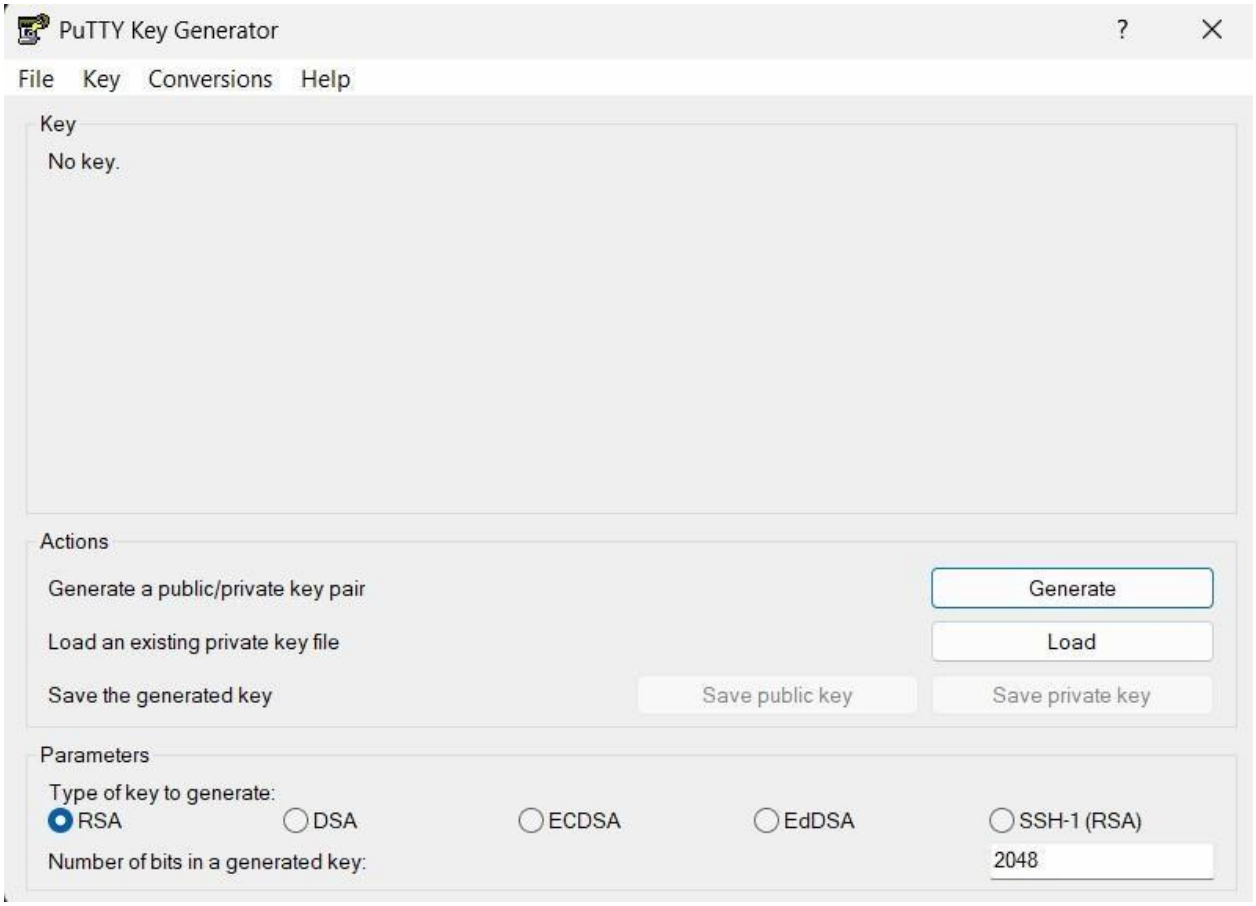
Objective:

To enable secure SSH access from PuTTY by converting an AWS-generated .pem key file into the .ppk format using PuTTYgen.

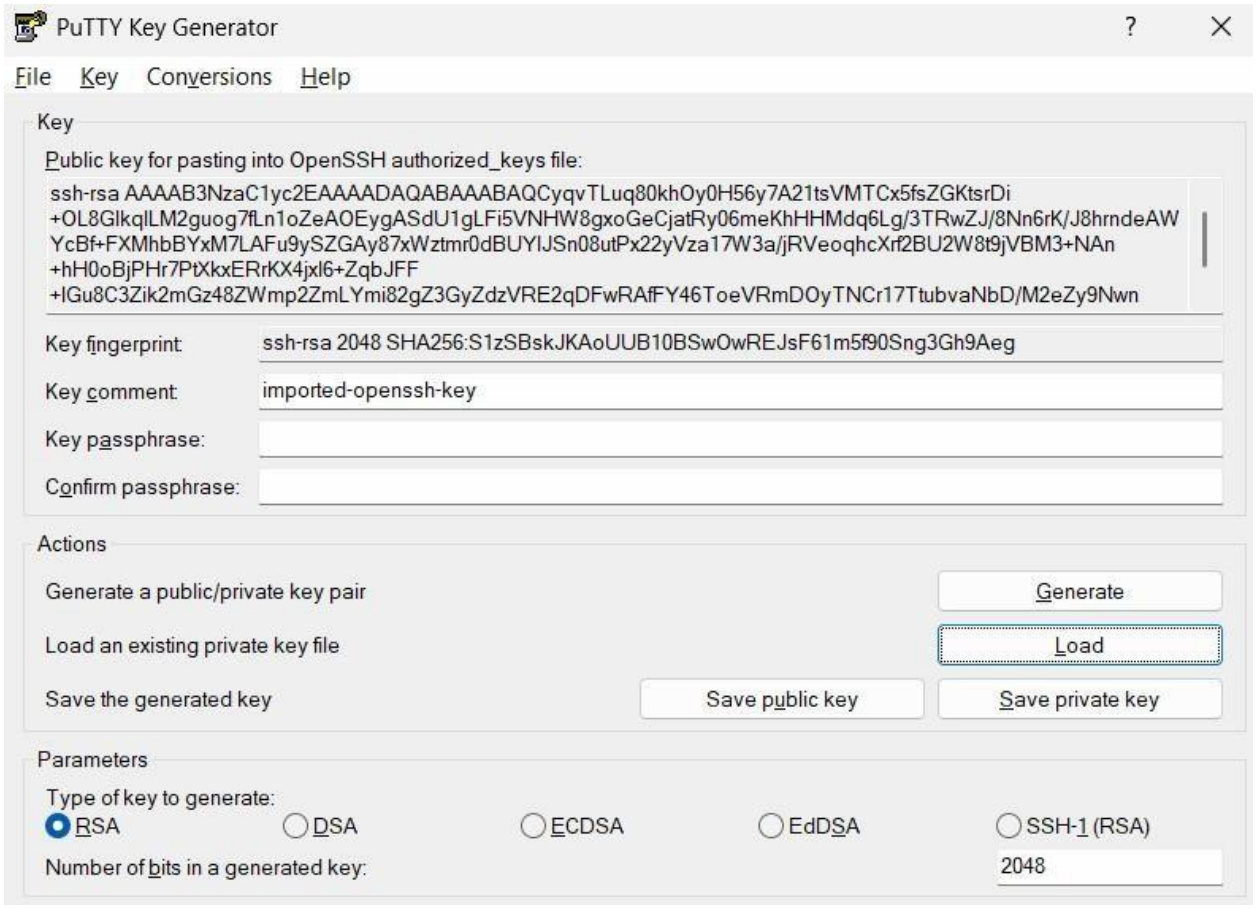
Prerequisite:

- PuTTYgen installed on the system.
- An existing .pem private key file downloaded from AWS.
- Basic understanding of SSH authentication.

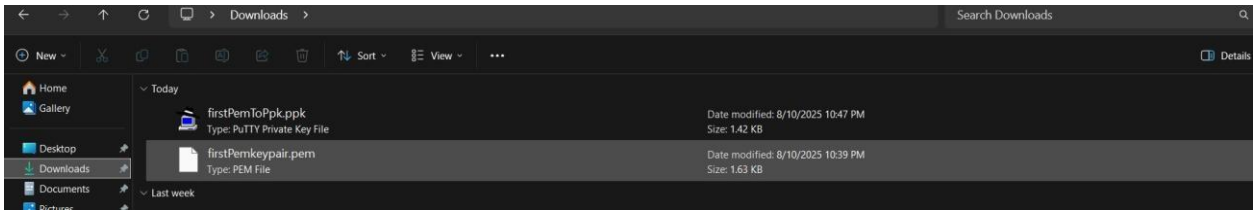
Step 1: Open PuTTYgen



Step 2: Load .pem file.



Step 3: Save the Key as .ppk Click Save private key and then check in downloads.



7.Create a simple text file on your EC2 instance named test.txt using the nano or vi editor.

Objective:

To practice basic file creation and editing on an EC2 instance by using command-line text editors like nano or vi to create a file named test.txt.

Prerequisite:

- An active AWS EC2 instance.
- SSH access to the EC2 instance.

- Basic knowledge of Linux command-line operations.
- Familiarity with nano or vi text editors.

Step 1: open putty

Step 2: open nano editor

Step 3: Type your content and then save and exit.

8. Access the environment's Configuration page. List any two resources (e.g., EC2, Load Balancer) that were automatically provisioned.

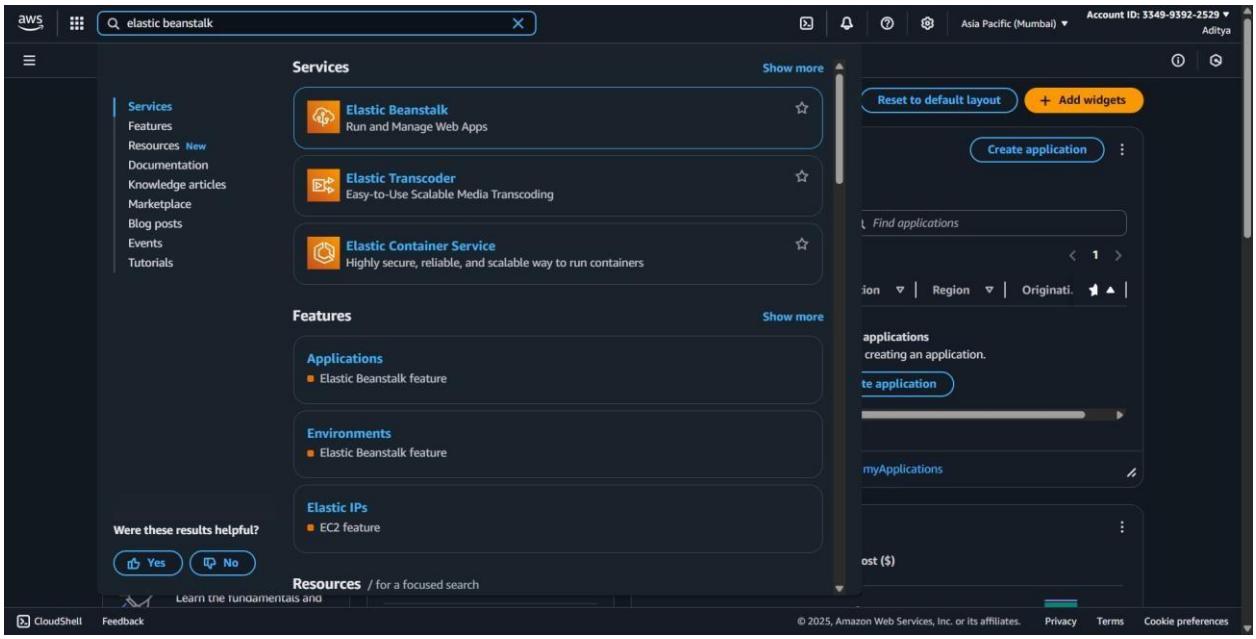
Objective:

To explore the environment's Configuration page and identify resources automatically provisioned by AWS, enhancing understanding of environment setup and resource management.

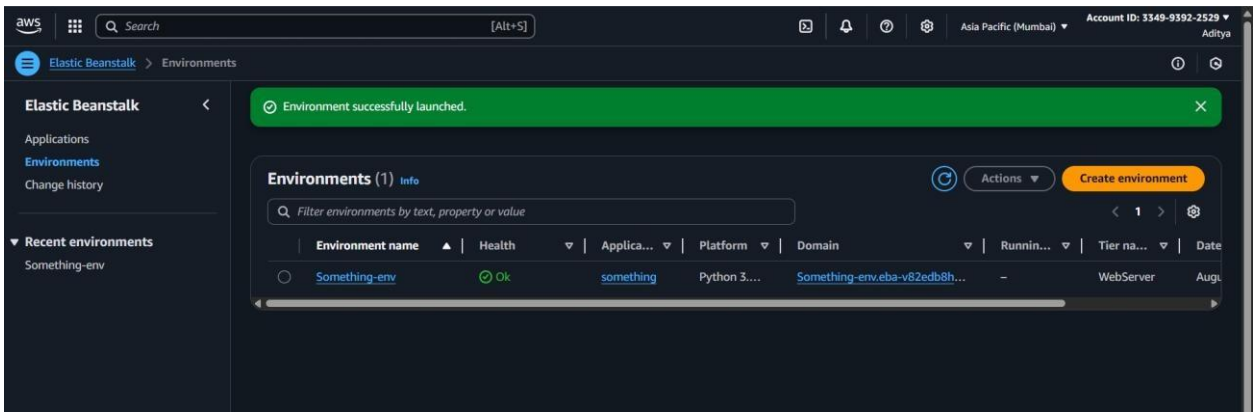
Prerequisite:

- An active AWS account.
- A deployed environment in AWS (e.g., Elastic Beanstalk).
- Permissions to view environment configuration details.
- AWS Management Console access.

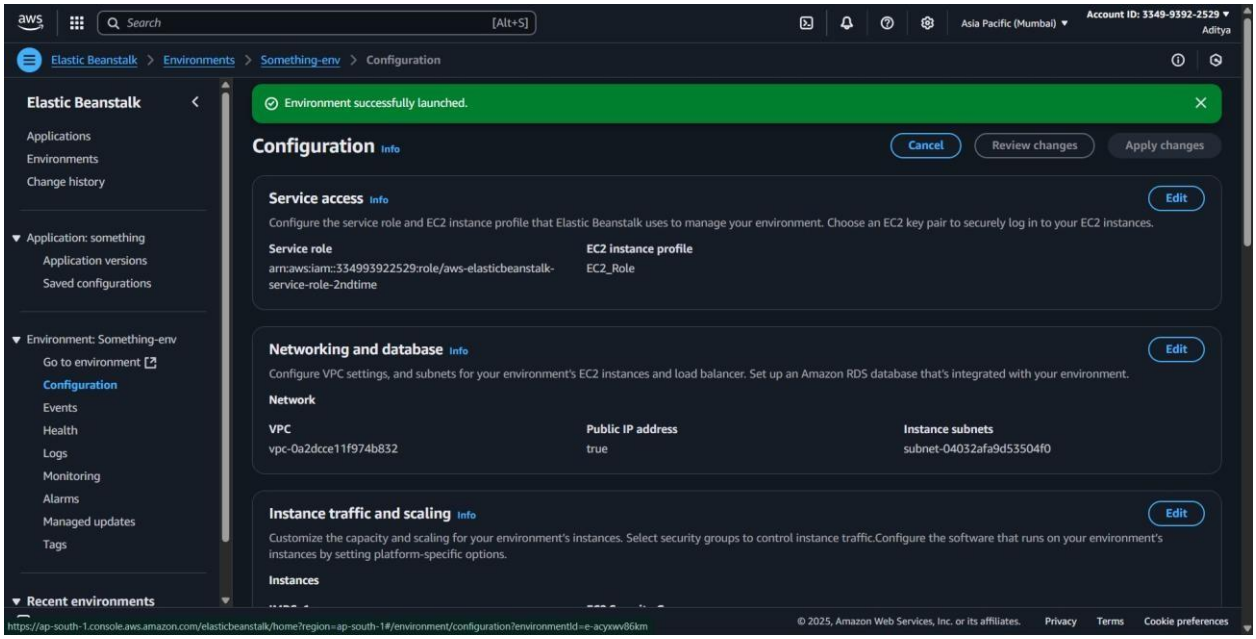
Step 1: Go to AWS Management Console and type elastic beanstalk.



Step 2: click the environment you have created.



Step 3: click configuration.



Step 4: On the **Configuration** page, scroll and check the categories such as:

Instances → shows the EC2 instances that were provisioned.

Load balancer → shows the Elastic Load Balancer created.

Security → shows security groups created for the environment.

9. Create a new volume from your snapshot.

Objective:

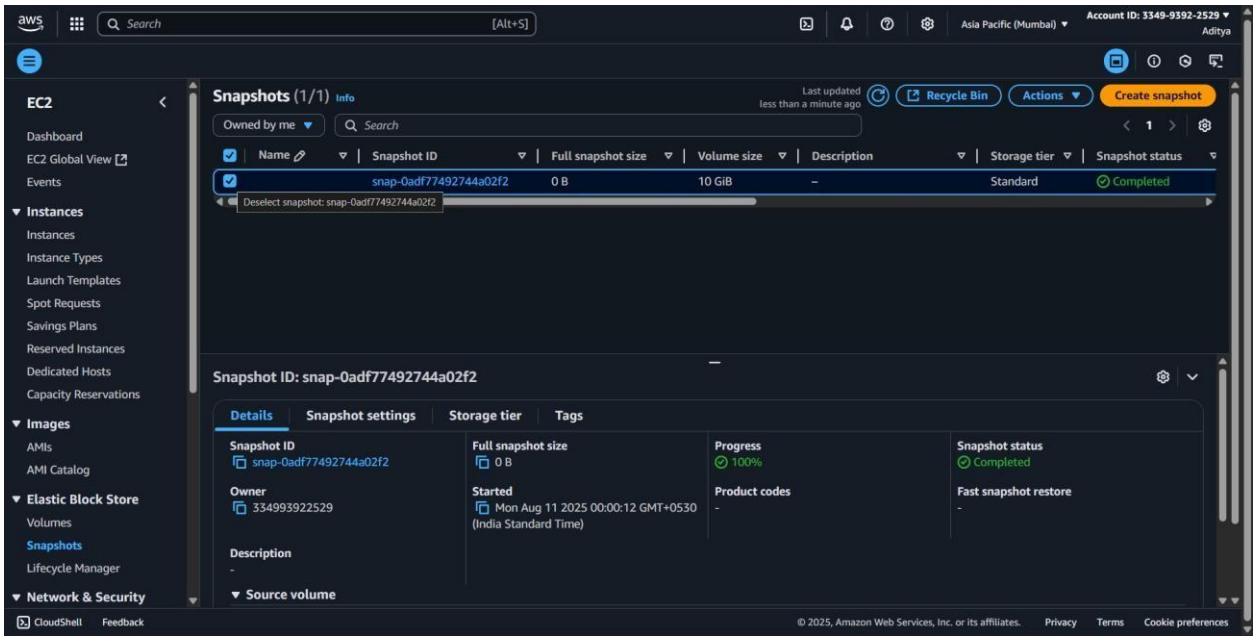
To utilize an existing EBS snapshot to create a new storage volume, demonstrating data restoration and replication capabilities in AWS.

Prerequisite:

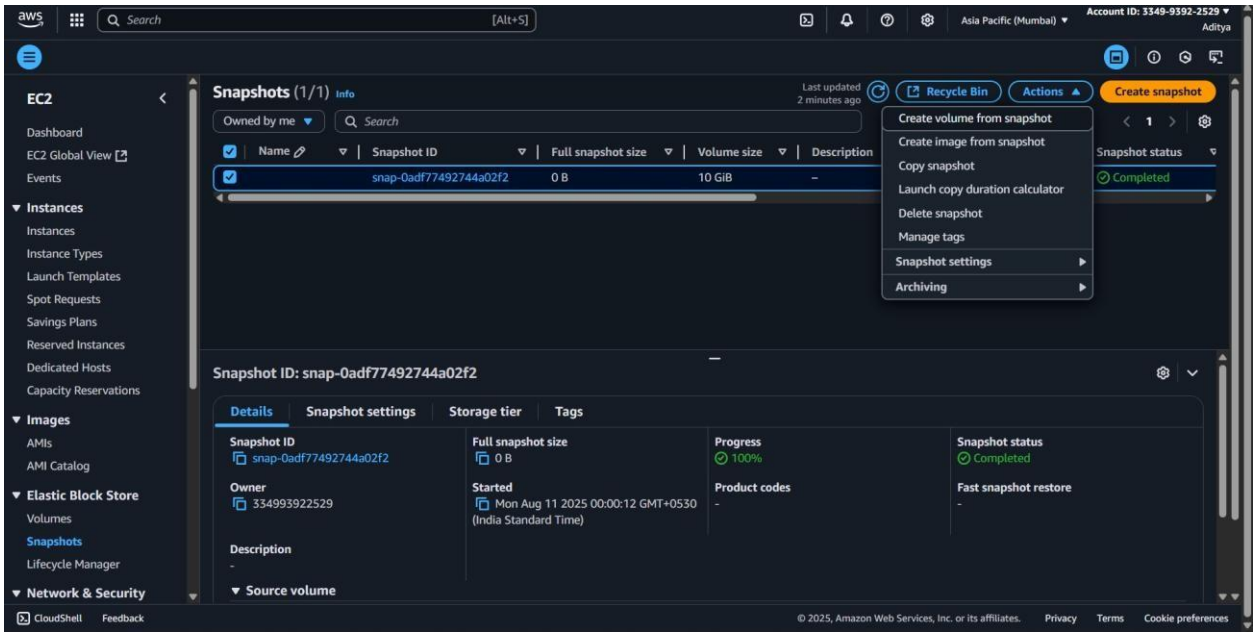
- An active AWS account.
- An existing EBS snapshot available.
- Permissions to create and manage EBS volumes.
- AWS Management Console or AWS CLI access.

Step 1: Go to the EC2 Dashboard

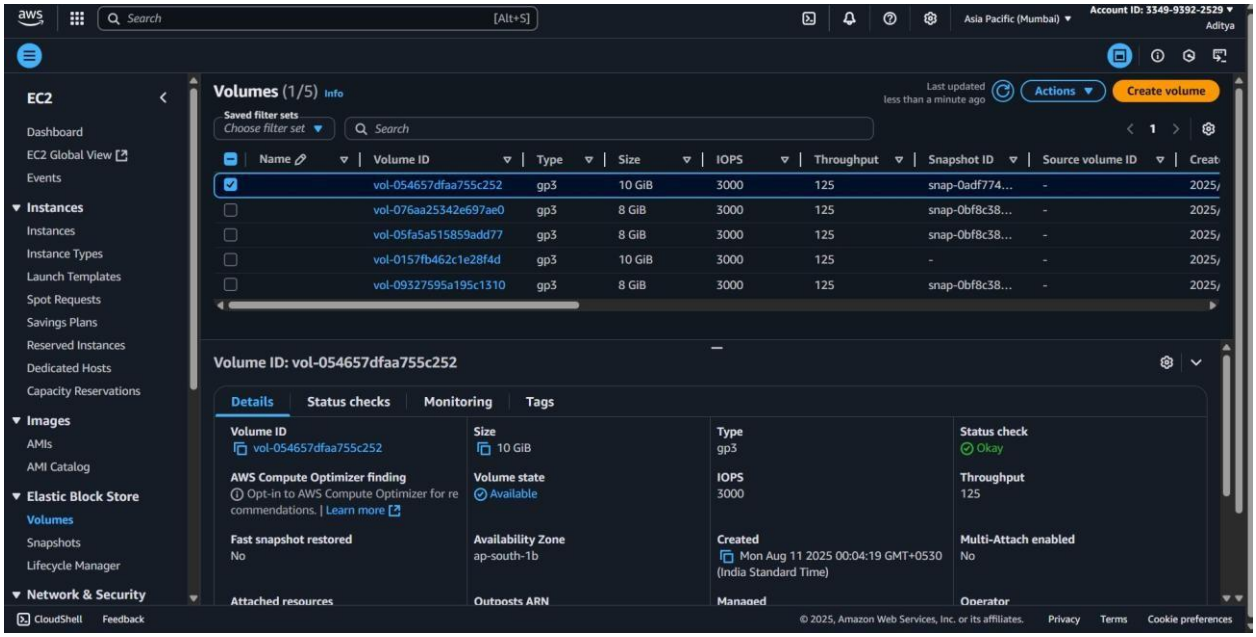
Step 2: open the snapshots page Click on the checkbox of your snapshot



Step 3: With the snapshot selected, click the **Actions** dropdown then choose **Create volume**.



Step 4: check in the volume section



10.Use the Amazon Data Lifecycle Manager (DLM) to create a scheduled backup policy.

Objective:

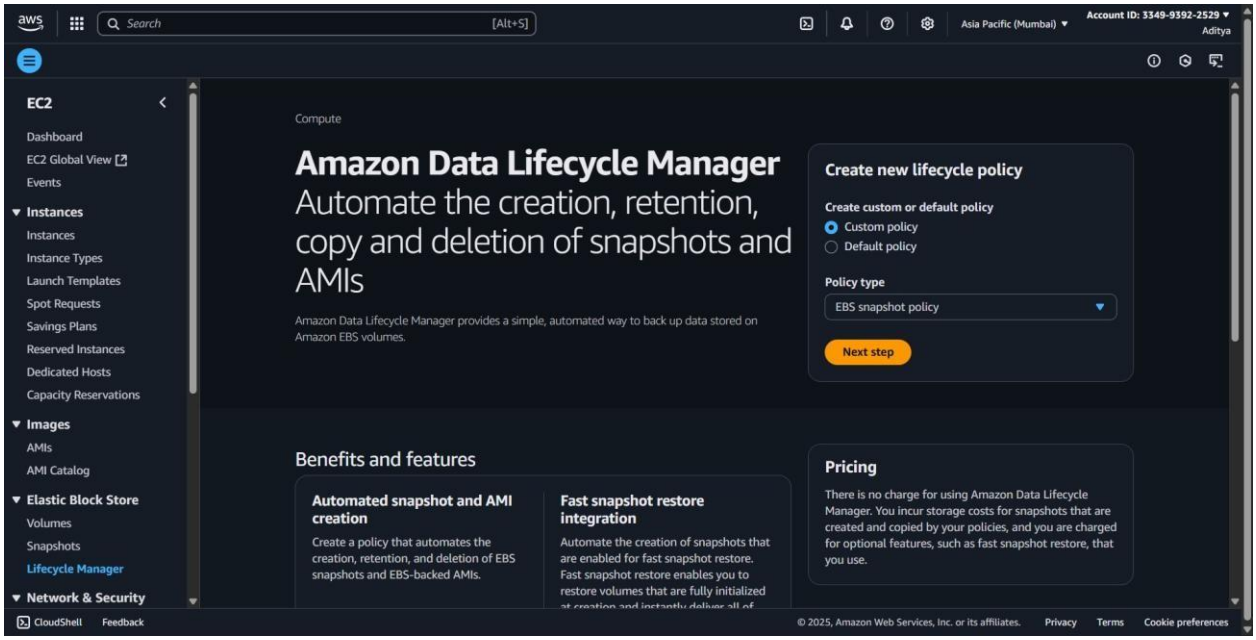
To automate EBS volume backups by creating a scheduled snapshot policy using Amazon Data Lifecycle Manager (DLM), ensuring regular data protection without manual intervention.

Prerequisite:

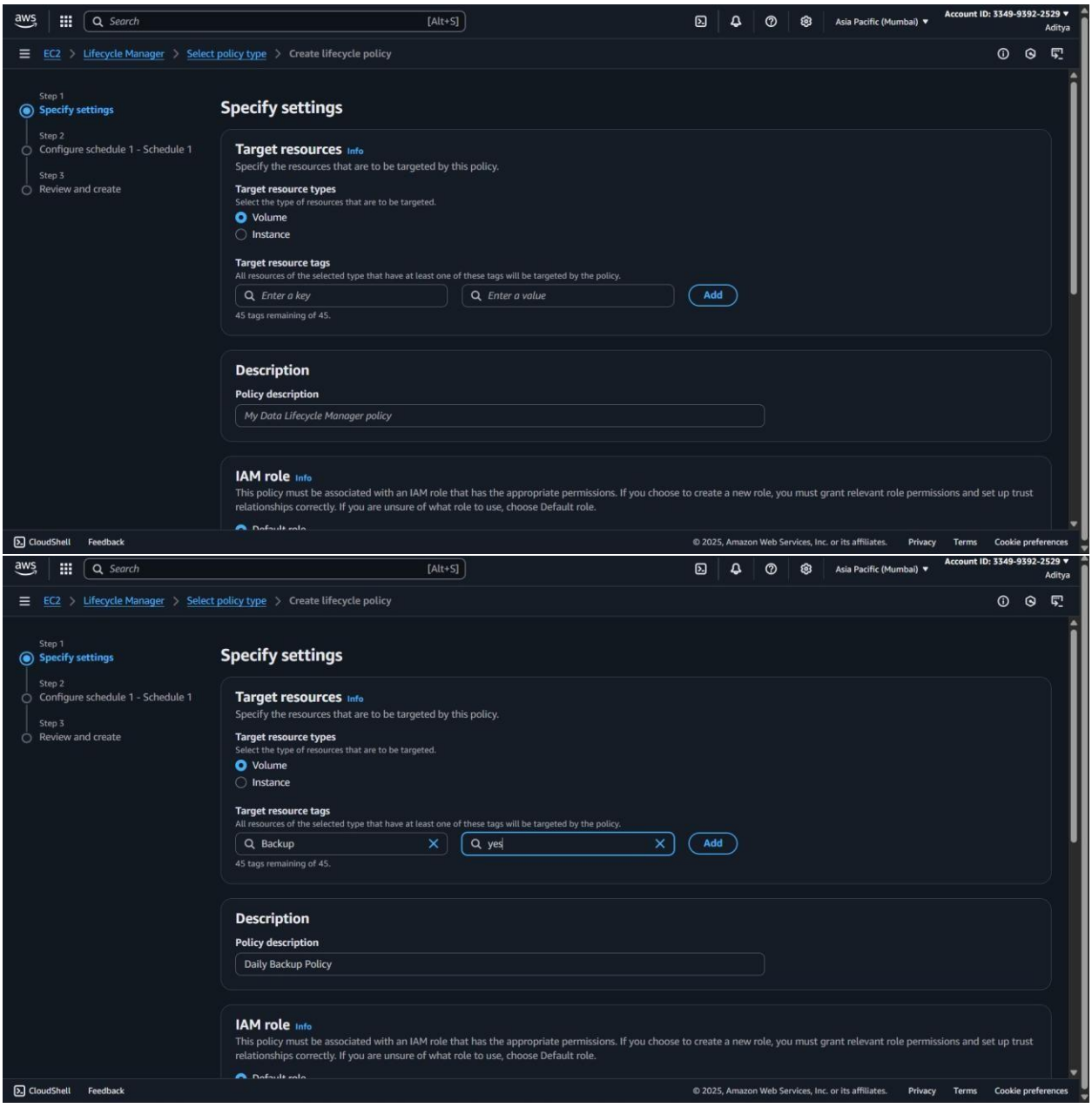
- An active AWS account.
- Existing EBS volumes to back up.
- Permissions to create and manage DLM lifecycle policies.
- AWS Management Console access.

Step 1: Go to EC2 dashboard

Step 2: Open Lifecycle Manager

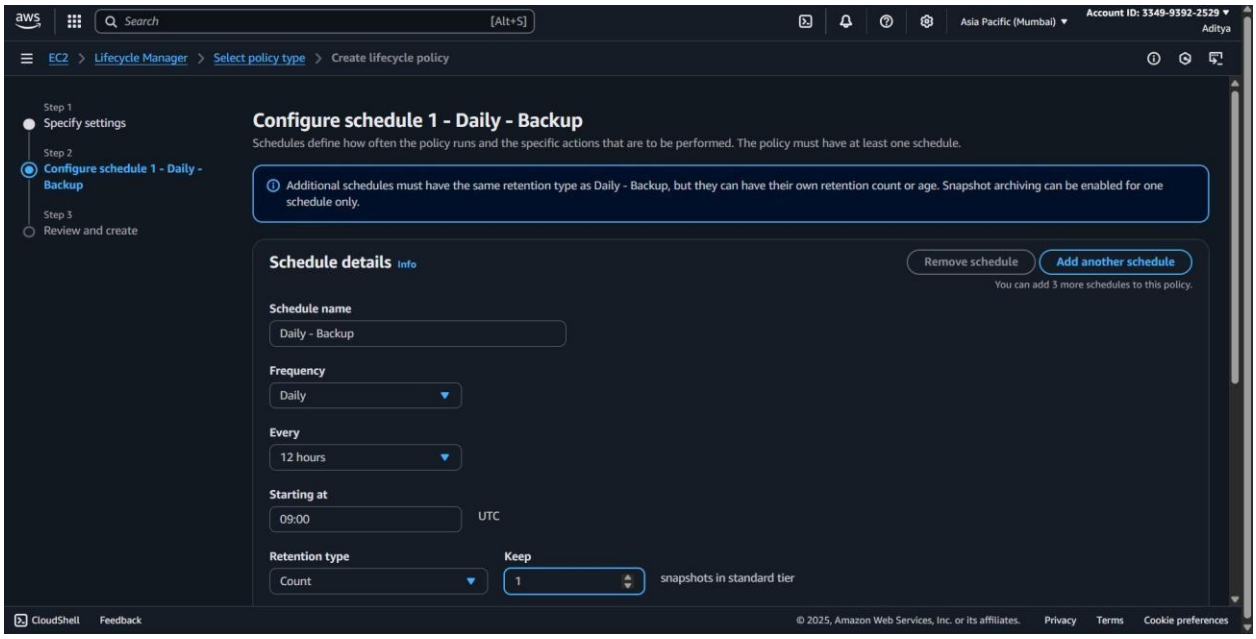


Step 3: click create lifecycle policy and choose EBS snapshot policy.



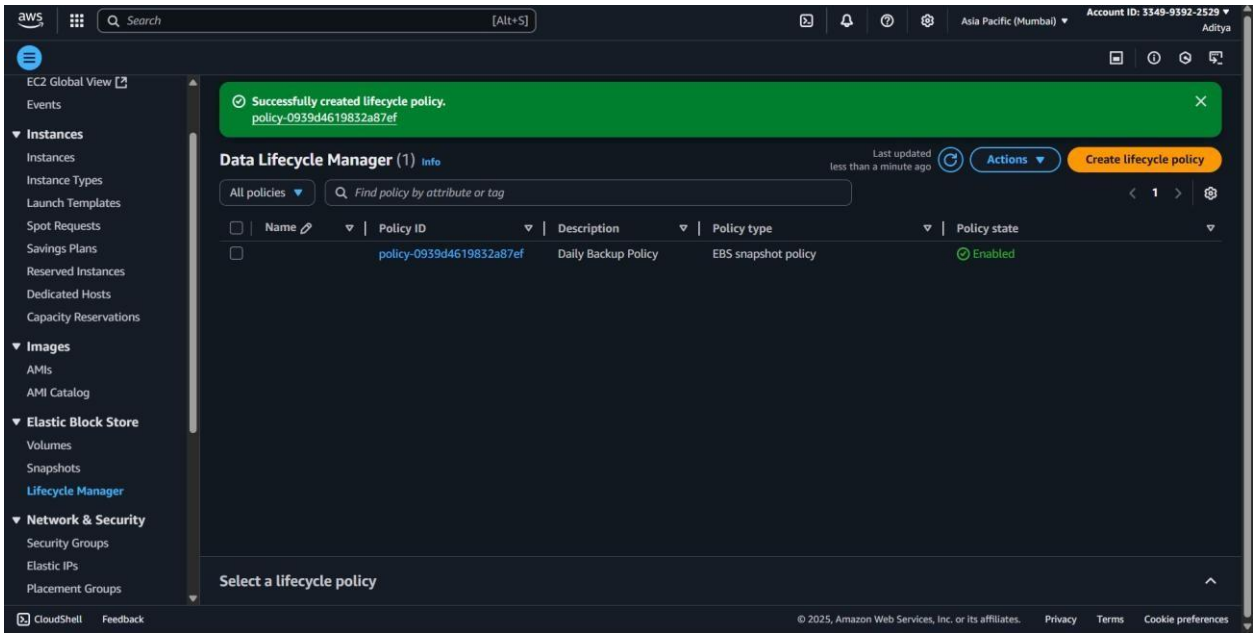
Step 4: Click next

Step 5: Configure Schedule.



Step 6: Review Policy Settings

Step 7: Click **Create policy**.



11.Upload another file and select S3 One Zone-IA or S3 Glacier as the storage class.

Objective:

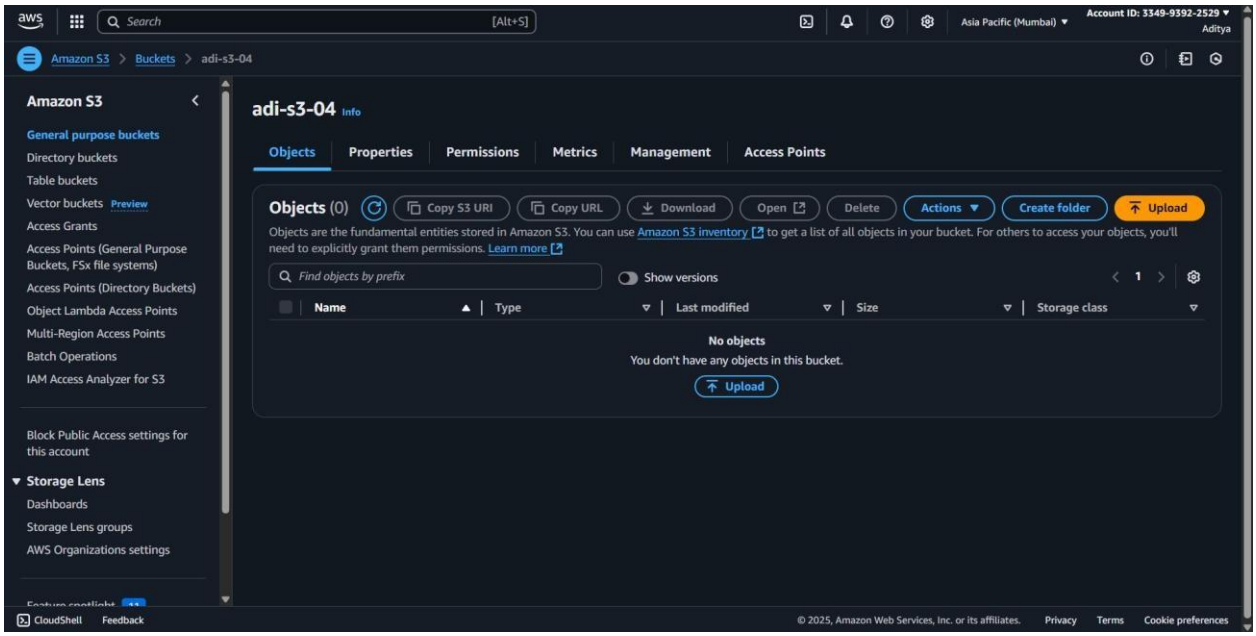
To optimize storage costs by uploading a file to Amazon S3 and selecting a cost-effective storage class such as S3 One Zone-IA or S3 Glacier based on data access needs.

Prerequisite:

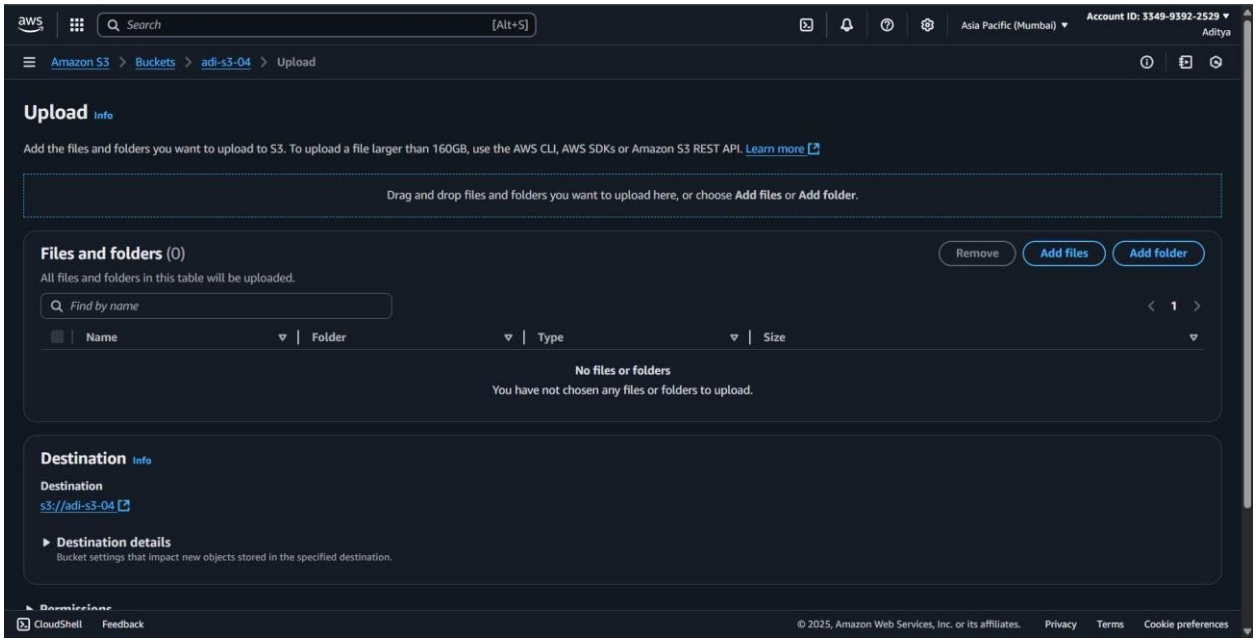
- An active AWS account.
- An existing S3 bucket.
- Permissions to upload objects and change storage classes.
- AWS Management Console or AWS CLI access.

Step 1: Go to the S3 Dashboard.

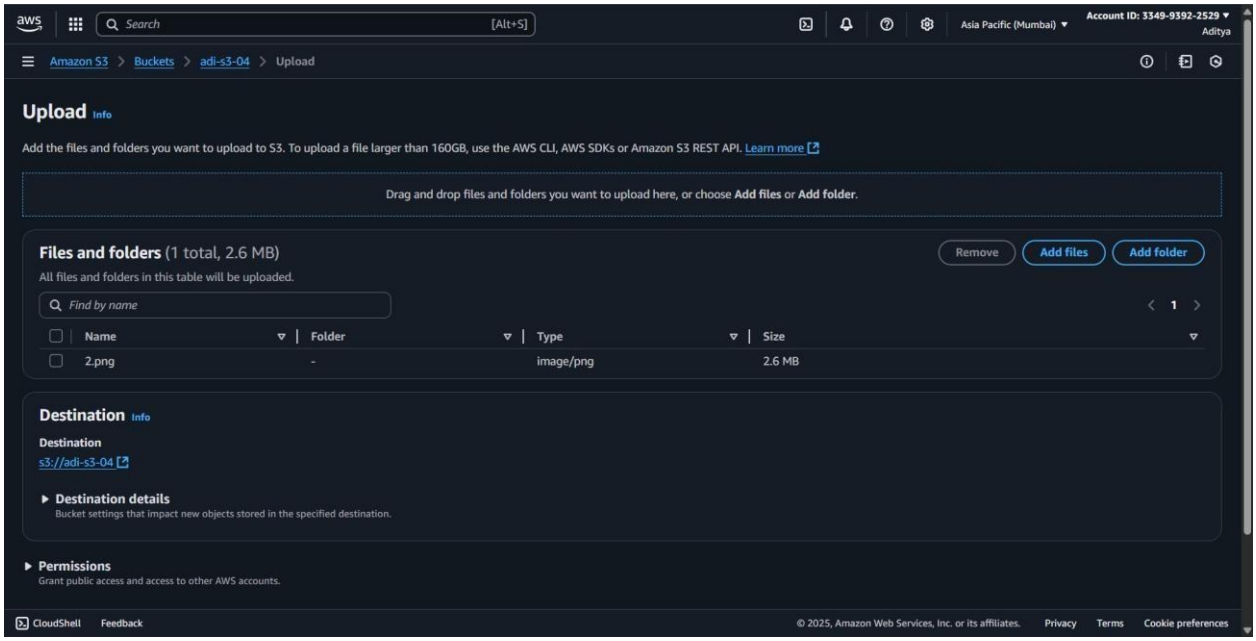
Step 2: Click the bucket name where you want to upload the file.



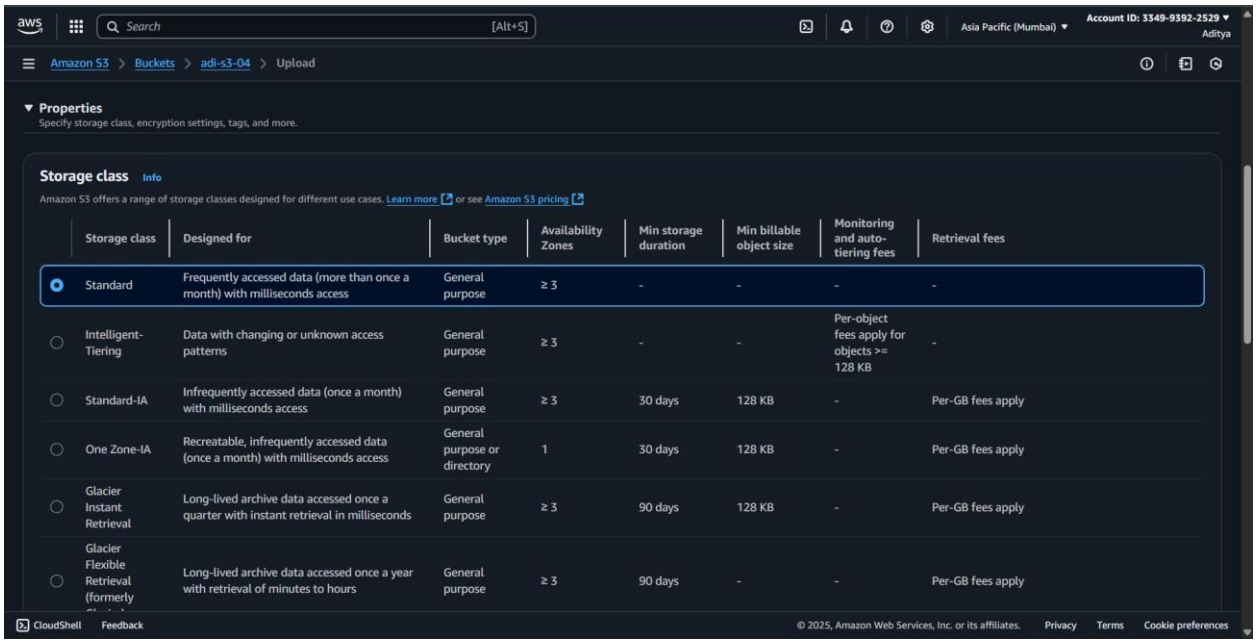
Step 3: On the bucket page, click the **Upload** button.



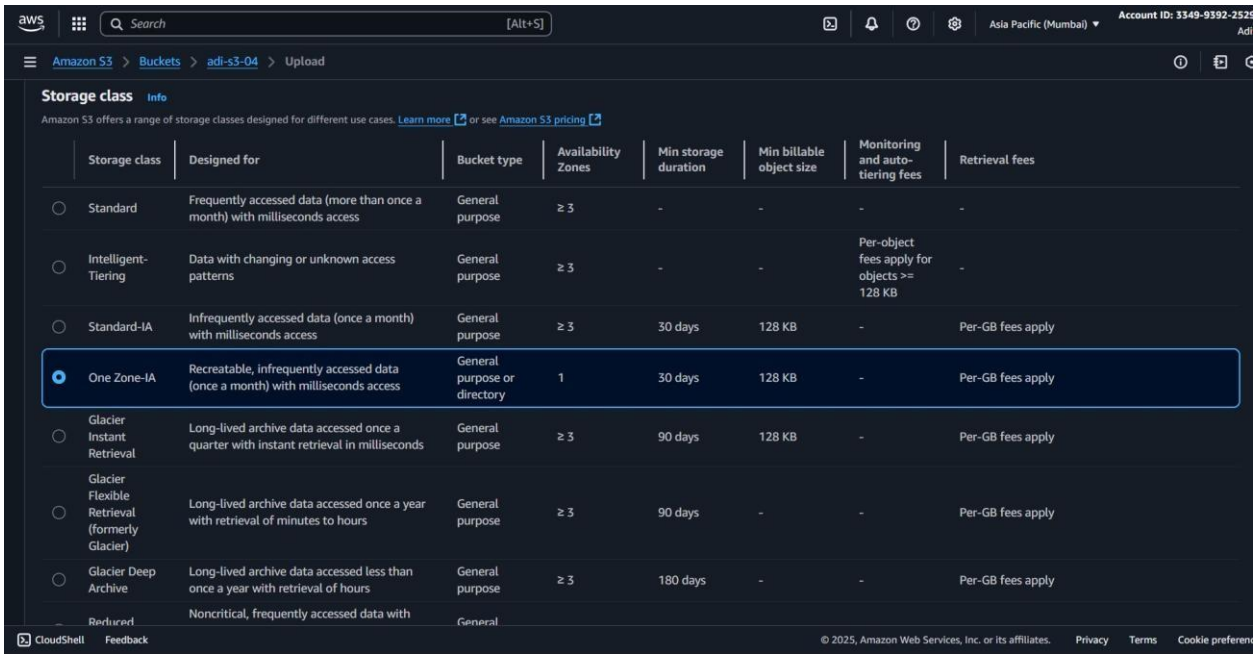
Step 4: In the **Files and folders** section, click **Add files**.



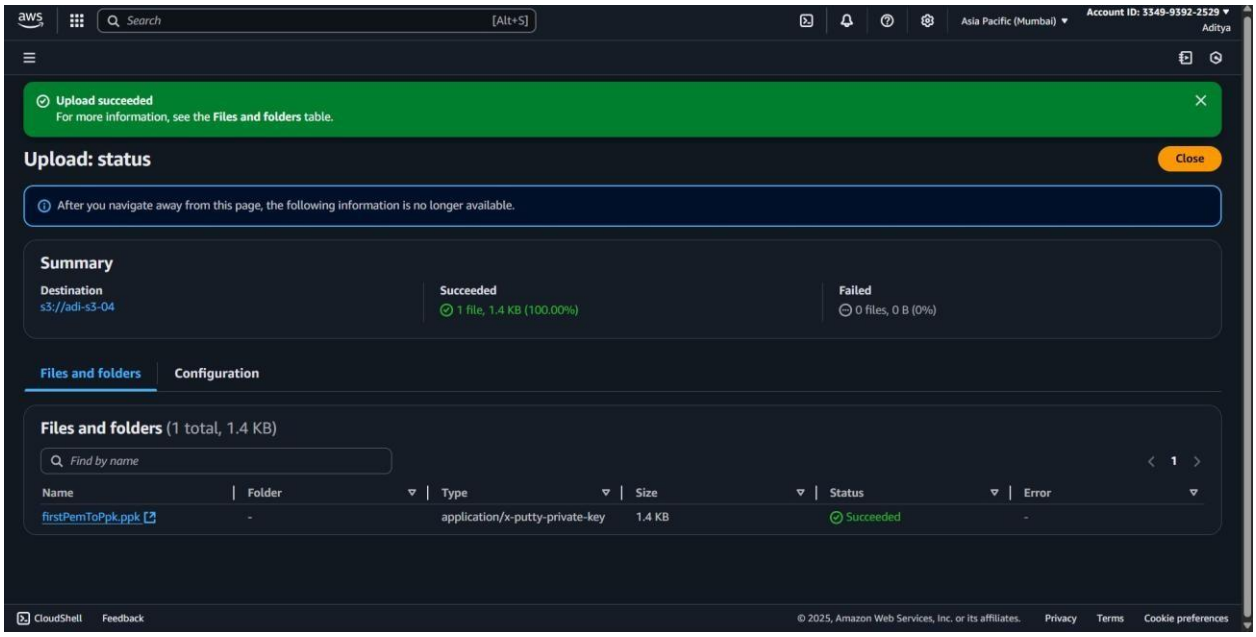
Step 5: Scroll down to **Properties.**



Step 6: select one zone



Step 7 : Scroll down and click **Upload**.



12.Change the storage class of an existing object to S3 Intelligent-Tiering.

Objective:

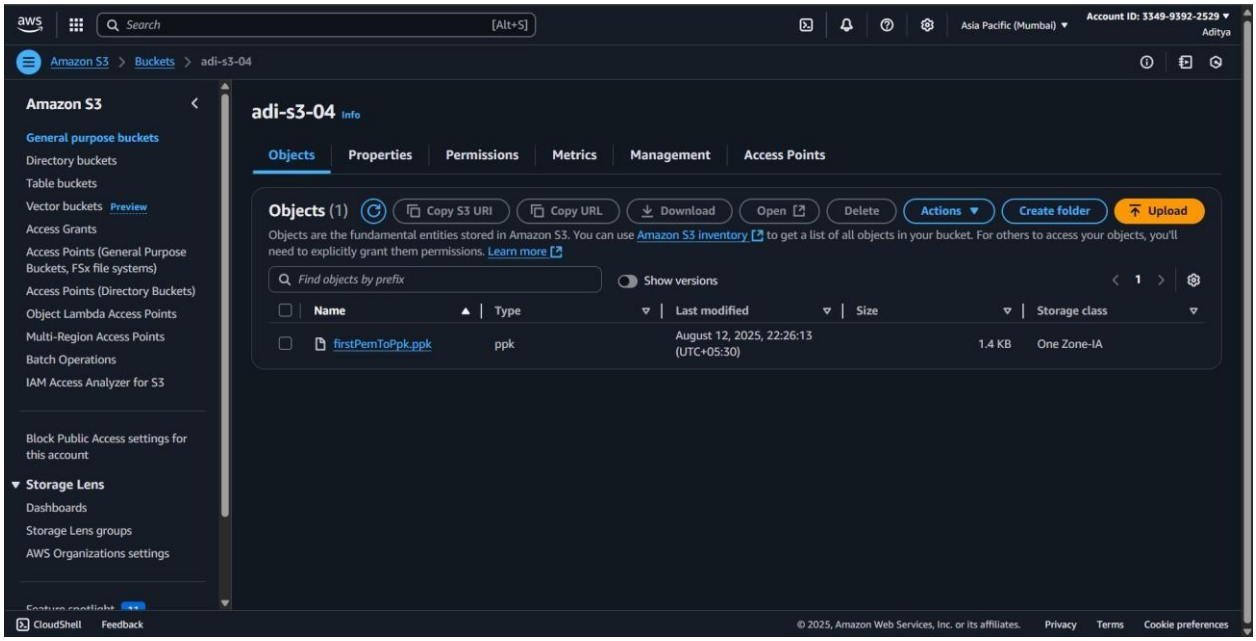
To improve storage efficiency by changing the storage class of an existing S3 object to S3 Intelligent-Tiering, enabling automatic cost optimization based on access patterns.

Prerequisite:

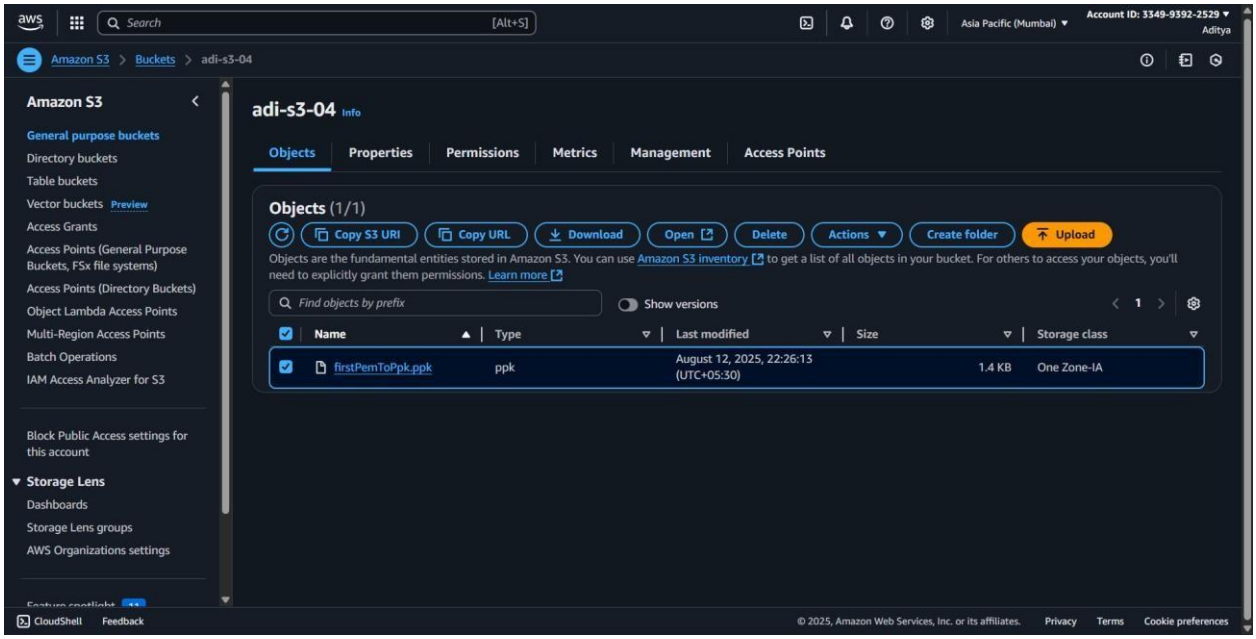
- An active AWS account.
- An existing S3 bucket with stored objects.
- Permissions to modify object storage classes.
- AWS Management Console or AWS CLI access.

Step 1: Go to the S3 Dashboard

Step 2: Open the Bucket

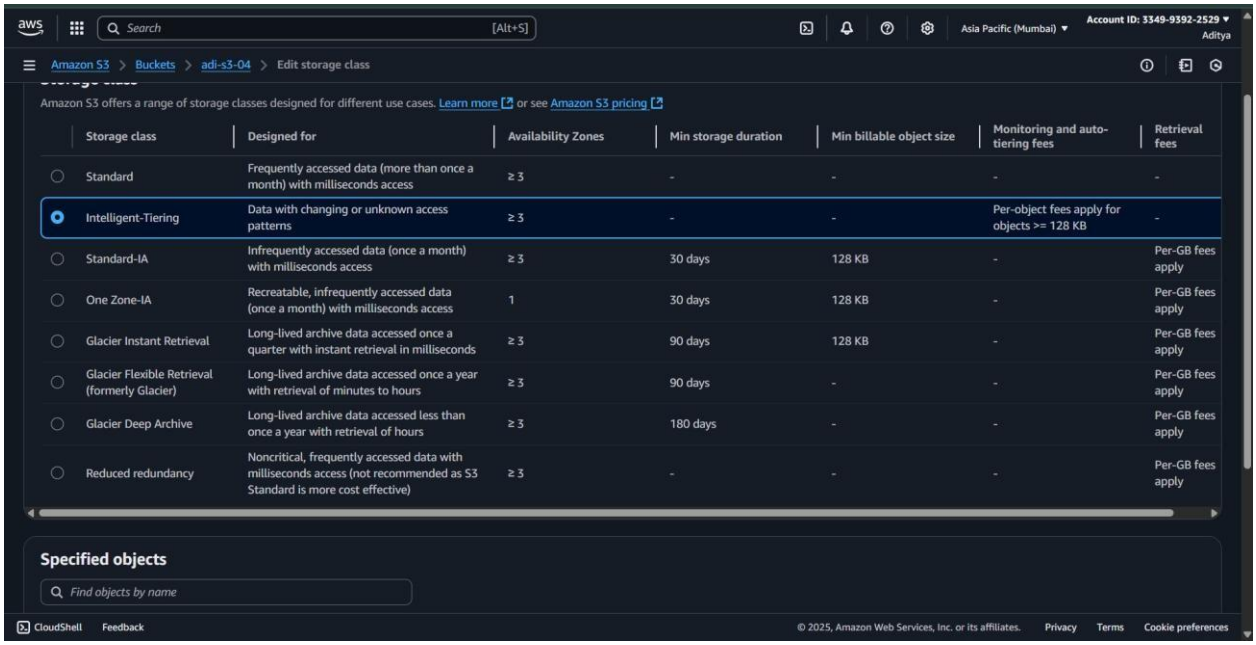


Step 3: Select bucket

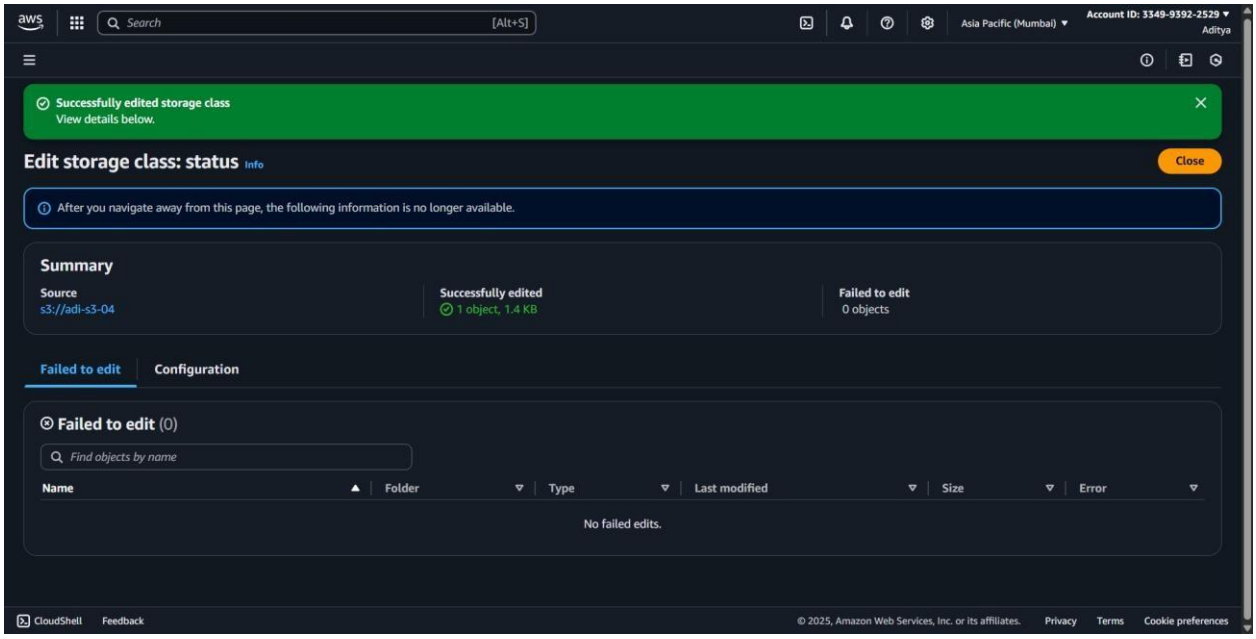


Step 4 : Click the **Actions** dropdown (top-right) and Select **Change storage class**.

Step 5: choose the s3 intelligent tiering



Step 6: save changes



13.Create a new IAM user with programmatic access and S3 read-only permissions.

Objective:

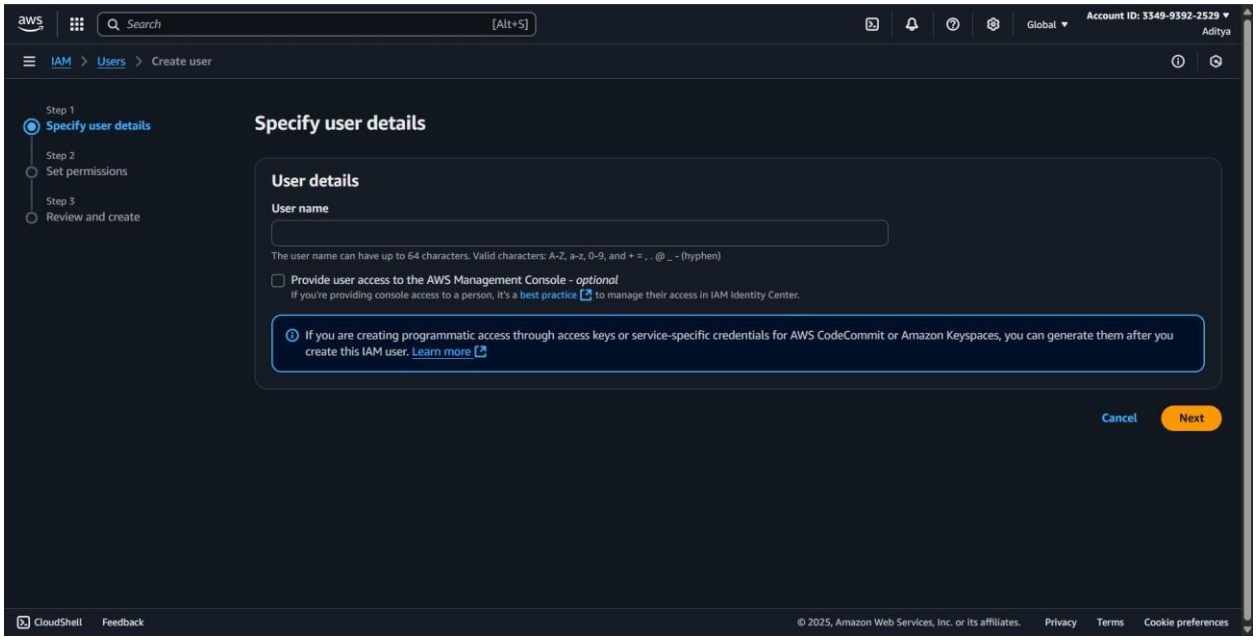
To set up a new IAM user with programmatic access and grant S3 read-only permissions, enabling secure API or CLI-based interactions without write privileges.

Prerequisite:

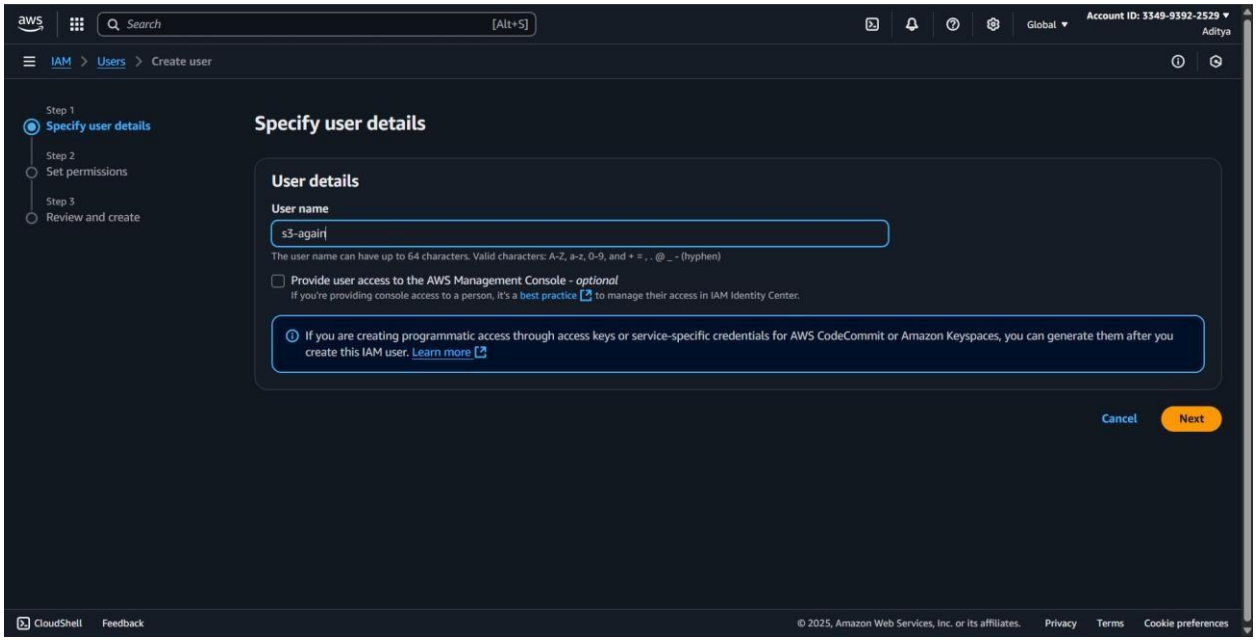
- An active AWS account.
- Permissions to create IAM users and attach policies.
- Basic understanding of programmatic access (Access Keys).
- AWS Management Console or AWS CLI access.

Step 1: In AWS Management Console, search for **IAM** and open it.

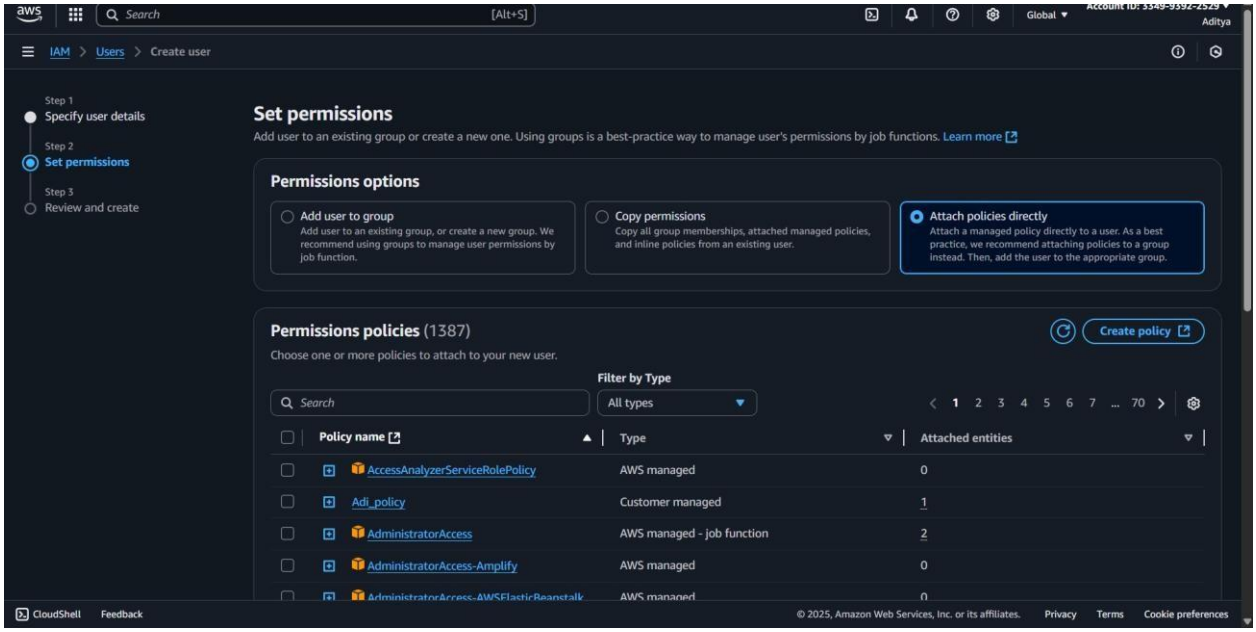
Step 2: In the left-hand menu, click **Users** and Click **Create user.**



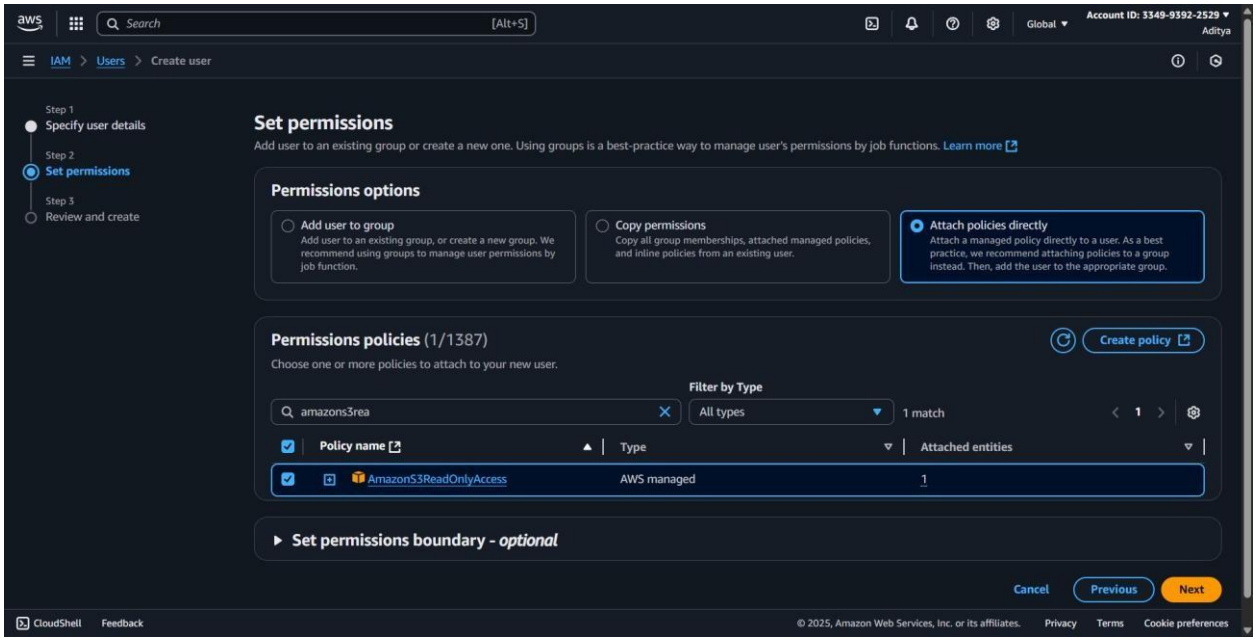
Step 3: set user details



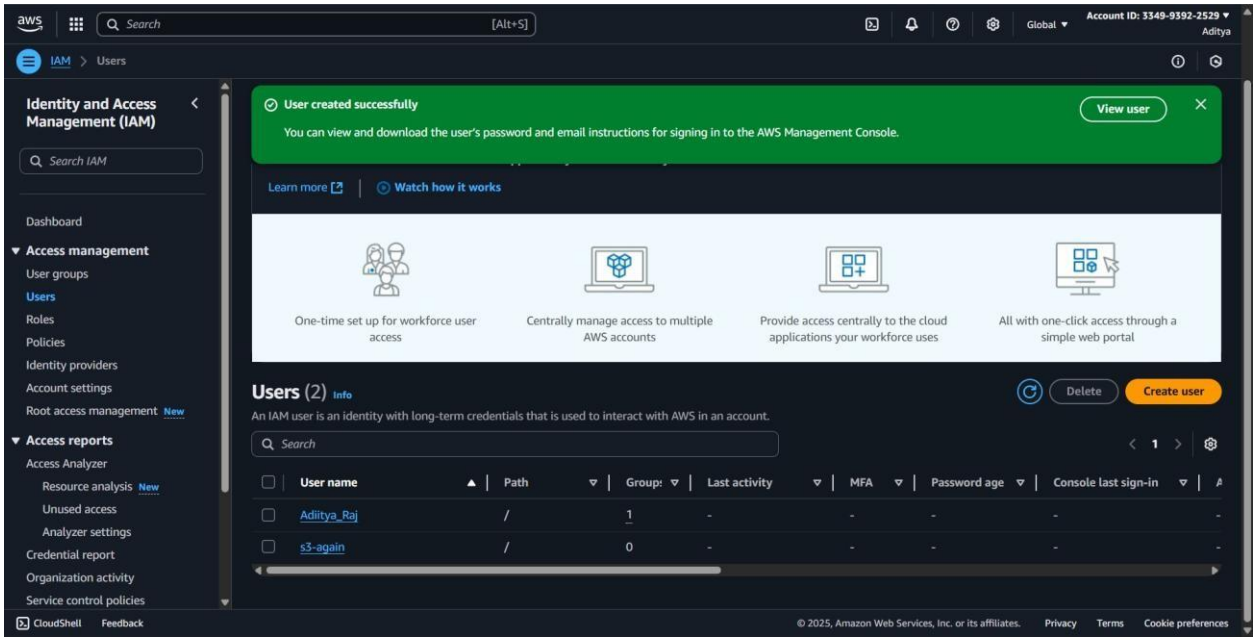
Step 4: Choose **Attach policies directly**.



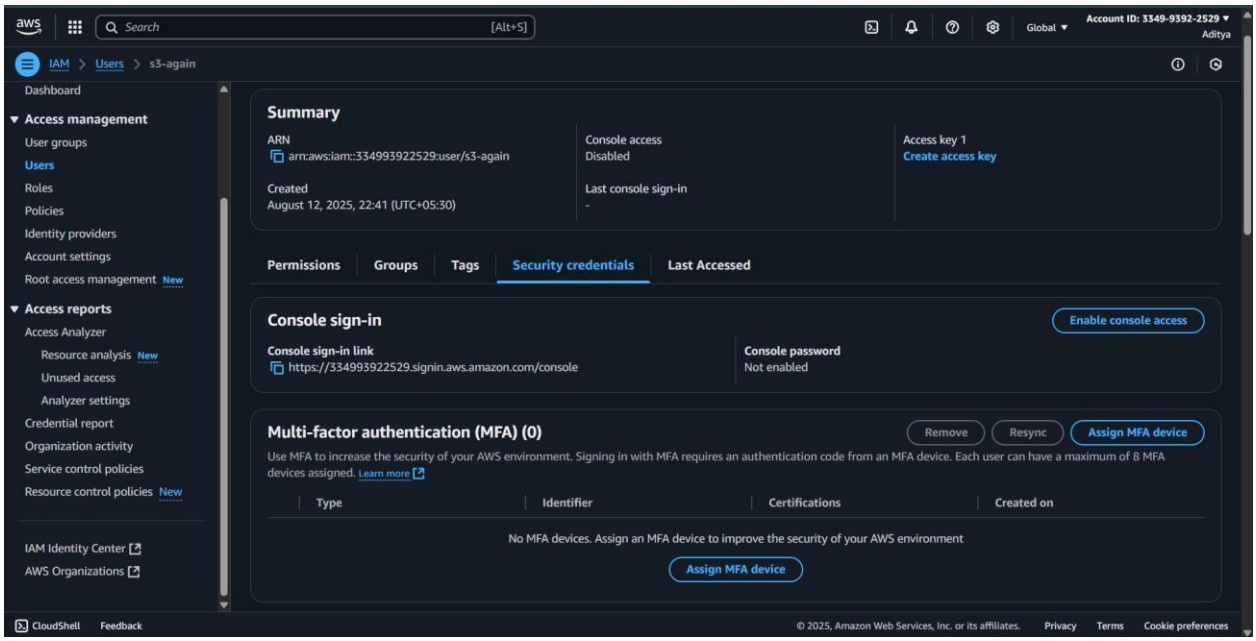
Step 5: Check the box for **AmazonS3ReadOnlyAccess**.



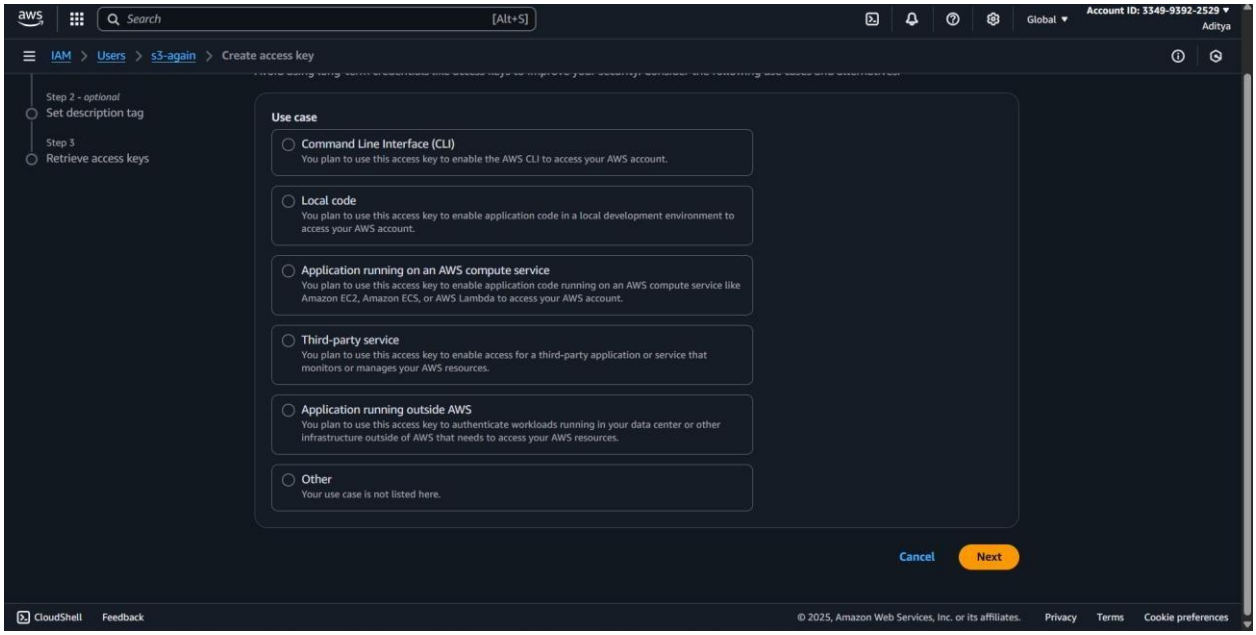
Step 6: click next and click create user



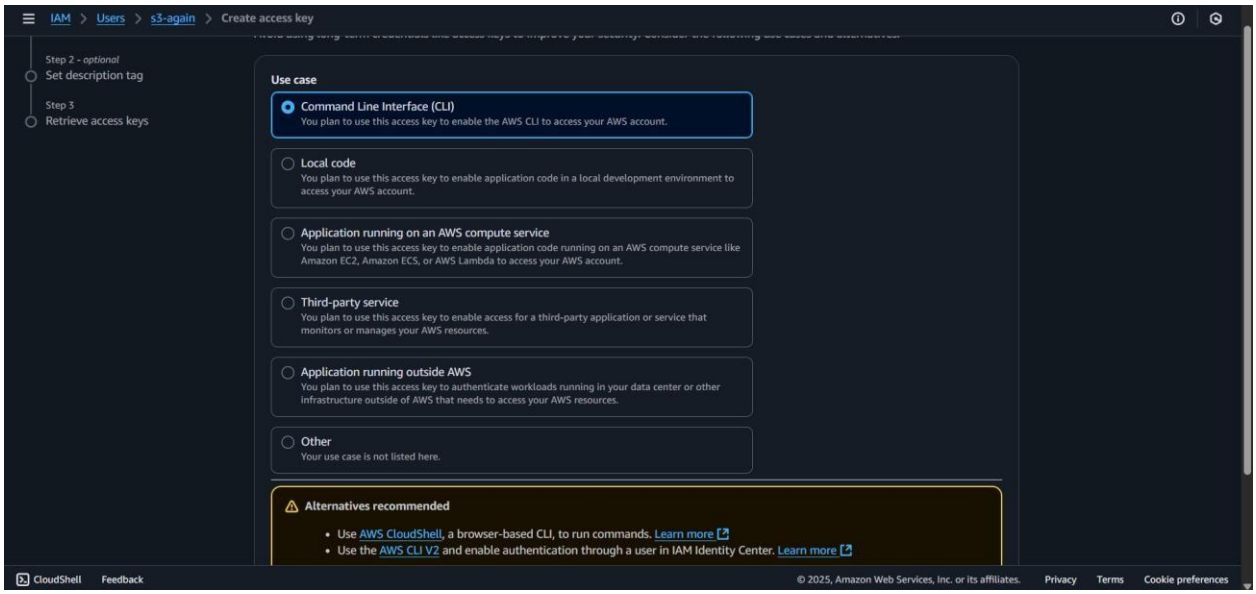
Step 7: After creation, click the user name you just created Go to **Security credentials** tab.



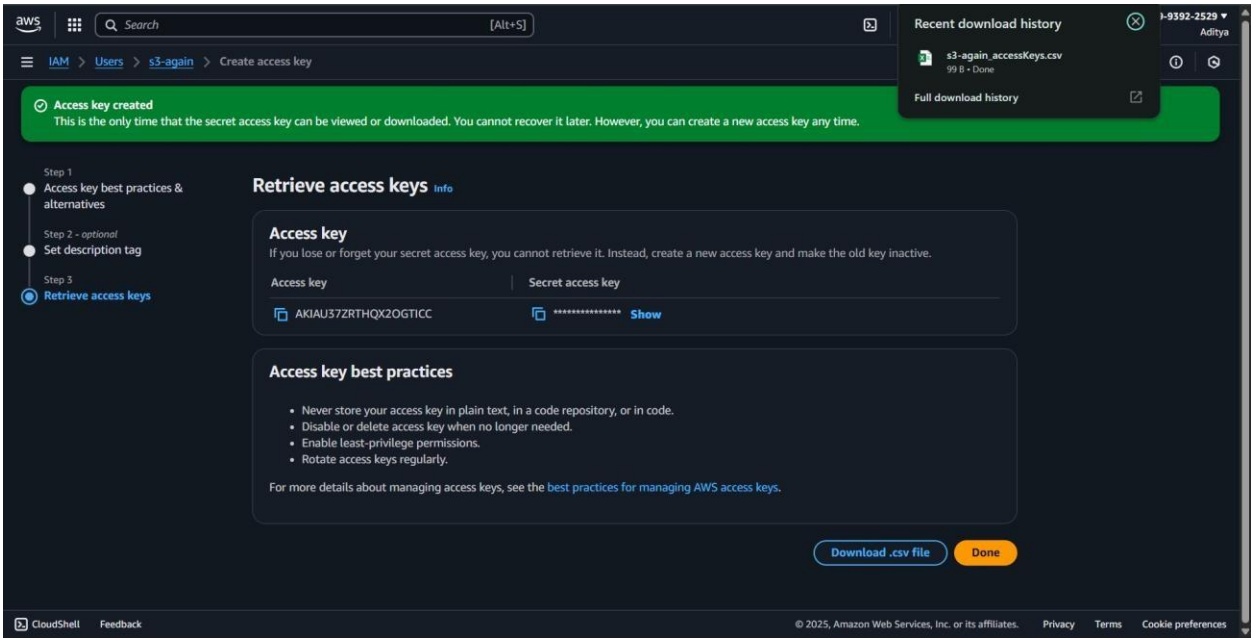
Step 8: In the **Access keys** section, click **Create access key**.



Step 9: select command line interface



Step 10: Download the .csv file containing the **Access key ID** and **Secret access key**.


Search
[Alt+S]
Recent download history
s3-again_accessKeys.csv
99 B • Done
Full download history
Aditya
IAM > Users > s3-again > Create access key
Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys
Retrieve access keys Info
Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.
Access key
Secret access key
AKIAU37ZRTHQX2OGTICC
********** Show
Access key best practices
Never store your access key in plain text, in a code repository, or in code.
Disable or delete access key when no longer needed.
Enable least-privilege permissions.
Rotate access keys regularly.
For more details about managing access keys, see the best practices for managing AWS access keys.
Download .csv file
Done
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

14.Generate Access Keys for this user. How would they use the AWS CLI to list the contents of your bucket?

Objective:

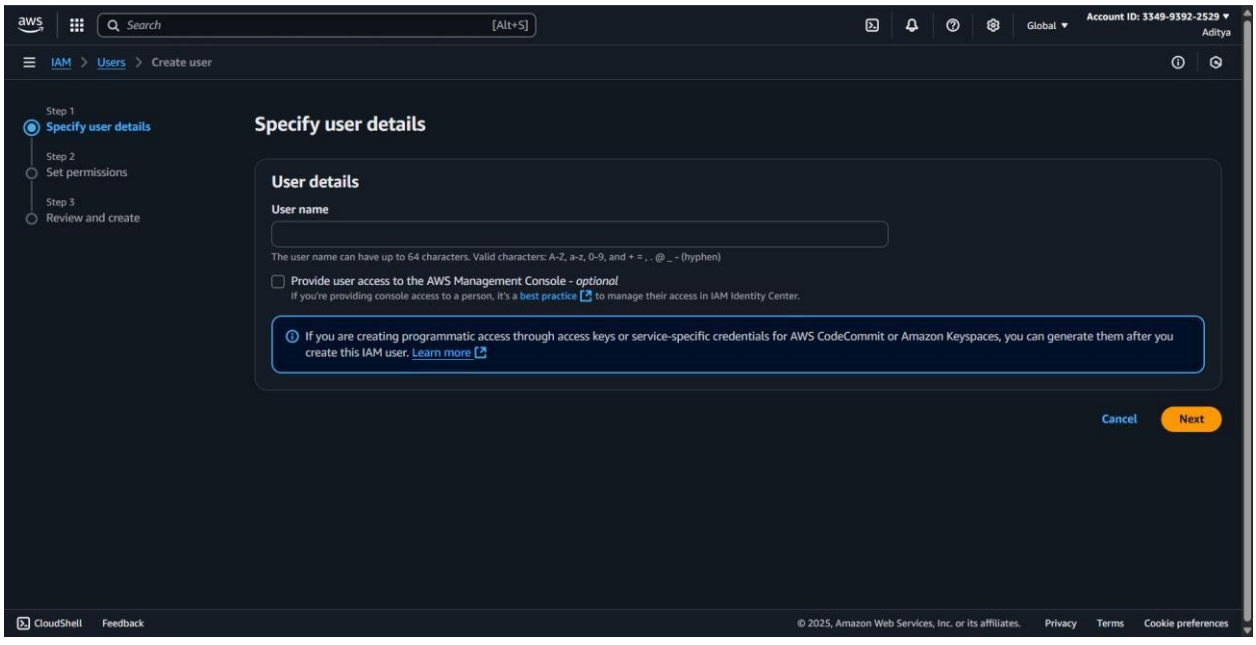
To generate access keys for an IAM user and demonstrate their usage in AWS CLI to securely list the contents of an S3 bucket.

Prerequisite:

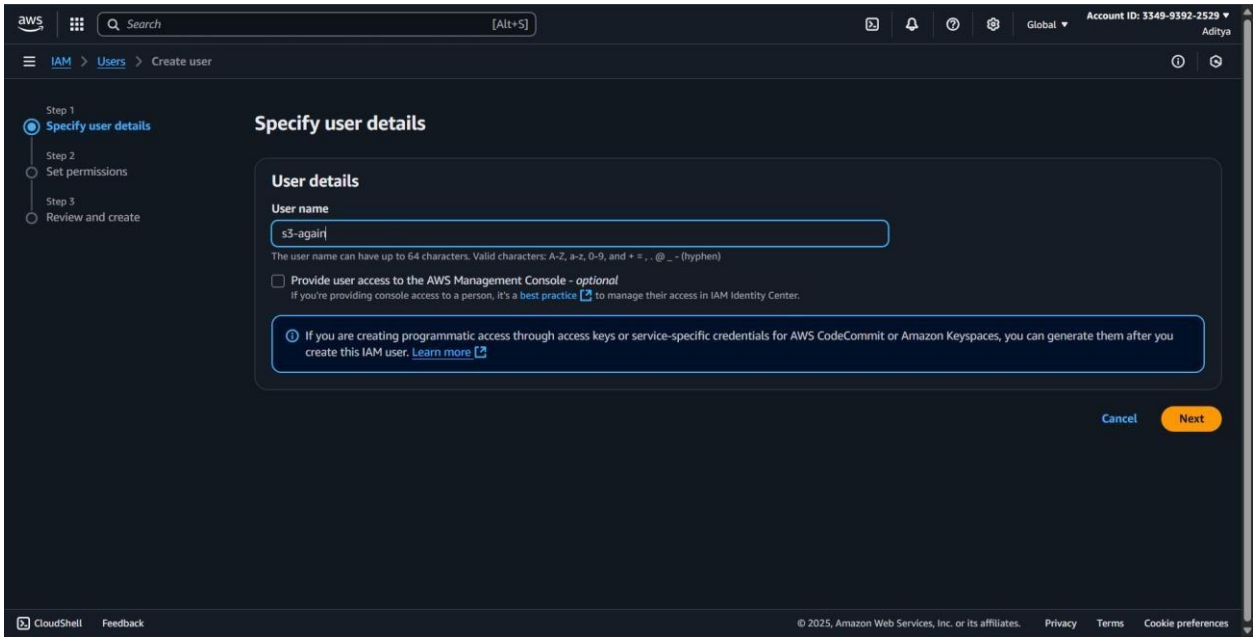
- An active AWS account.
- An IAM user with S3 read permissions.
- AWS CLI installed and configured.
- Basic knowledge of AWS CLI commands.

Step 1: In AWS Management Console, search for **IAM** and open it.

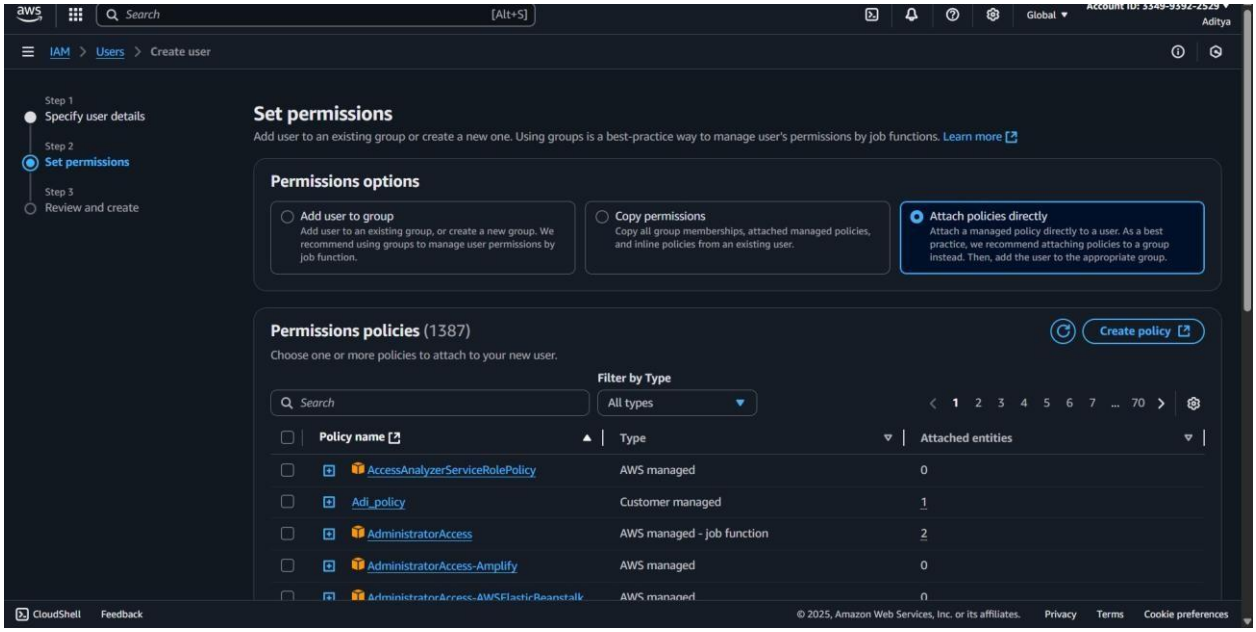
Step 2: In the left-hand menu, click **Users** and Click **Create user.**



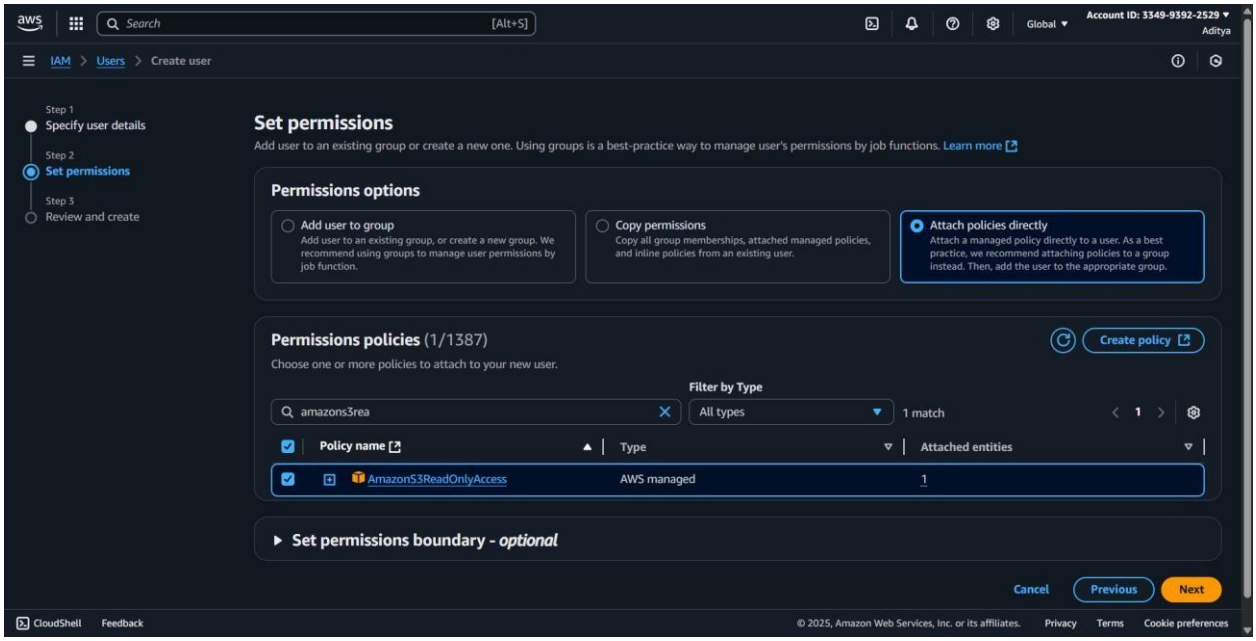
Step 3: set user details



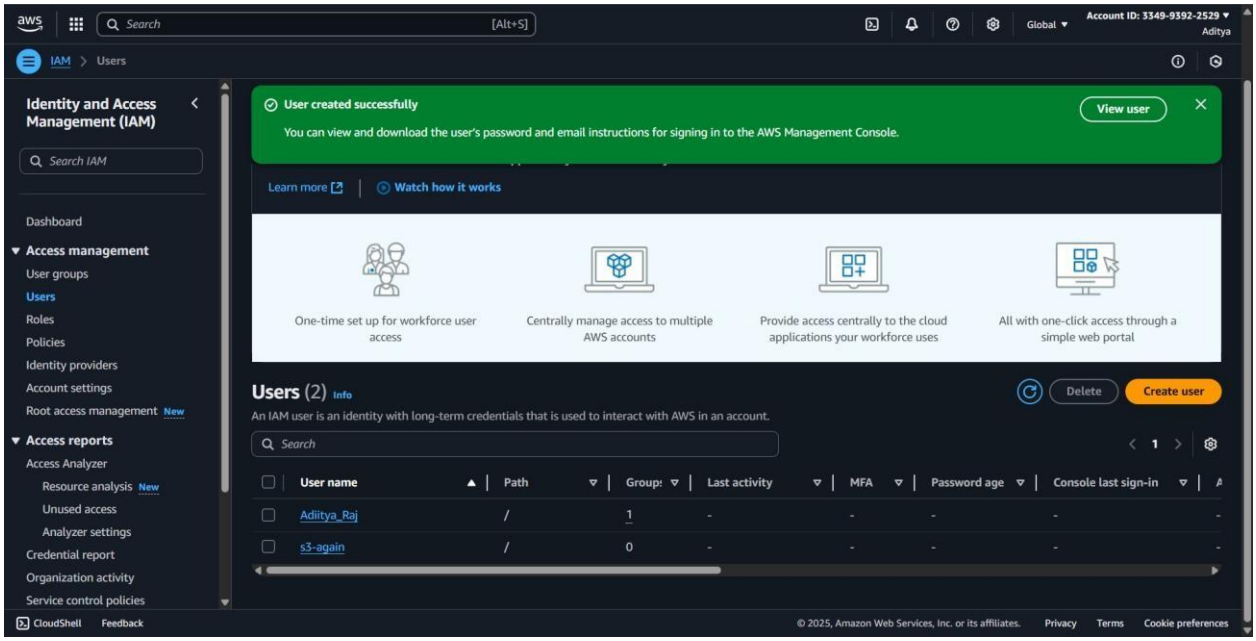
Step 4: Choose **Attach policies directly**.



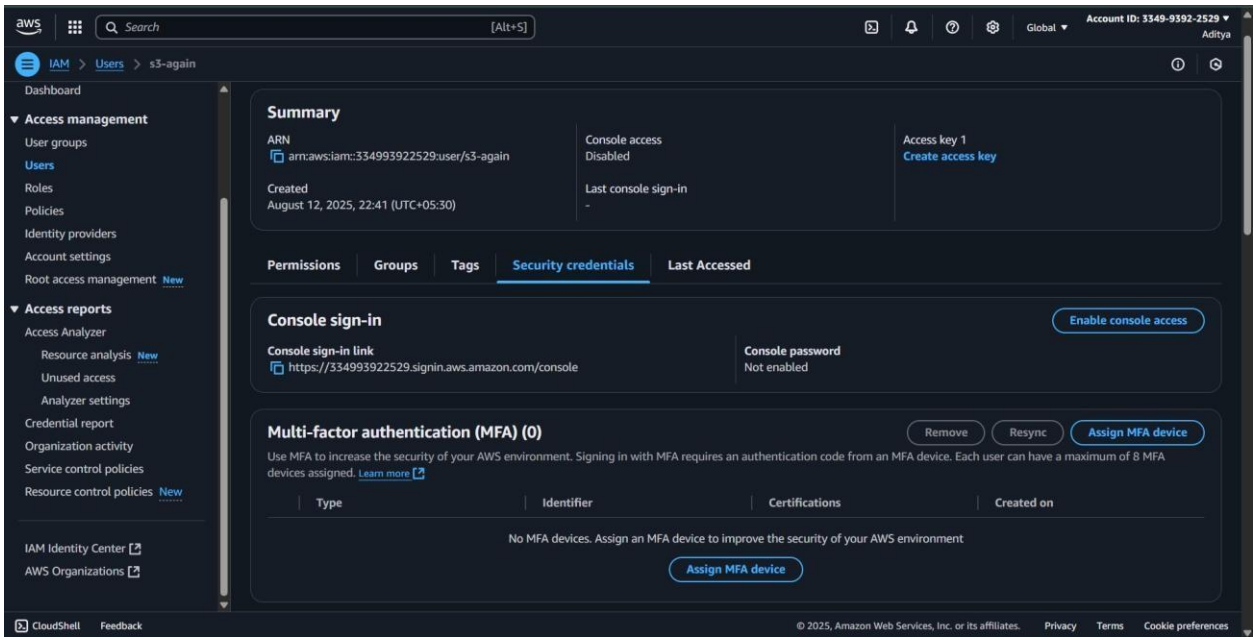
Step 5: Check the box for **AmazonS3ReadOnlyAccess**.



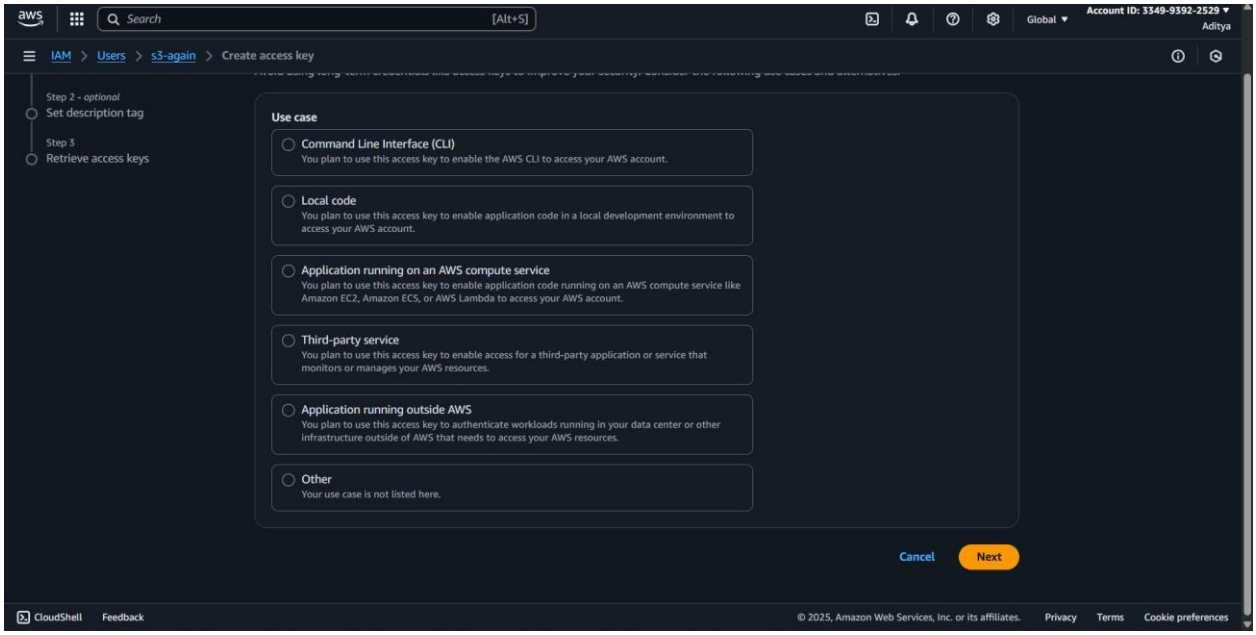
Step 6: click next and click create user



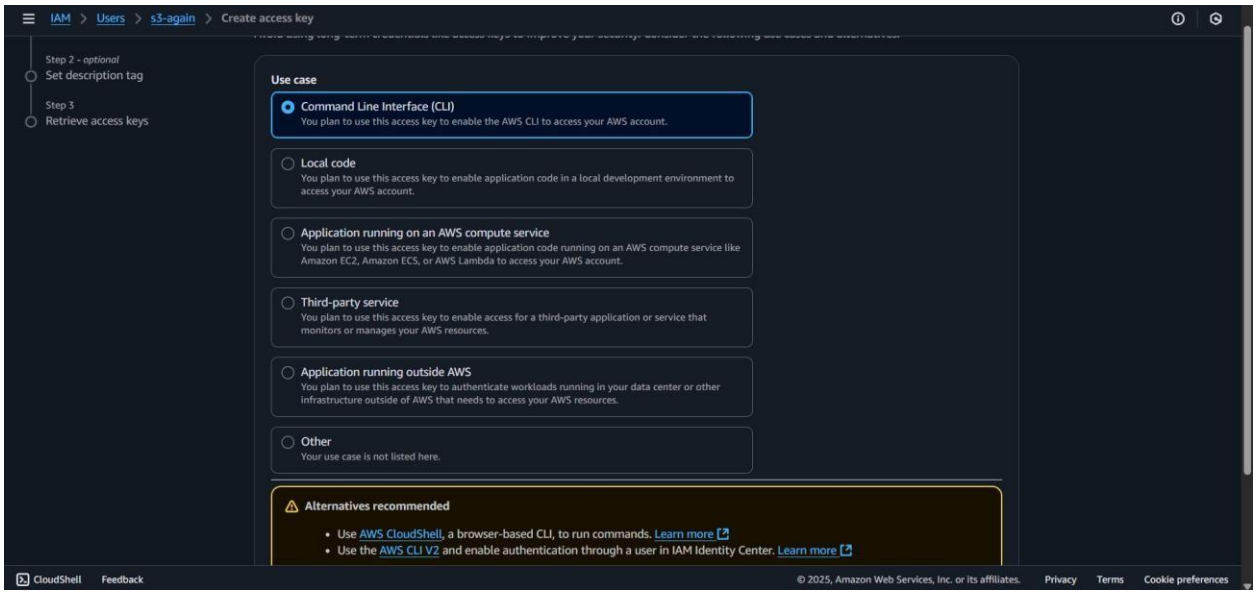
Step 7: After creation, click the user name you just created Go to **Security credentials** tab.



Step 8: In the **Access keys** section, click **Create access key**.

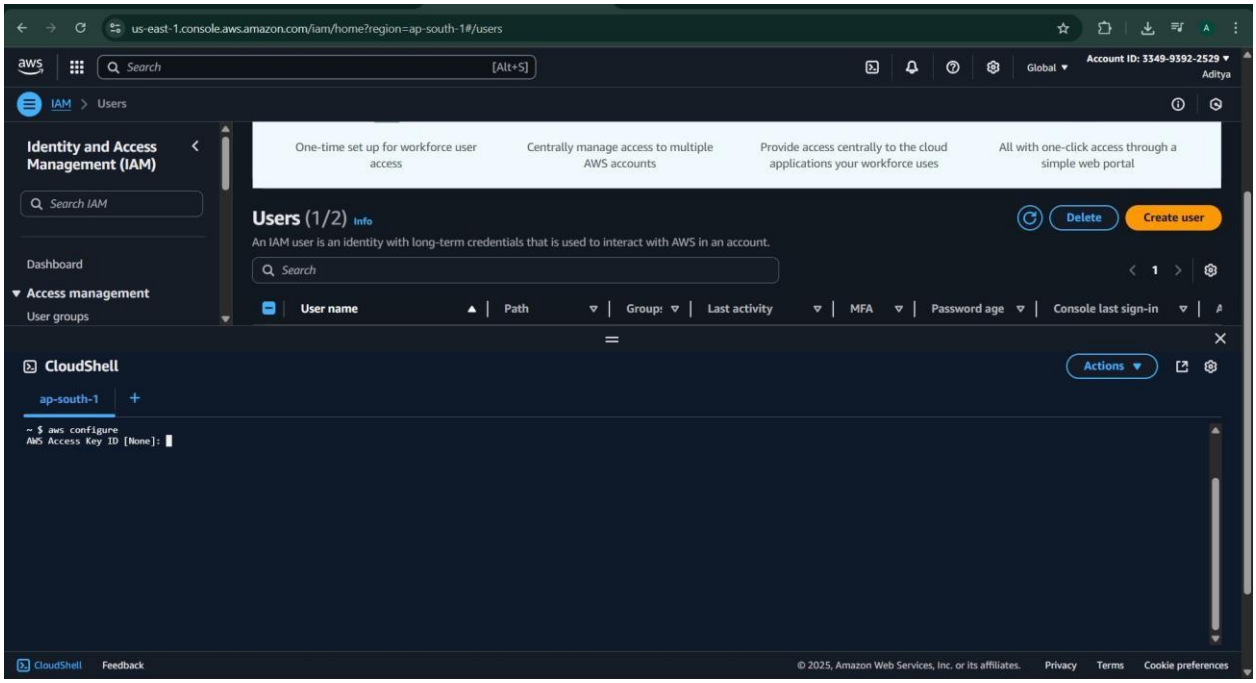


Step 9: select command line interface



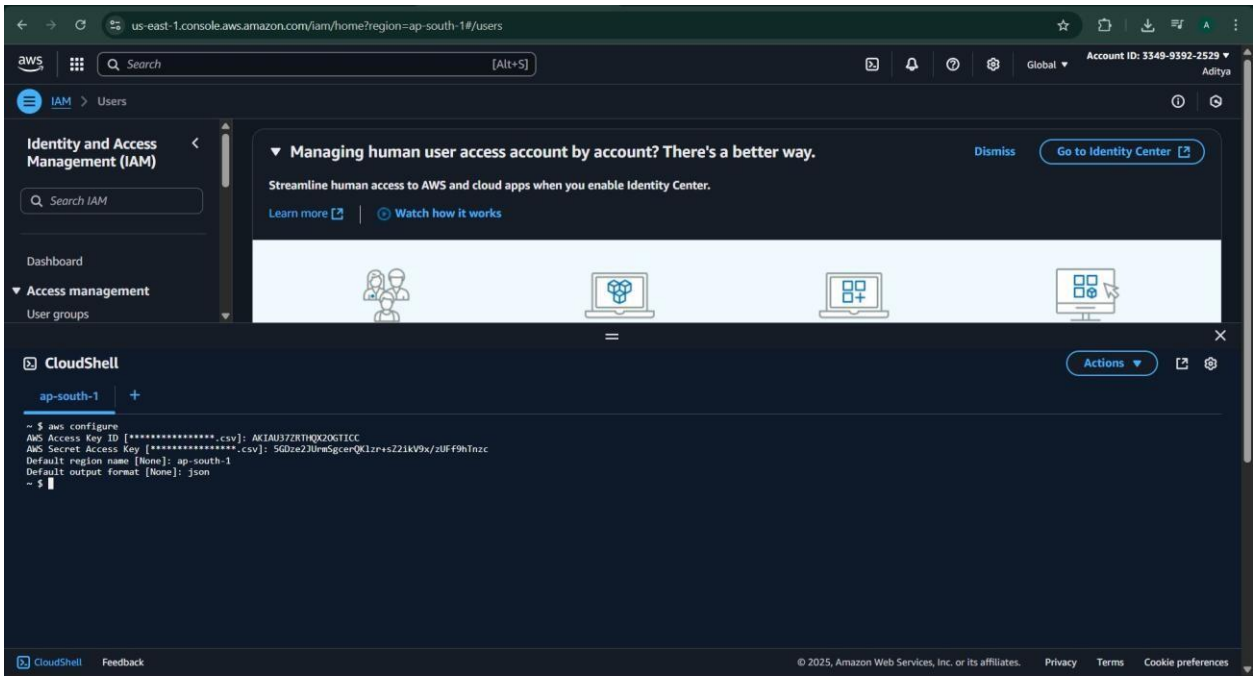
Step 10: Download the .csv file containing the **Access key ID** and **Secret access key**.

Step 11: **Open a Terminal or Command Prompt**.

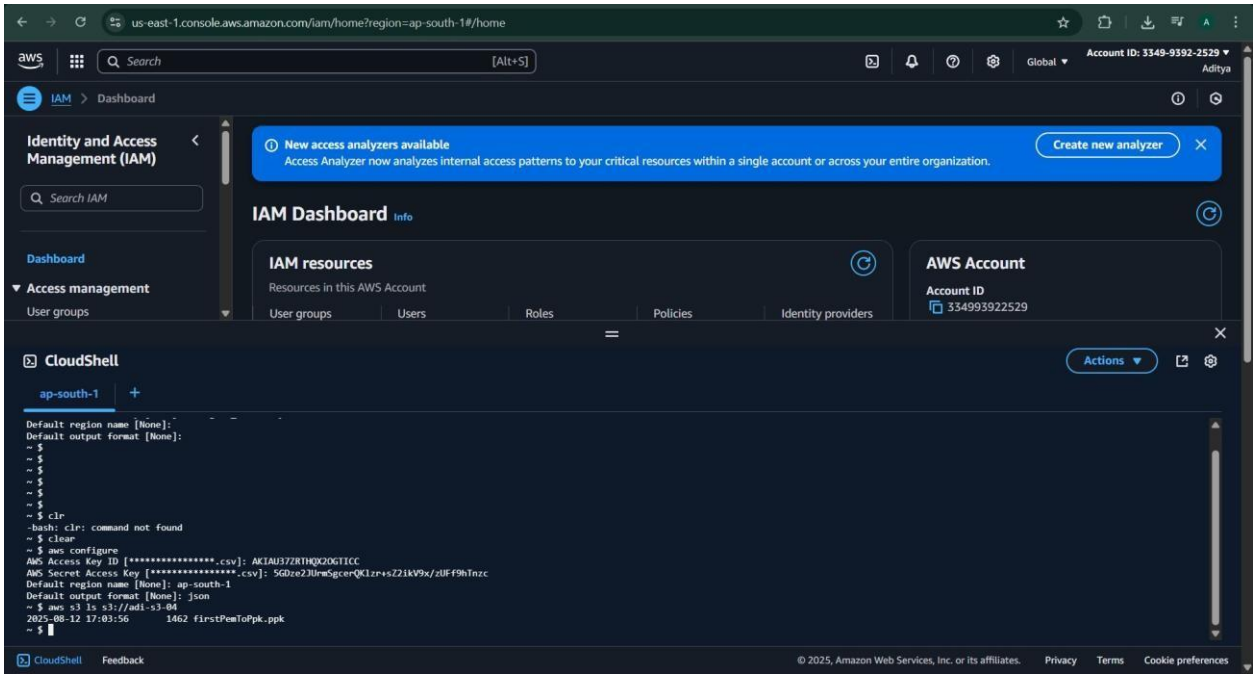


Step 12: Run aws configure.

Step 13: fill essential details in the cloudshell



Step 14: aws s3 ls s3://your-bucket-name



15.Create a bucket policy to allow public read access to all objects in the bucket.

Objective:

To configure a bucket policy that grants public read access to all objects in an S3 bucket, enabling unrestricted object viewing over the internet.

Prerequisite:

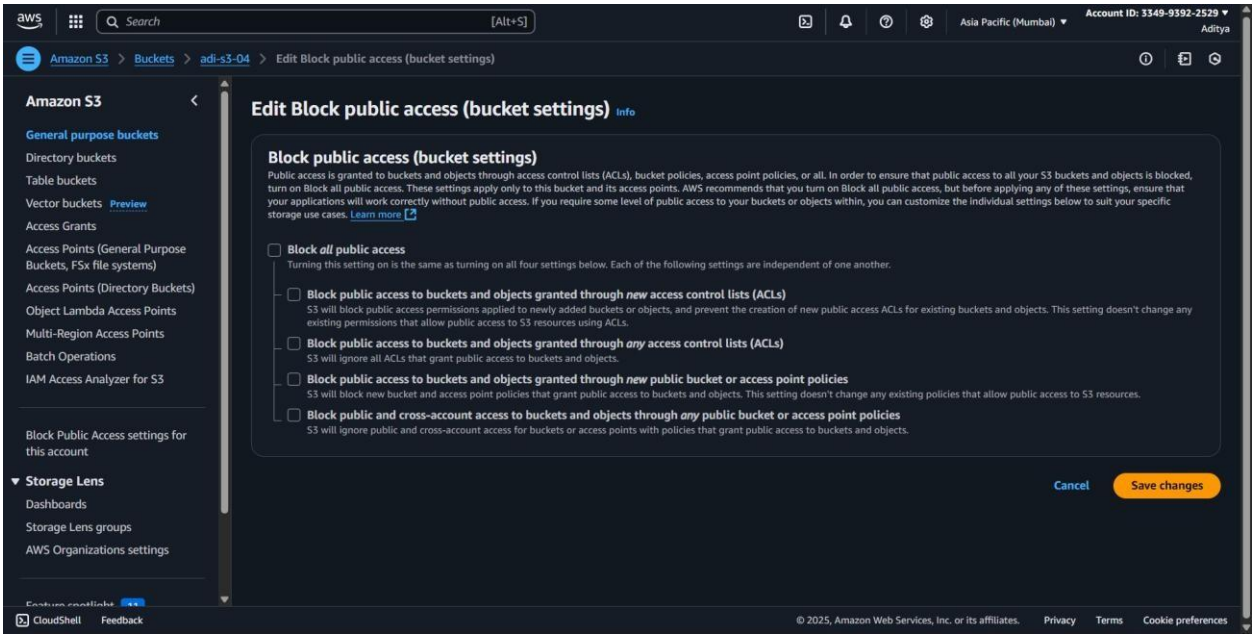
- An active AWS account.
- An existing S3 bucket.
- Permissions to edit bucket policies.
- Basic knowledge of JSON policy syntax.
- AWS Management Console or AWS CLI access.

Step 1: Go to the S3 Dashboard.

Step 2: Open Your Bucket

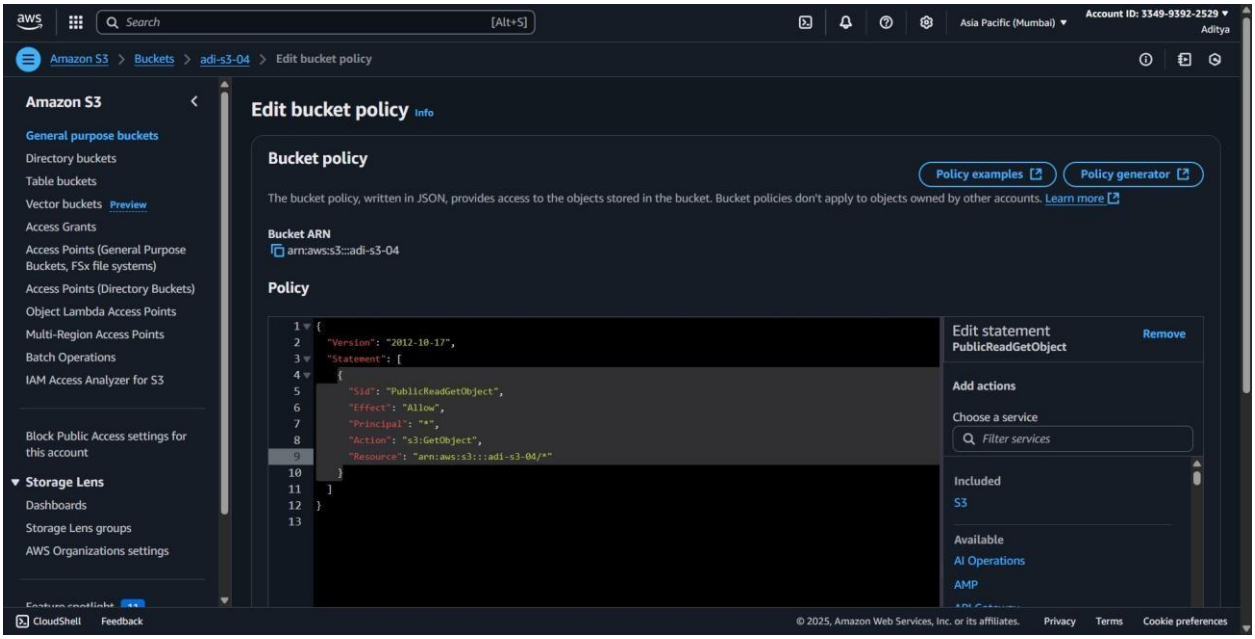
Step 3: **Enable Public Access**

- Go to **Permissions** tab.
- Under **Block public access (bucket settings)**, click **Edit**.
- Turn off **Block all public access** (uncheck it) → confirm by typing **confirm** → Save changes.

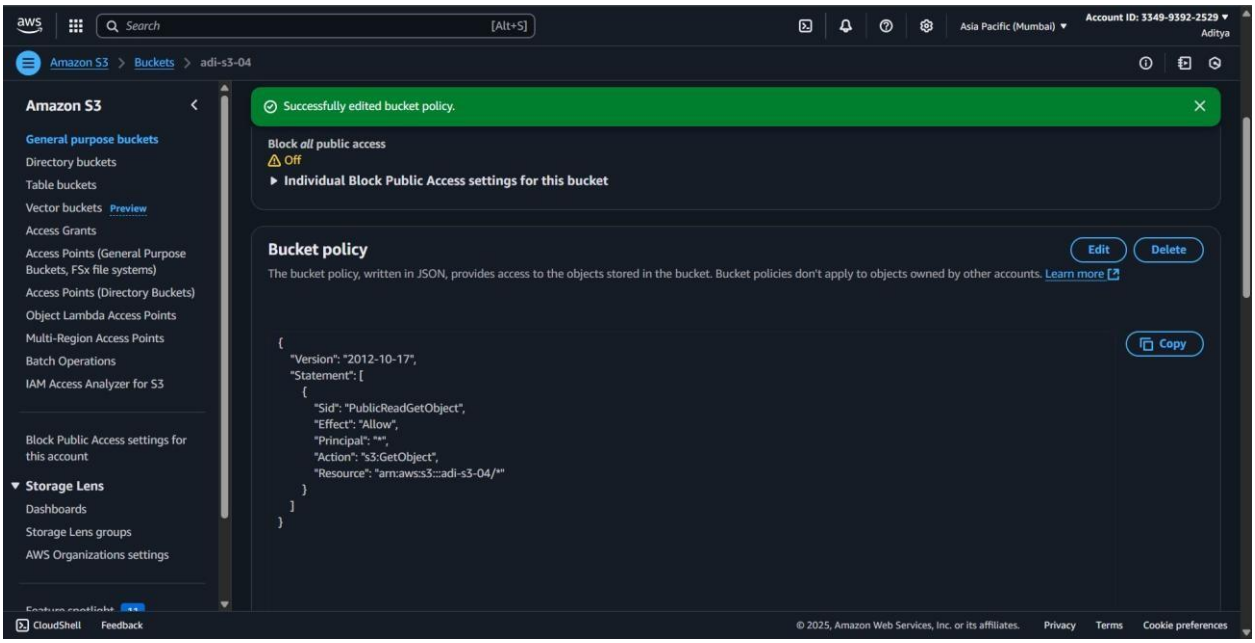


Step 4: **Go to Bucket Policy**

In **Permissions** tab, scroll to **Bucket policy** section → click **Edit**.



Step 5: save policy



16.Enable server access logging for your S3 bucket. Where are the logs stored?

Objective:

To enable server access logging for an S3 bucket in order to track detailed access requests, and identify the location where these logs are stored.

Prerequisite:

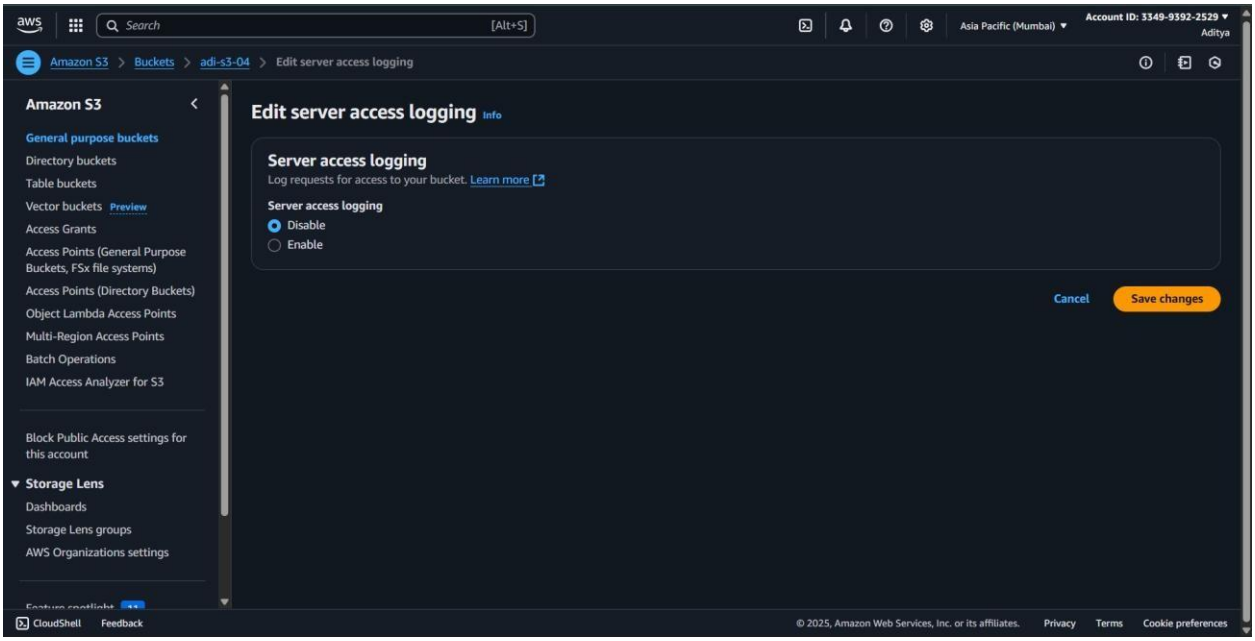
- An active AWS account.
- An existing S3 bucket.
- A target S3 bucket to store access logs.
- Permissions to modify bucket settings and enable logging.
- AWS Management Console or AWS CLI access.

Step 1: Go to s3 dashboard Step

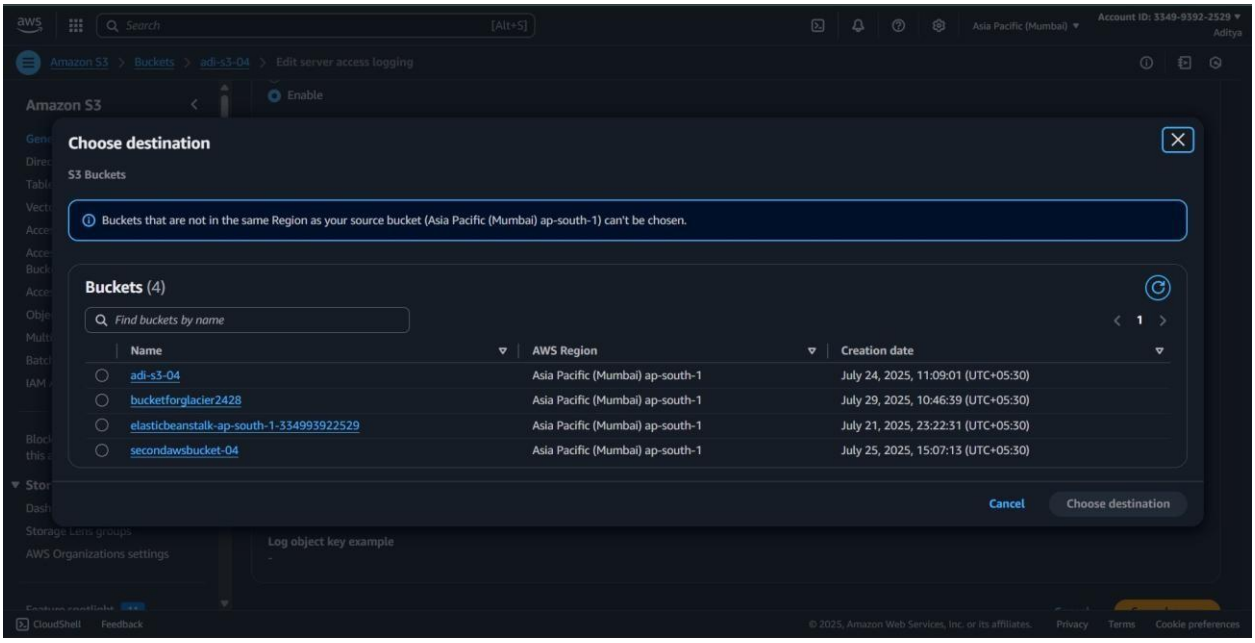
2: Open Your Bucket.

Step 3: Go to Properties

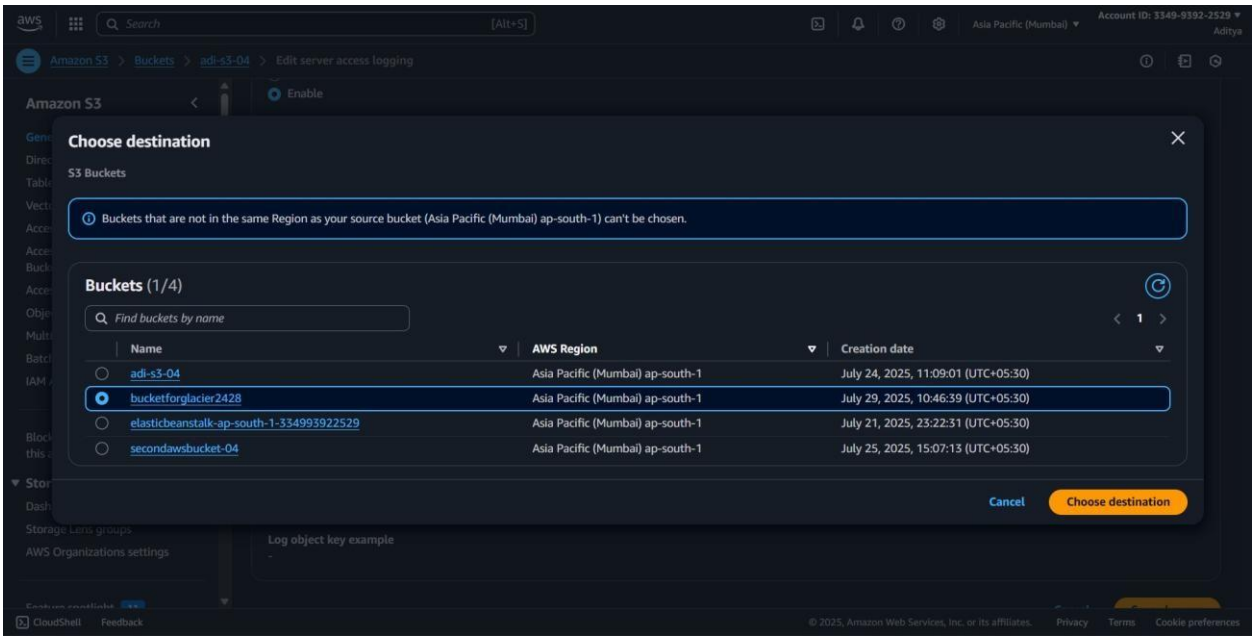
Step 4: Scroll down to **Server access logging** → click **Edit**.



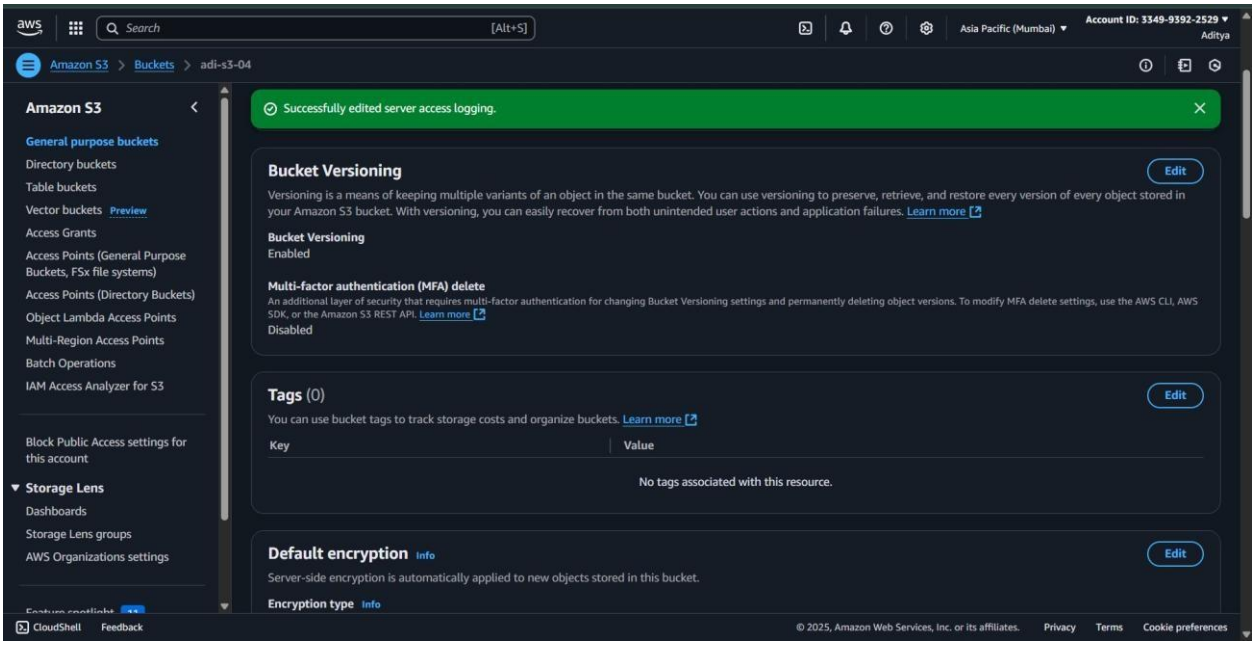
Step 5: Enable Logging.



Step 6: choose another S3 bucket where logs will be stored.



Step7: save changes:



17.Write a simple handler function that returns: "Welcome to VIT BHOPAL!"

Objective:

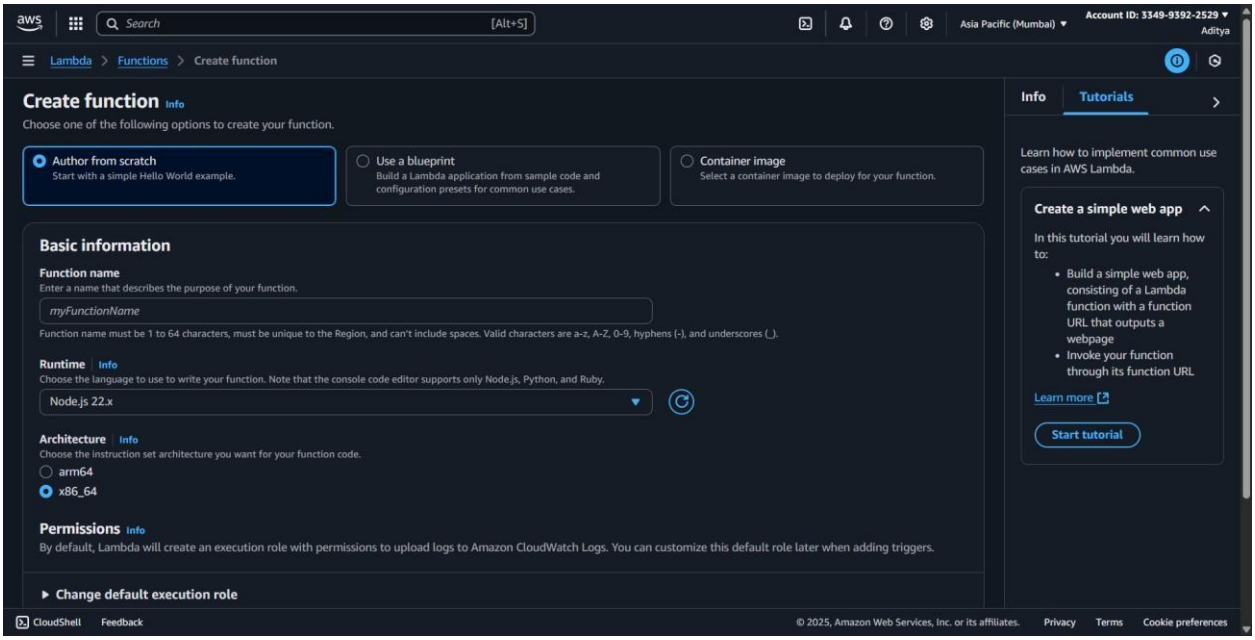
To create a basic handler function that outputs the message *"Welcome to VIT BHOPAL!"*, demonstrating the fundamentals of function creation and return values.

Prerequisite:

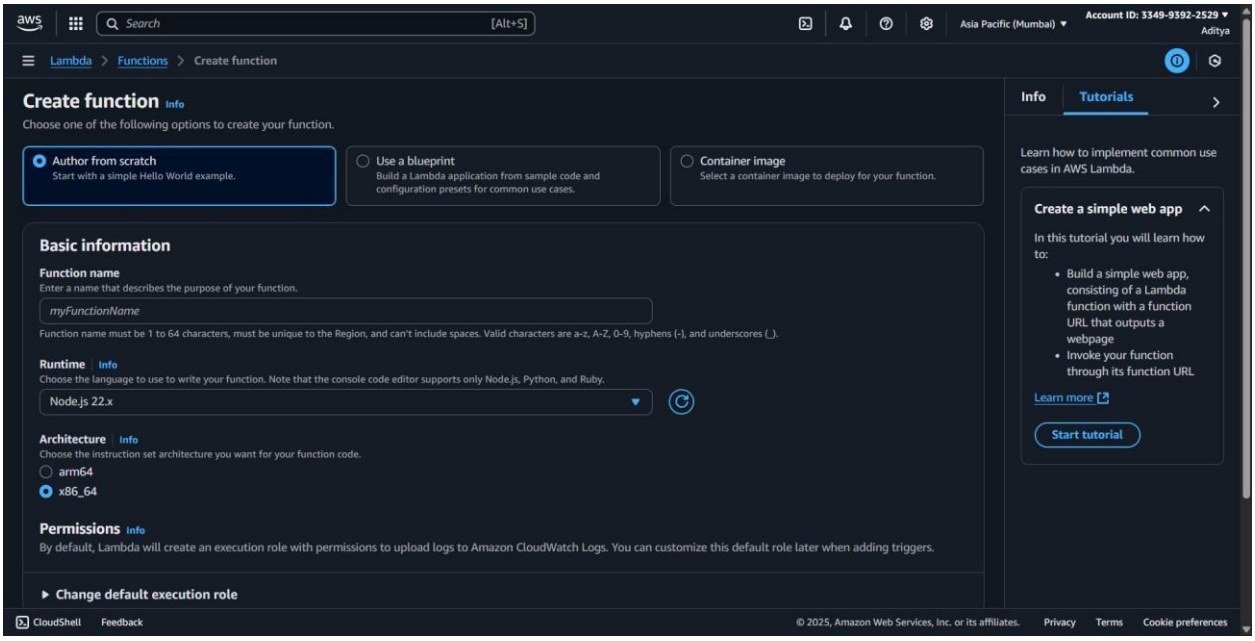
- A programming environment set up (e.g., AWS Lambda, Python, or Node.js runtime).
- Basic understanding of function syntax in the chosen language.
- Access to the development or deployment platform.

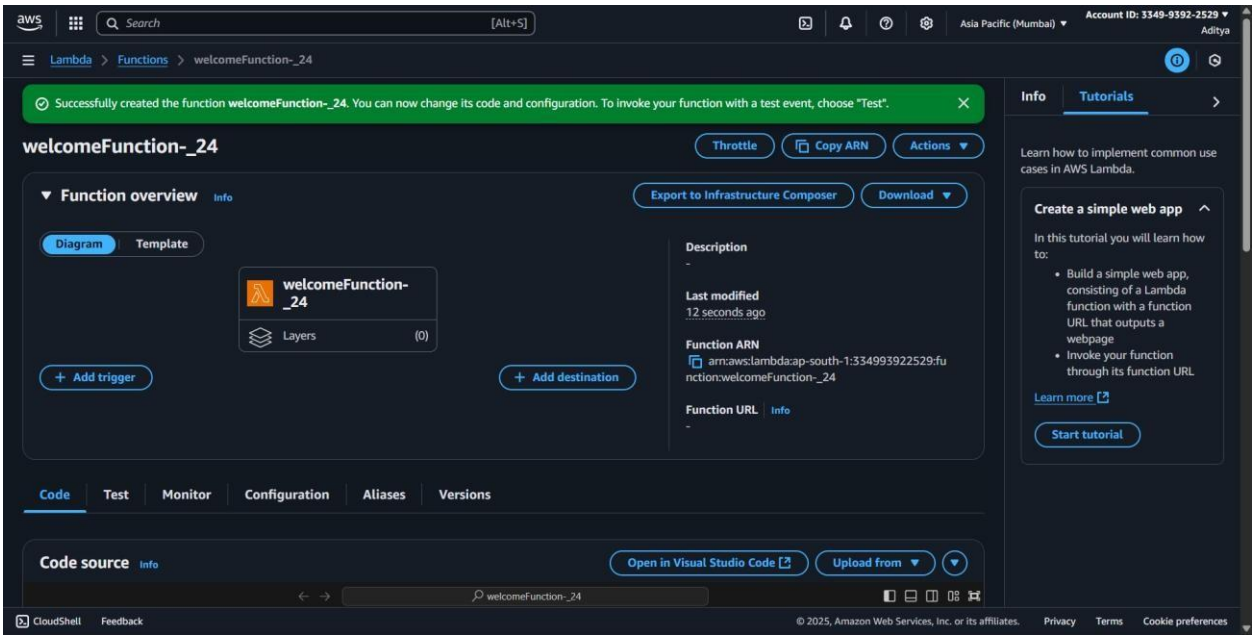
Step 1: Go to **AWS Management Console** → search **Lambda**.

Step 2: Click **Create function** → **Author from scratch**.

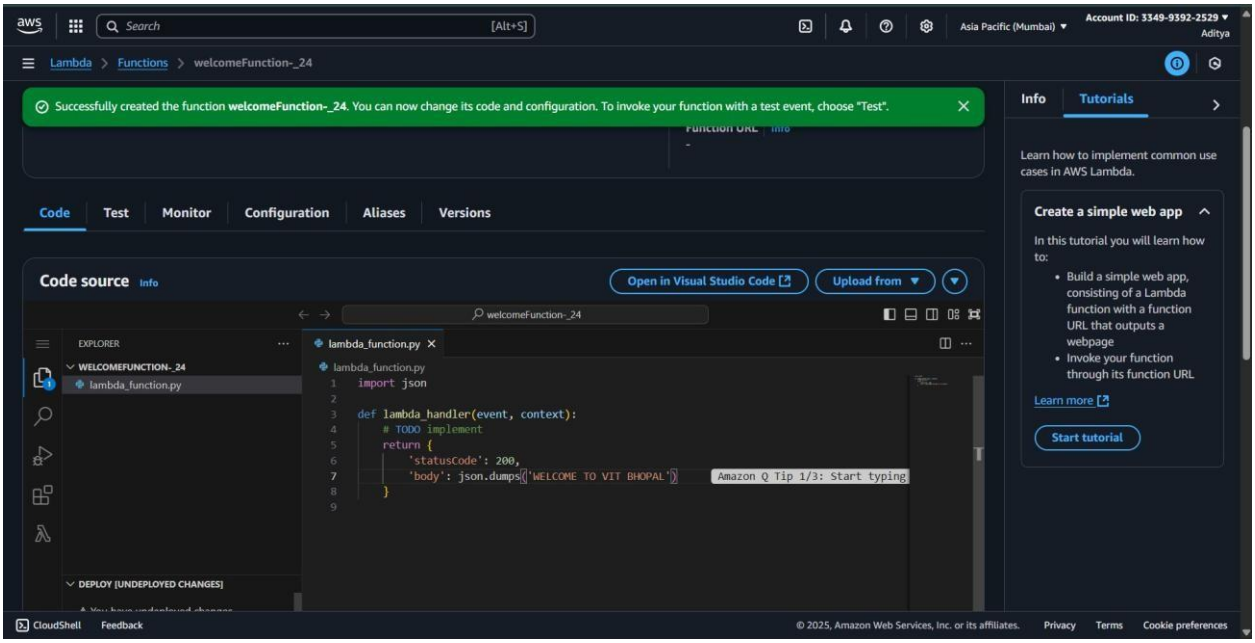


Step 3: create function

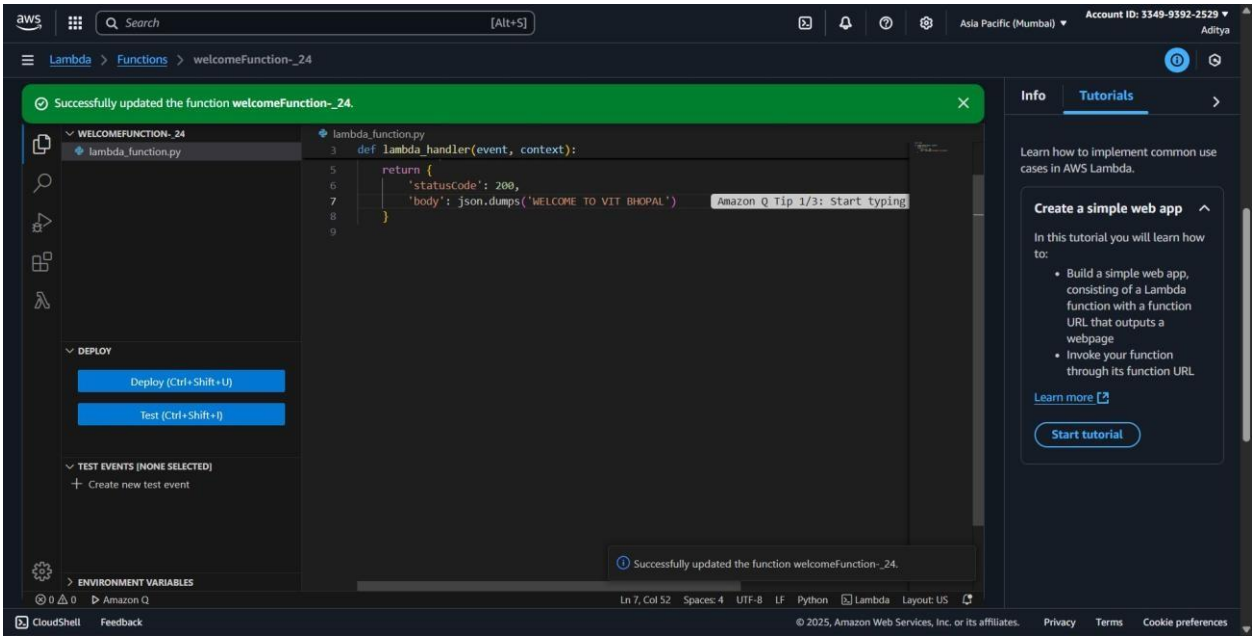




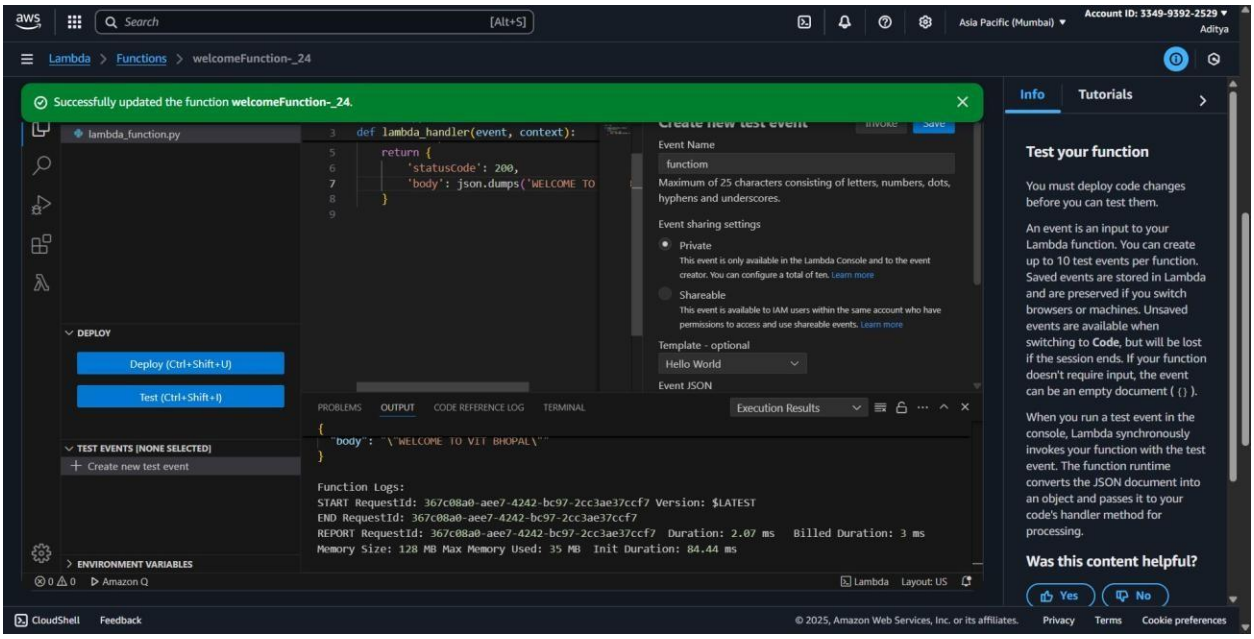
Step 4: change the code



Step 5: click deploy



Step 6: Click **Test**


Search
[Alt+S]
Asia Pacific (Mumbai)
Account ID: 3349-9392-2529
Aditya
Lambda > Functions > welcomeFunction-_24
Successfully updated the function welcomeFunction-_24.
lambda_function.py
def lambda_handler(event, context):
return {
'statusCode': 200,
'body': json.dumps('WELCOME TO
}
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
Create new test event
ENVIRONMENT VARIABLES
Create new test event
Invoke
Save
Event Name
functiom
Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.
Event sharing settings
Private
This event is only available in the Lambda Console and to the event creator. You can configure a total of ten. Learn more
Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more
Template - optional
Hello World
Event JSON
PROBLEMS
OUTPUT
CODE REFERENCE LOG
TERMINAL
Execution Results
{
"body": "\"WELCOME TO VIT BHOPAL\""
}
Function Logs:
START RequestId: 367c08a0-aee7-4242-bc97-2cc3ae37ccf7 Version: $LATEST
END RequestId: 367c08a0-aee7-4242-bc97-2cc3ae37ccf7
REPORT RequestId: 367c08a0-aee7-4242-bc97-2cc3ae37ccf7 Duration: 2.07 ms Billed Duration: 3 ms
Memory Size: 128 MB Max Memory Used: 35 MB Init Duration: 84.44 ms
0 0
Amazon Q
Lambda
Layout: US
Info
Tutorials
Test your function
You must deploy code changes before you can test them.
An event is an input to your Lambda function. You can create up to 10 test events per function. Saved events are stored in Lambda and are preserved if you switch browsers or machines. Unsaved events are available when switching to Code, but will be lost if the session ends. If your function doesn't require input, the event can be an empty document ({}).
When you run a test event in the console, Lambda synchronously invokes your function with the test event. The function runtime converts the JSON document into an object and passes it to your code's handler method for processing.
Was this content helpful?
Yes
No
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

18.Launch a Linux EC2 instance into the Public Subnet.

Objective:

To deploy a Linux-based EC2 instance within a public subnet, enabling direct internet access for management and application hosting.

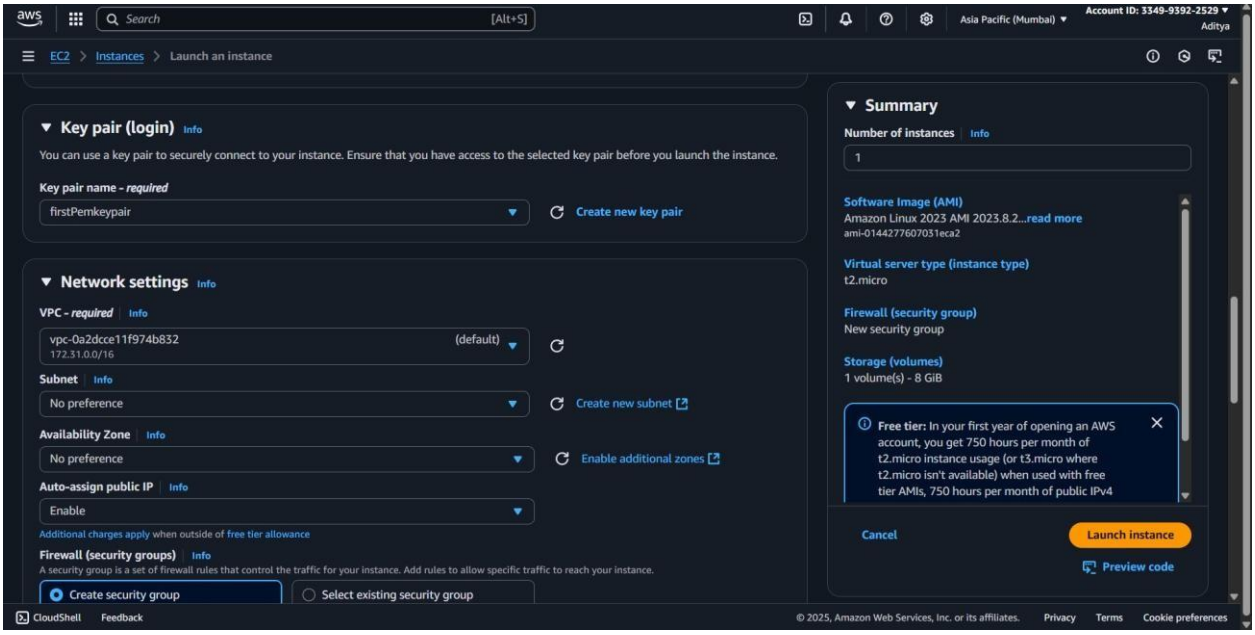
Prerequisite:

- An active AWS account.
- A VPC with a configured public subnet.
- Permissions to launch EC2 instances.
- An existing key pair for SSH access.
- AWS Management Console or AWS CLI access.

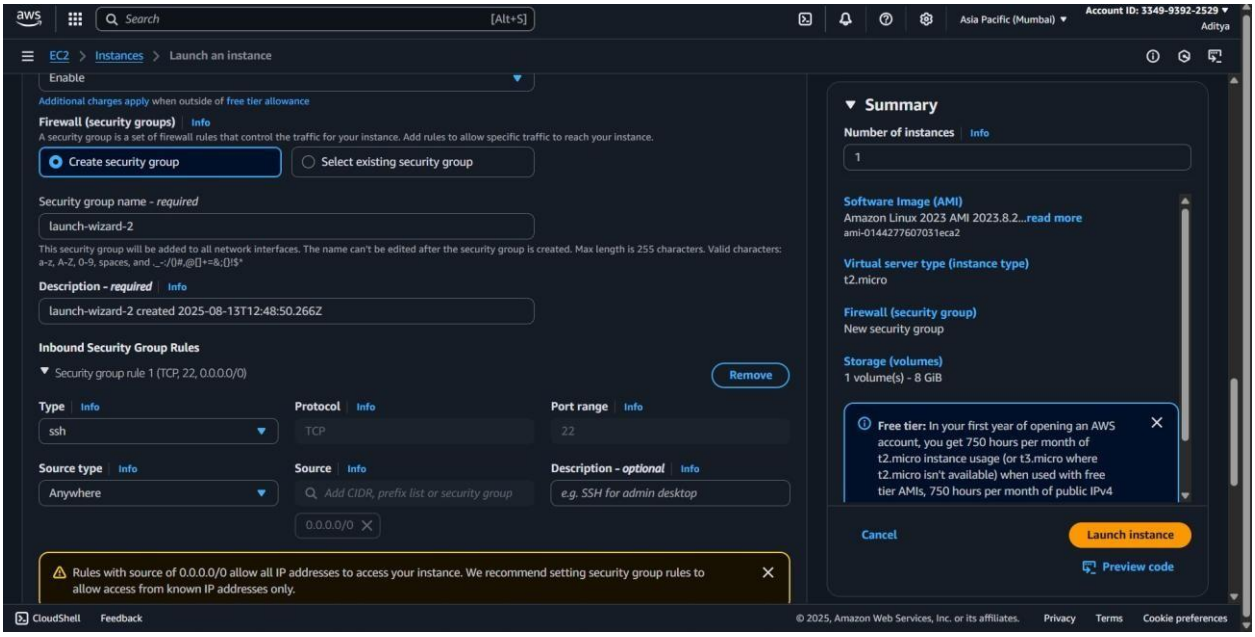
Step 1: create an ec2 instance.

Step 2: create with the help of a keypair

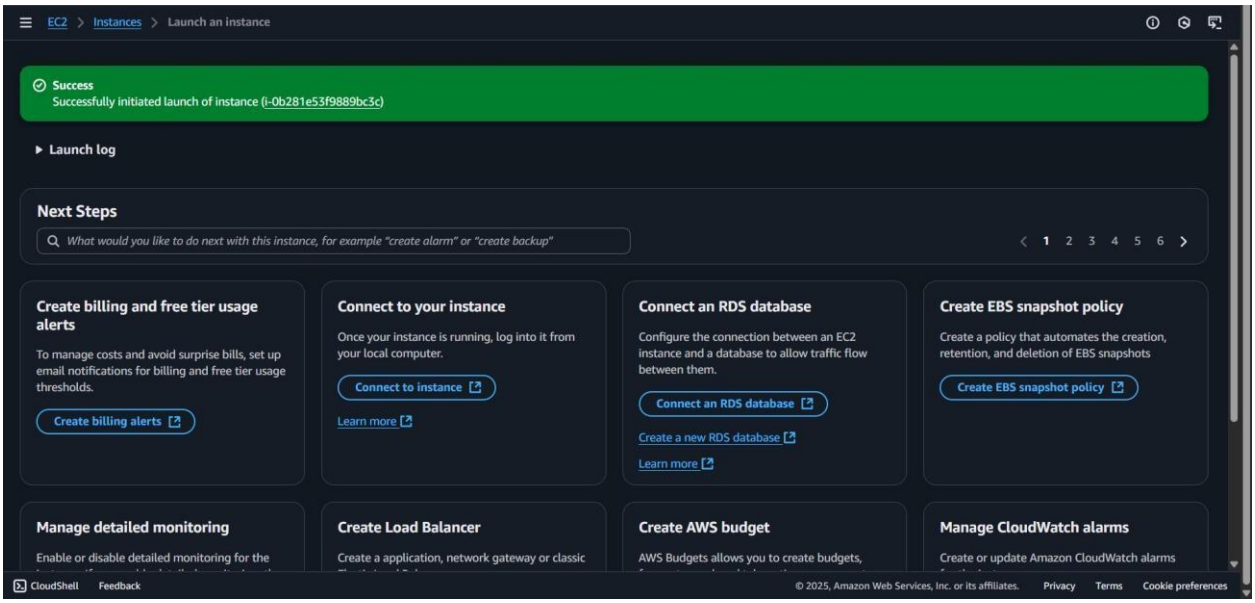
Step 3: Click **Edit** in the “Network settings” section.



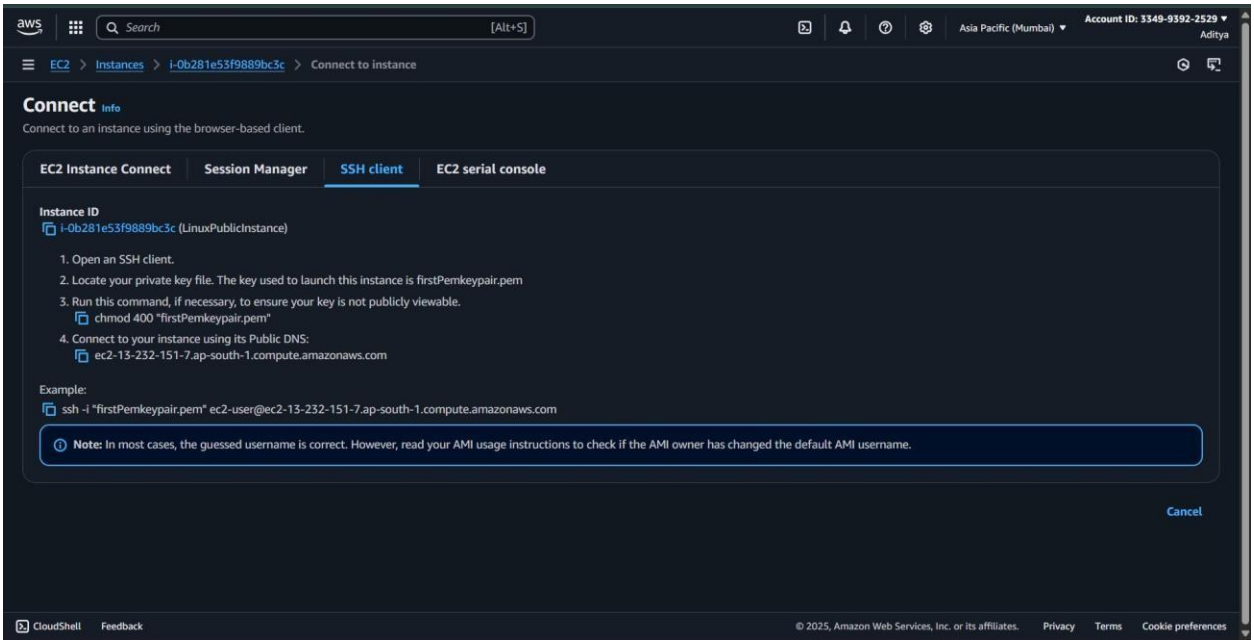
Step 4: Choose your **VPC** (the one that contains the Public Subnet).



Step 5: launch the instance



Step 6: connect to instance and proceed to ssh client option


aws
Search
[Alt+S]
Asia Pacific (Mumbai)
Account ID: 3349-9392-2529
Aditya
EC2 > Instances > i-0b281e53f9889bc3c > Connect to instance
Connect Info
Connect to an instance using the browser-based client.
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-0b281e53f9889bc3c (LinuxPublicInstance)
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is firstPemkeypair.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "firstPemkeypair.pem"
4. Connect to your instance using its Public DNS:
ec2-13-232-151-7.ap-south-1.compute.amazonaws.com
Example:
ssh -i "firstPemkeypair.pem" ec2-user@ec2-13-232-151-7.ap-south-1.compute.amazonaws.com
Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.
Cancel
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

19.Modify the security group of the public EC2 instance to allow only your IP on port 22.

Objective:

To secure SSH access for a public EC2 instance by configuring its security group to allow port 22 connections exclusively from the user's own public IP address.

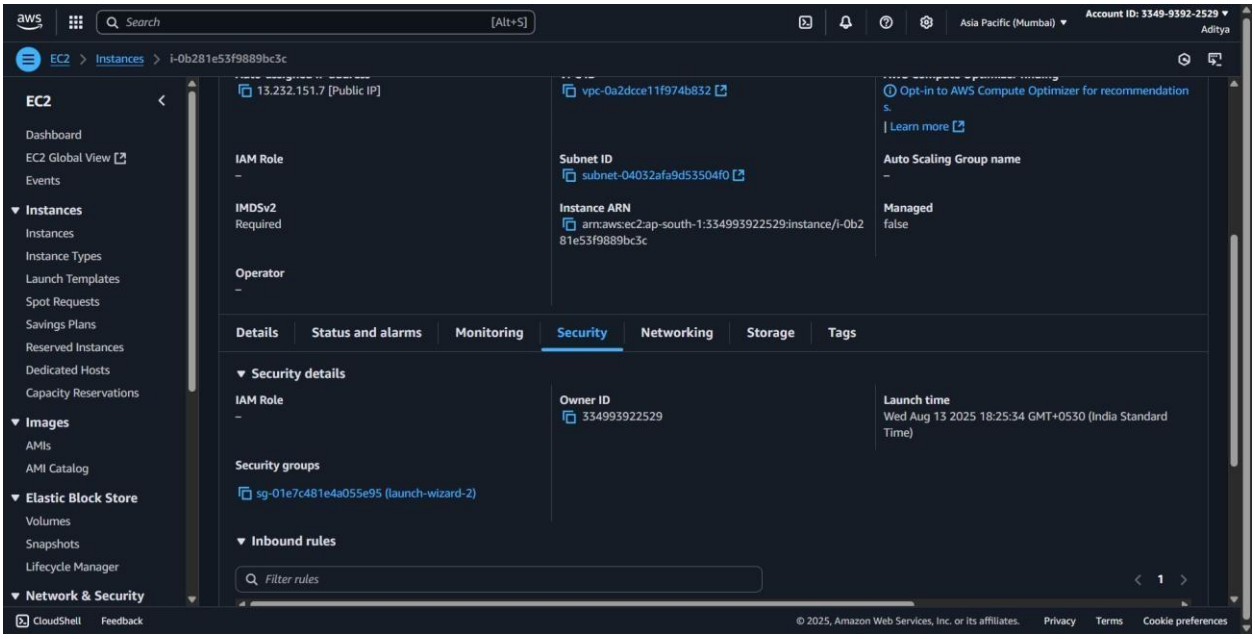
Prerequisite:

- An active AWS account.
- A running EC2 instance in a public subnet.
- Knowledge of the current public IP address.
- Permissions to edit security group inbound rules.
- AWS Management Console or AWS CLI access.

Step 1: **Find the Security Group** Select

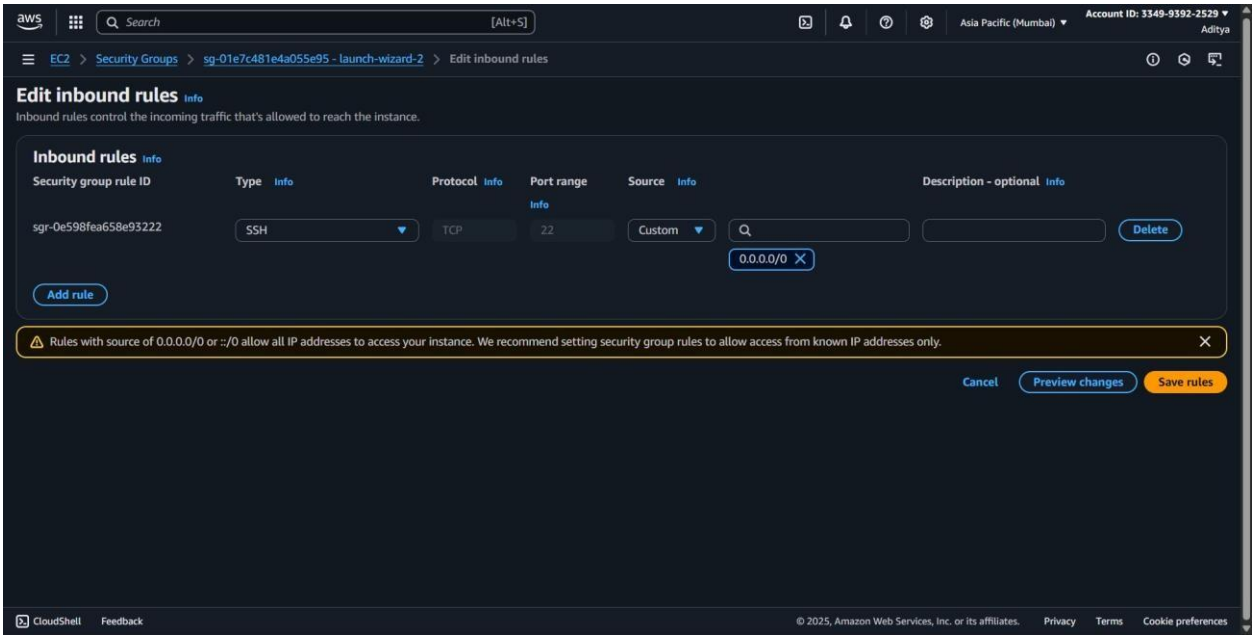
your EC2 instance.

In the **Description** tab, note the **Security group ID** under **Security**.



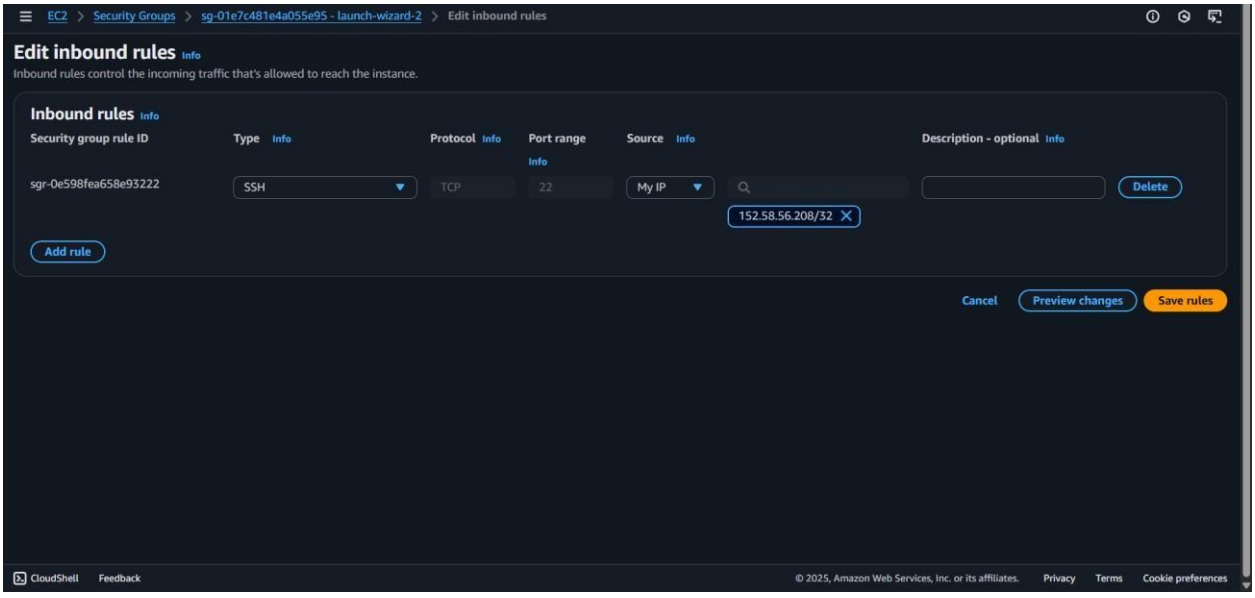
Step 2: **Edit Inbound Rules**

Click on the Security Group ID → Go to the **Inbound rules** tab. Click **Edit inbound rules**.

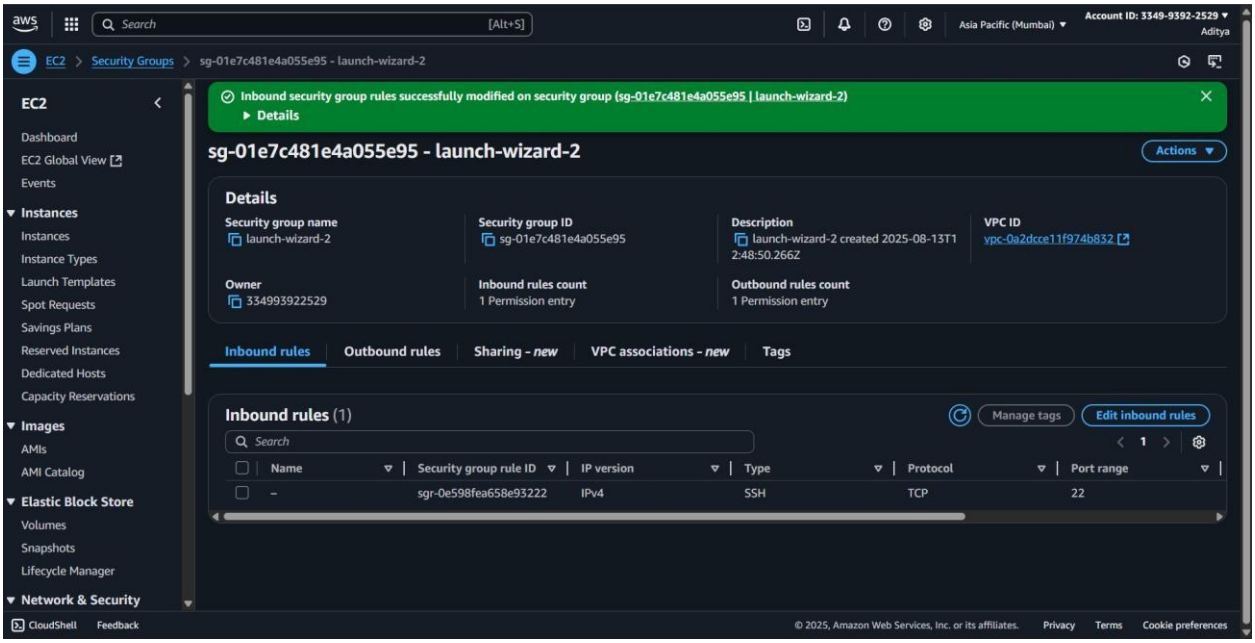


Step 3: **Update SSH Rule**

- Remove any existing **SSH** rules allowing access from **0.0.0.0/0** (which means open to everyone).
- Add a new rule: ○ **Type:** SSH ○ **Protocol:** TCP ○ **Port Range:** 22 ○ **Source: My IP** (AWS auto-detects your current public IP).



Step 4: save changes



20.Create and attach an Internet Gateway (IGW) to your VPC.

Objective:

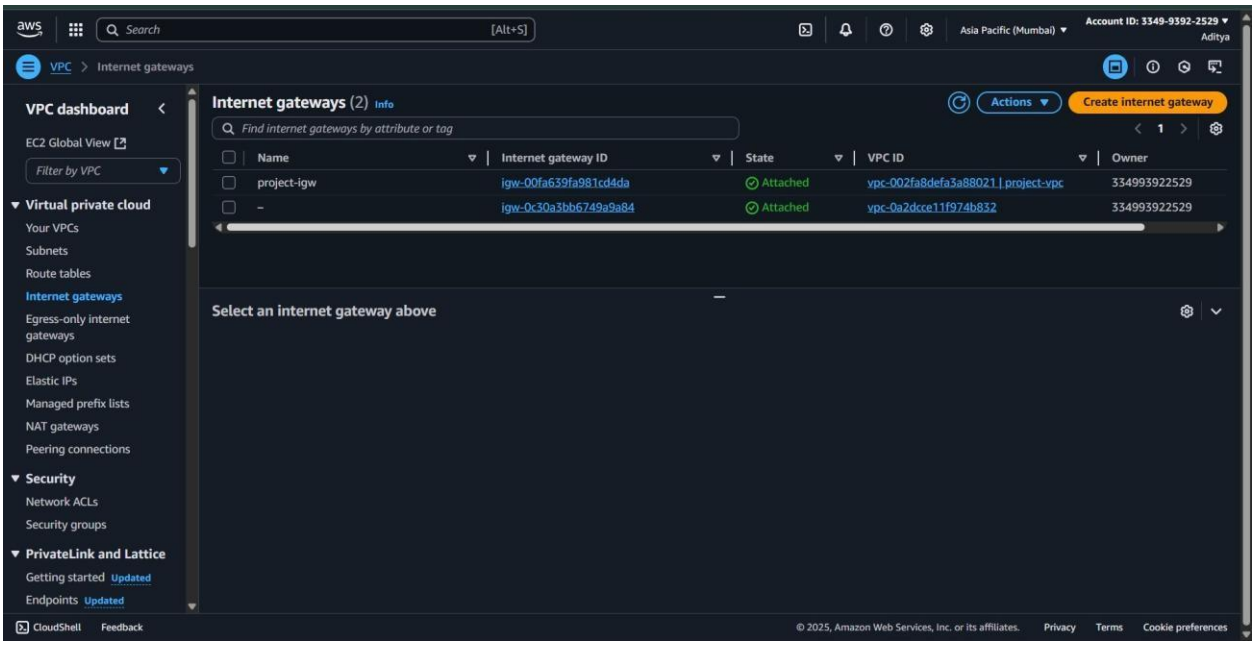
To enable internet connectivity for resources within a VPC by creating an Internet Gateway (IGW) and attaching it to the VPC.

Prerequisite:

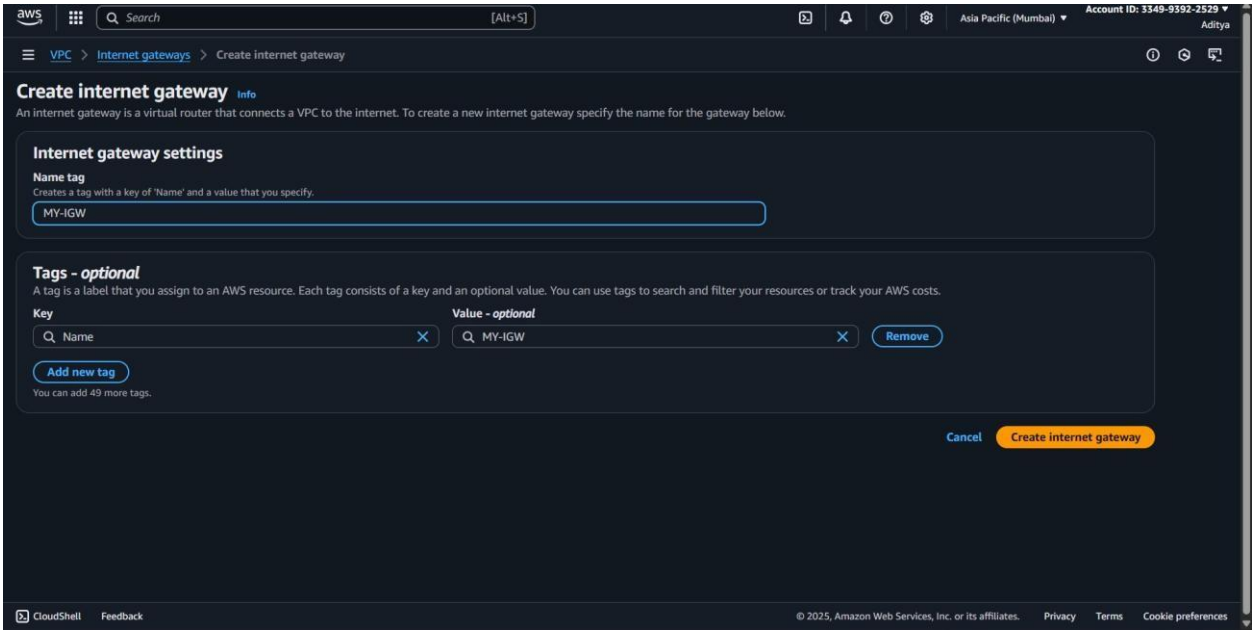
- An active AWS account.
- An existing VPC.
- Permissions to create and attach Internet Gateways.
- AWS Management Console or AWS CLI access.

Step 1: Go to vpc dashboard.

Step 2: Create an Internet Gateway



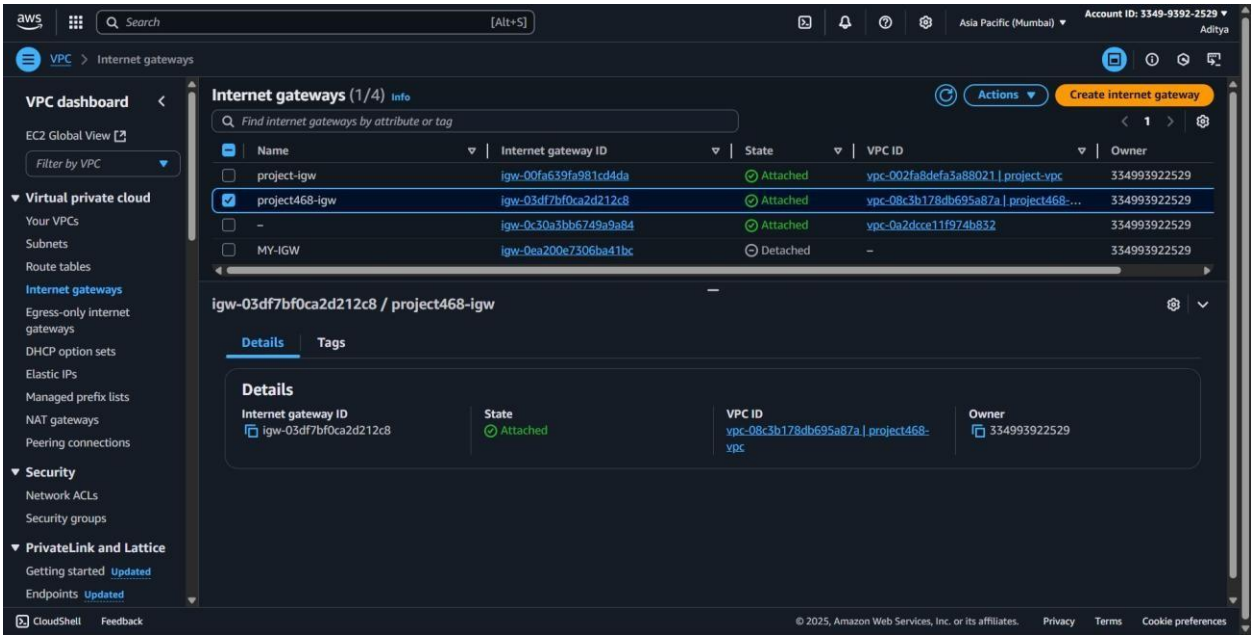
Step 3: click create internet gateway



Step 4: **Attach the IGW to Your VPC**

Select the newly created IGW.

- Click **Actions → Attach to VPC**.
- Choose your **VPC** (e.g., My-VPC).
- Click **Attach internet gateway**



File 2

Index

S. No.	Name of experiment	Experiment date	Submission date	Page no.

1.	Deploy and configure Storage Gateway appliance. Set up File, Volume, and Tape Gateway modes. Integrate on-premises environment with AWS storage. Monitor and manage gateway operations.	15/09/25	24/09/25	4
2.	Design and implement custom VPC architecture. Configure subnets, route tables, and gateways. Implement network security with Security Groups and NACLs. Set up VPC peering and transit gateway.	15/09/25	24/09/25	7
3.	Implement comprehensive IAM strategy. Configure users, groups, roles, and policies. Set up MFA and access controls. Audit and monitor IAM activities.	15/09/25	24/09/25	11
4.	Deploy and configure RDS instances. Implement backup and recovery strategies. Set up read replicas and Multi-AZ deployments. Monitor and optimize database performance.	15/09/25	24/09/25	14

5.	Create and configure DynamoDB tables. Design efficient partition and sort keys. Implement Global Secondary Indexes. Set up DynamoDB Streams for realtime processing.	17/09/25	24/09/25	16

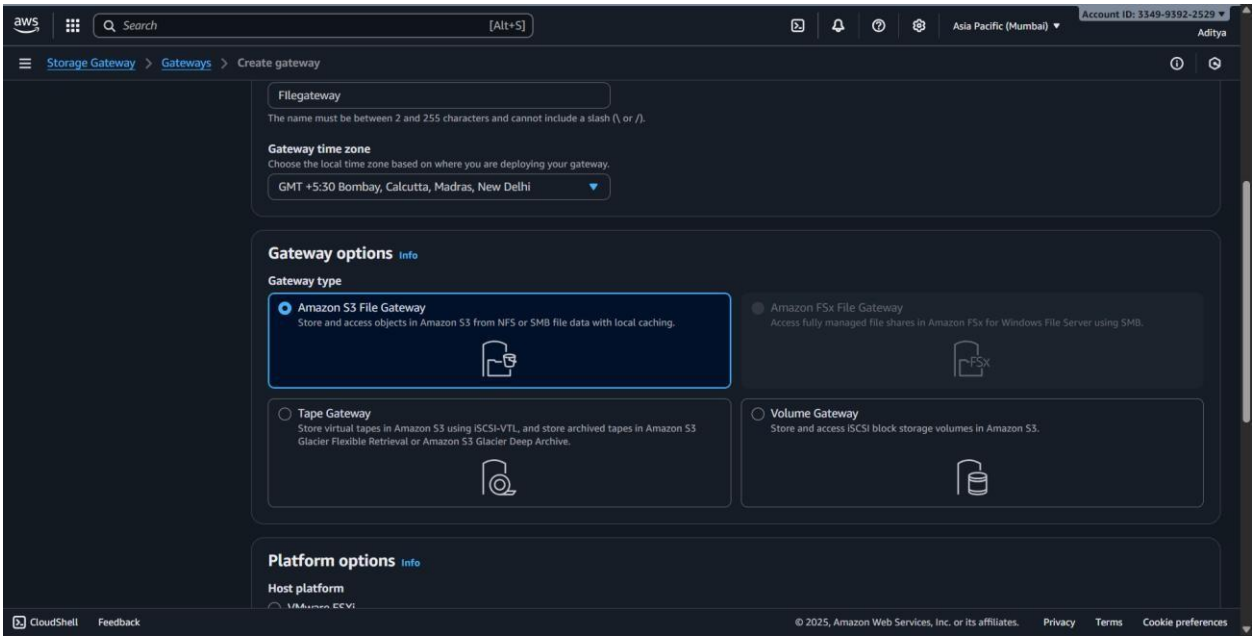
6.	Create CloudFormation templates for infrastructure. Implement nested stacks and cross-stack references. Manage stack updates and rollbacks. Integrate with CI/CD pipelines.	17/09/25	24/09/25	19
7.	Deploy applications using Elastic Beanstalk. Configure application environments. Implement blue-green deployments. Monitor application health and performance.	17/09/25	24/09/25	21
8.	Set up comprehensive logging and monitoring. Create custom metrics and alarms. Implement log analysis and alertingBuild operational dashboards.	19/09/25	24/09/25	22
9.	Design and implement complex workflows. Handle error conditions and retries. Integrate multiple AWS services. Monitor workflow execution.	20/09/25	24/09/25	24
10.	Set up comprehensive cost monitoring. Create budgets and alerts. Analyze spending patterns. Implement cost optimization strategies	20/09/25	24/09/25	26

1. Deploy and configure Storage Gateway appliance. Set up File, Volume, and Tape Gateway modes. Integrate on-premises environment with AWS storage. Monitor and manage gateway operations.

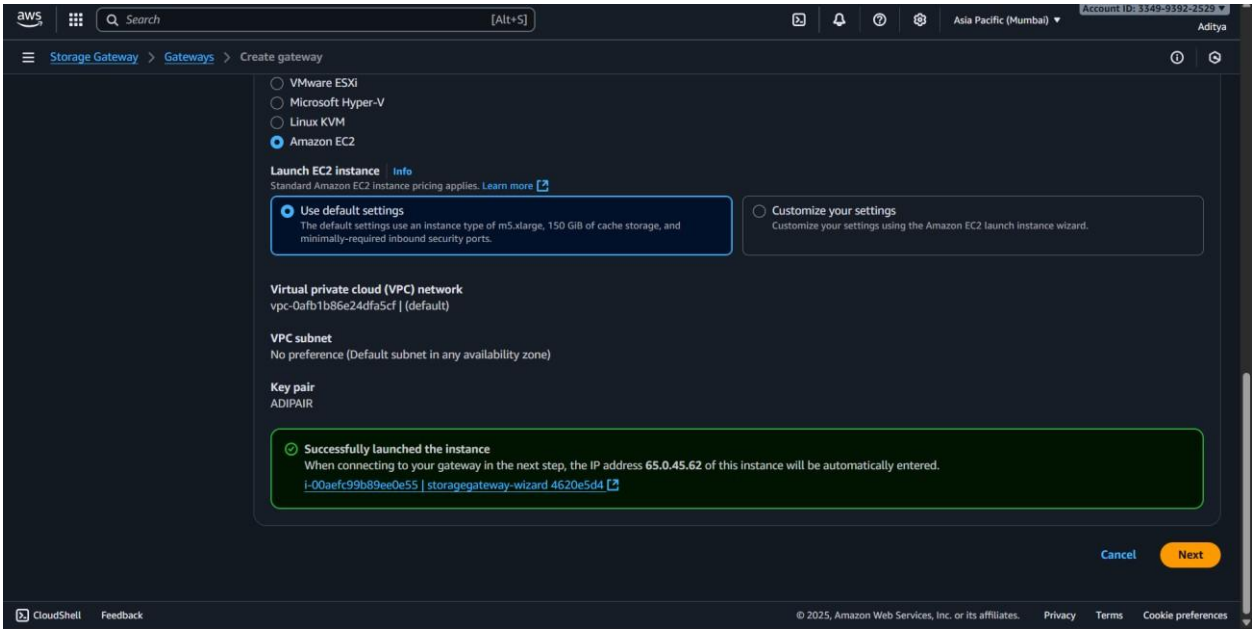
Steps:

1. Deploy AWS Storage Gateway Appliance

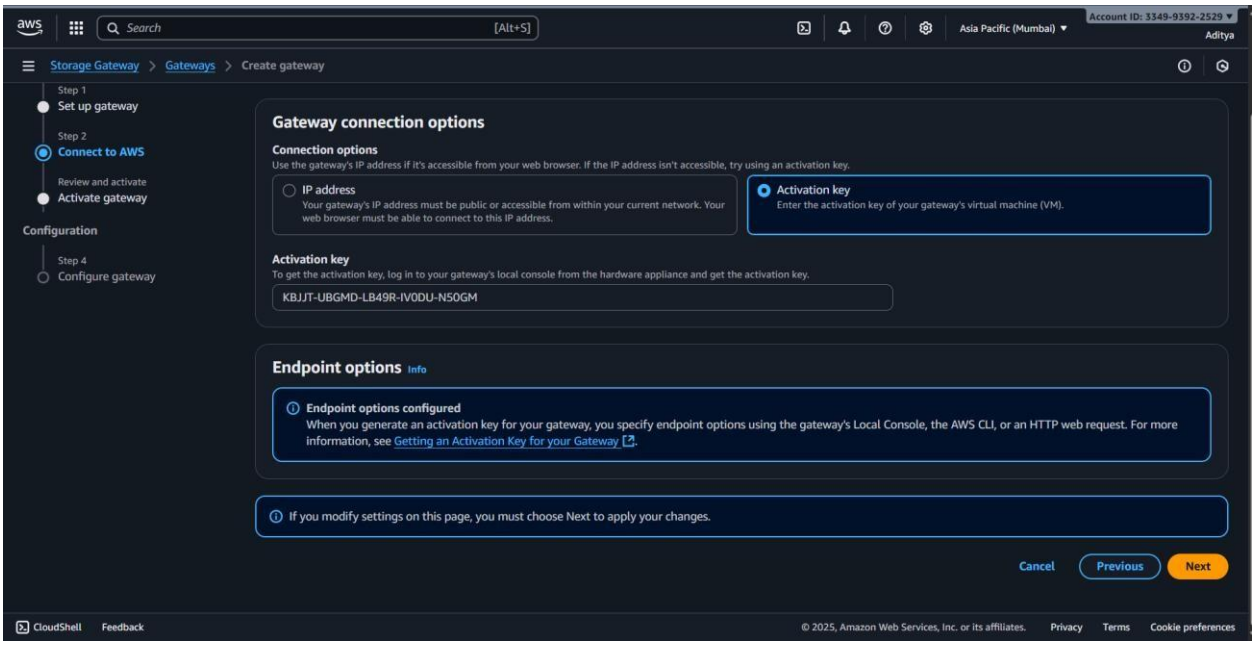
1. **Login to AWS Console** → search **Storage Gateway**.
2. Click **Create gateway**.
3. Select **Gateway type** → choose one (File / Volume / Tape).
4. Choose **Host platform**:
    - **VMware ESXi / Hyper-V / KVM** → if you have virtualization.
    - **Amazon EC2** → simplest for lab environment.



5. Deploy the gateway:
    - If using **EC2** → select an **instance type (m5.xlarge recommended)**. ○ Launch with **default settings**.

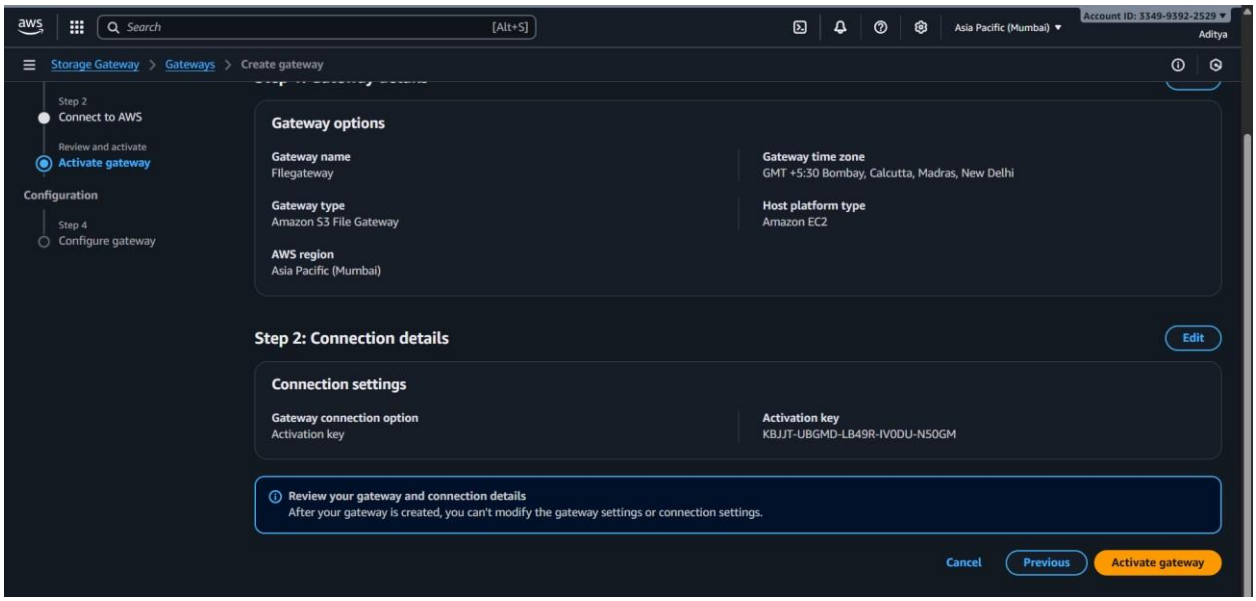


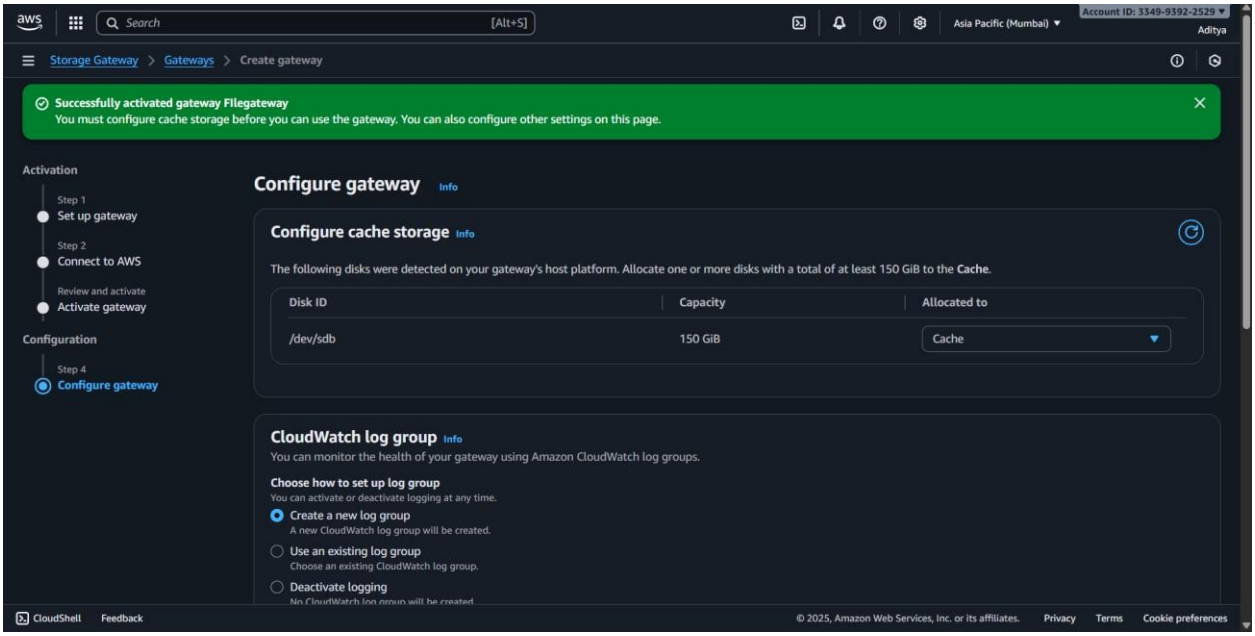
Once deployed → copy paste the activation key .



2. Activate the Gateway

1. Go back to **AWS Console → Storage Gateway**.
2. Paste the **activation key**.
3. Give a **name** (example: *MyFileGateway*).
4. Choose **time zone and AWS region**.
5. Click **Activate gateway**.





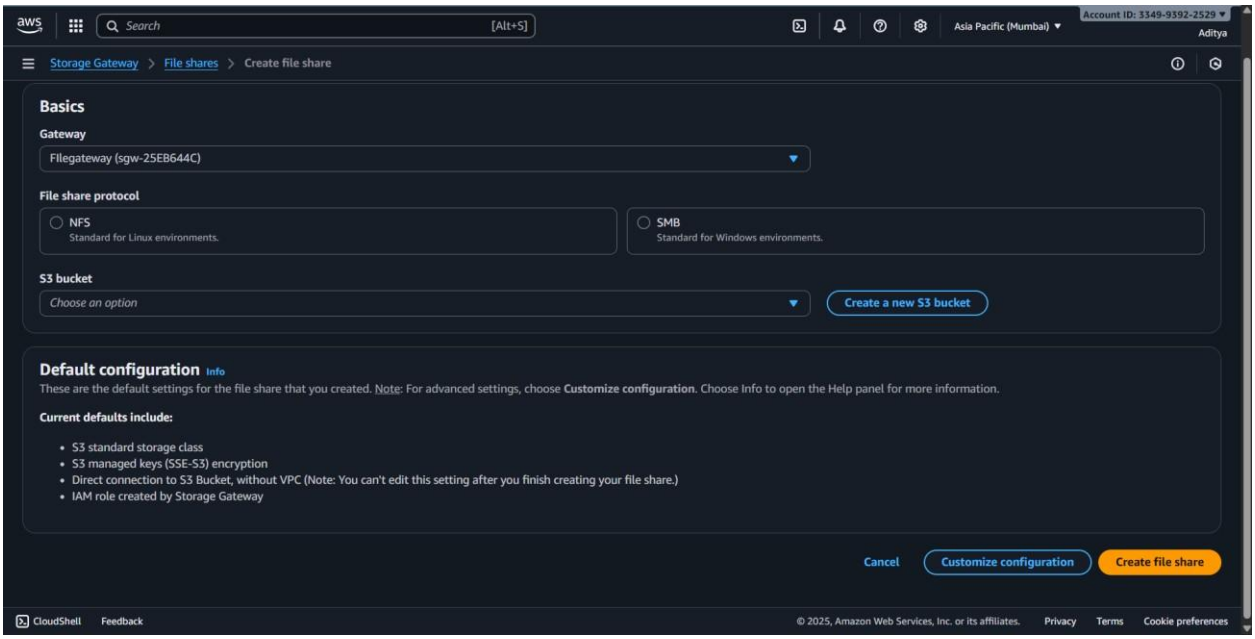
3. Configure Gateway Modes

(A) File Gateway

- Provides **NFS (Linux/Unix)** or **SMB (Windows)** file shares.

1. In the Storage Gateway console → choose **File Gateway**.
2. Create a **File Share**:
    - Backend storage = **Amazon S3 bucket**.
    - Protocol = **NFS / SMB**.

3. Attach this file share to your **on-premises client**.
4. Now, files written to the share automatically sync to **S3**.



(B) Volume Gateway

- Provides **iSCSI block storage** (two modes: Cached / Stored volumes).

1. Choose **Volume Gateway** during setup.
2. Configure **iSCSI connection** to on-premises servers.
3. Select **Cached Volume** → data stored in S3, frequently accessed in cache.
4. Select **Stored Volume** → data stored locally, backup copies to S3.
5. Attach volume to on-premises server → use like a local disk.

(C) Tape Gateway

- Used for backup/archival with **Virtual Tape Library (VTL)**.

1. Choose **Tape Gateway** in setup.
2. Create **Virtual Tapes** (each tape stored in **Amazon S3**).
3. Integrate with **backup software** (like Veeam, Commvault, NetBackup).
4. Tapes can be archived to **Amazon S3 Glacier** for cost efficiency.

4. Integrate On-Premises with AWS

1. Install Storage Gateway appliance **on-premises (VMware/Hyper-V)**.
2. Connect the appliance to AWS Storage Gateway service via **Internet or Direct Connect**.
3. Mount File Shares (NFS/SMB) or connect Volumes (iSCSI) or Tapes (VTL) on local servers.
4. Verify data transfer → data moves securely to **AWS S3, EBS, or Glacier**.

5. Monitor and Manage Gateway 1. Go to **AWS Console →**

CloudWatch.

2. Check metrics like: ○ **Read/Write throughput** ○ **Cache hit ratio** ○ **Disk utilization**
3. Use **CloudWatch Alarms** to alert on performance/storage issues.
4. In **Storage Gateway Console**, you can:
    - Expand volumes or file shares. ○ Add/remove tapes.
    - Stop/Start the gateway.

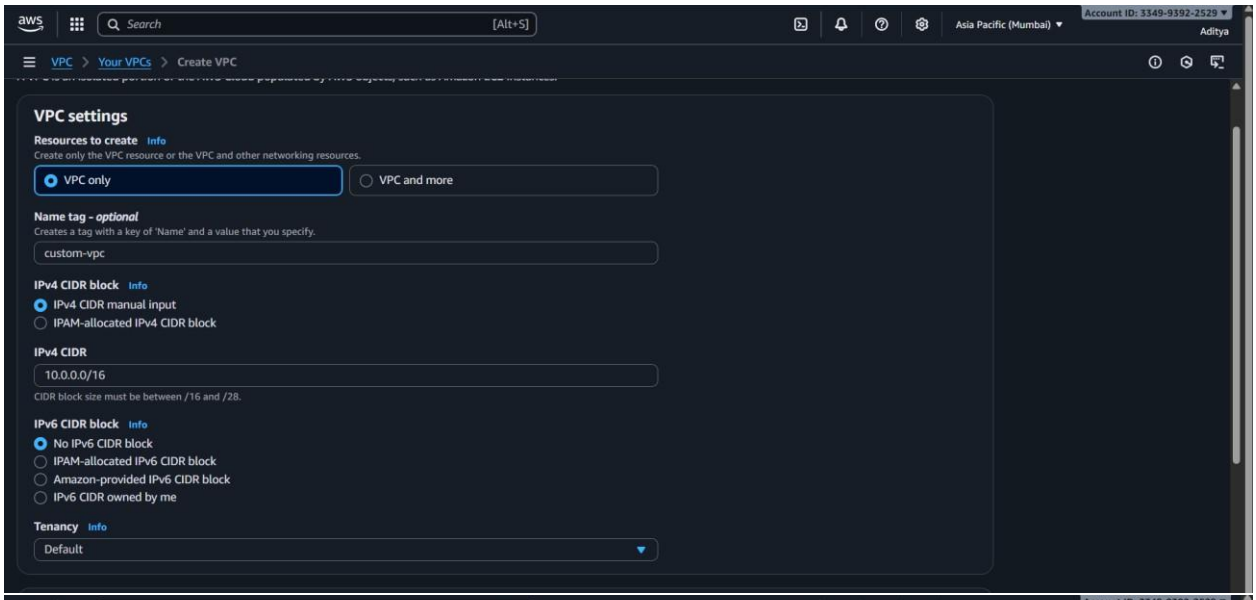
2.Design and implement custom VPC architecture. Configure subnets, route tables, and gateways. Implement network security with Security Groups and NACLs. Set up VPC peering and transit gateway.

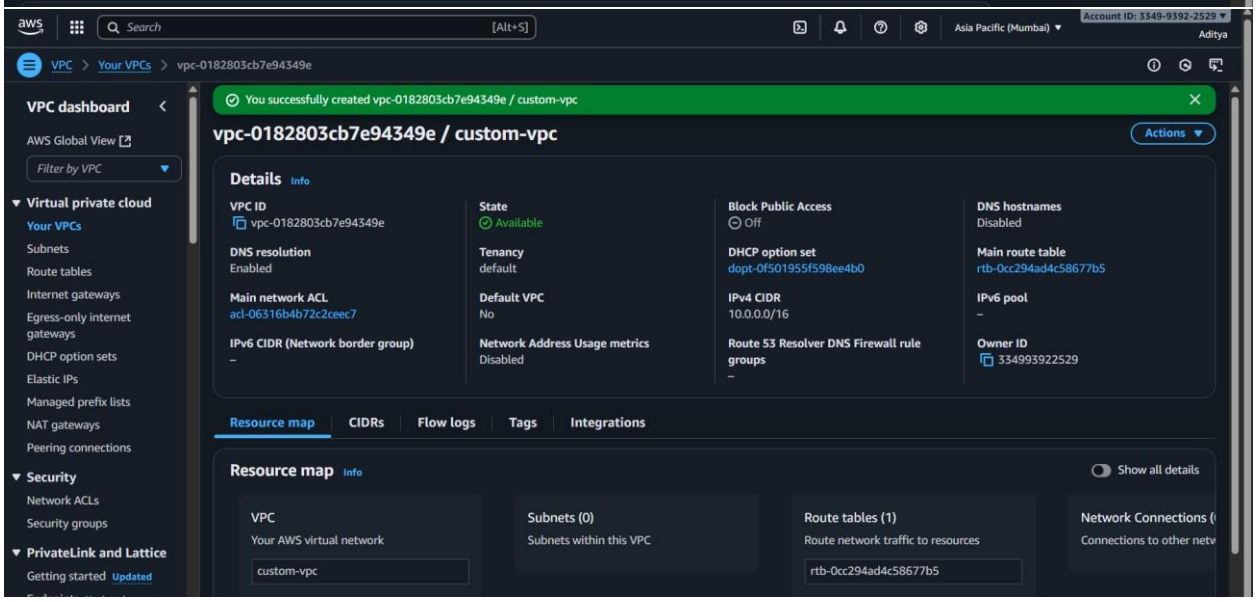
Steps-

Steps to Design and Implement Custom VPC Architecture

2. Log in to the AWS Management Console.
3. Navigate to VPC service.

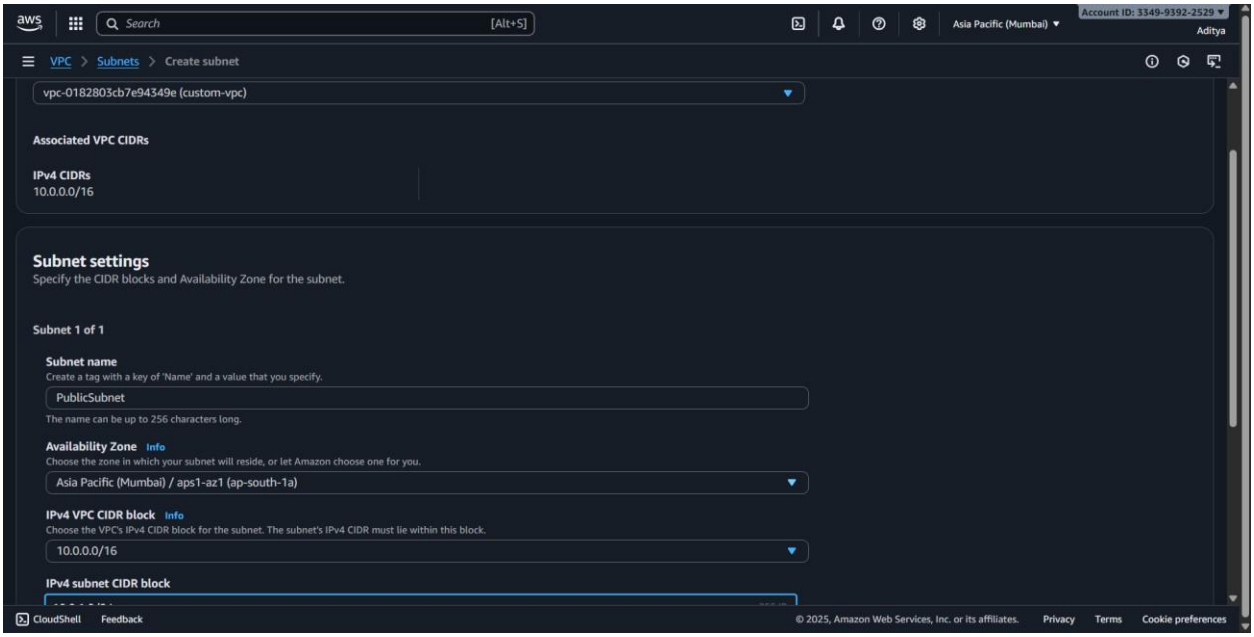
4. Click Create VPC.
5. Select VPC only.
6. Enter:
    - Name: CustomVPC
    - IPv4 CIDR block: 10.0.0.0/16 ○ Leave other defaults and click Create VPC.

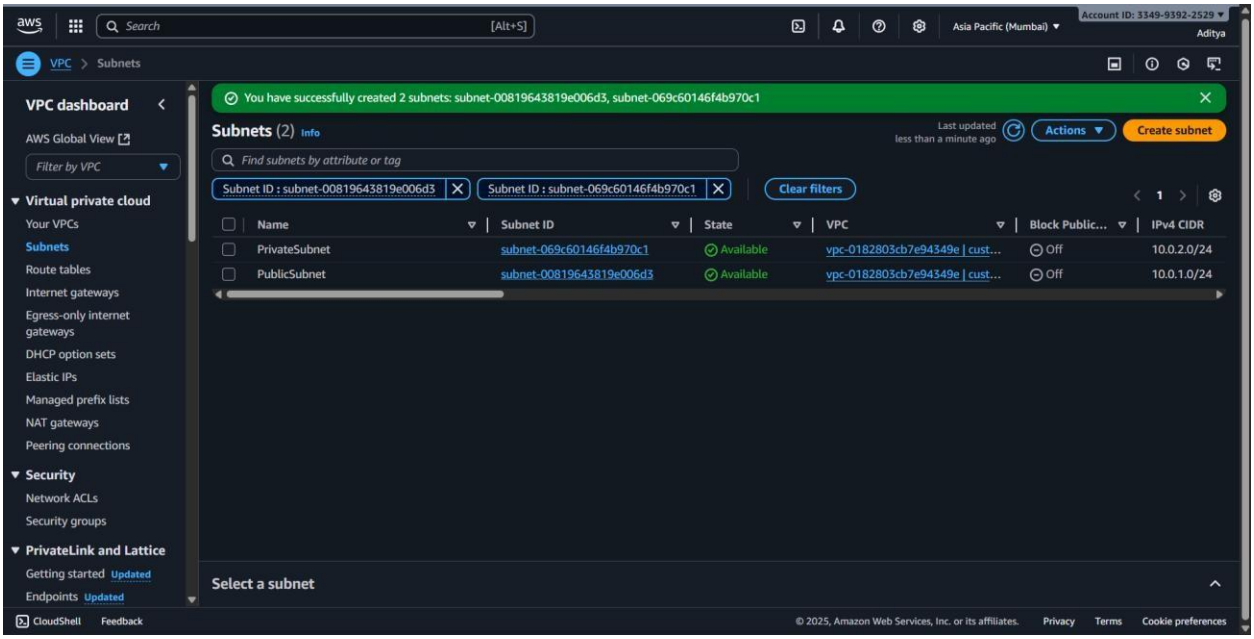




Steps to Configure Subnets

1. In the VPC dashboard, go to Subnets → Create subnet.
2. Select CustomVPC.



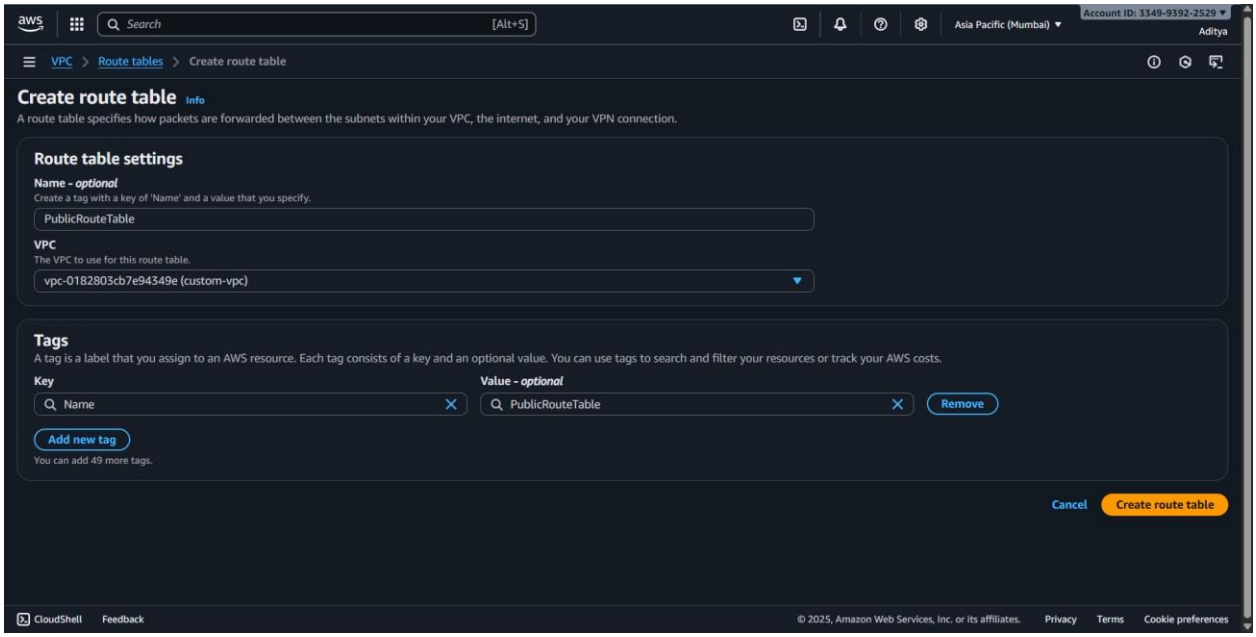


3. Create two subnets:
    - PublicSubnet: 10.0.1.0/24 (Availability Zone 1).
    - PrivateSubnet: 10.0.2.0/24 (Availability Zone 2).
4. Click Create subnet

Steps to Configure Route Tables

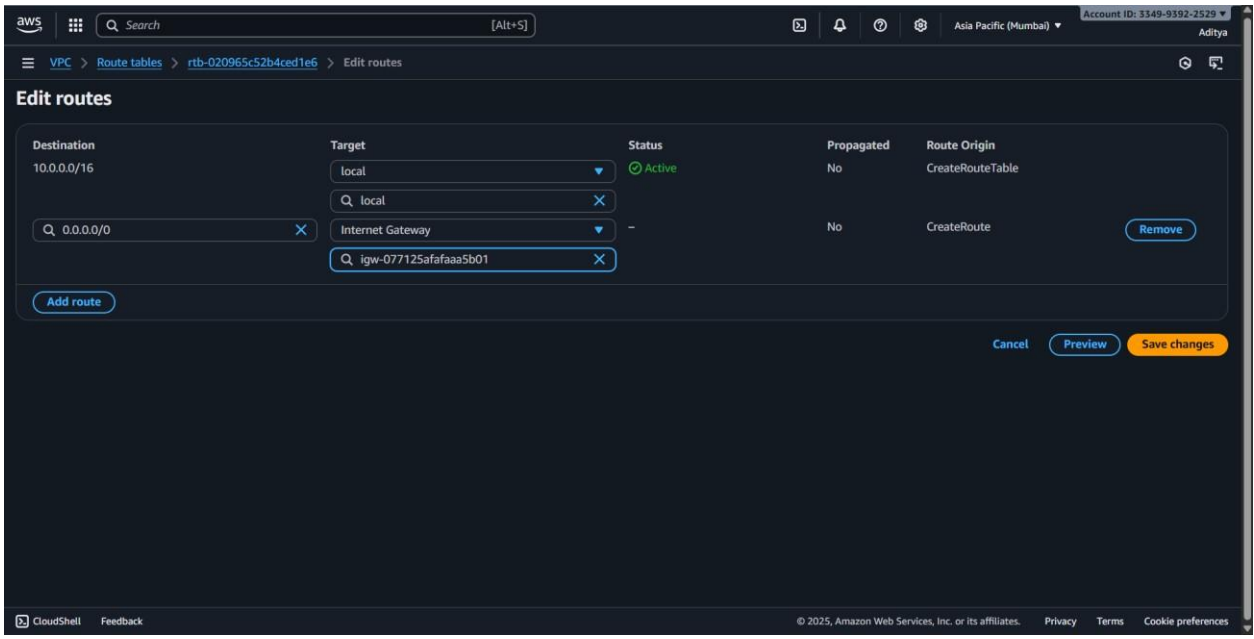
1. In the **VPC dashboard**, go to **Route tables → Create route table**.

- Name: PublicRouteTable → Associate with **CustomVPC**.



2. Add a route:

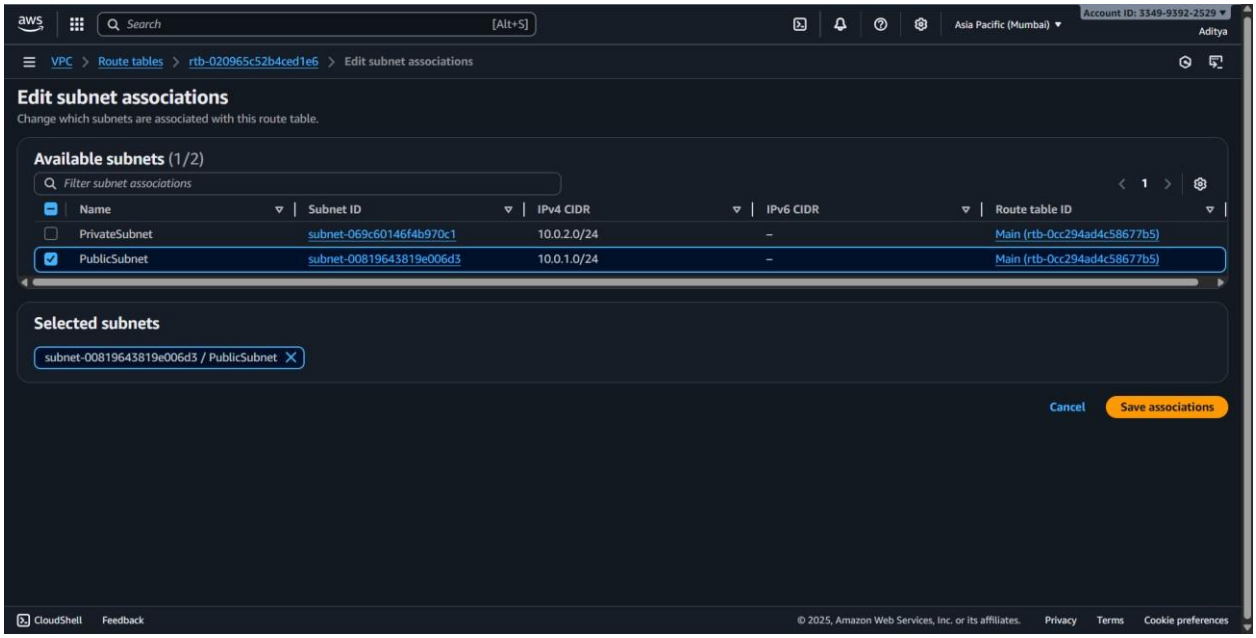
- Destination: 0.0.0.0/0 ○



Target: **Internet Gateway**

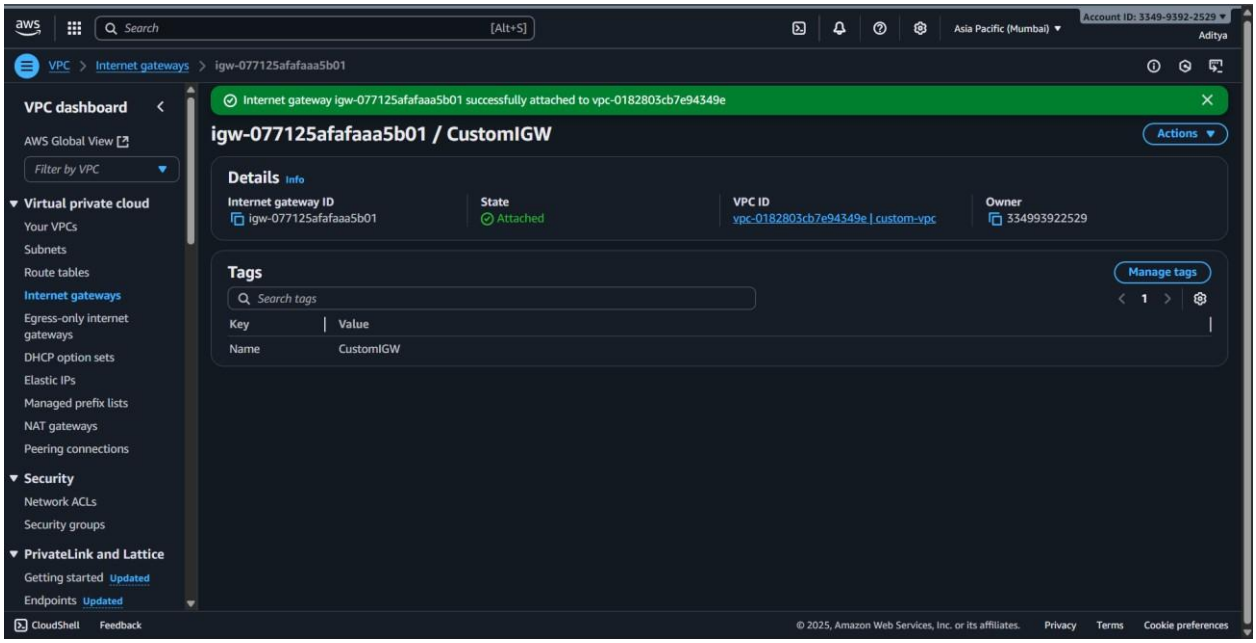
(IGW).

3. Associate this route table with **PublicSubnet**.



4. For **PrivateSubnet**, keep the **Main Route Table** (no direct Internet access).

5. **Steps to Configure Gateways**
6. Go to **Internet Gateways → Create internet gateway**.
7. Name: CustomIGW.
8. Attach the IGW to **CustomVPC**.



9. (Optional for private subnet) Create **NAT Gateway**:
10. Place it inside **PublicSubnet**.
11. Associate with **Elastic IP**.
12. Add route in **PrivateSubnet's Route Table** to point 0.0.0.0/0 → NAT Gateway.

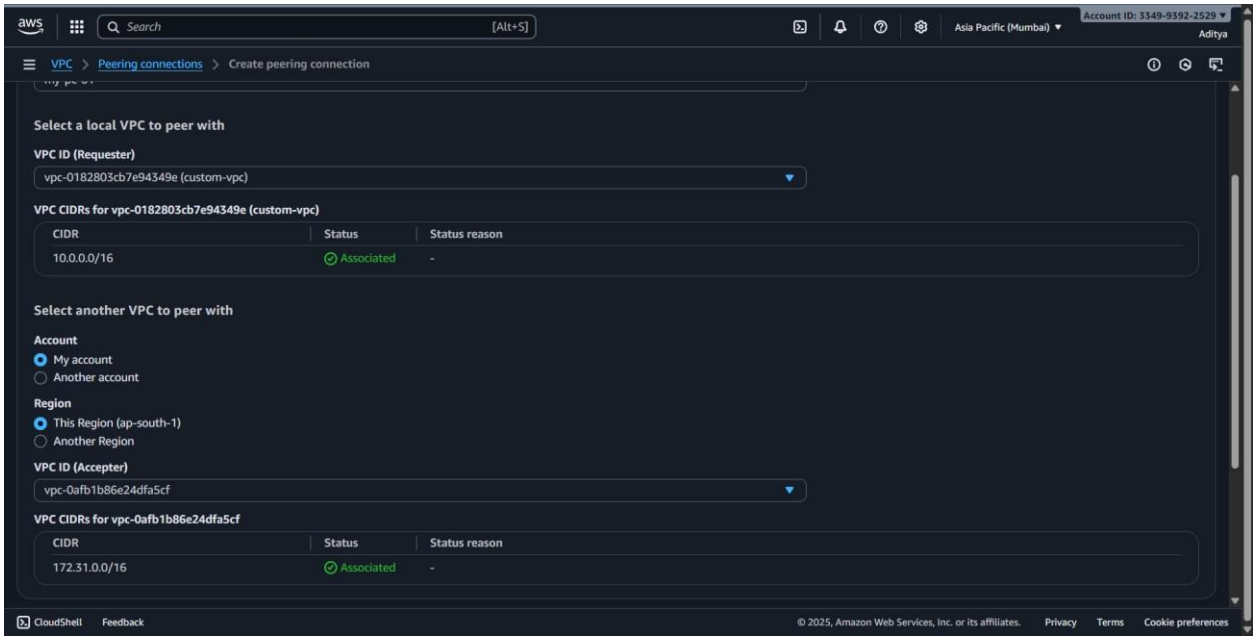
Steps to Implement Network Security

1. **Security Groups**:
    - Create a **WebSG** for public instances: Allow inbound HTTP (80), HTTPS (443), and SSH (22).
    - Create a **DBSG** for database instances: Allow inbound MySQL (3306) only from **WebSG**.
2. **Network ACLs (NACLs)**:
    - For **PublicSubnet** NACL: Allow inbound 80, 443, 22 and outbound all.
    - For **PrivateSubnet** NACL: Allow inbound 3306 and outbound traffic restricted to application/web tier.

Steps to Set Up VPC Peering

1. In the **VPC dashboard**, select **Peering Connections → Create peering connection**.

2. Select **CustomVPC** as requester, and another VPC (e.g., OtherVPC) as accepter.



3. Click **Create peering connection**.
4. Update **route tables** in both VPCs to enable communication.

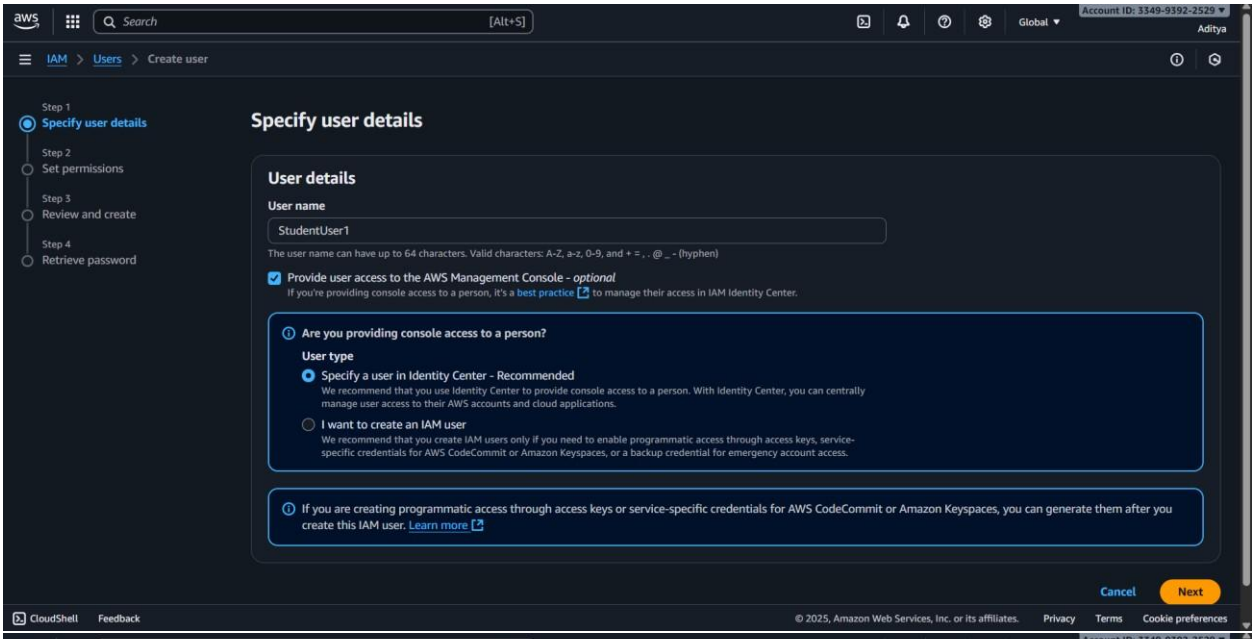
Steps to Set Up Transit Gateway

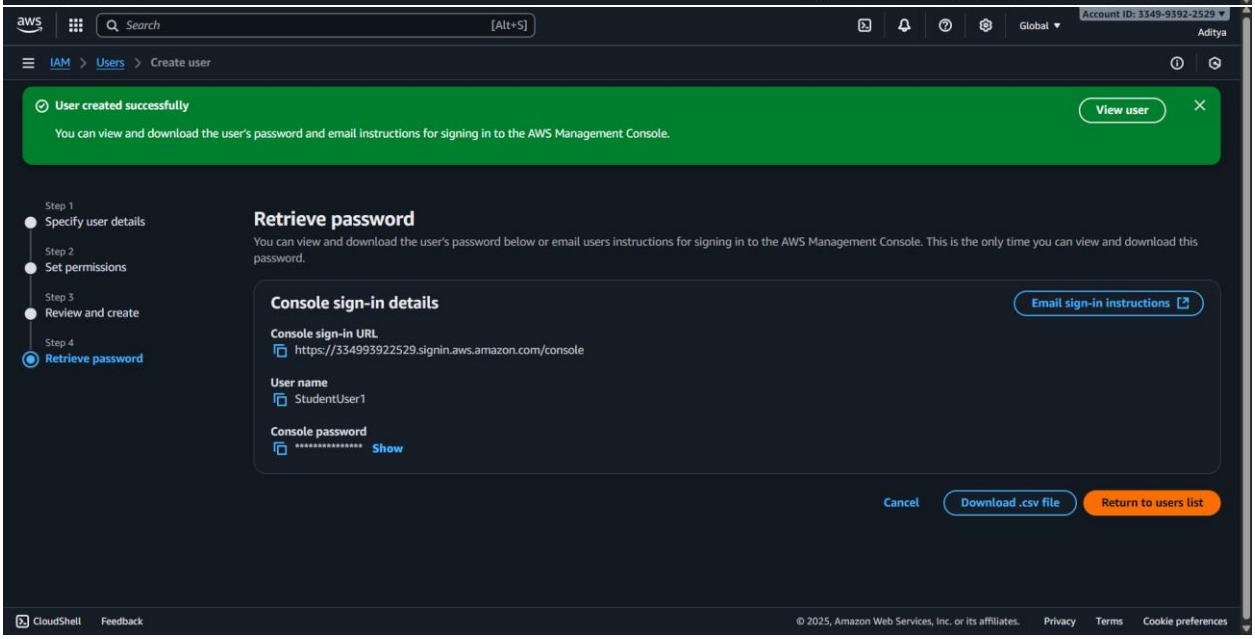
1. Navigate to **Transit Gateways → Create transit gateway**.
    - Name: CustomTransitGW.
    - Keep default ASN or assign custom.
2. Attach **CustomVPC** to the Transit Gateway.
3. Attach other VPCs or VPN connections as needed.
4. Update route tables in each VPC to forward traffic via **Transit Gateway**.

3.Implement comprehensive IAM strategy. Configure users, groups, roles, and policies. Set up MFA and access controls. Audit and monitor IAM activities.

Create Users

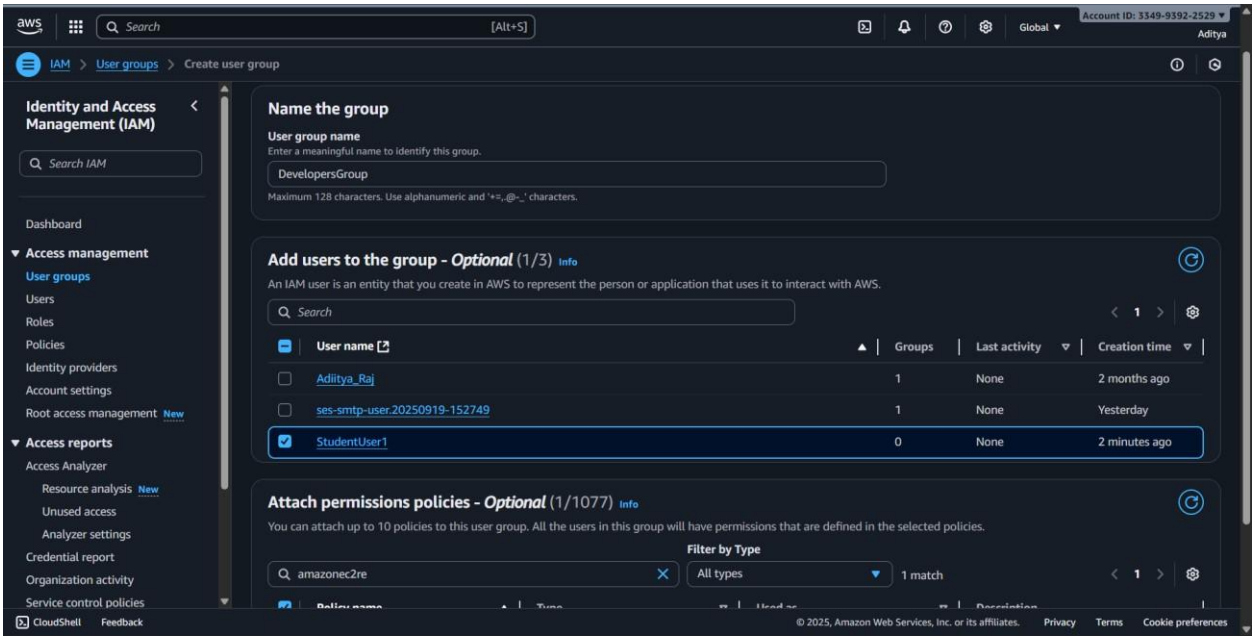
3. Go to Users → Add users.
4. Enter username (e.g., StudentUser1).
5. Select Provide user access:
    - Check AWS Management Console access (with password).
    - Optionally check Programmatic access (for CLI/SDK).
6. Click Next.





Create Groups

7. Go to **User groups → Create group**.
8. Enter group name: DevelopersGroup.
9. Attach a managed policy like AmazonEC2ReadOnlyAccess.
10. Add the user (StudentUser1) to this group.

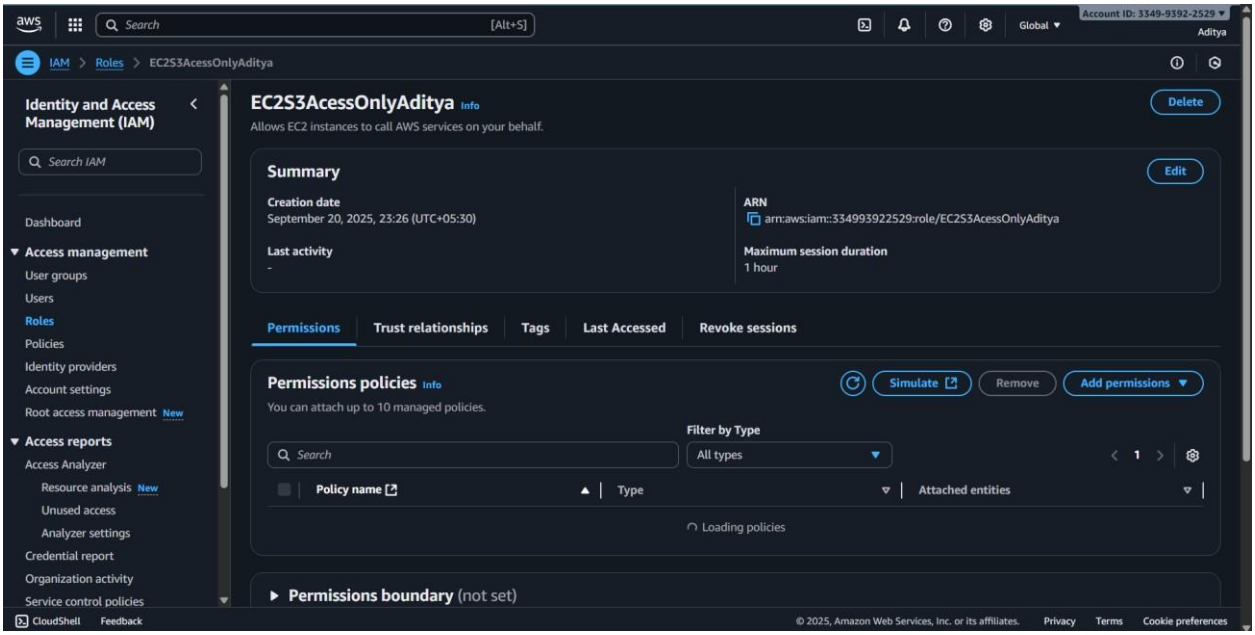


Create Roles

11. Go to **Roles → Create role**.
12. Select trusted entity:

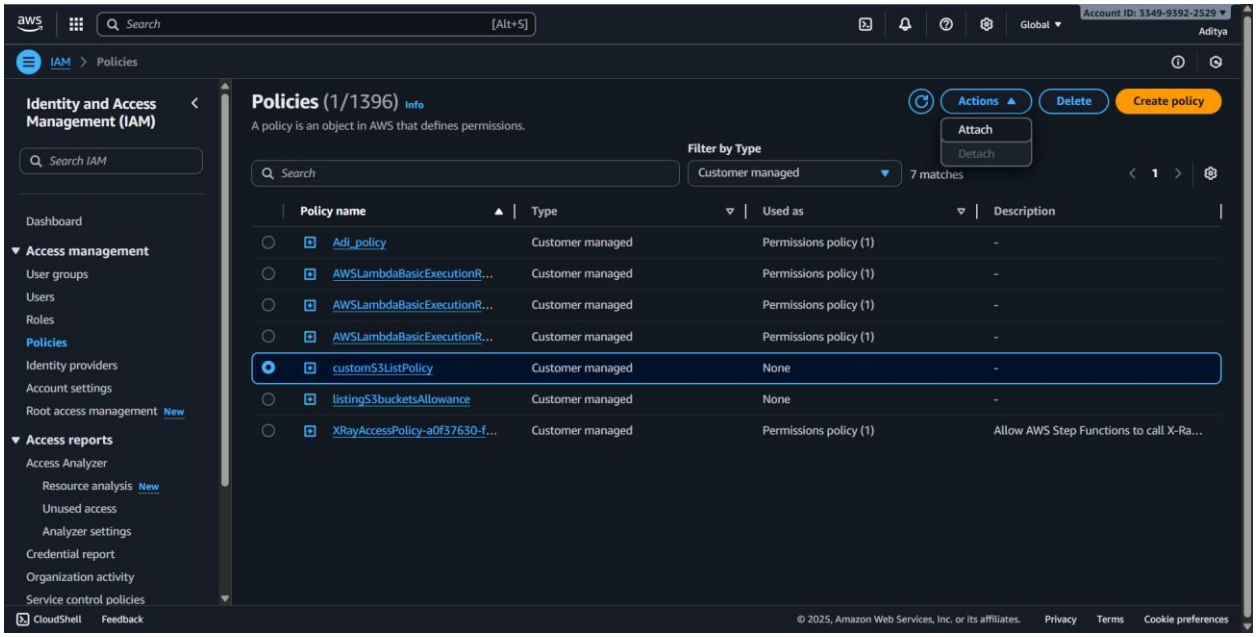
- For AWS service, choose **EC2** (so EC2 can assume this role).

13. Attach policy (e.g., AmazonS3FullAccess).
14. Name the role: EC2S3AccessRole.
15. Launch an EC2 instance and attach this role to allow S3 access.



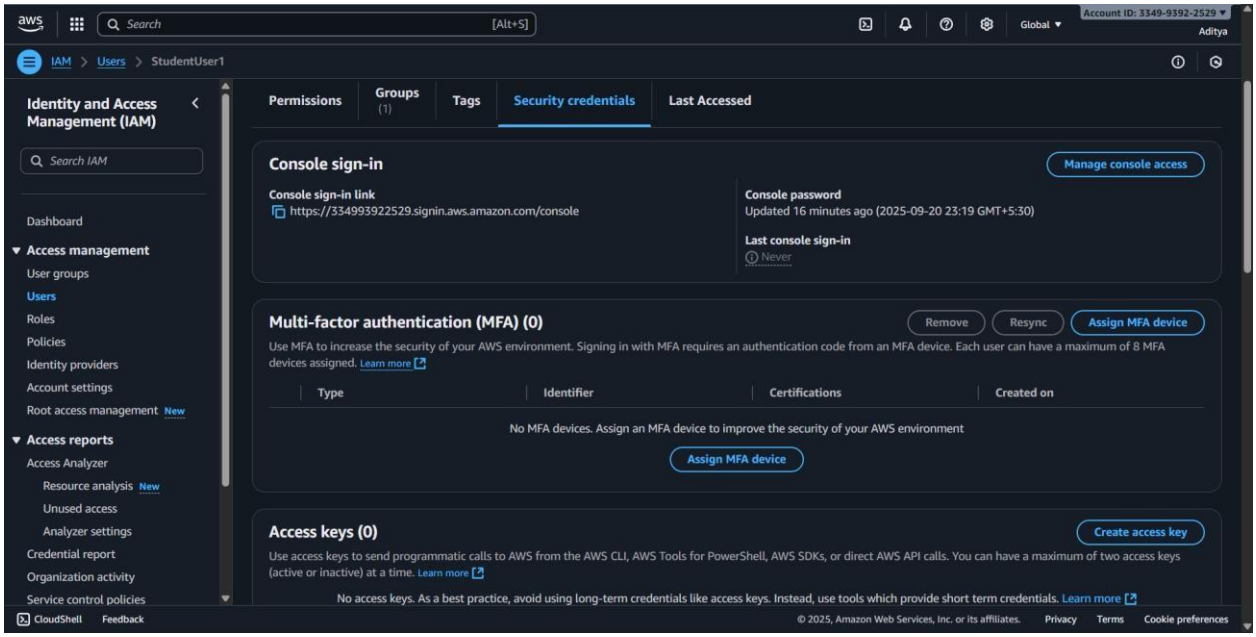
Create Custom Policies

16. Go to **Policies → Create policy**.
17. Choose **JSON editor** and define a custom policy, e.g., to allow only s3:ListBucket on a specific bucket.
18. Save policy as CustomS3ListPolicy.
19. Attach this policy to the desired user or group.



Steps to Set Up MFA and Access Controls

20. Go to **Users → select StudentUser1 → Security credentials**.

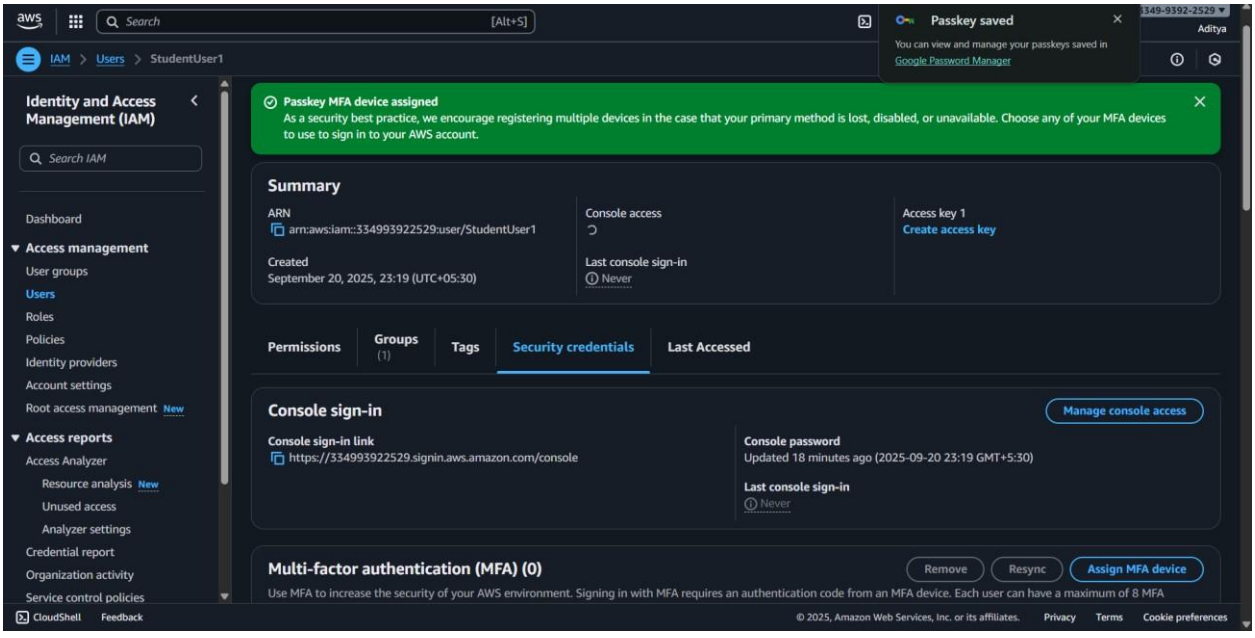


21. Under **Multi-factor authentication (MFA)**, choose **Assign MFA device**.

22. Select **Authenticator app** → scan QR code with Google Authenticator/Authy.

23. Enter the 2 codes generated → Activate MFA.

24. Now login requires username, password, and MFA code.



Steps to Audit and Monitor IAM Activities

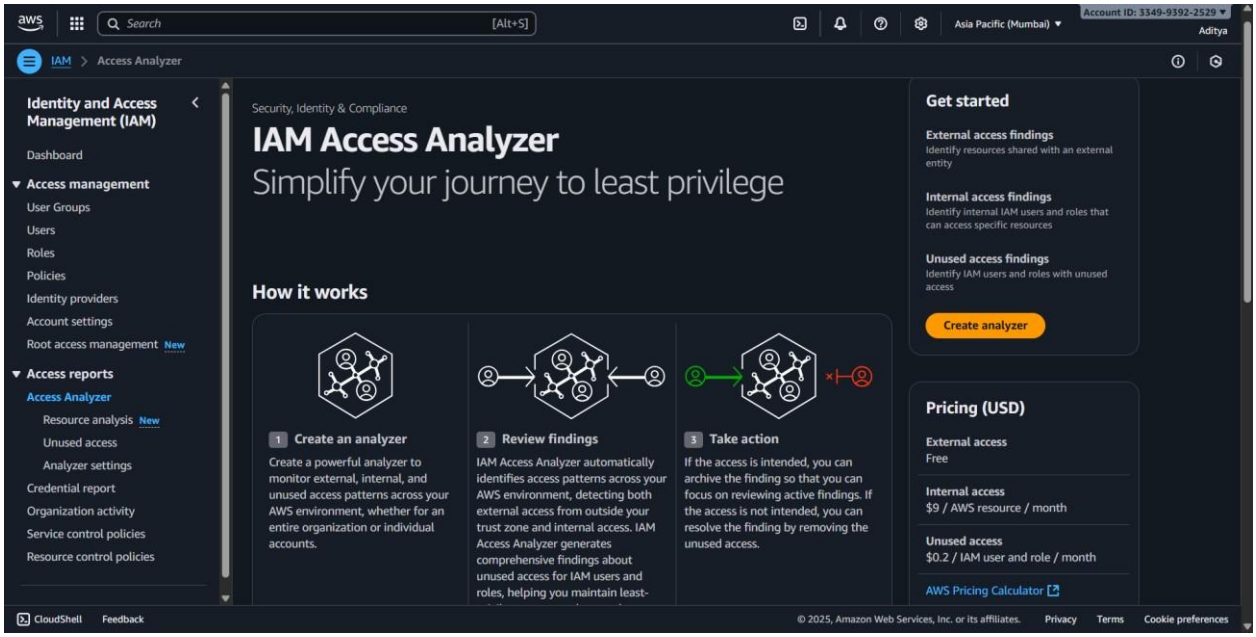
25. Go to **IAM → Access analyzer** to review policies granting public or cross-account access.

26. Enable **CloudTrail** service:

- This records all IAM API activities (user logins, policy changes, role assumptions).

27. Enable **AWS Config** to track configuration changes in IAM (e.g., user/group creation).

28. Review logs in **CloudWatch Logs** for monitoring unusual login attempts or denied actions.



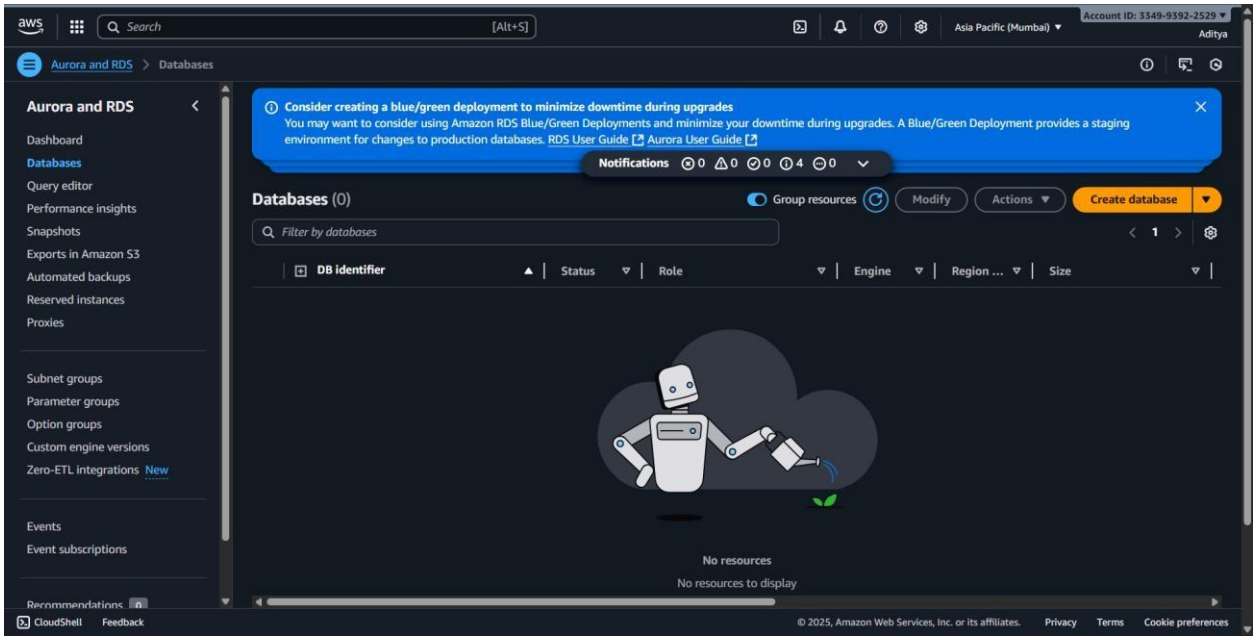
4.Deploy and configure RDS instances. Implement backup and recovery strategies.

Set up read replicas and Multi-AZ deployments. Monitor and optimize database Performance

Steps:

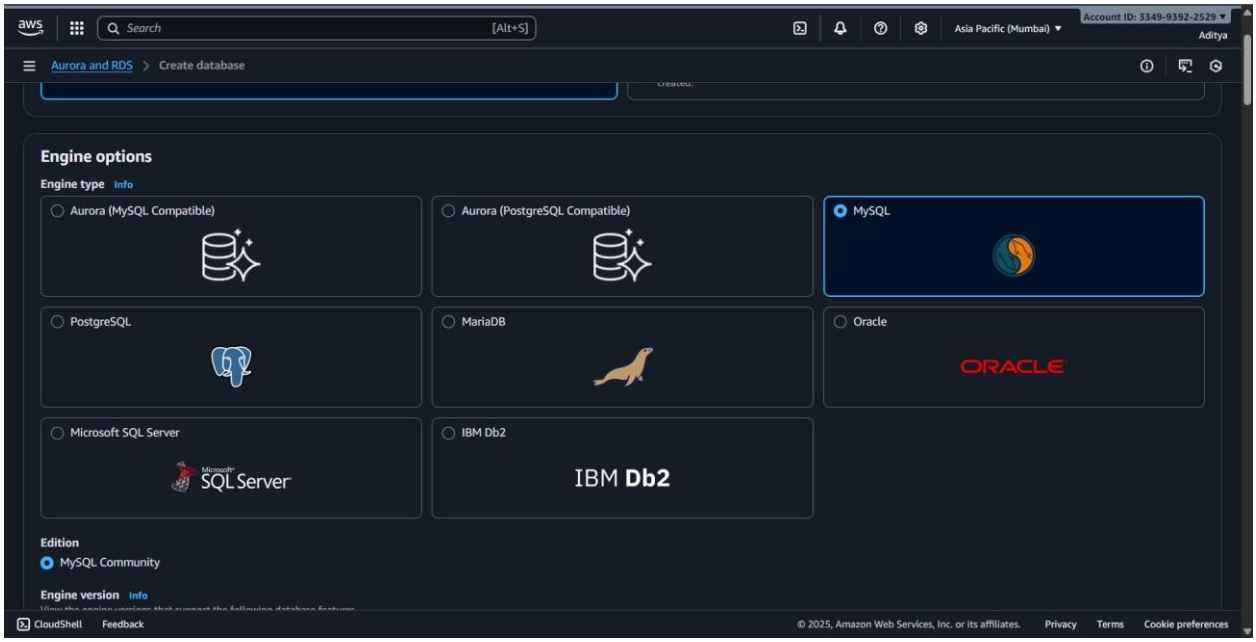
1.Go to **RDS** service.

2. Click **Create database**.

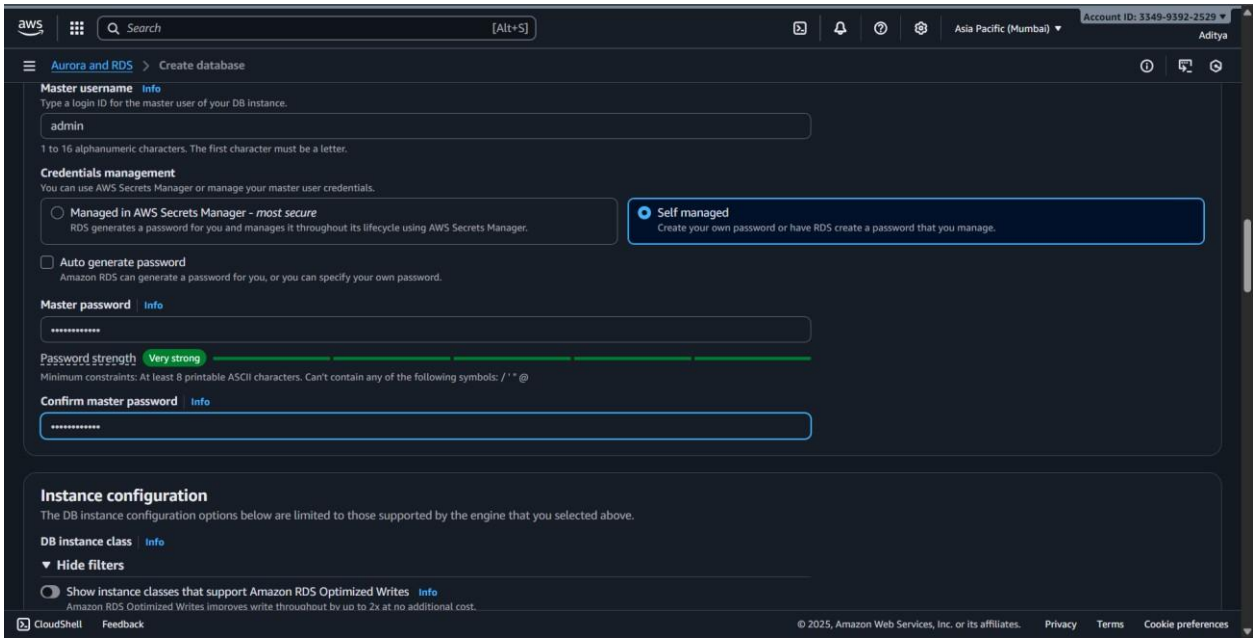


3. Choose **Standard create**.

5. Select **Engine**: MySQL or PostgreSQL (free-tier eligible).



6. Choose **Templates**: Select **Free tier** (to reduce cost).
7. Set details:
    - DB identifier: StudentDB.
    - Username: admin.
    - Password: set a secure one.

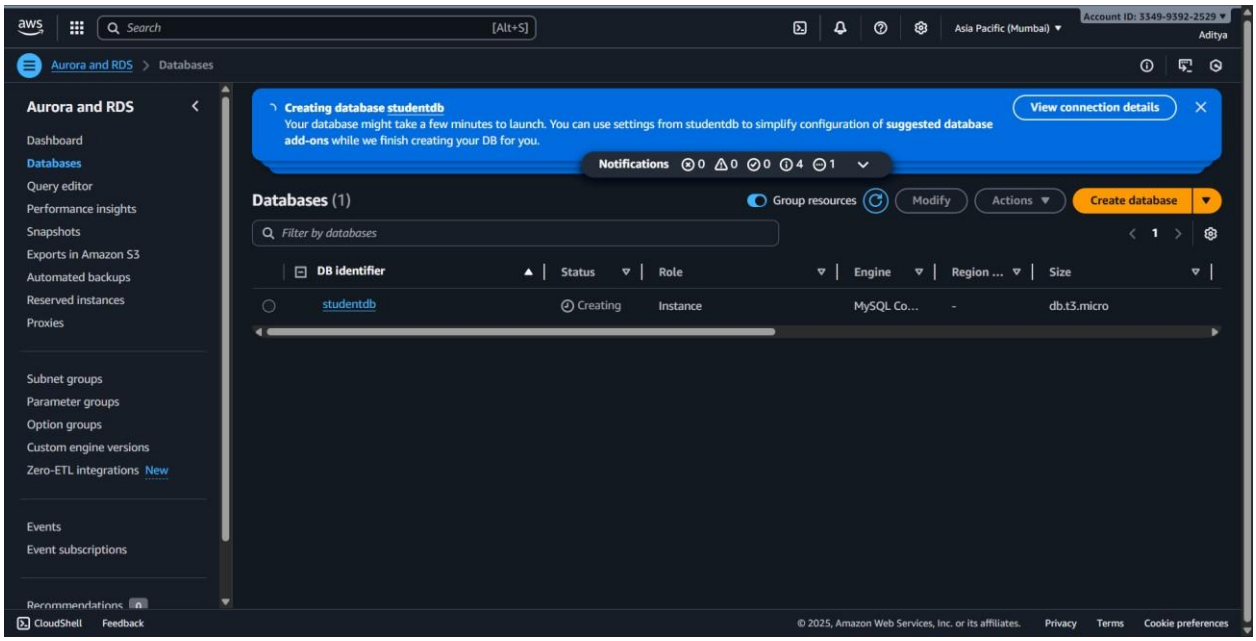


8. Instance size: choose **db.t3.micro** (free-tier).
9. Storage: Select **20 GB General Purpose SSD** (free-tier).

10. VPC: Use your default or custom VPC.

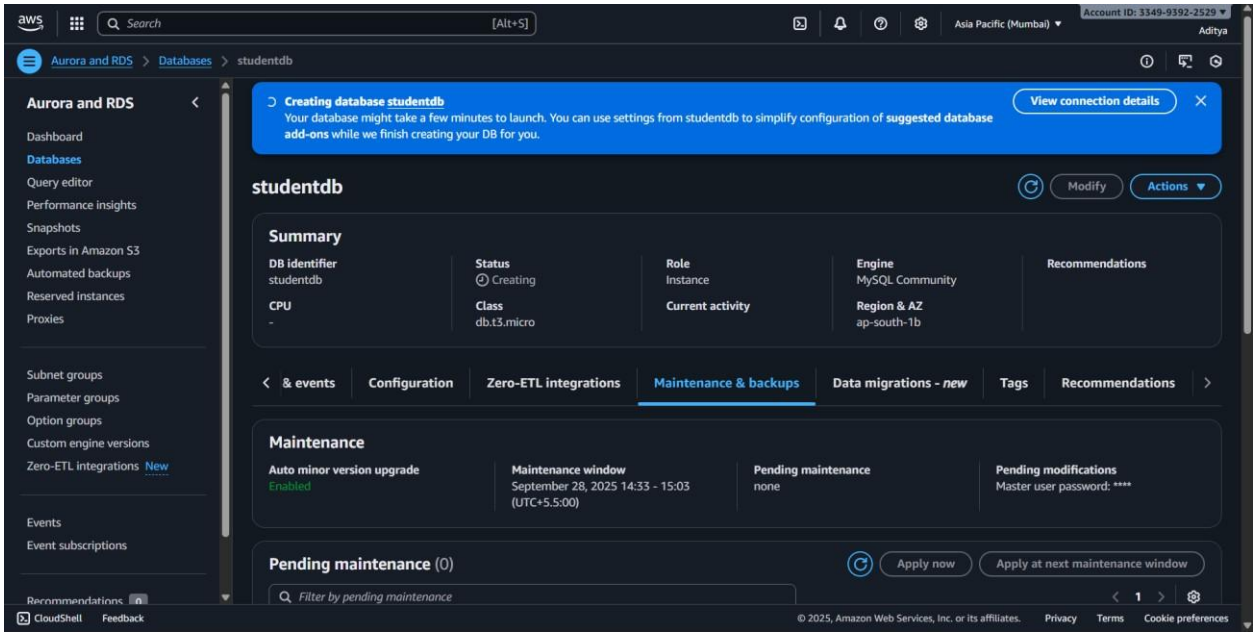
11. Connectivity: Enable public access (for lab only, not production).

12. Click **Create database**.

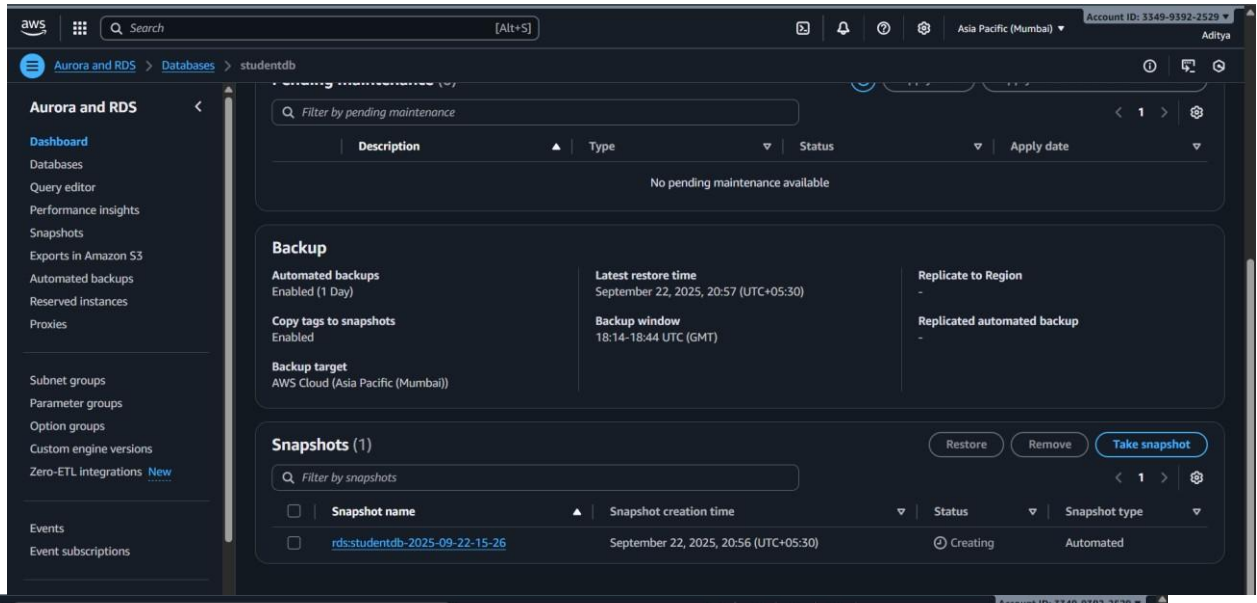


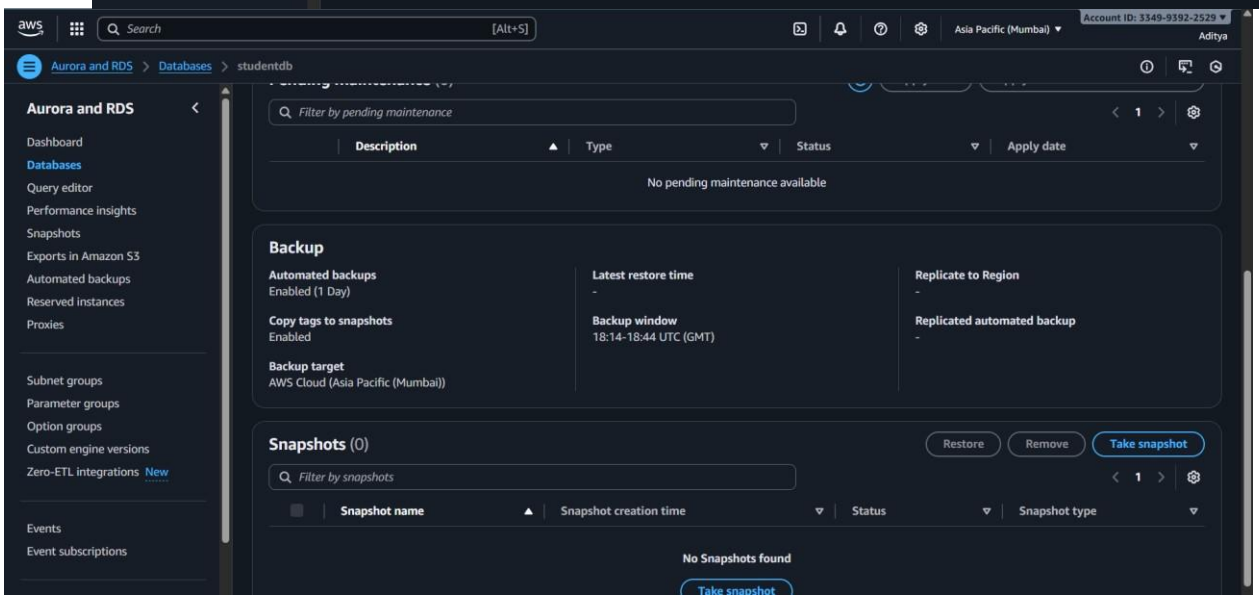
Steps to Implement Backup and Recovery Strategies

1. Go to your RDS instance → **Maintenance & backups** tab.

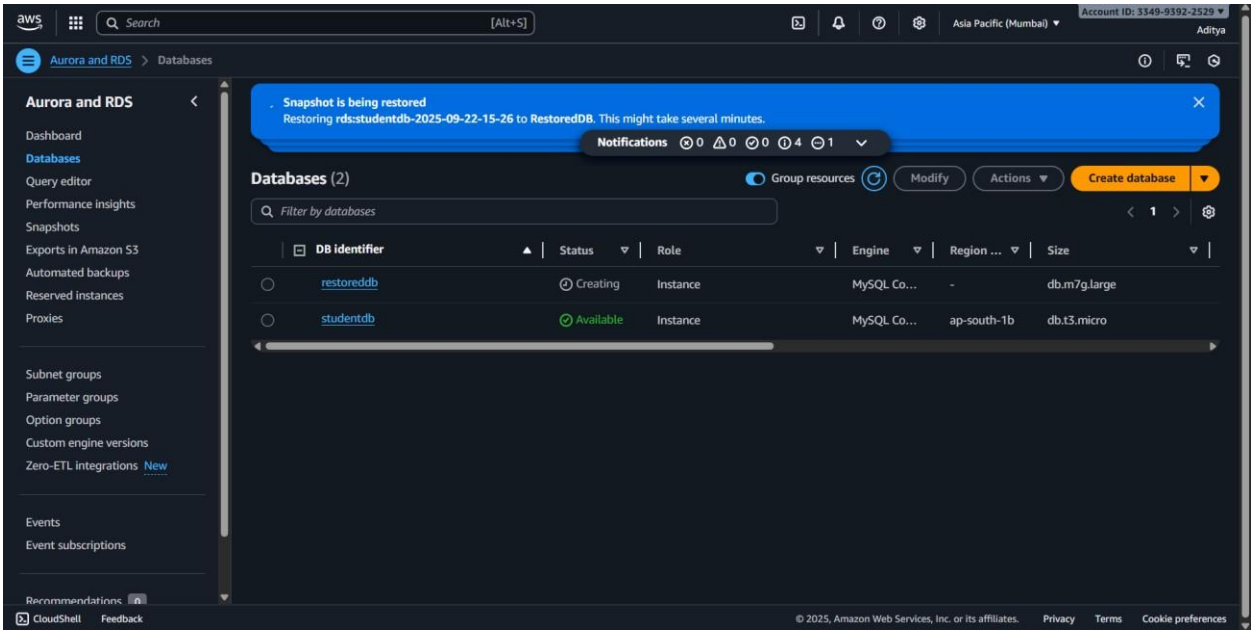


2. Enable **Automated backups**.



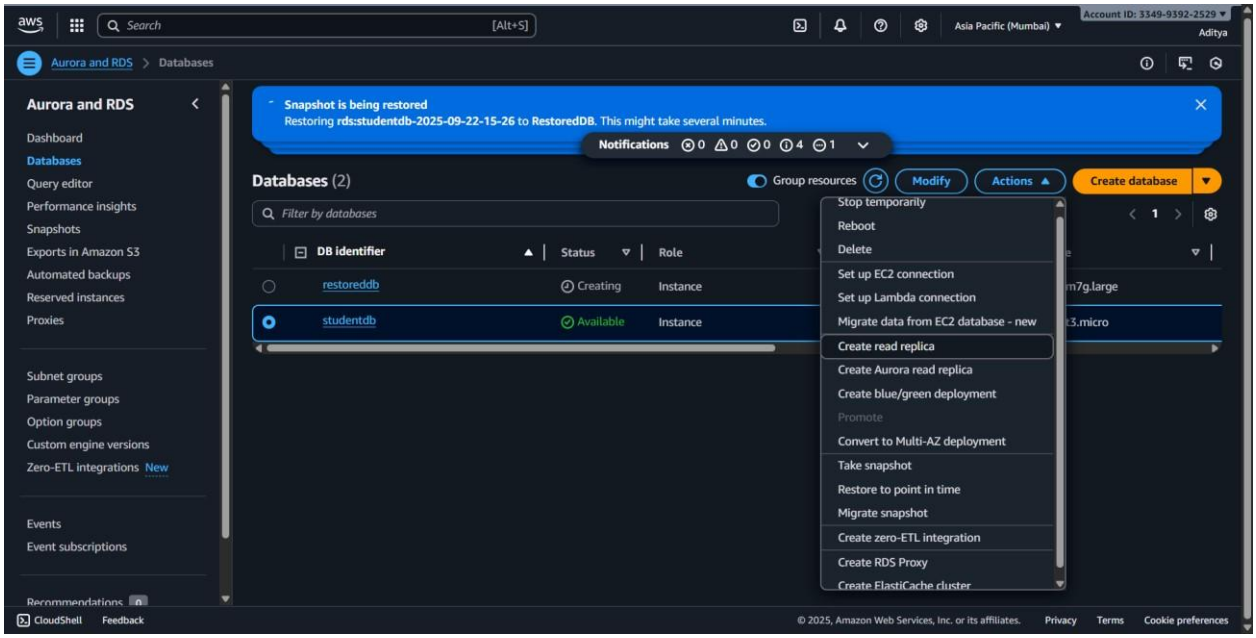


3. Take a **Manual Snapshot**:
    - Actions → Take snapshot.
4. For **Recovery**:
    - Go to Snapshots → Select snapshot → **Restore snapshot** → Launch new RDS instance from snapshot.

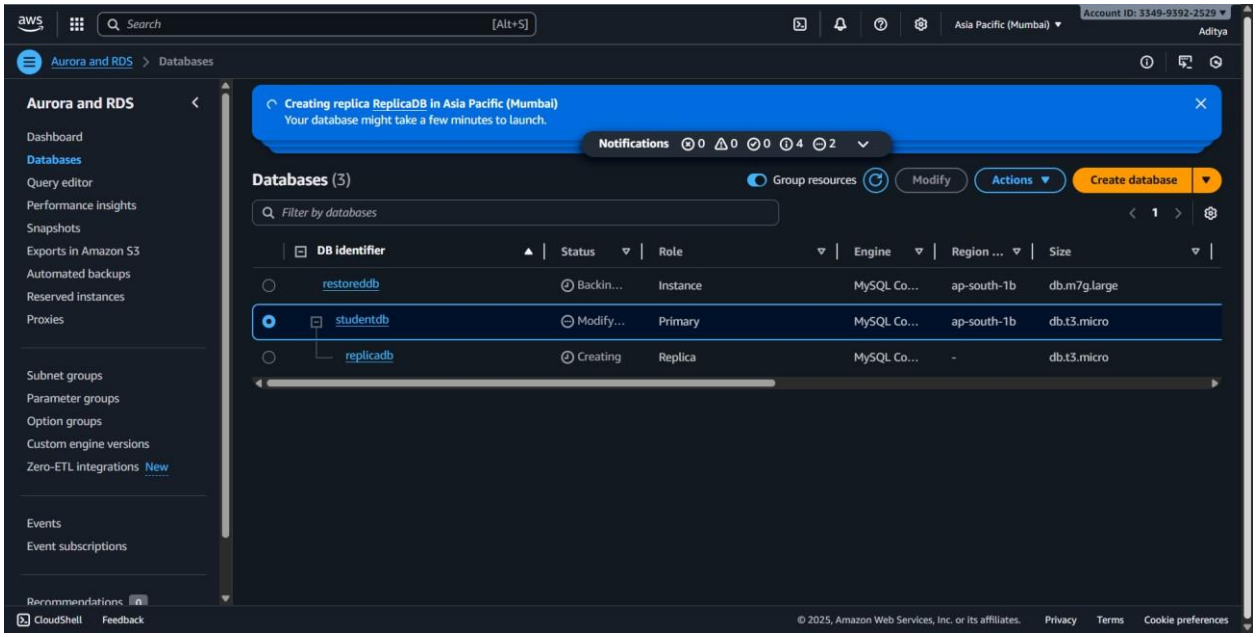


Steps to Set Up Read Replicas

1. Go to your RDS instance.
2. Click **Actions → Create read replica**.



3. Choose instance size → db.t3.micro (to save cost).
4. Keep in same region (lower cost) unless cross-region is required.
5. Create the replica.

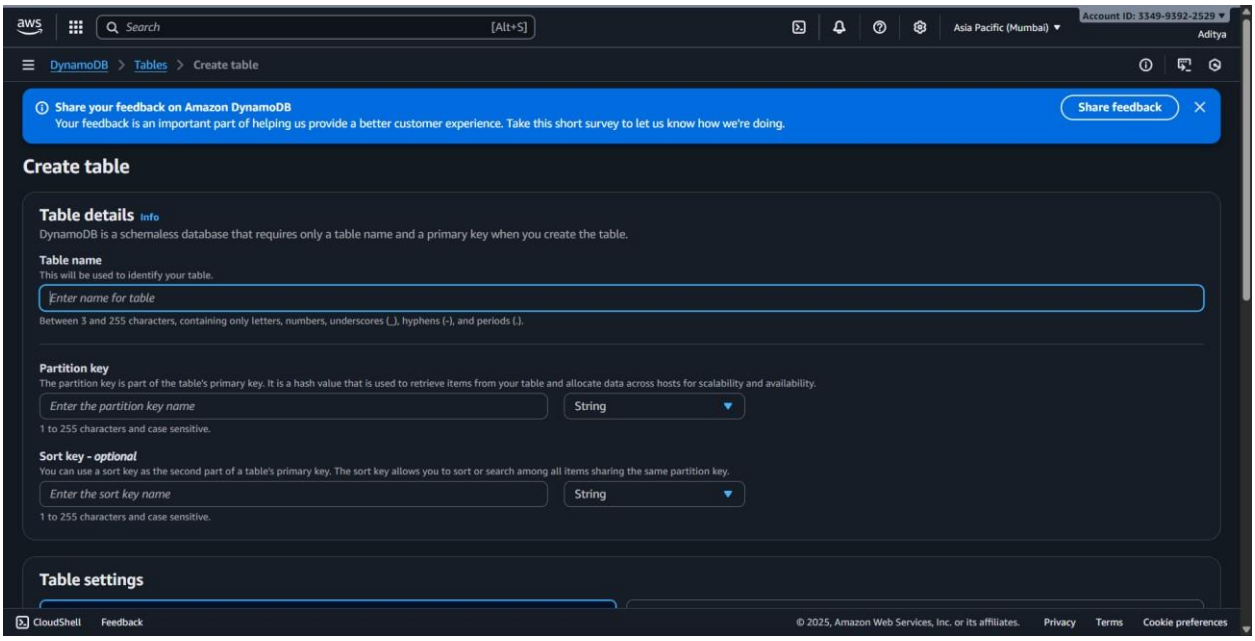


6. Use the **replica endpoint** for read queries (reporting, analytics).

5.Create and configure DynamoDB tables. Design efficient partition and sort keys. Implement Global Secondary Indexes. Set up DynamoDB Streams for real-time processing.

STEPS:

Navigate to **DynamoDB** service and Click on **Create table**.



Enter **Table name**: StudentData.

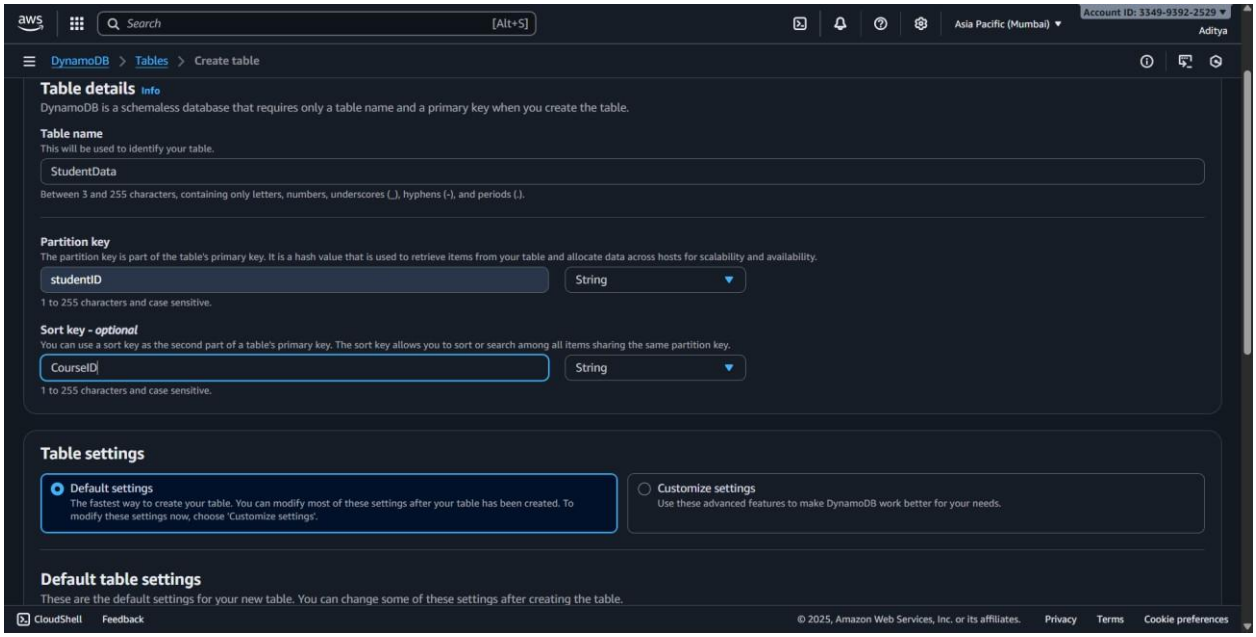
Define the **Partition key**:

- Attribute name: StudentID (String).

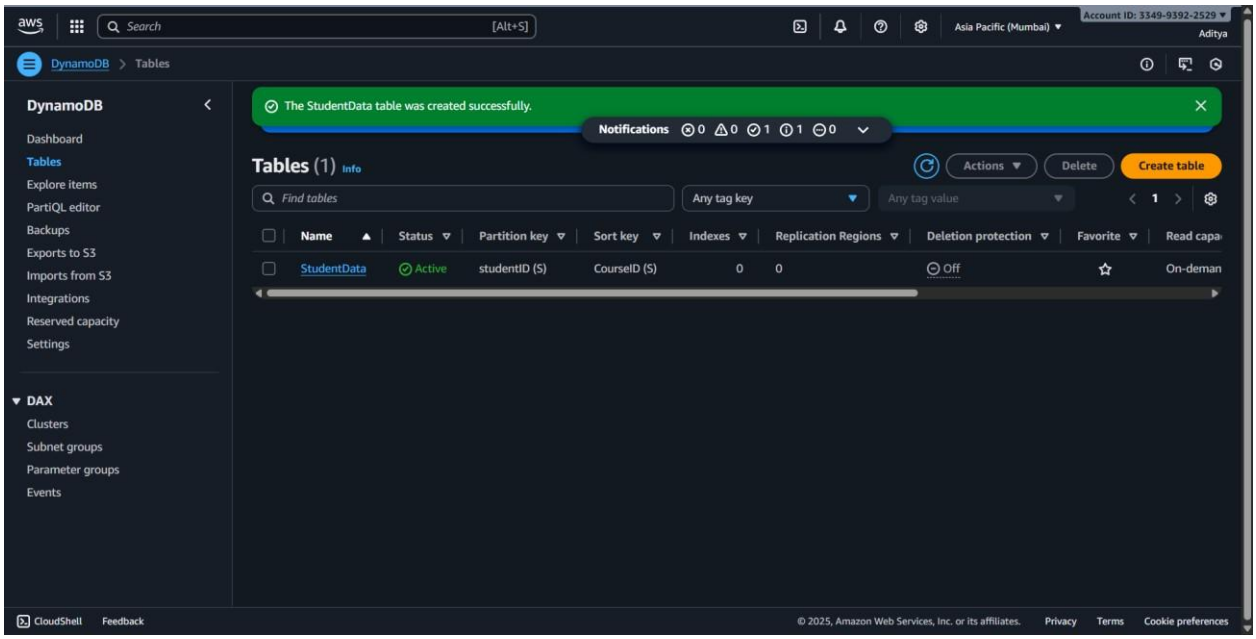
Define the **Sort key**:

- Attribute name: CourseID (String).
  (This helps uniquely identify students enrolled in different courses.)

Leave default settings for **Table settings** or enable **Provisioned capacity** if you want to define read/write units.



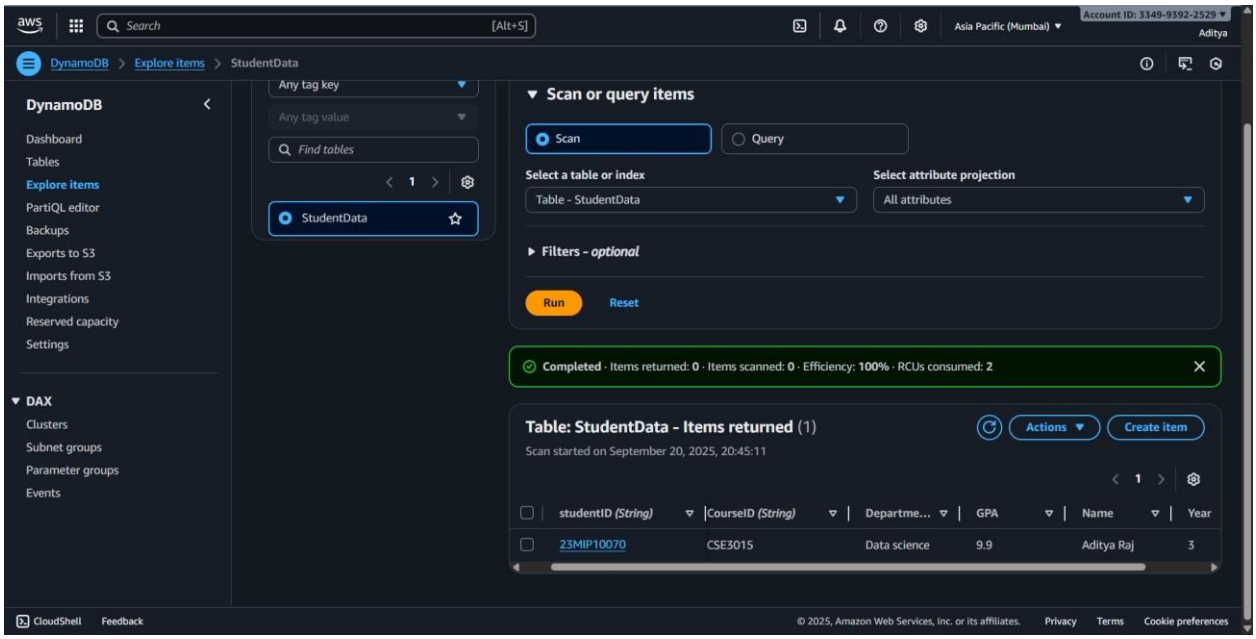
Click **Create table**.



Steps to Design Efficient Partition and Sort Keys

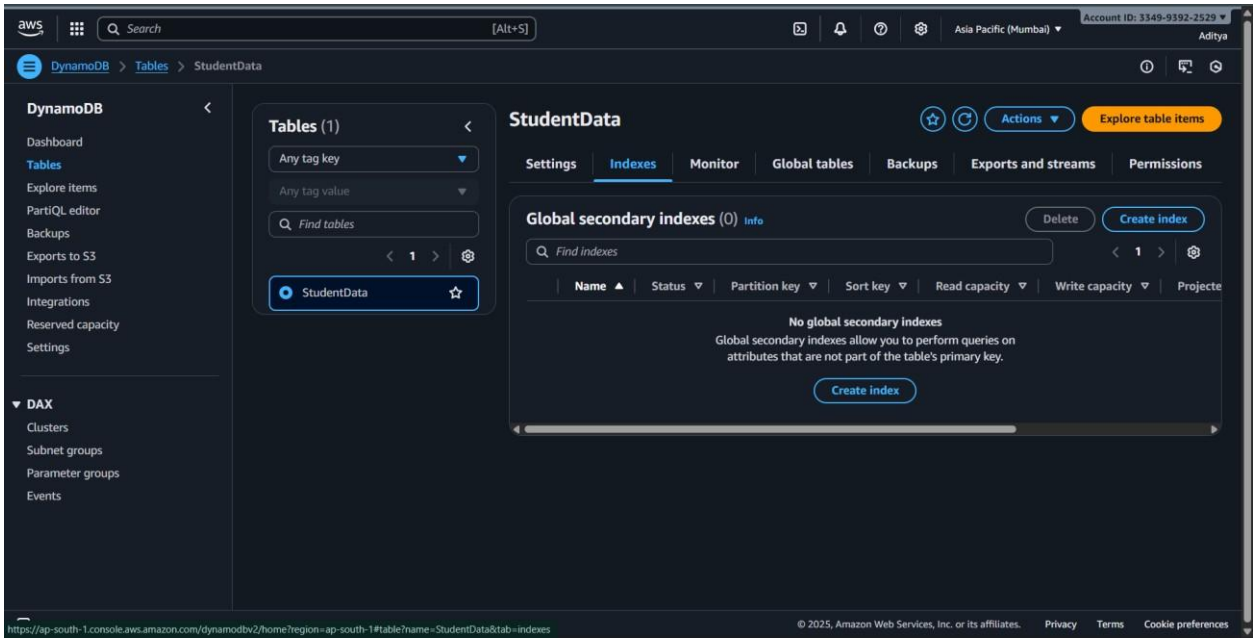
1. Choose StudentID as **Partition Key** → ensures even distribution of student records.

2. Choose CourseID as **Sort Key** → allows multiple course enrollments per student.
3. Add attributes such as:
    - Name (String) ○ Department (String)
    - Year (Number) ○ GPA (Number).



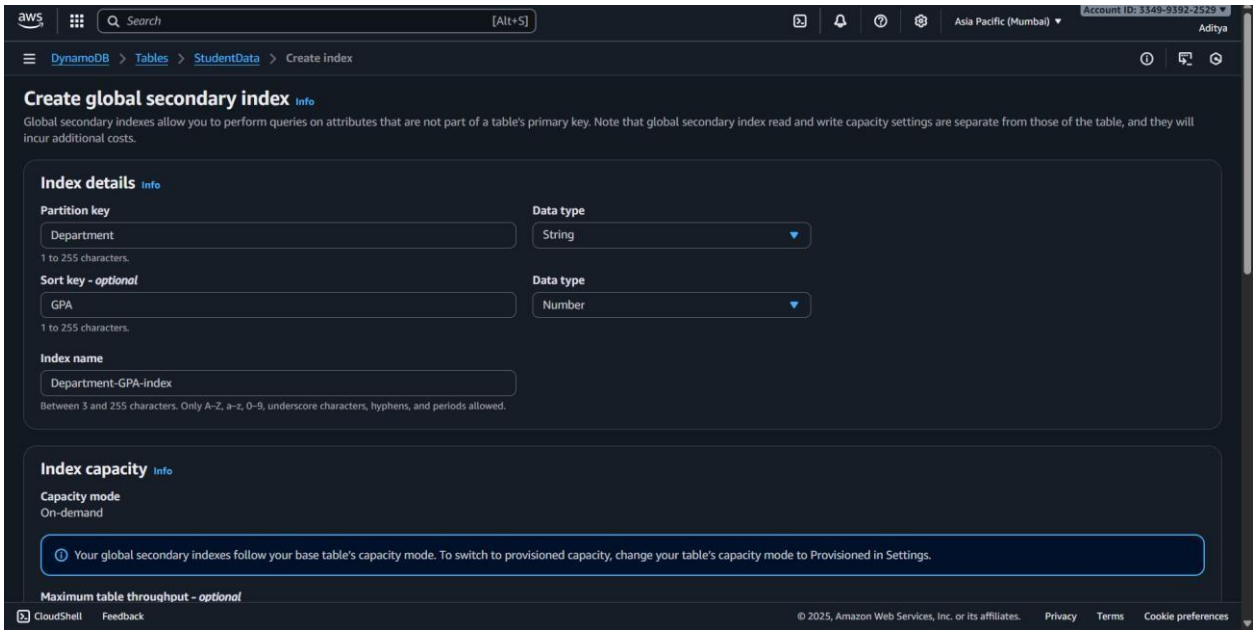
Steps to Implement Global Secondary Indexes (GSI)

1. Open the created **StudentData** table.
2. Go to **Indexes → Create index**.

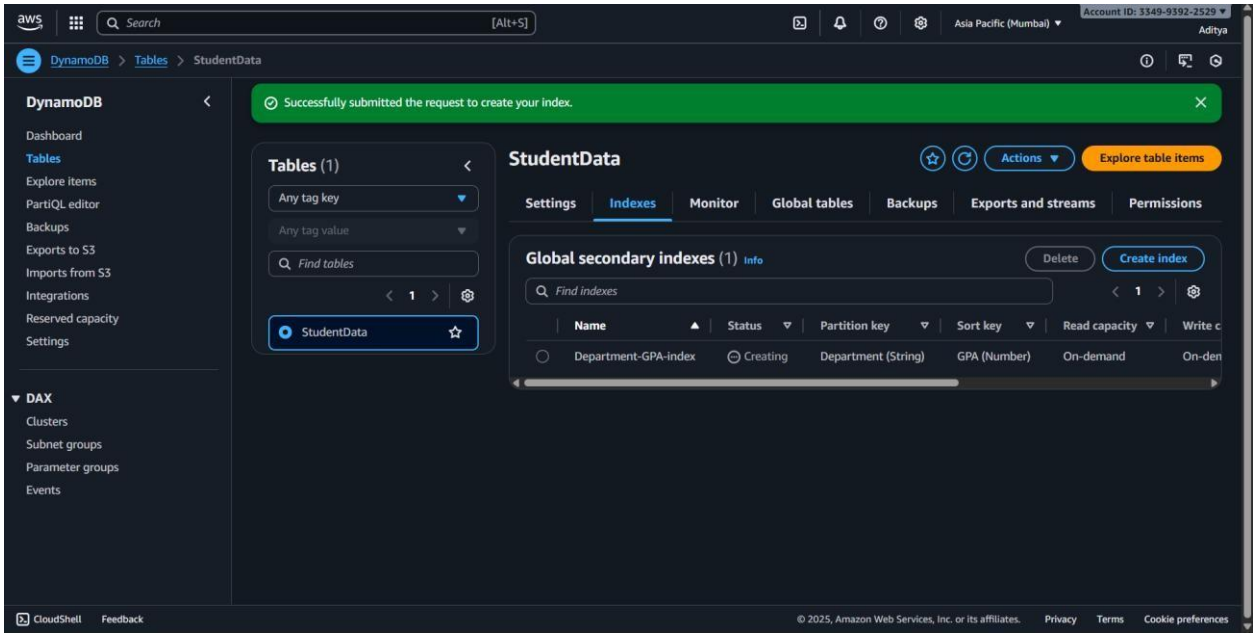


3. Define **Partition key** for the GSI: ○ Example: Department (String).
4. Define **Sort key** for the GSI:
    - Example: GPA (Number).
      (This allows queries like "Find students by Department and sort by GPA.") 5.

Select **Projected attributes**: All or specific (Name, Year, GPA).



6. Click **Create index**.

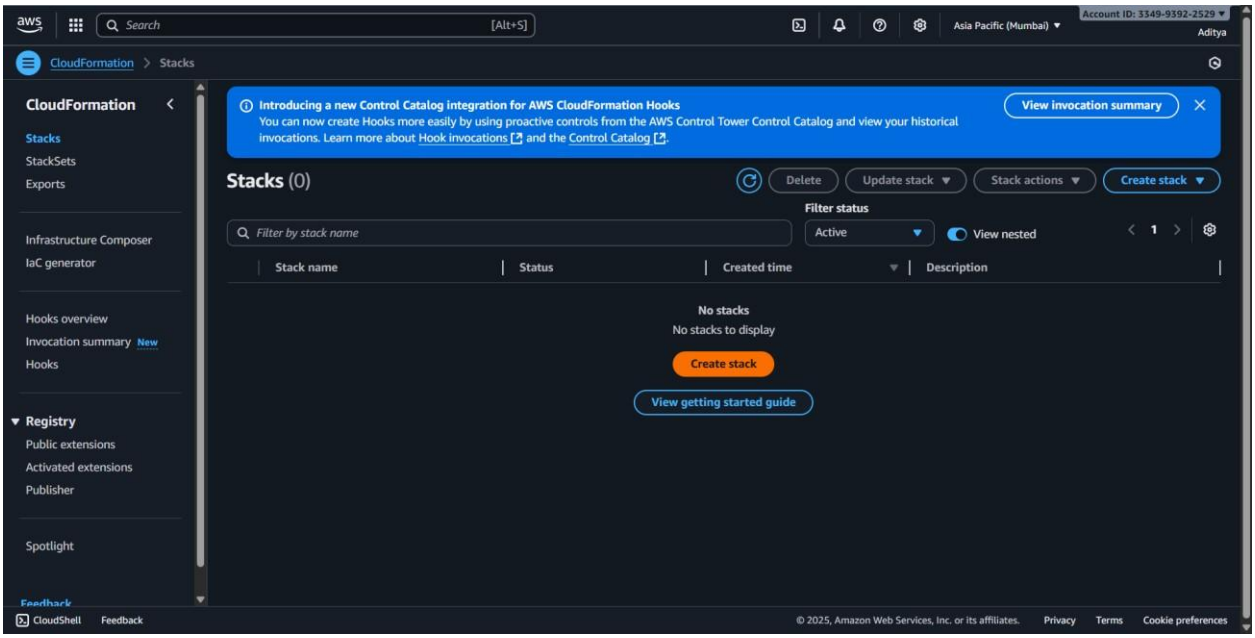


6.Create CloudFormation templates for infrastructure. Implement nested stacks and cross-stack references. Manage stack updates and rollbacks. Integrate with CI/CD pipelines.

Steps:

Open AWS Console → CloudFormation

- Click Create stack → With new resources (standard).



- **Upload Root Template** ○ Upload `root-stack.yml` (this will call child stacks).

- **Nested Stacks** ○ In `root-stack.yml`, reference child templates (e.g., `vpc.yml`, `ec2.yml`).

- **Cross-Stack References** ○ In `vpc.yml`, define `Outputs` (e.g., VPC ID).
    - ○ In `ec2.yml`, use `Fn::ImportValue` to fetch that VPC ID.

- **Deploy Stack** ○ Click **Next → Next → Create stack**.

- **Update & Rollback** ○ Modify a child template (e.g., change EC2 type). ○ Deploy again → CloudFormation shows update in progress.
    - ○ If error, AWS automatically rolls back to last good state.

- **CI/CD Integration** ○ In CodePipeline, add a stage to deploy CloudFormation template after build.

7.Deploy applications using Elastic Beanstalk. Configure application environments.

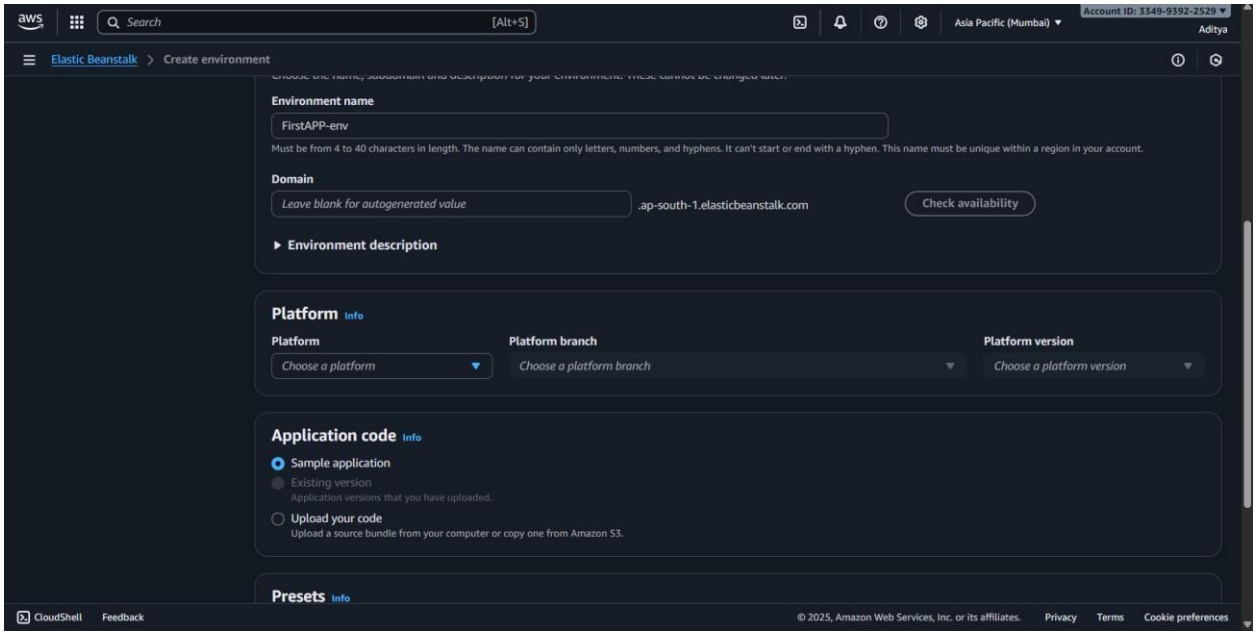
Implement blue-green deployments. Monitor application health and performance.

Step 1: Open Elastic Beanstalk

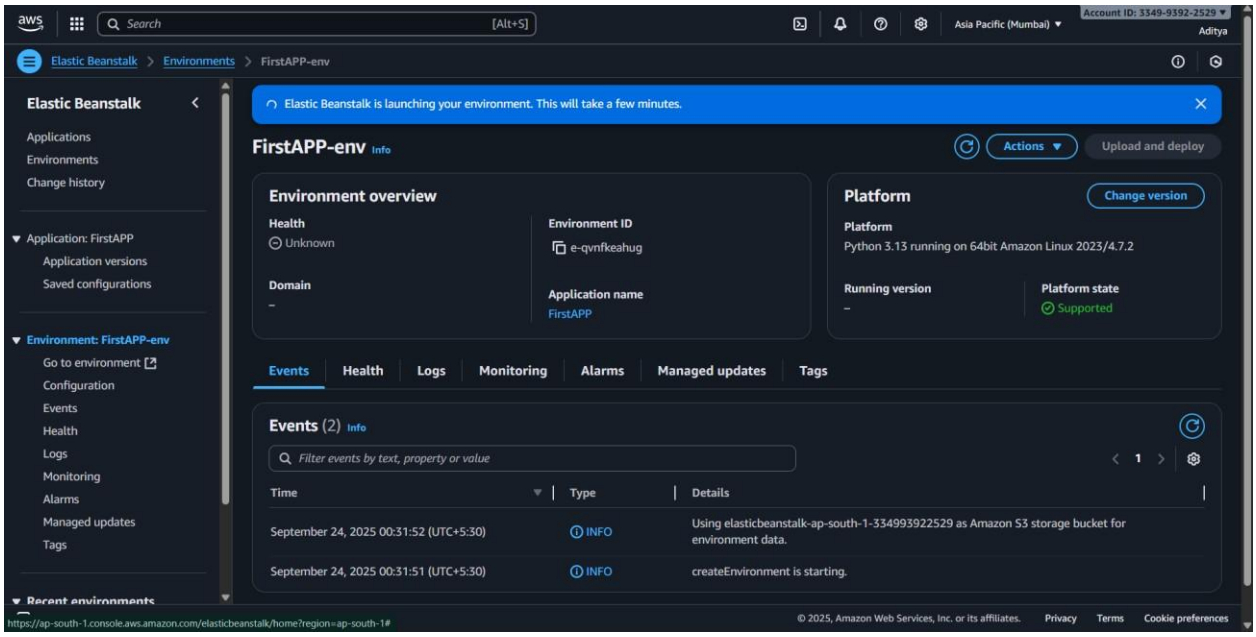
1. Sign in to **AWS Management Console**.
2. Search for **Elastic Beanstalk** in the search bar.
3. Click **Create Application**.

Step 2: Create an Application

1. Enter **Application Name** (e.g., my-eb-app).
2. Choose **Platform** (e.g., Python, Node.js, Java, or Tomcat).
3. For **Application code**, you can either:
    - Upload your **.zip** application code, OR
    - Choose **Sample application** (easier for lab testing).



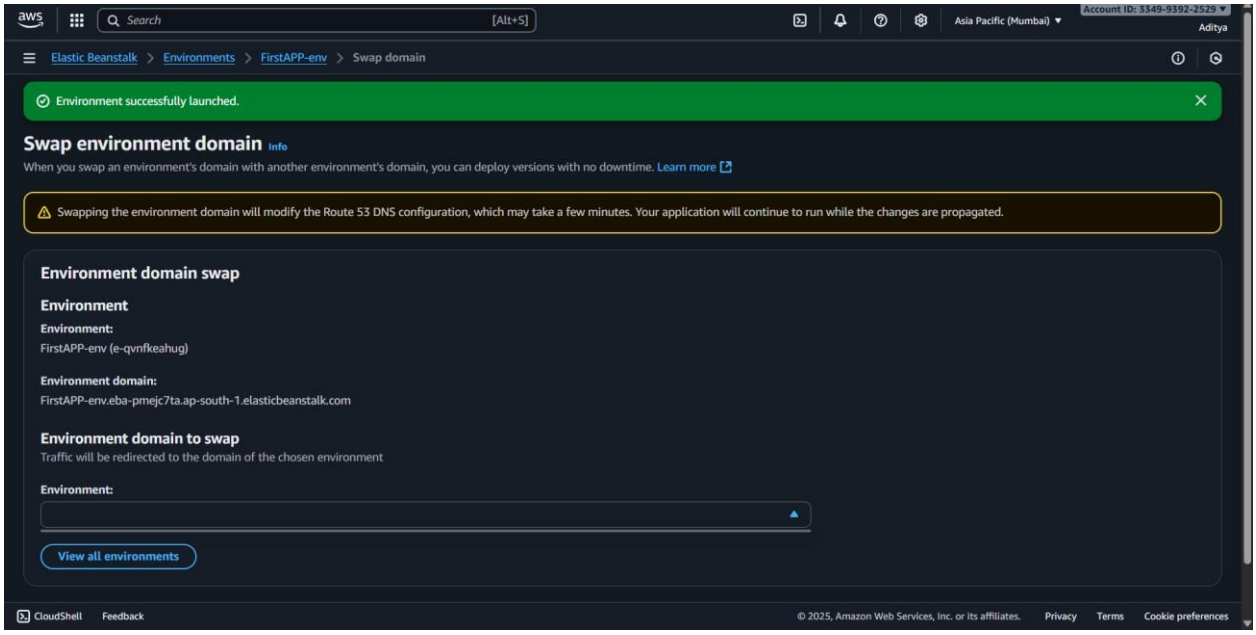
4. Click **Create application** → AWS will automatically create an **Environment** with EC2, Load Balancer, Auto Scaling, and S3 bucket.



Step 3: Configure the Environment

Step 4:

1. Click on your environment (e. Create a **new environment** → choose **Web server environment**.
    - Example: my-eb-app-green.
    - Upload the updated version of your app here.
2. Once it is deployed, test the **URL of green environment**.
3. To switch traffic:
    - Go to **Elastic Beanstalk → Environments**. ○ Select **Swap environment URLs** between **blue** and **green**.



8.Set up comprehensive logging and monitoring. Create custom metrics and alarms.

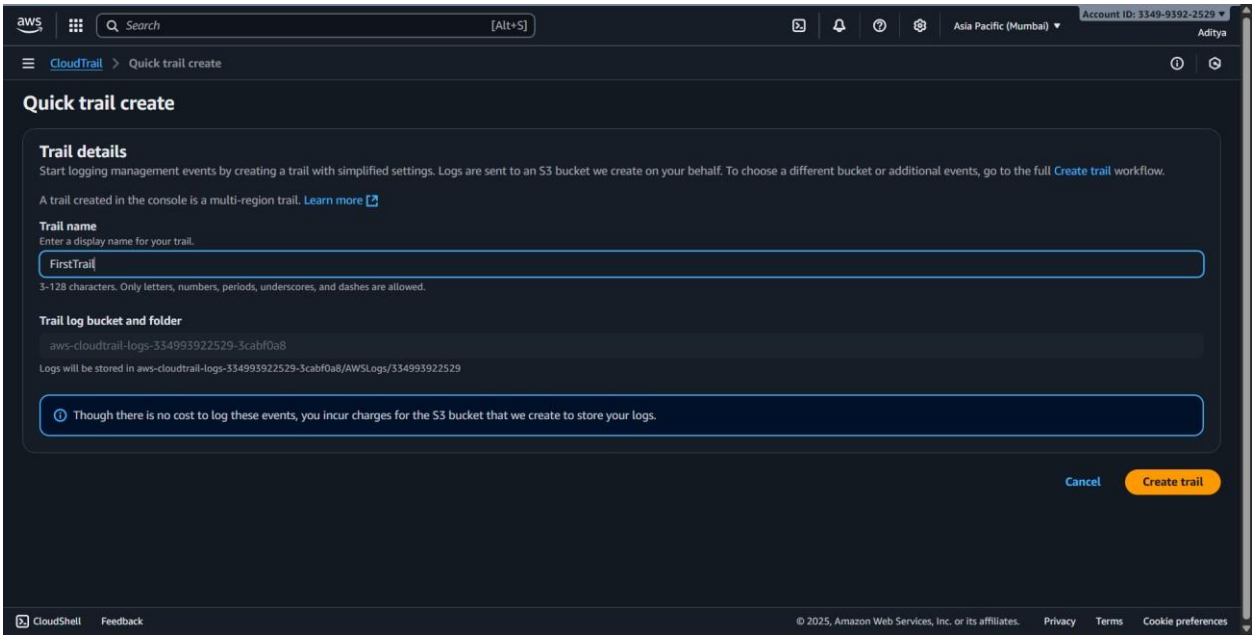
Implement log analysis and alerting, Build operational dashboards.

Steps:

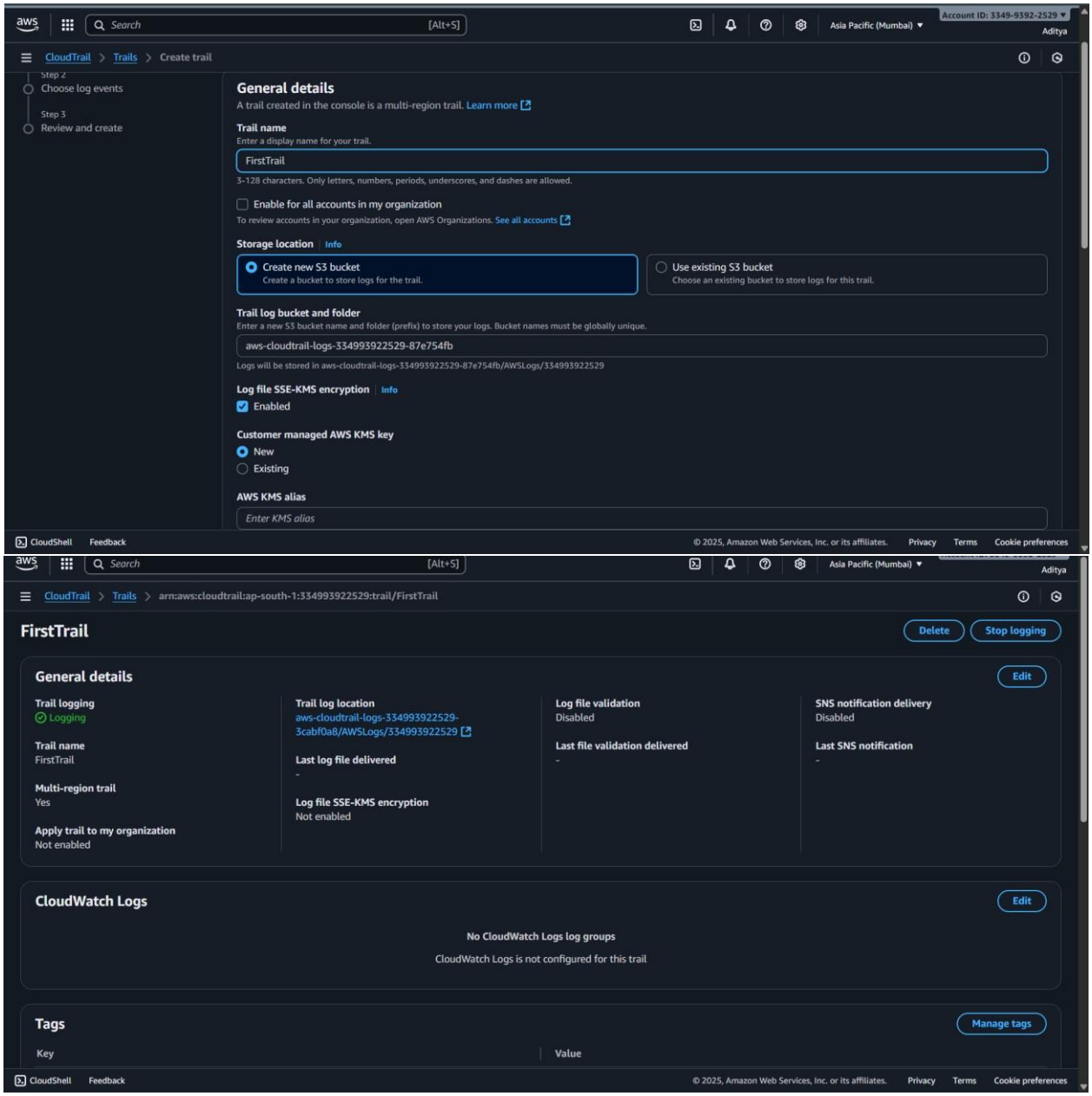
1. **Enable CloudTrail Logging:**

- Open AWS CloudTrail → Click Create trail.

- Give a name for the trail.

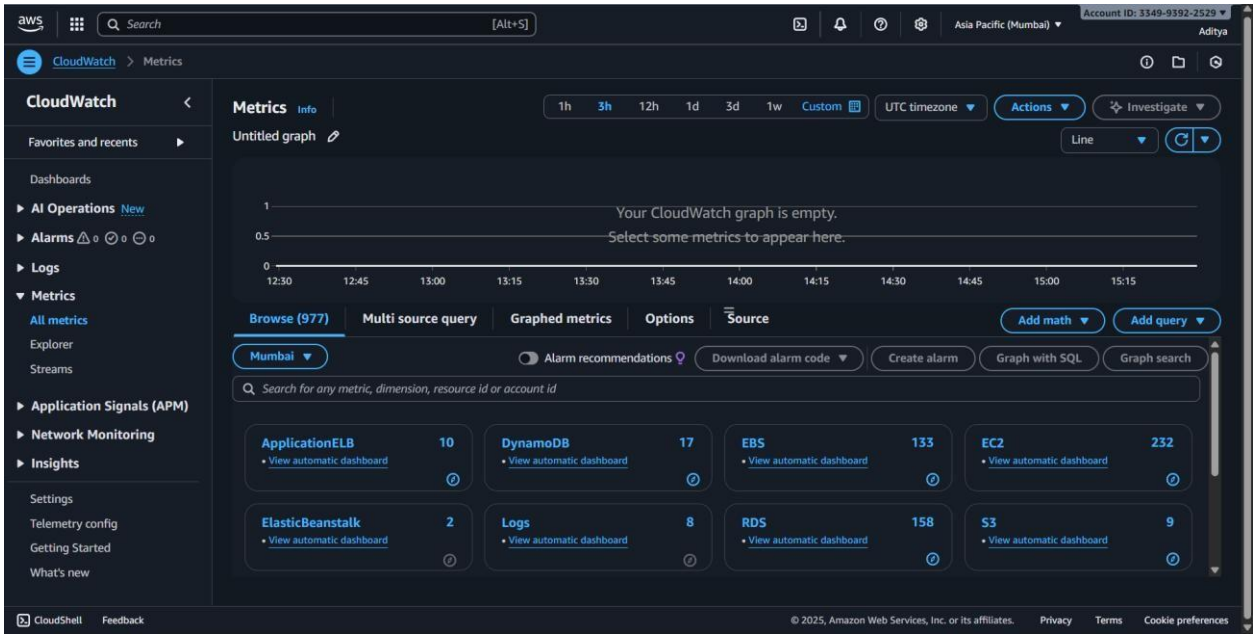


- Choose or create an S3 bucket to store logs.
- Enable CloudWatch Logs integration (optional for real-time monitoring).
- Click Create.

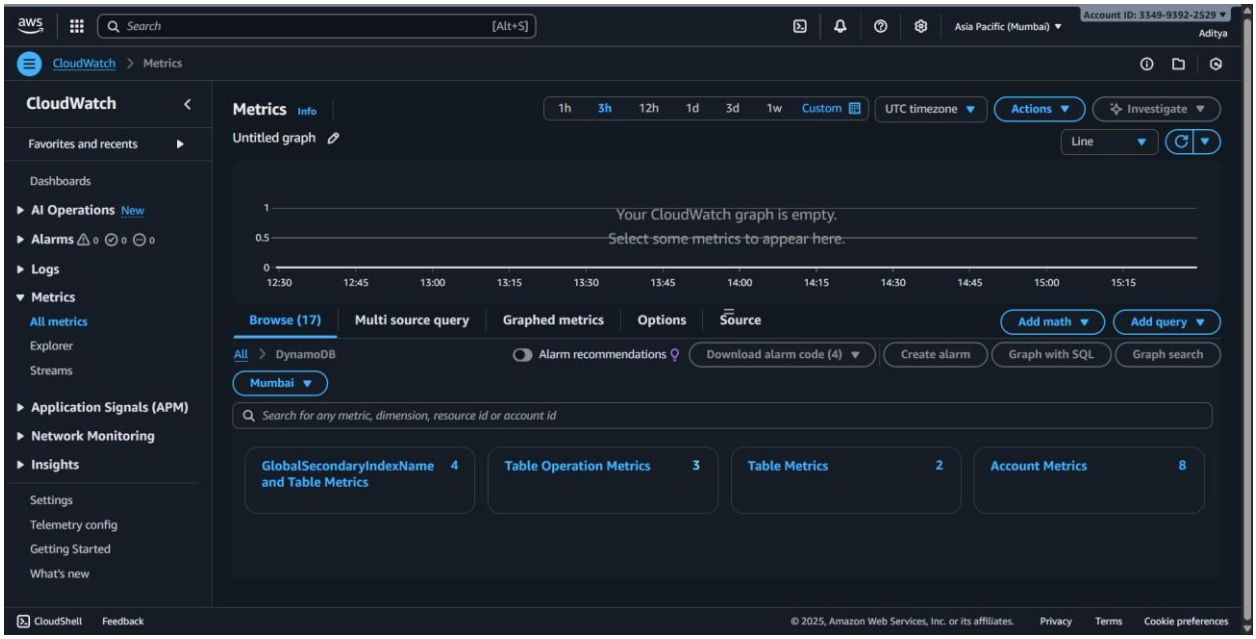


2. Enable CloudWatch Monitoring:

- Open AWS CloudWatch.
- Check default metrics for EC2, RDS, Lambda, etc.

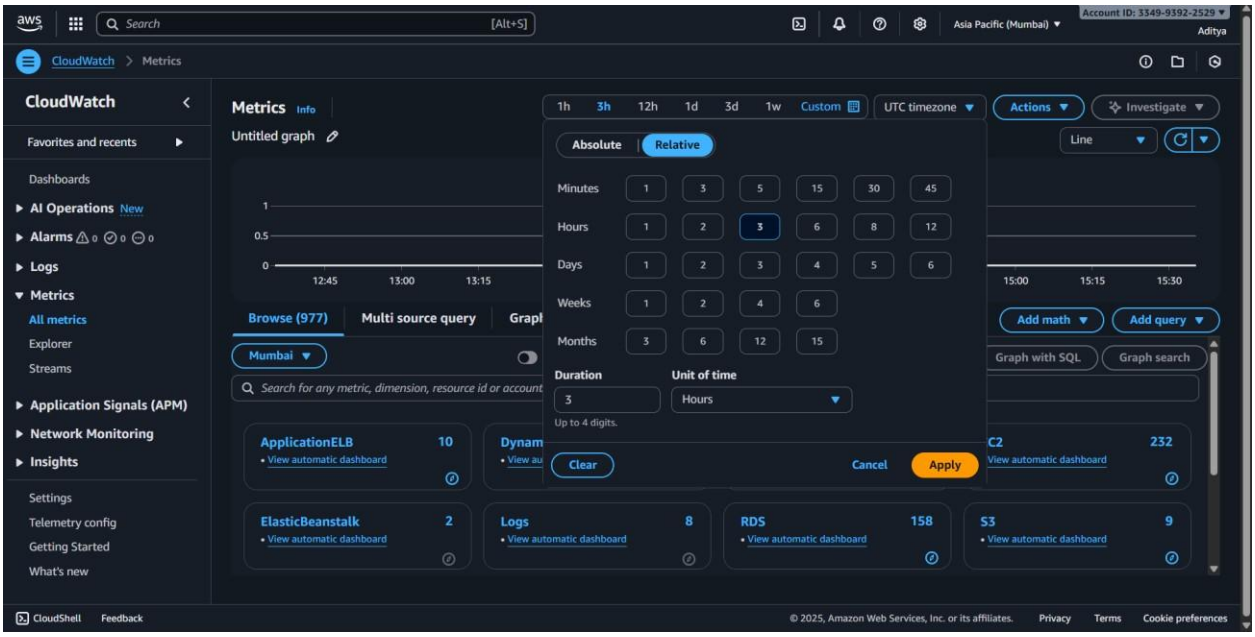


- Make sure monitoring is enabled for resources you want to track.



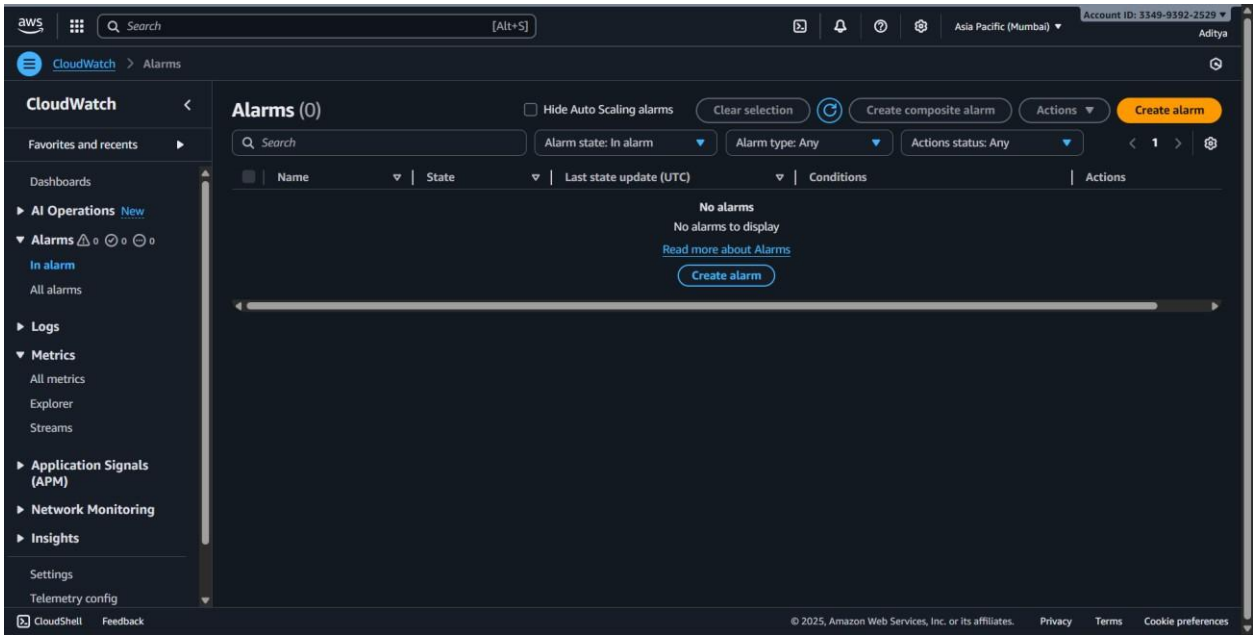
3. Create Custom Metric:

- In CloudWatch → Metrics → Click All Metrics → Create custom metric.
- Publish a metric using AWS CLI or console (e.g., number of requests or applicationspecific data).

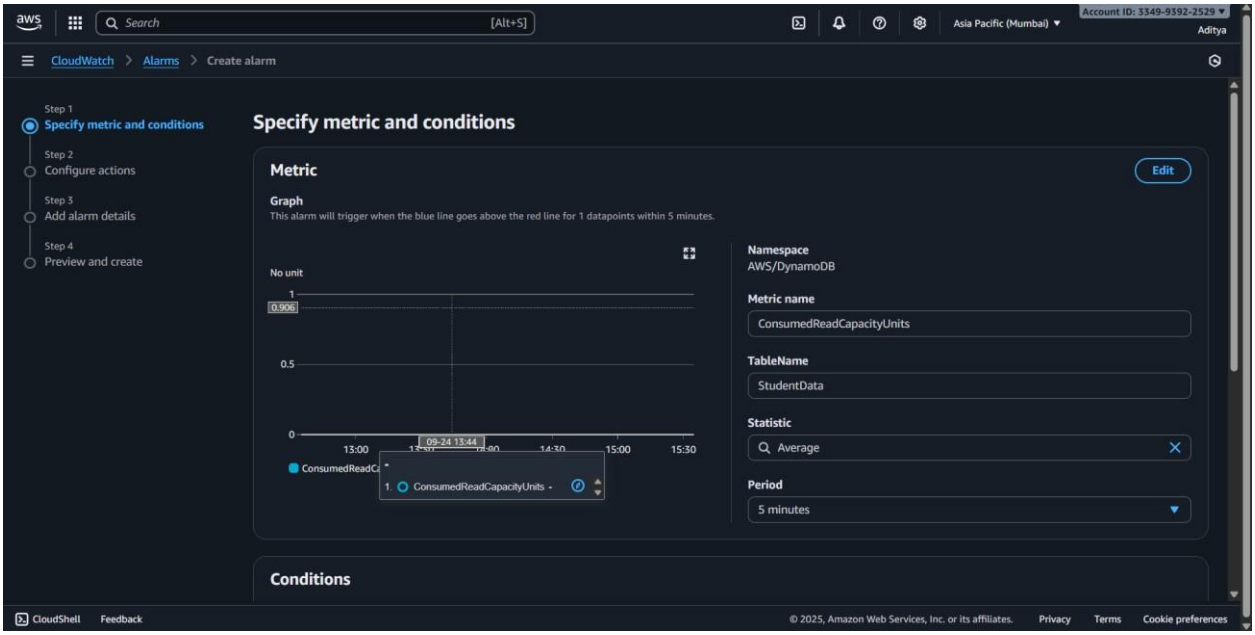


4. Create Alarms:

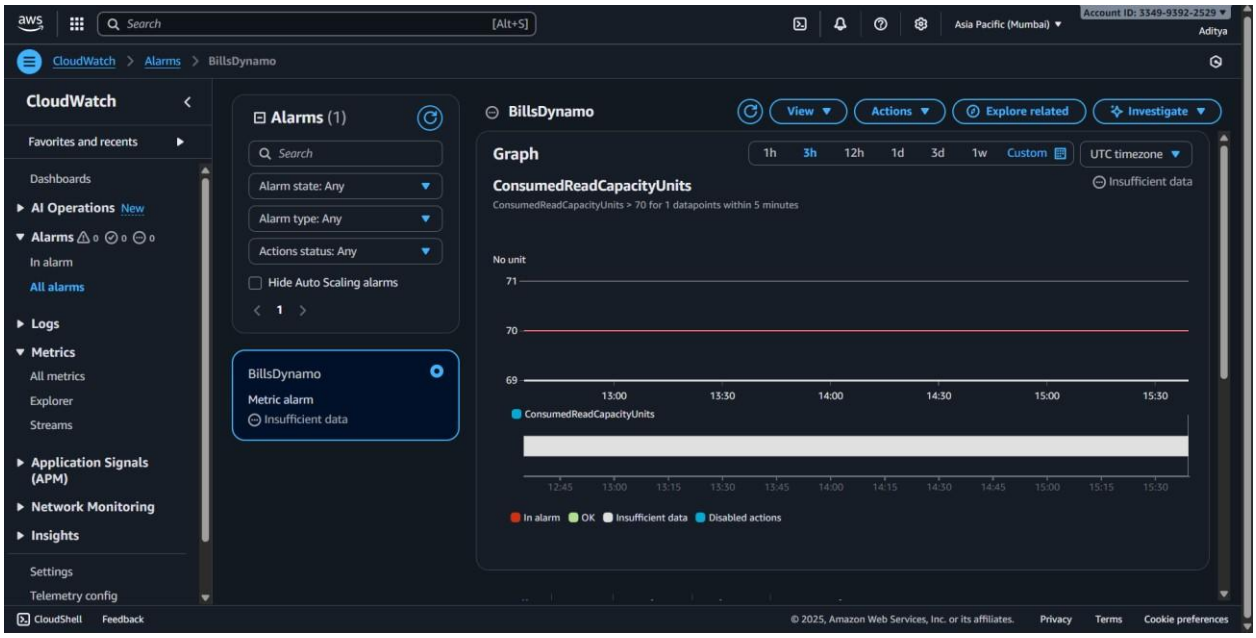
- CloudWatch → Alarms → Create alarm.



- Select the metric (CPU Utilization, custom metric, etc.).

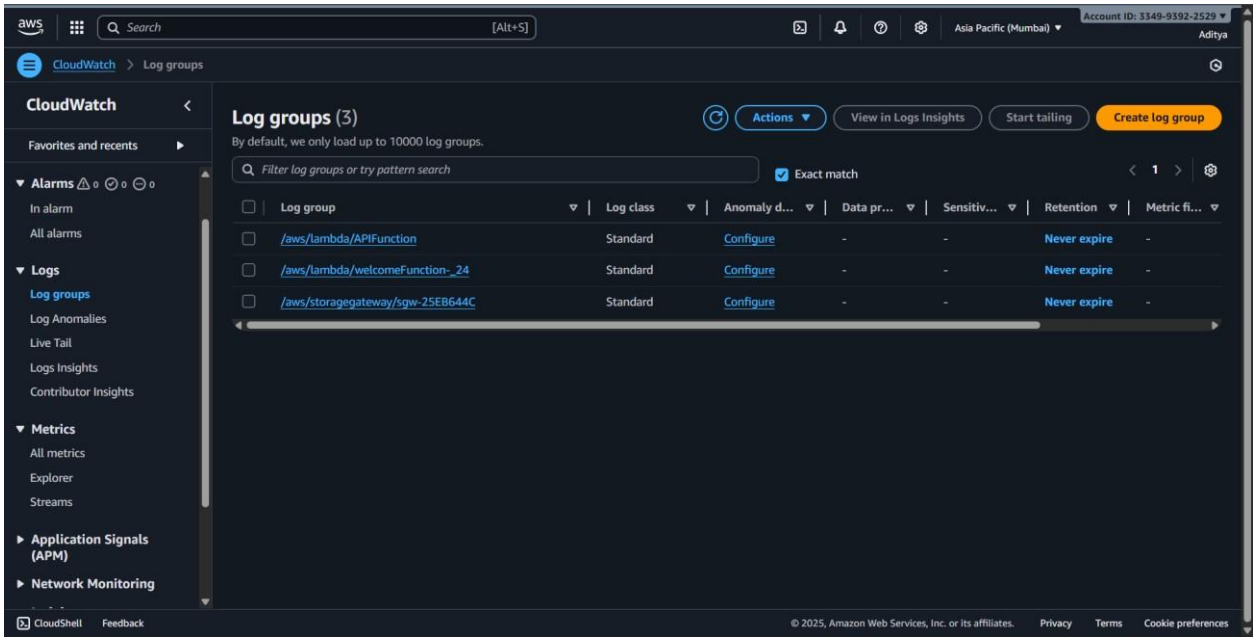


- Set threshold (e.g., > 70%).
- Configure notification via SNS (email or SMS).
- Click Create alarm.

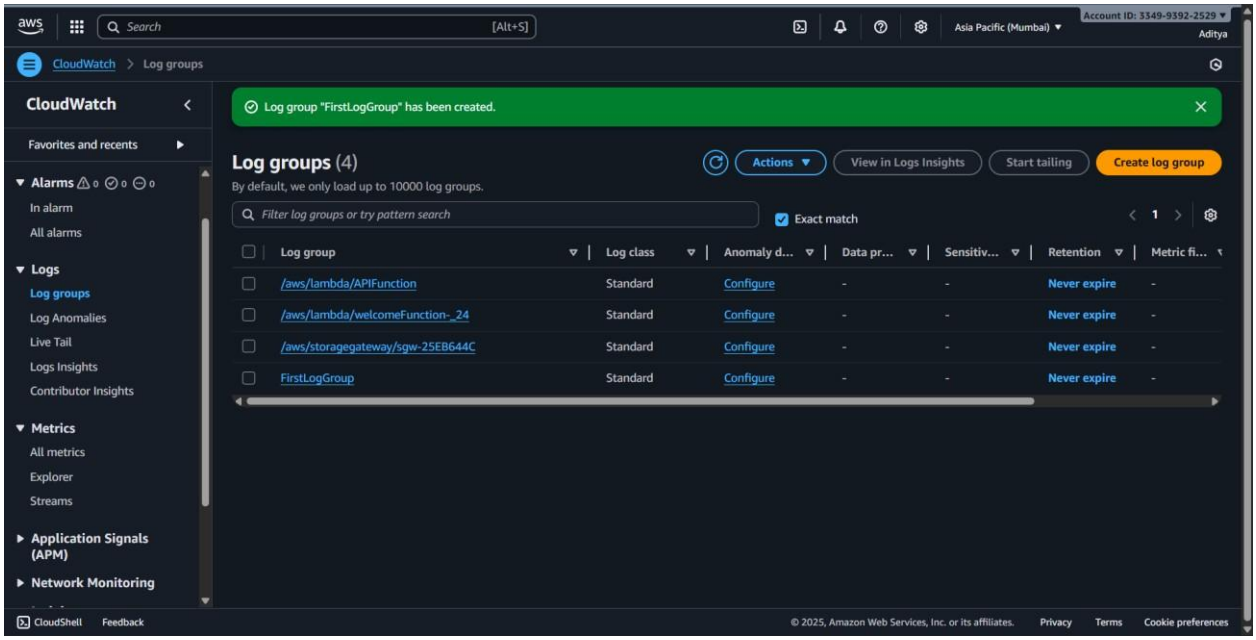


5. Log Analysis:

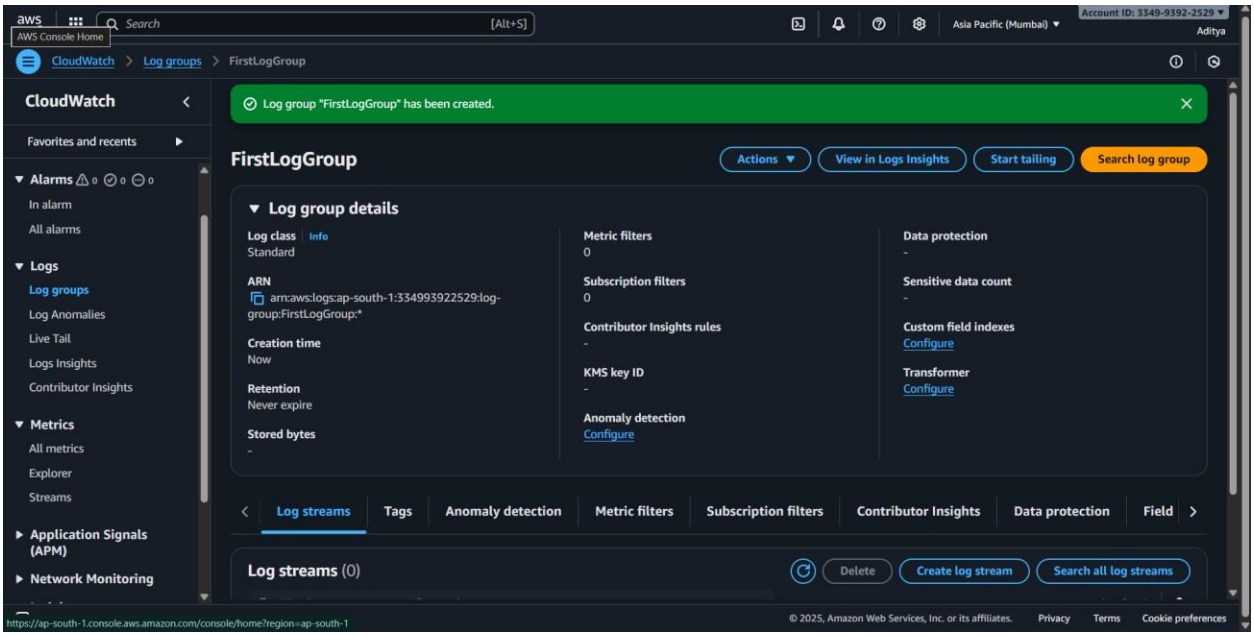
- CloudWatch → Logs → Create Log Group.



- Stream logs from EC2, Lambda, or CloudTrail.

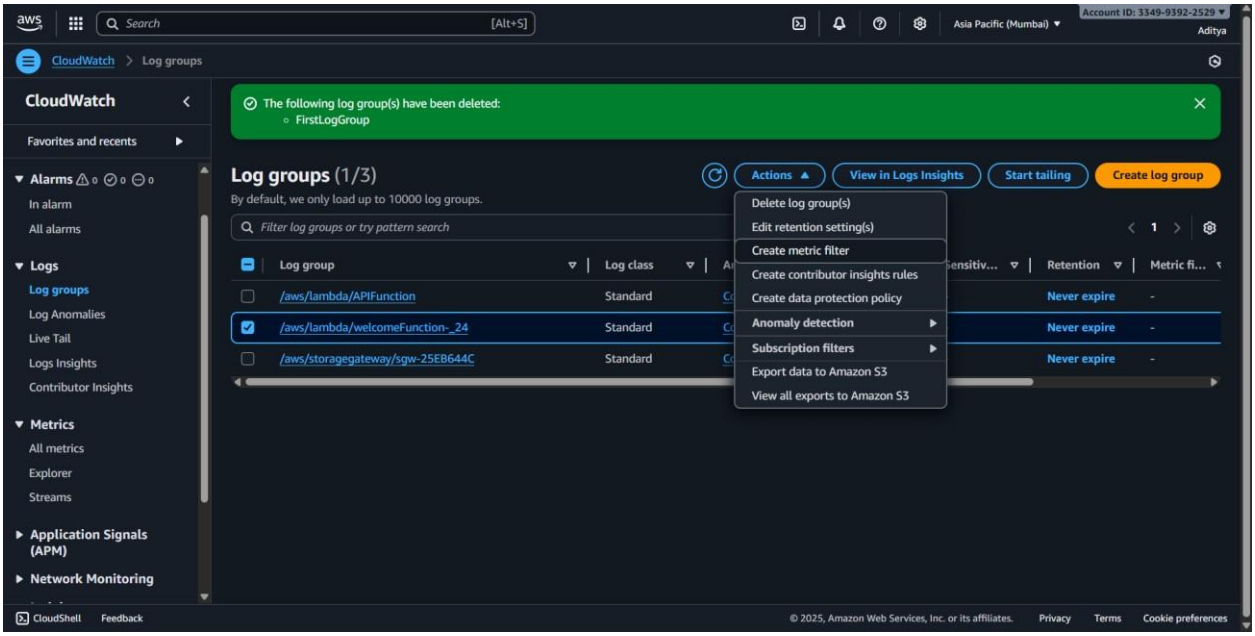


- Use CloudWatch Logs Insights → Run queries like fields @timestamp, @message | filter @message like /ERROR/.

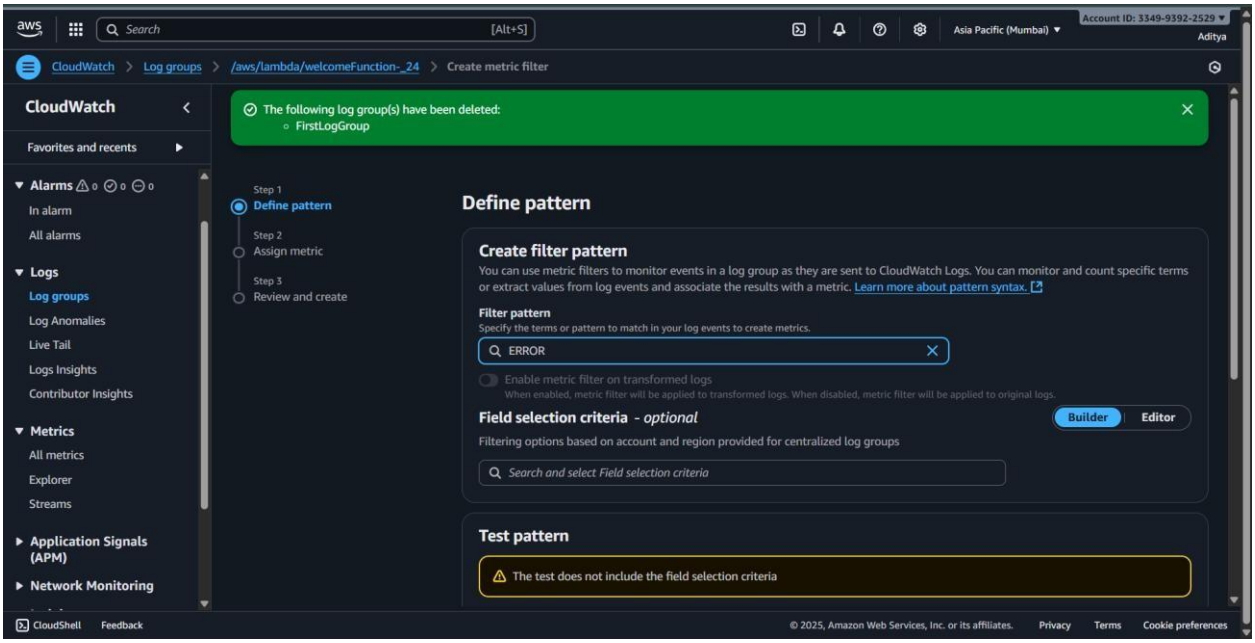


6. Alerting via Log Filters:

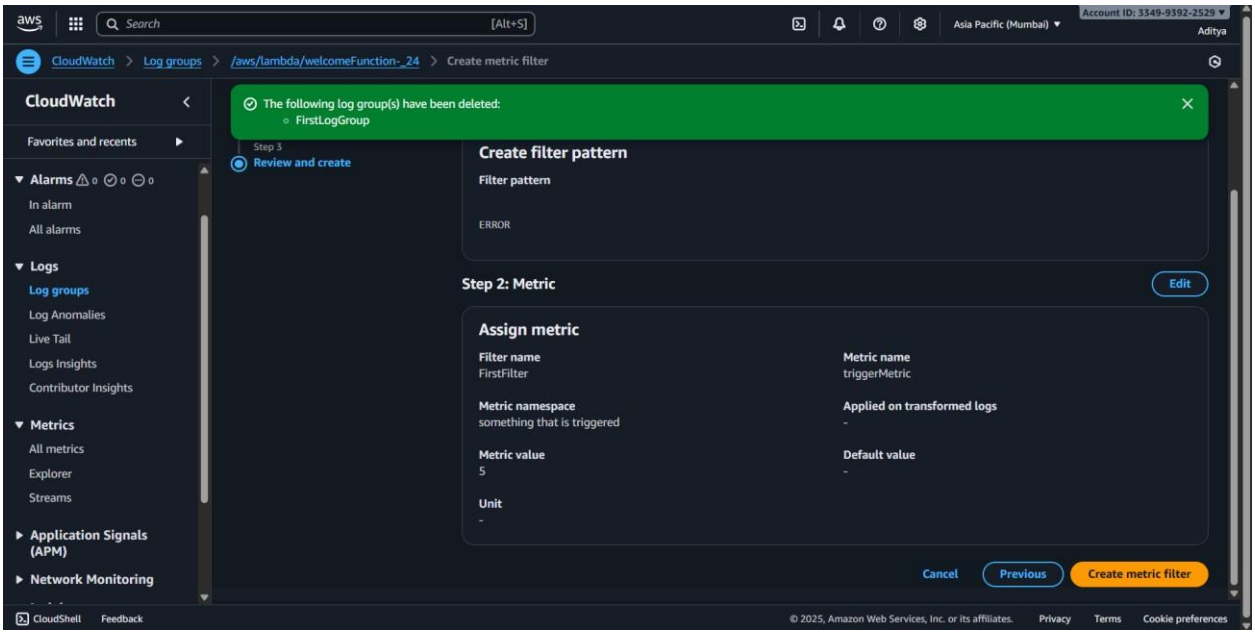
- CloudWatch Logs → **Create metric filter**.



- Define pattern (e.g., “ERROR” in logs).

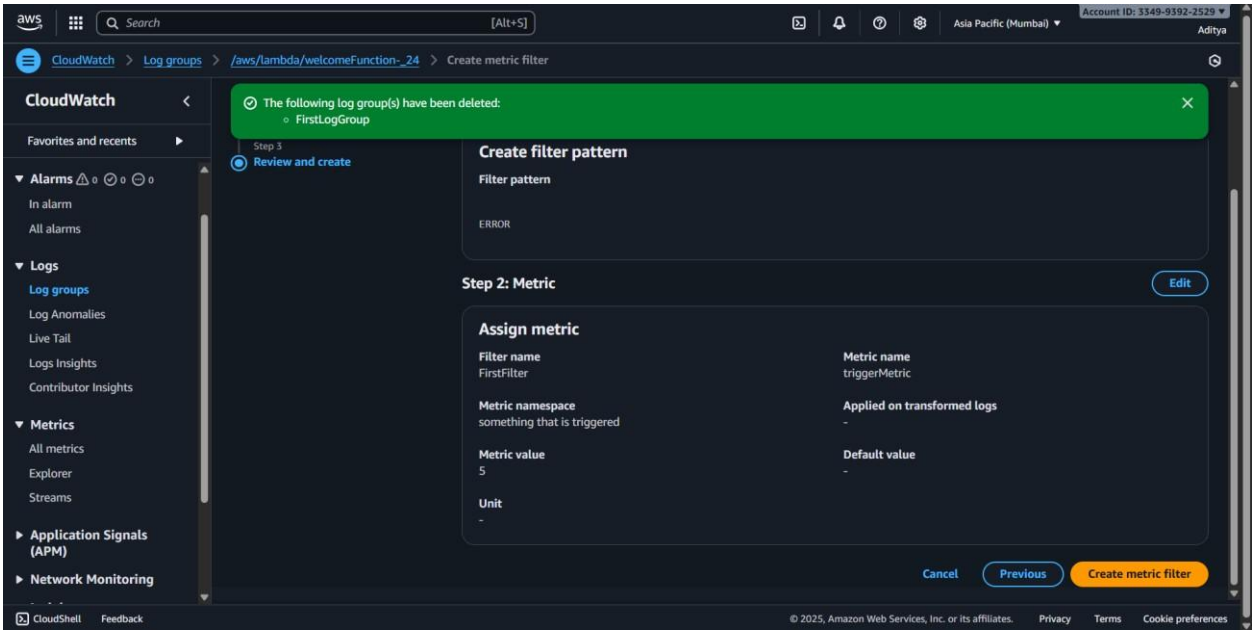


- Assign **alarm** to the filter → Choose SNS topic for notifications.

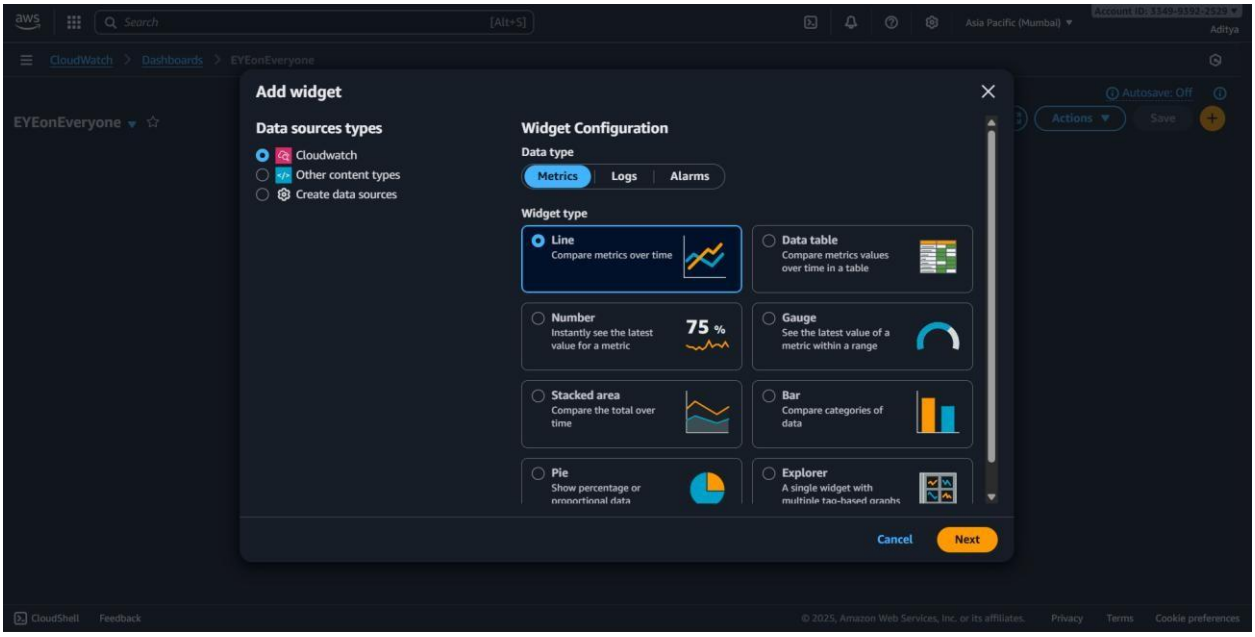


7. Build Operational Dashboard:

- CloudWatch → **Dashboards** → **Create dashboard**.



- Add **widgets** for metrics, alarms, and log visualizations.



- Save the dashboard for monitoring overall system health.

9. Design and implement complex workflows. Handle error conditions and retries.

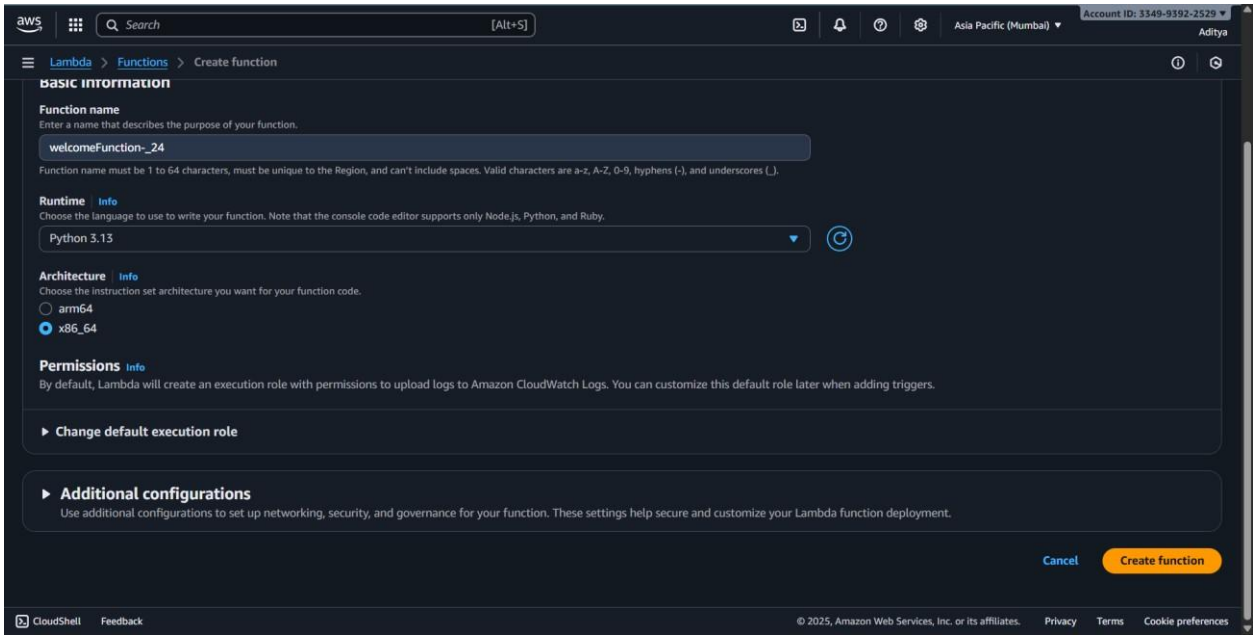
Integrate multiple AWS services. Monitor workflow execution.

Steps

Experiment: Designing and Implementing Complex Workflows in AWS

Steps:

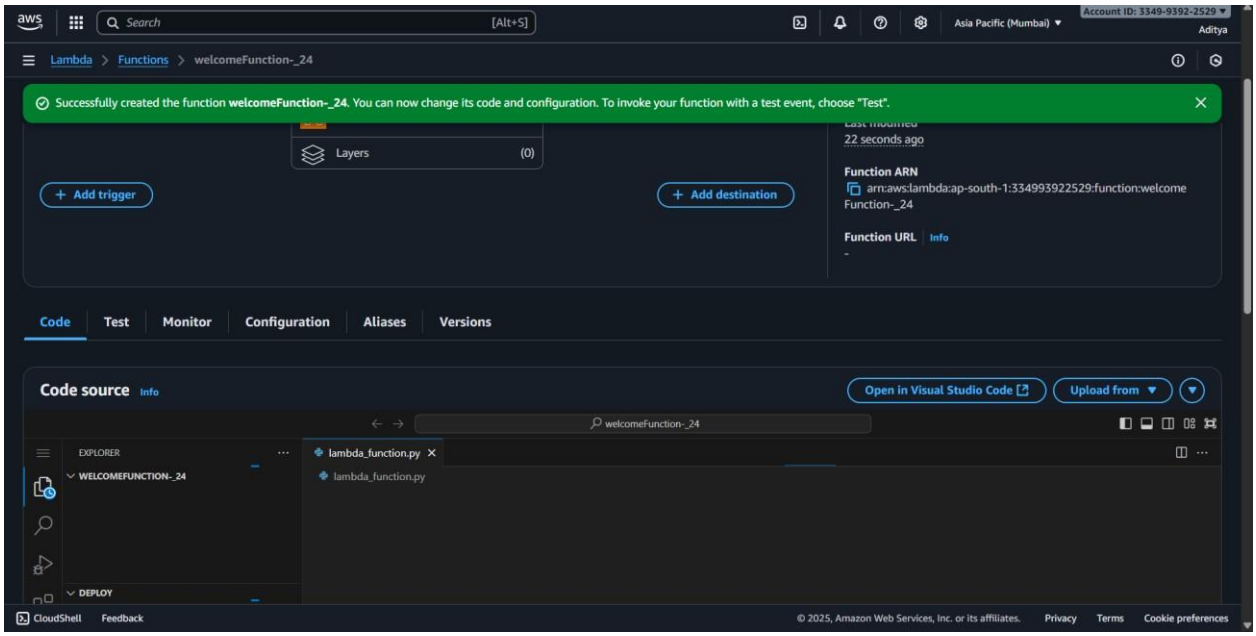
1. **Create a Lambda Function** ○ Open AWS Console → Lambda → Create function.



- Choose Author from scratch, give function name LabLambda, runtime Python

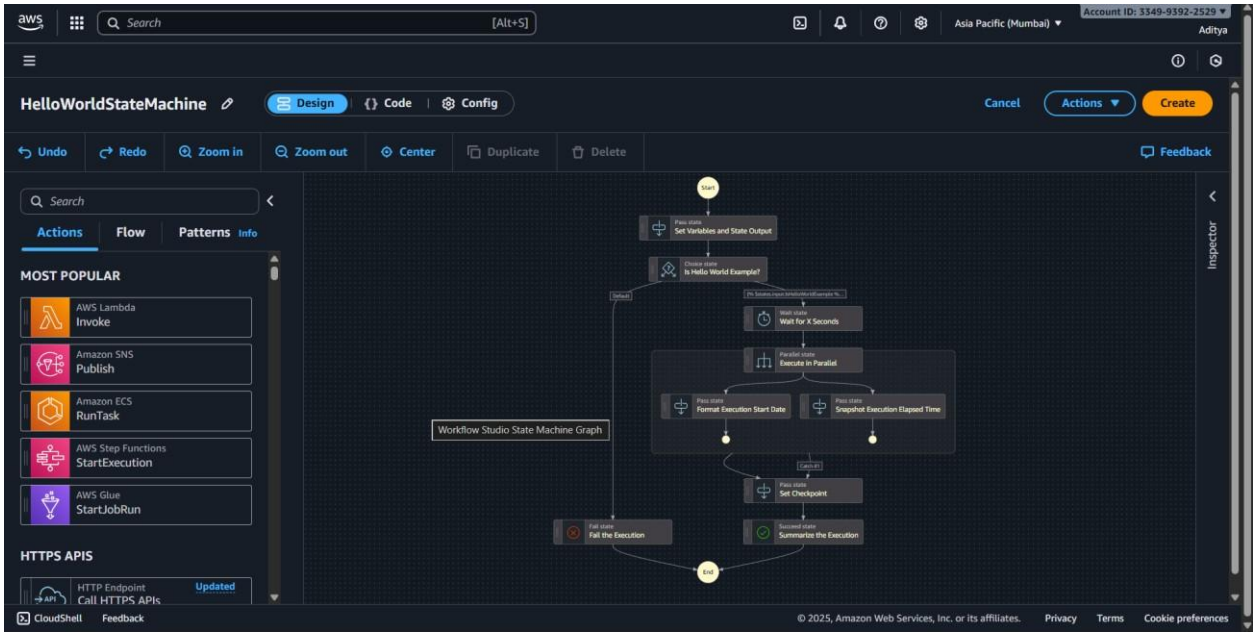
3.11.

- Use a small function that randomly succeeds or fails:
- import random ○ def lambda_handler(event, context): ○ if random.choice([True, False]):
- raise Exception("Simulated Error")
- return "Success"



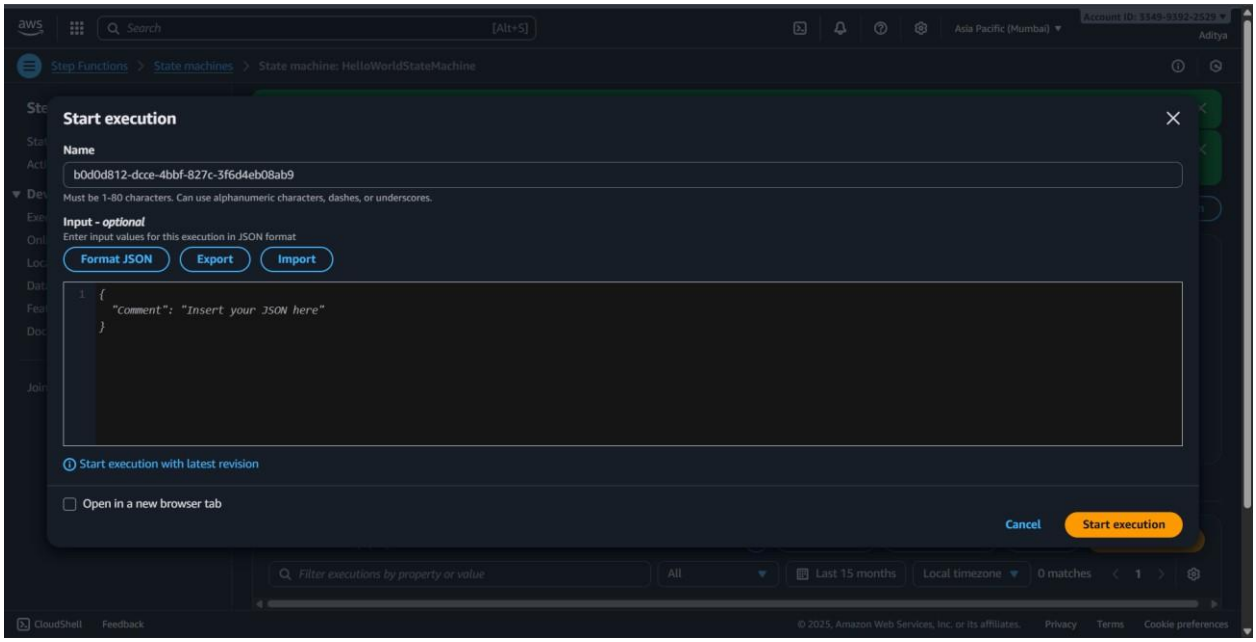
- Click Create function.

2. **Open Step Functions** ○ Go to AWS Console → Step Functions → Create state machine.



   - Choose Standard Workflow → Design your workflow visually.

3. **Add Workflow Steps** ○ Add Task state → Choose Lambda function LabLambda. ○ Add Retry: ErrorEquals = ["States.ALL"], IntervalSeconds = 2, MaxAttempts = 2, BackoffRate = 2.0. ○ Add Catch: ErrorEquals = ["States.ALL"] → Next = FailState or error handling step.
   - Add Succeed state at the end.
4. **Assign IAM Role** ○ Use/create a role that allows Step Functions to invoke Lambda.
5. **Start Execution** ○ Click Start execution → Input {} (JSON).



- Observe Execution Graph → green = success, red = error/retry.

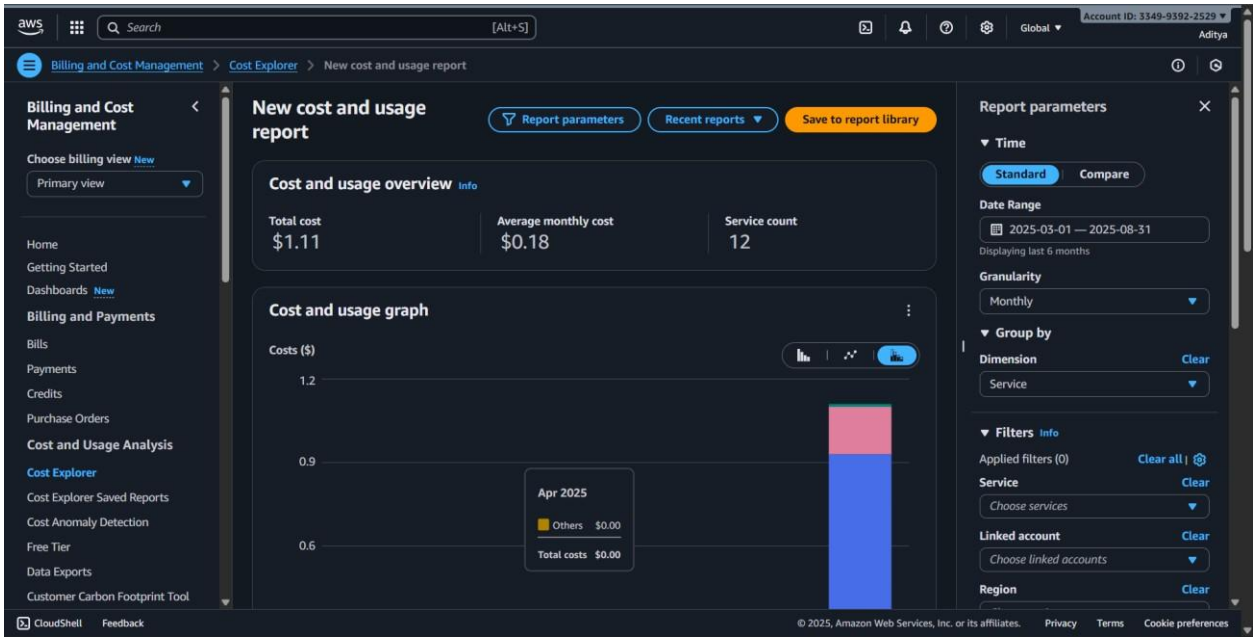
6. **Monitor Workflow** ○ Open CloudWatch → Logs → Check Lambda and Step Functions logs. ○ Optionally, create CloudWatch dashboard to track executions and success/failures.
7. **Cleanup** ○ Delete Step Function state machine. ○ Delete Lambda function. ○ Delete CloudWatch log groups to reduce cost

10.Set up comprehensive cost monitoring. Create budgets and alerts. Analyze spending patterns. Implement cost optimization strategies

1. Set Up Comprehensive Cost Monitoring

1. Login to AWS Management Console.
2. Search for Cost Explorer in the search bar.
3. Enable Cost Explorer (if not already enabled).

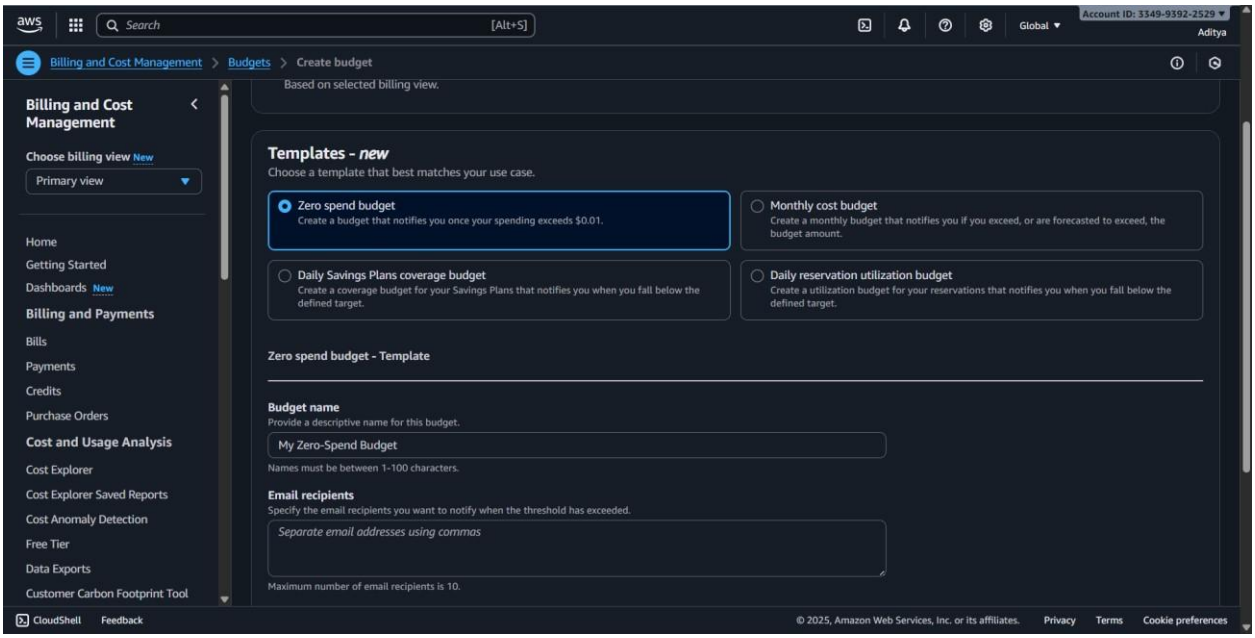
4. This gives you detailed cost and usage reports for services, regions, and linked accounts.
5. Explore views such as:
    - Monthly/Hourly Costs ○ Service-wise Cost Breakdown ○ Linked Account Usage



2. Create Budgets and Alerts

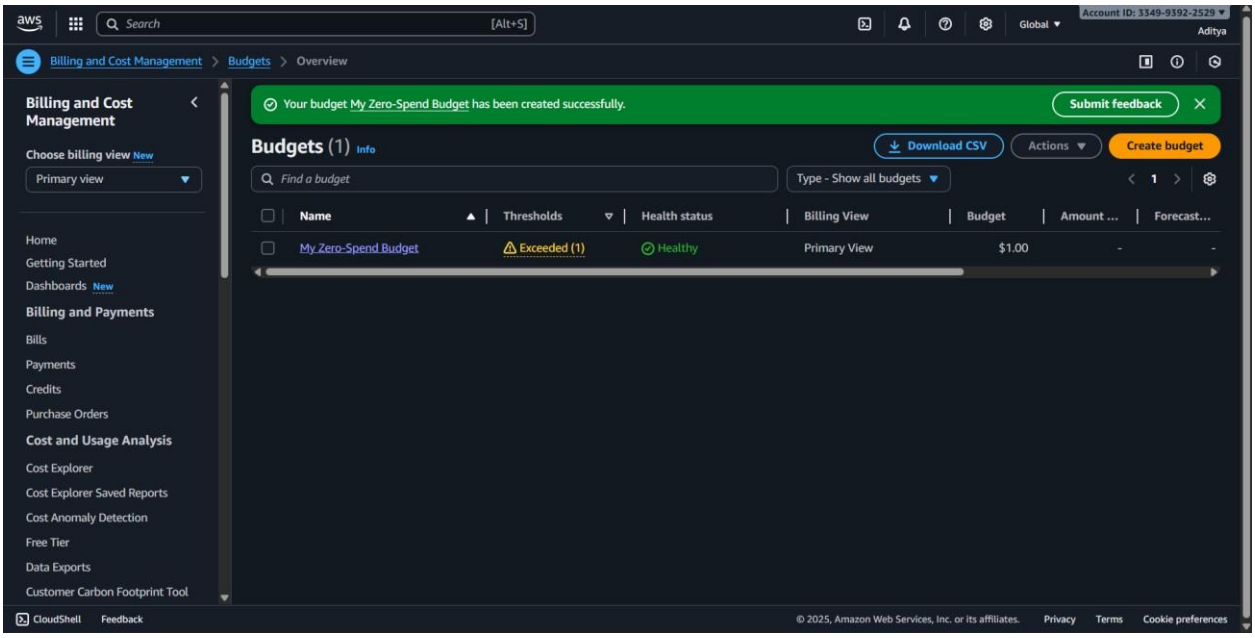
1. Go to AWS Budgets (in Billing & Cost Management).
2. Click Create Budget.

3. Select Cost budget → Next.



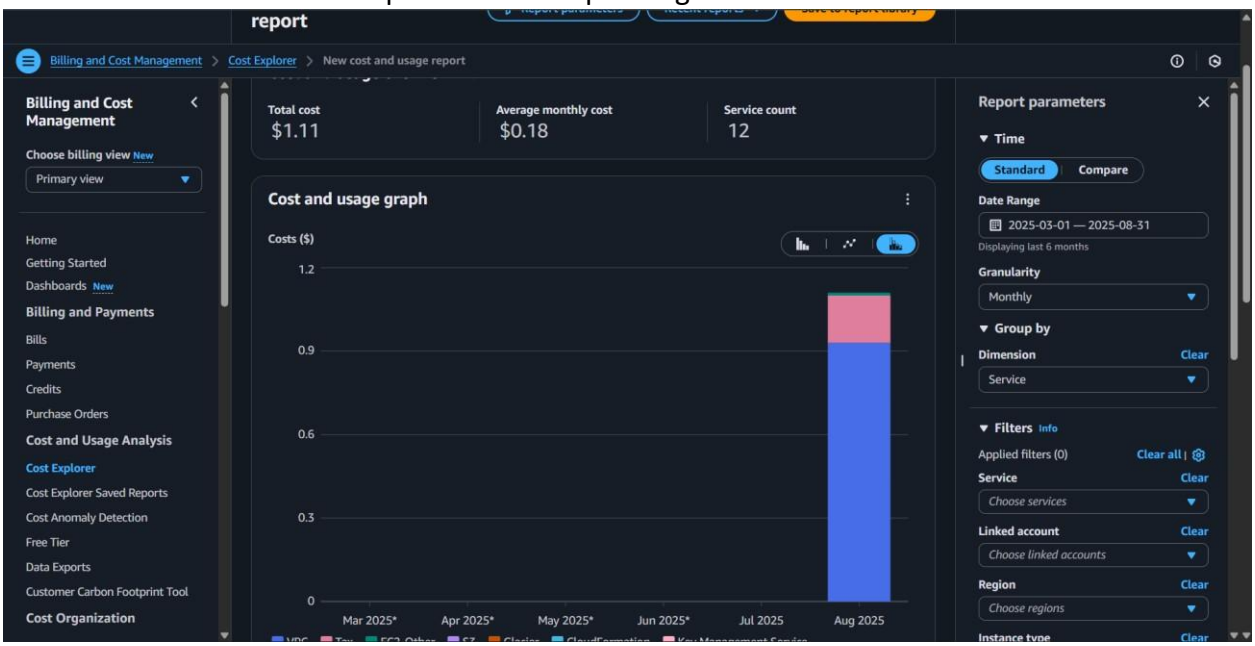
4. Set:
    - Period (Monthly/Quarterly/Yearly).
    - Budgeted amount (e.g., $10 for Free Tier experiments).
5. Configure alerts:
    - Add email address.
    - Set threshold (e.g., 80% of budget).
6. Review and click Create Budget.

Now you'll get an alert email if spending exceeds the threshold.



3. Analyze Spending Patterns

1. Go to Cost Explorer → Reports.
2. Analyze usage by:
    - Service (EC2, S3, RDS, etc.). ○ Region (e.g., US East vs Asia Pacific).
    - Usage type (On-Demand, Reserved, Data transfer).
3. Identify which services cost the most.
4. Use Trends & Forecasts to predict future spending.



4. Implement Cost Optimization Strategies

1. EC2 Optimization ○ Use right-sizing recommendations (smaller instance types). ○ Stop or terminate unused instances.
    - Use Savings Plans or Reserved Instances for long-term workloads.
2. Storage Optimization ○ Move rarely used data to S3 Glacier or Infrequent Access.
    - Delete unused snapshots and EBS volumes.
3. Networking Optimization ○ Minimize data transfer across regions.
    - Use CloudFront CDN to reduce data transfer cost.
4. Automation ○ Use AWS Instance Scheduler to stop dev/test servers at night. ○ Use Trusted Advisor for cost-saving recommendations.

5. Monitoring and Continuous Improvement

1. Set regular reviews (weekly/monthly) in Cost Explorer.
2. Adjust budgets based on usage.
3. Continuously apply cost-saving best practices.