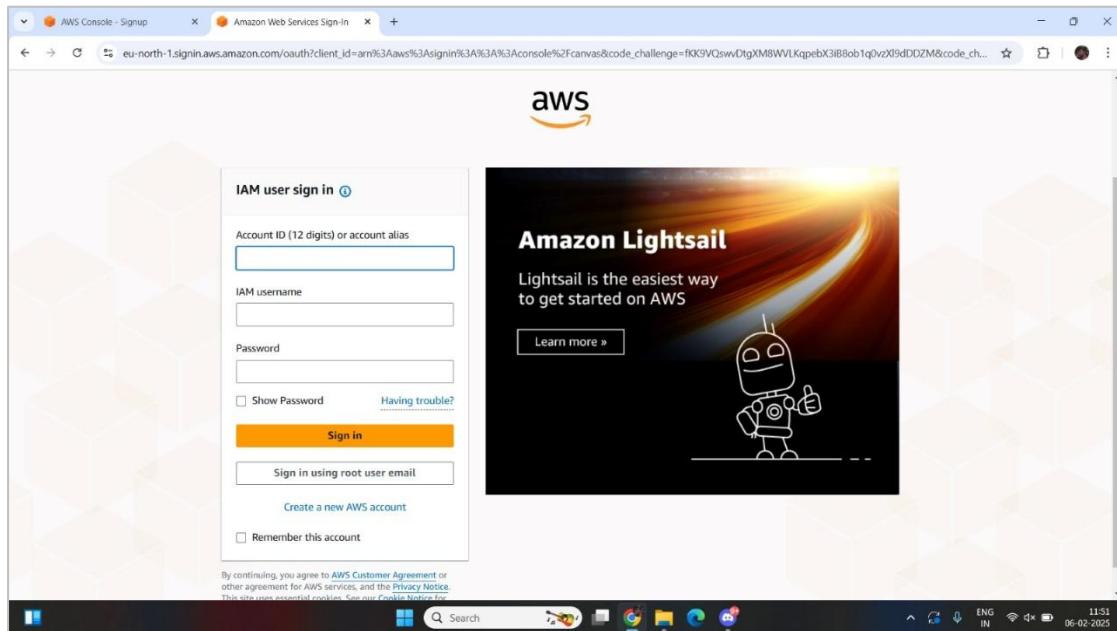


ASSIGNMENT 1:

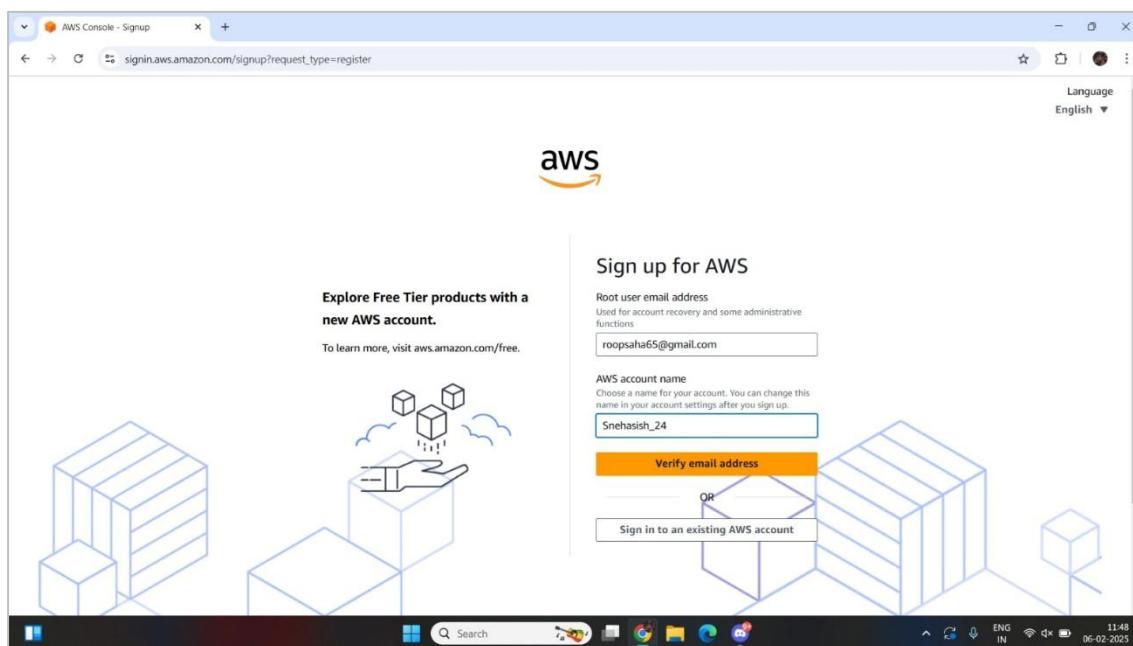
Problem Definition: Create an account in AWS and configure a budget

Account Creation:

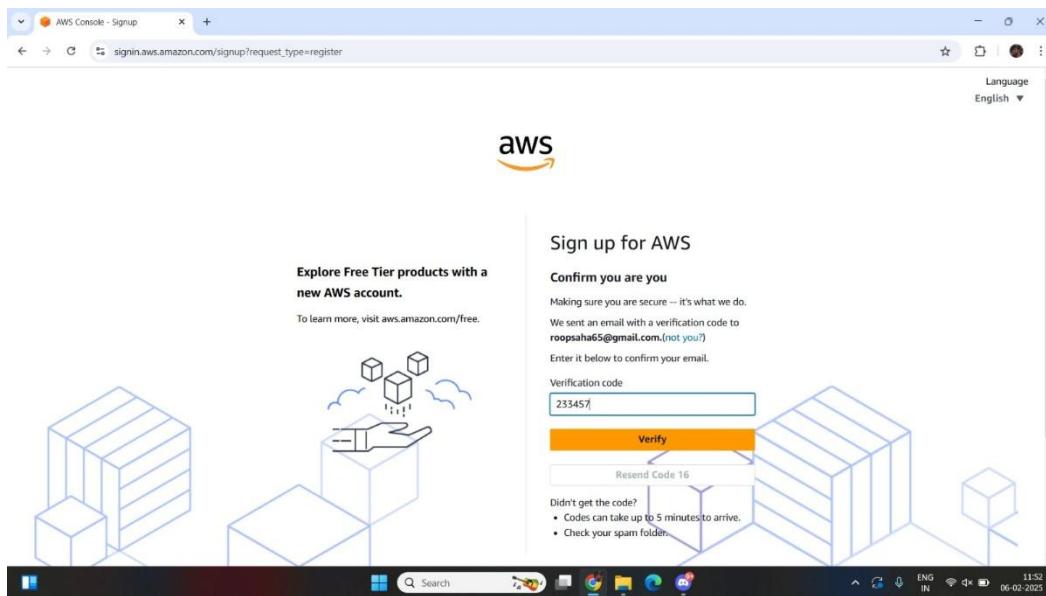
1. Go to **AWS Management Console** and click on “**Sign Up**”.



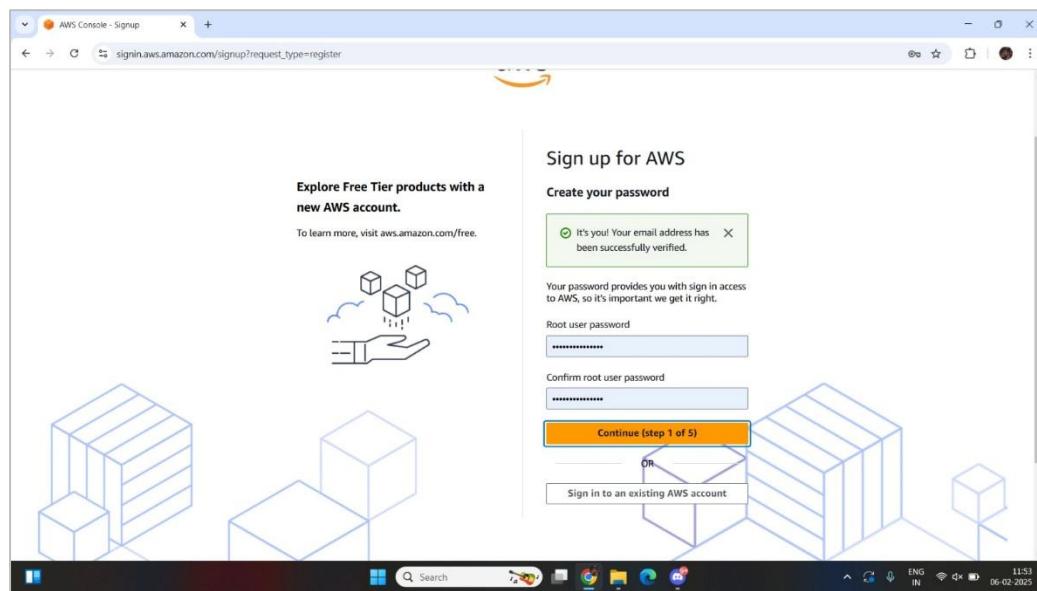
2. Enter root user email address and AWS account name.



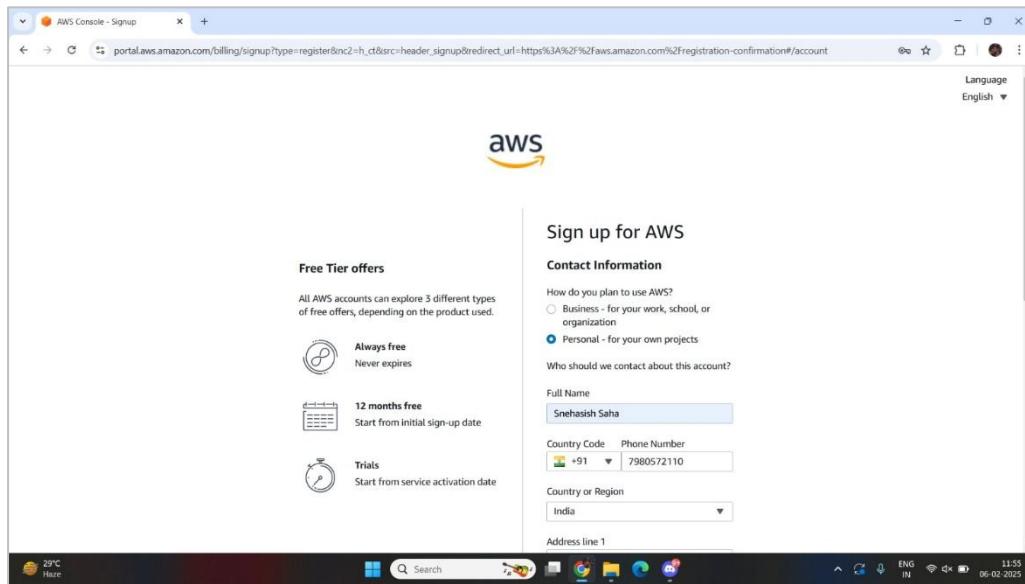
3. Verify your email address.



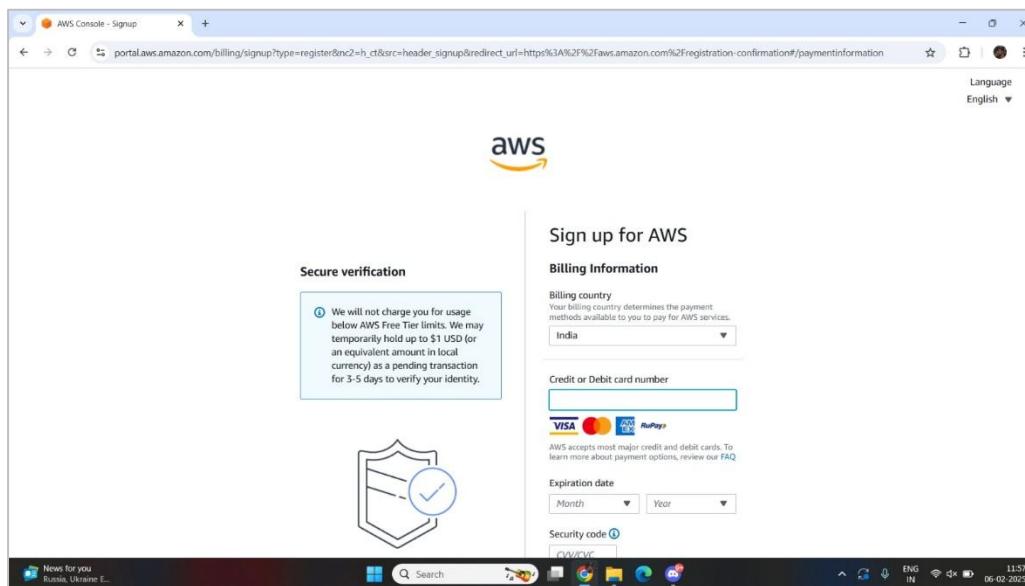
4. Set your password, then click on "**Continue (step 1 of 5)**".



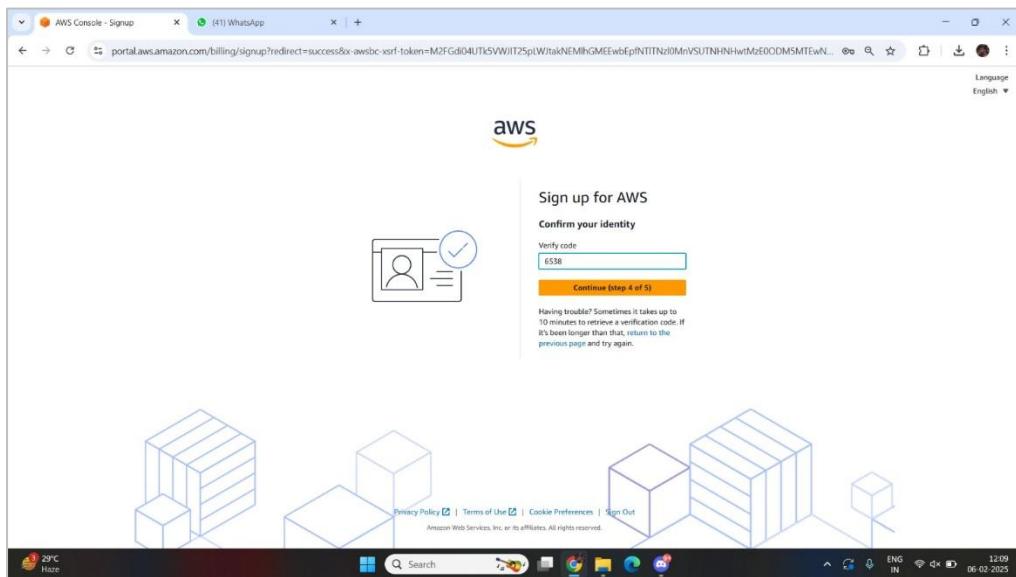
5. Fill the required details, tick off the check box and then click on “**Continue (step 2 of 5)**”.



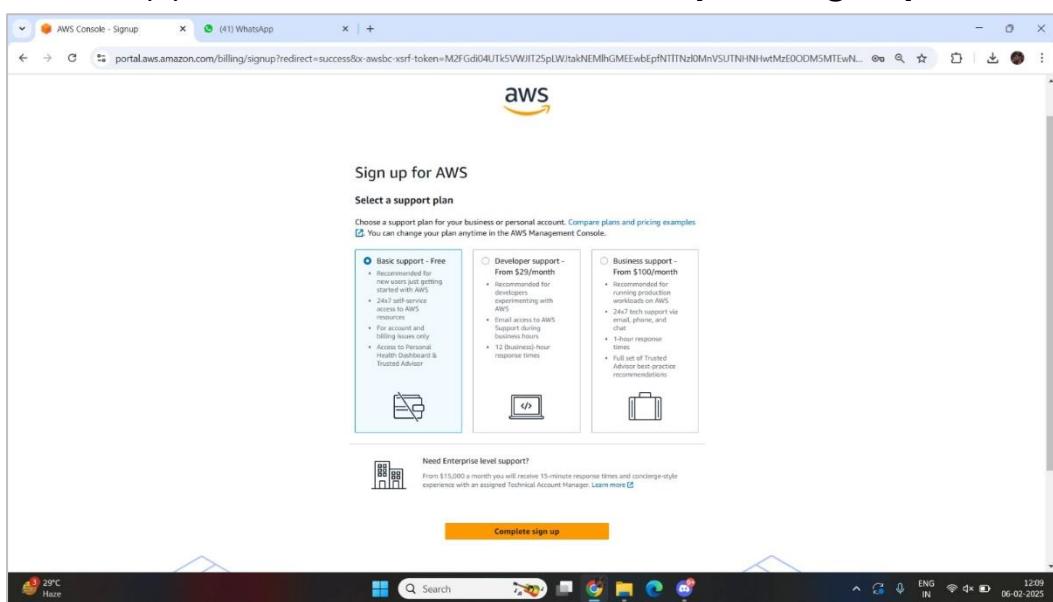
6. Enter billing information and complete payment, then click on “**Continue (step 3 of 5)**”.



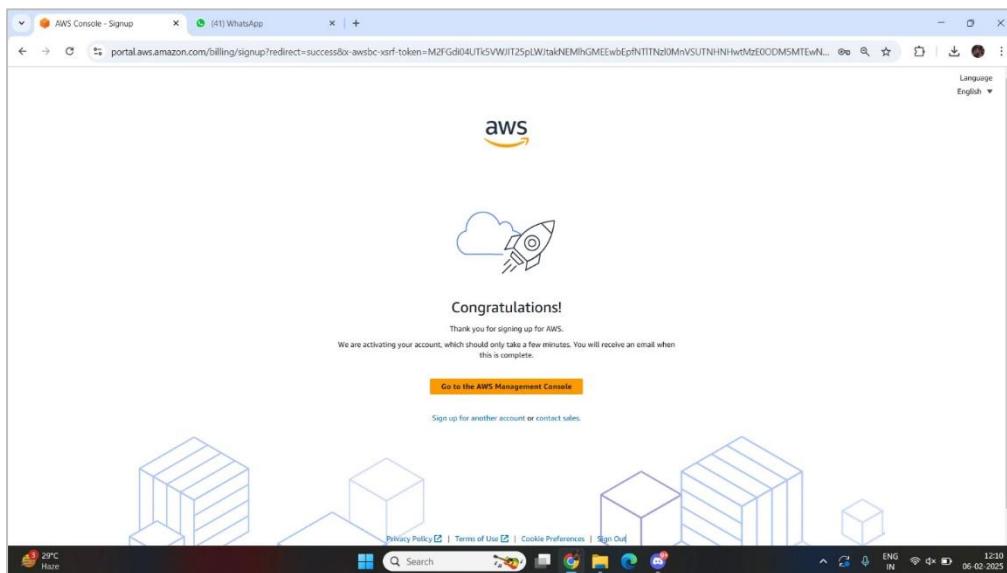
7. Verify your phone number then click “**Continue (step 4 of 5)**”.



8. Select a Support Plan then click on “**Complete sign up**”.

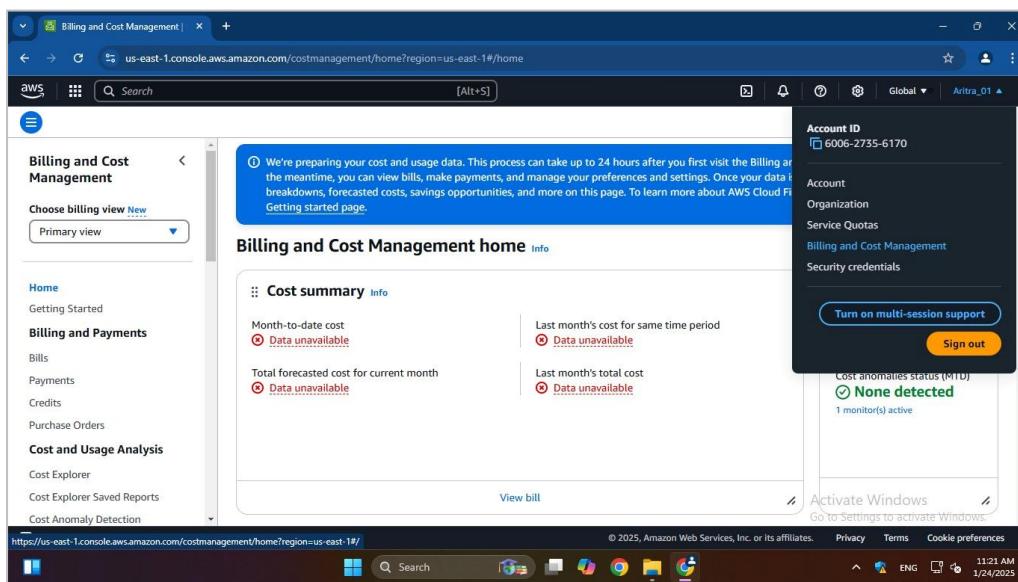


9. Congratulation page will be shown if account is created successfully.



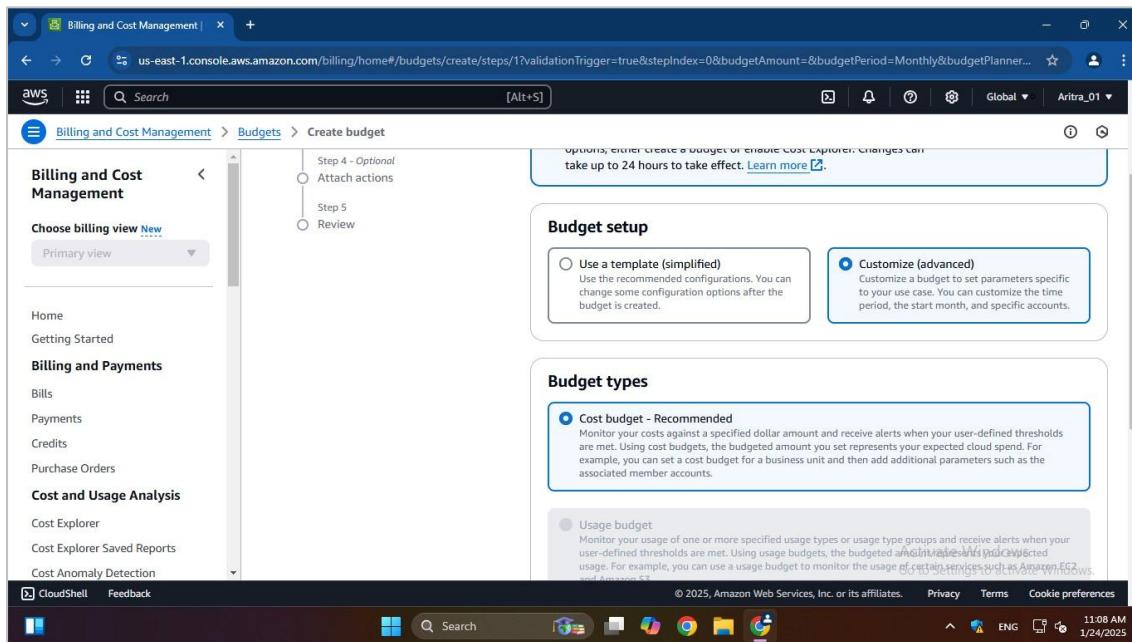
Budget Configuration:

1. Login to **AWS Management Console**, click on your username at the top right corner and click on "**Billing and Cost Management**".

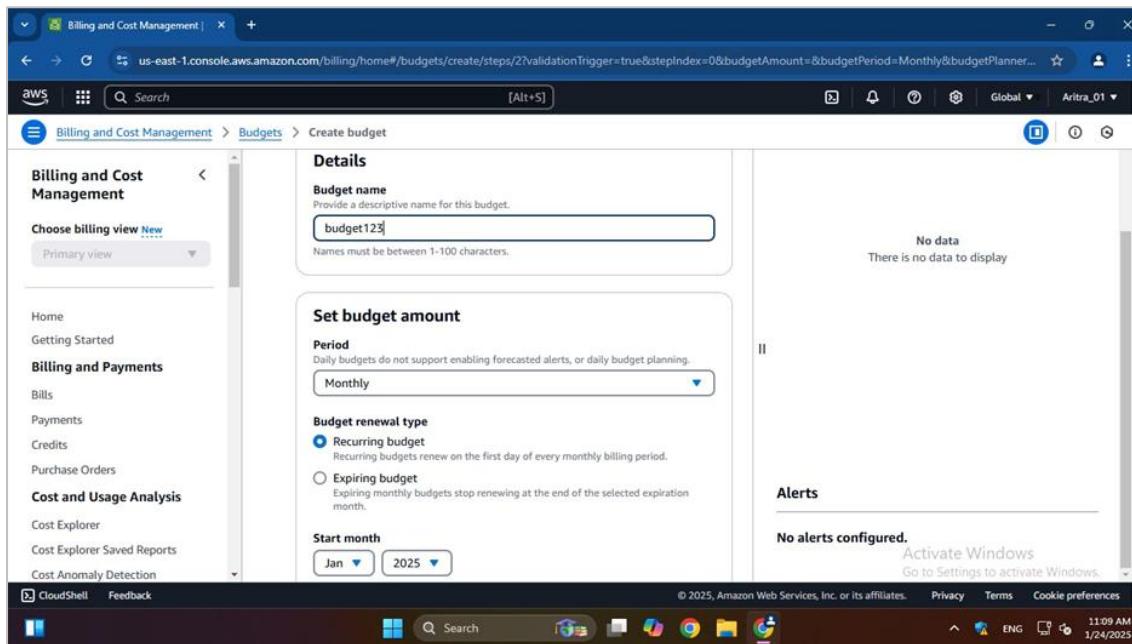


2. Under “**Budgets and Planning**”, click on “**Budgets**” and then click on “**Click on budget**”.

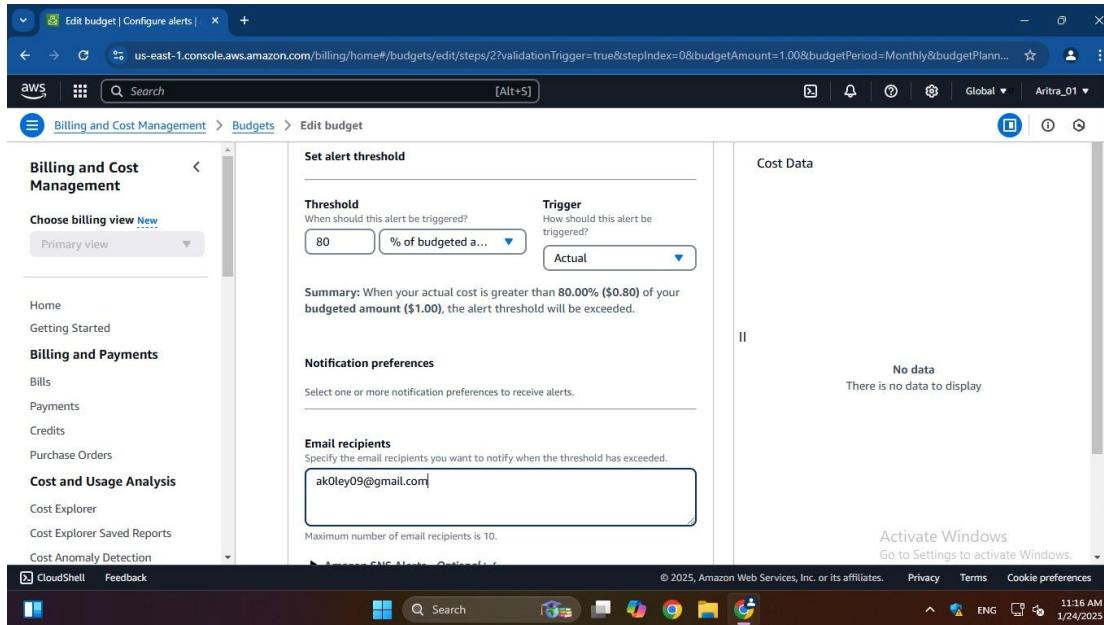
3. Configure budget setup by selecting “**Customize (advanced)**” then click on “**Next**”.



4. Add budget details.

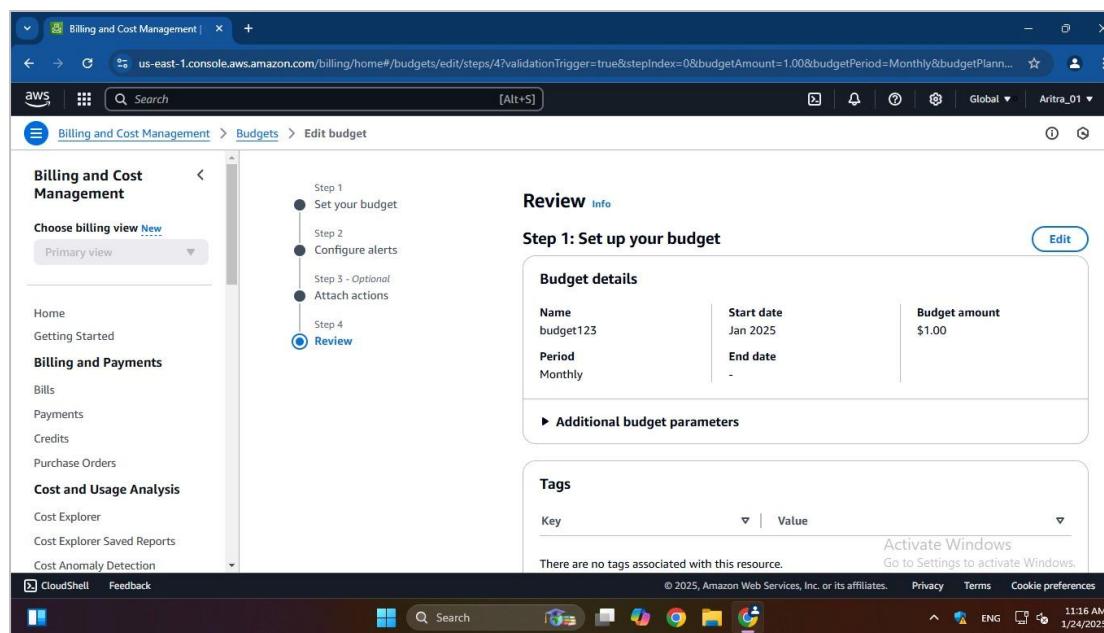


5. After selecting “**Add threshold**”, fill-in the necessary details and then click on “**Next**”.



The screenshot shows the AWS Billing and Cost Management console. On the left, there's a sidebar with navigation links like Home, Getting Started, Billing and Payments, and Cost and Usage Analysis. The main area is titled "Edit budget". Step 2 is selected, specifically "Set alert threshold". A summary states: "When your actual cost is greater than 80.00% (\$0.80) of your budgeted amount (\$1.00), the alert threshold will be exceeded." Below this, there's a "Notification preferences" section which is currently empty. To the right, there's a "Cost Data" panel that says "No data" and "There is no data to display". At the bottom, there are standard browser controls and a status bar indicating the date and time.

6. Review the budget details and click on “**Save**” to finalize the budget.



The screenshot shows the AWS Billing and Cost Management console at step 4 of the budget creation process. The sidebar and main navigation are similar to the previous screenshot. The main area shows a flowchart with four steps: Step 1 (Set your budget), Step 2 (Configure alerts), Step 3 - Optional (Attach actions), and Step 4 (Review). Step 4 is currently active. The "Review" section contains "Budget details" with fields for Name (budget123), Start date (Jan 2025), End date (blank), and Budget amount (\$1.00). There's also a "Tags" section which is currently empty. The status bar at the bottom indicates the date and time.

7. Budget is successfully created.

The screenshot shows the AWS Billing and Cost Management console. The left sidebar includes sections for Home, Getting Started, Billing and Payments (Bills, Payments, Credits, Purchase Orders), and Cost and Usage Analysis (Cost Explorer, Cost Explorer Saved Reports, Cost Anomaly Detection). The main content area displays a success message: "Your budget budget123 has been updated successfully." Below this, a table lists one budget entry:

Name	Thresholds	Budget	Amount used	Forecasted ...	Current vs. bu...
budget123	OK	\$1.00	\$0.00		

At the bottom right, there are links for "Activate Windows", "CloudShell", "Feedback", and system status indicators (ENG, 11:17 AM, 1/24/2025).

ASSIGNMENT 2:

Problem Definition: Create MFA (Multi-Factor Authentication) for authentication.

Instructions:

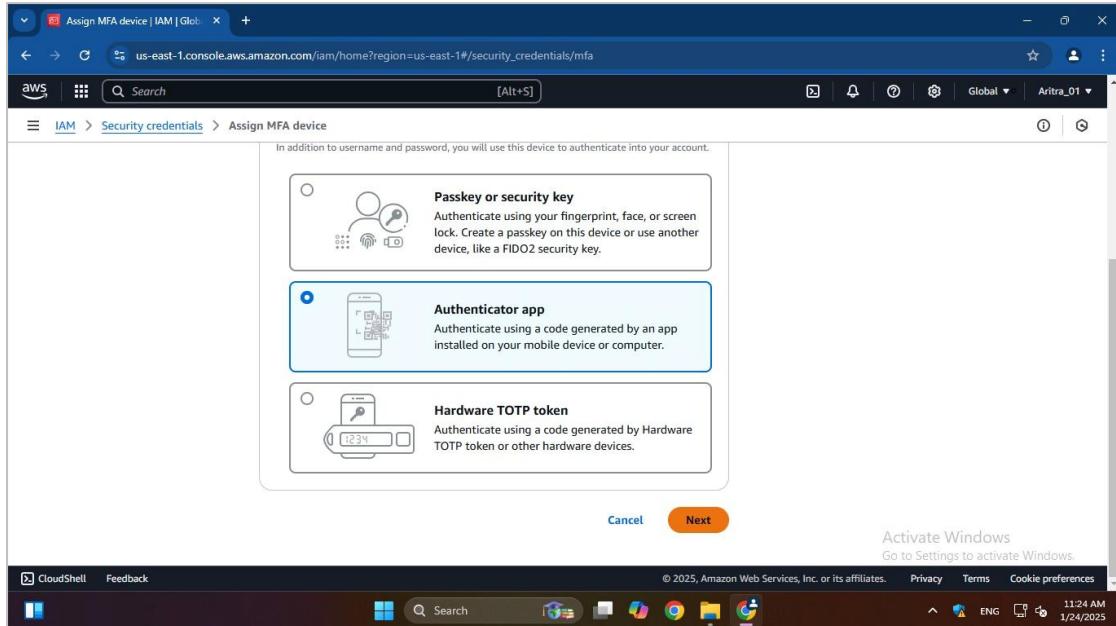
1. Log into AWS account and go to “**Security credentials**”.

The screenshot shows the AWS IAM Security Credentials page. In the 'Multi-factor authentication (MFA)' section, there is a button labeled 'Assign MFA device' which is highlighted with a yellow box. Other buttons like 'Remove' and 'Resync' are also visible.

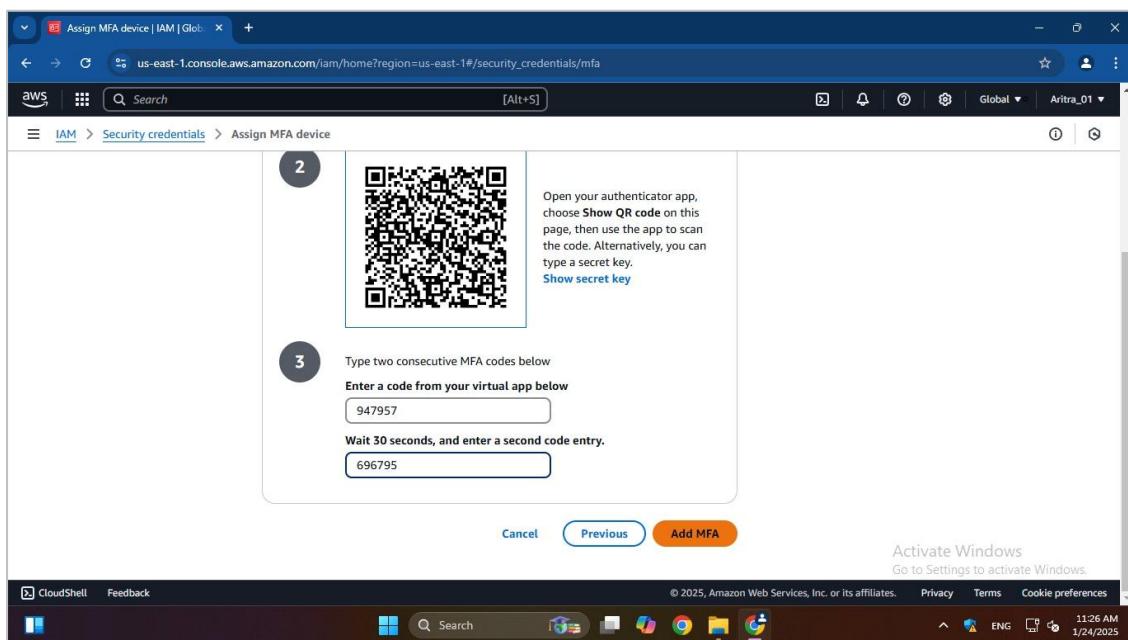
2. Add MFA device by clicking on “**Assign MFA device**”.

The screenshot shows the same AWS IAM Security Credentials page after clicking the 'Assign MFA device' button. The button is now highlighted with a yellow box. The other buttons ('Remove', 'Resync') are also visible.

3. Enter a desired device name, MFA device and then click on “**Next**”.



4. Enter code using authenticator app and click on “**Add MFA**” to complete the process.



5. MFA device assigned successfully.

The screenshot shows the AWS IAM Security Credentials page. A green success message at the top states: "MFA device assigned. You can register up to 8 MFA devices of any combination of the currently supported MFA types with your AWS account root and IAM user. With multiple MFA devices, you only need one MFA device to sign in to the AWS console or create a session through the AWS CLI with that user." Below this, the "Multi-factor authentication (MFA) (1)" section displays a single entry:

Type	Identifier	Certifications	Created on
Virtual	arn:aws:iam::600627356170:mfa/Aritra@123	Not Applicable	Fri Jan 24 2025

Below this, the "Access keys (0)" section indicates "No access keys".

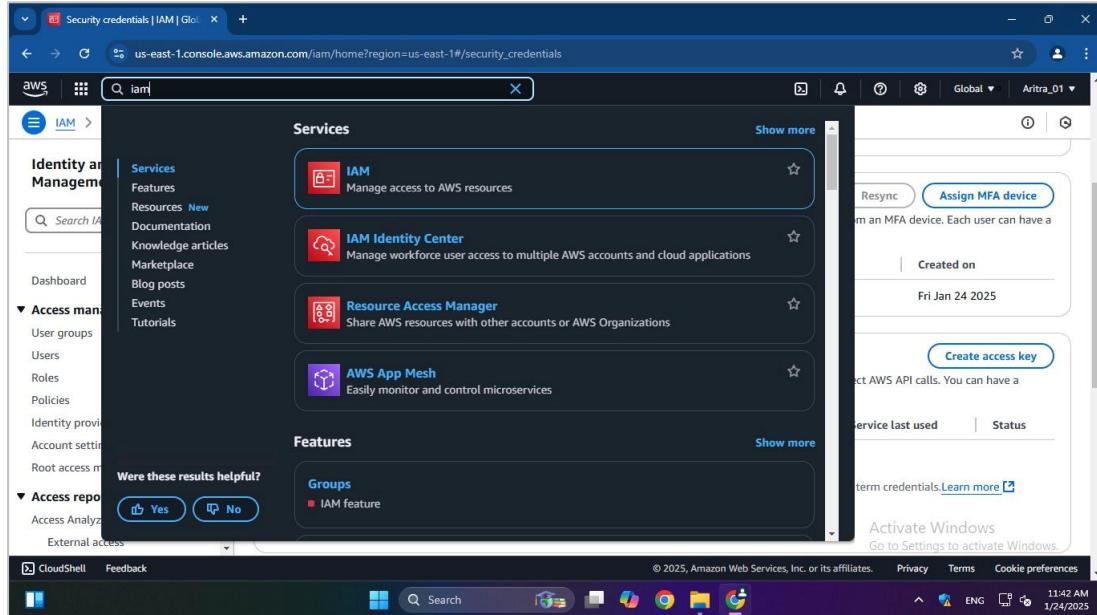
The left sidebar shows navigation options under "Identity and Access Management (IAM)".

ASSIGNMENT 3:

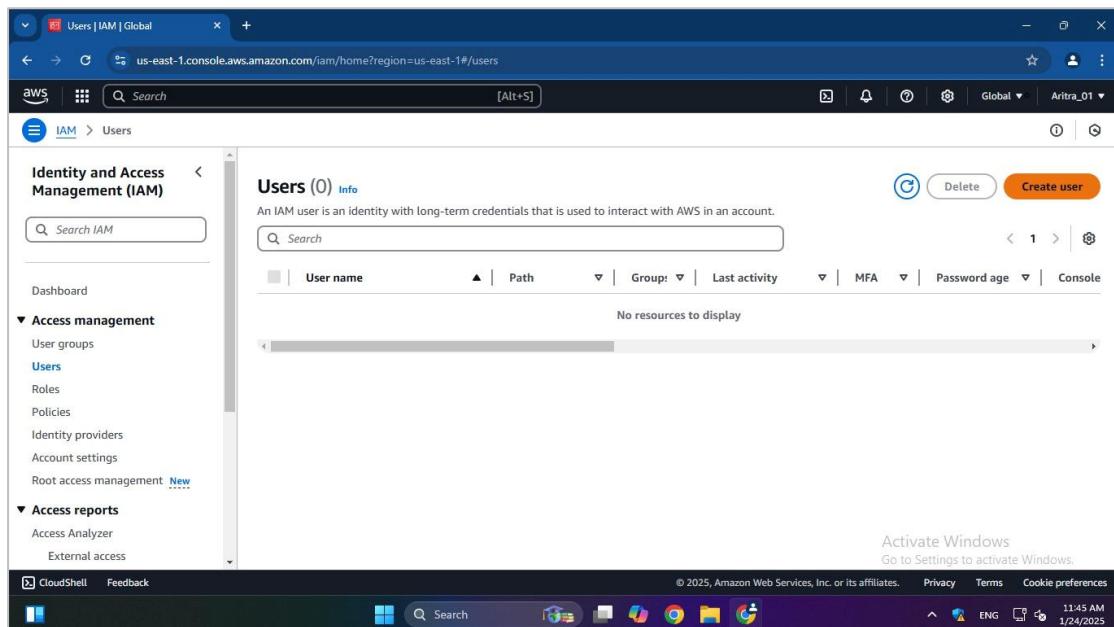
Problem Definition: Create IAM user and give full access to S3.

Instructions:

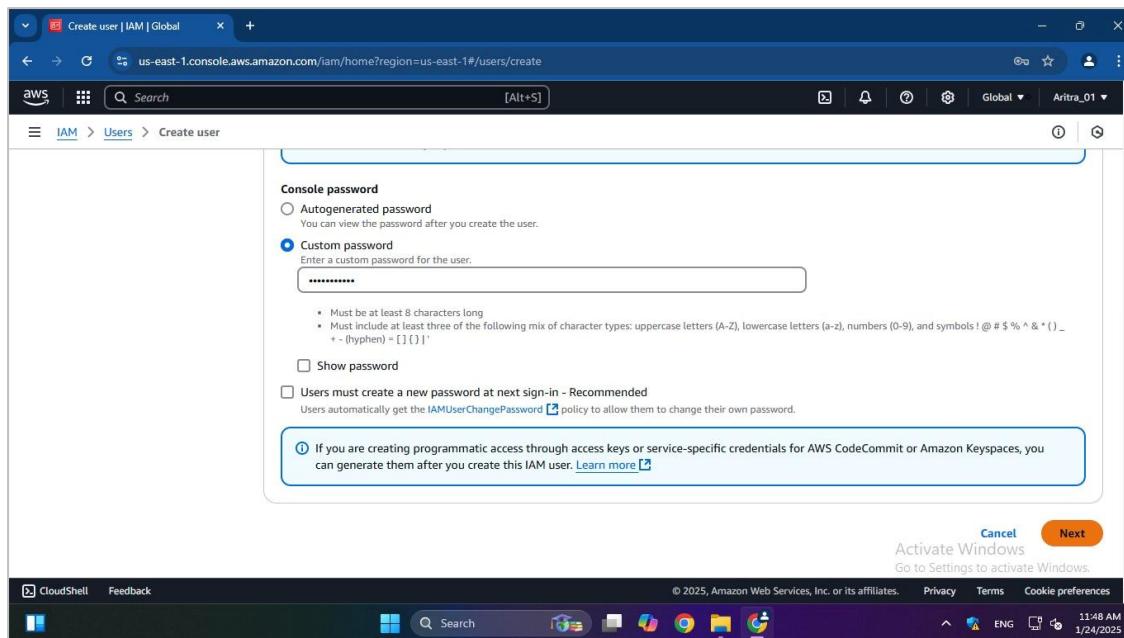
1. From AWS console, search for “**IAM**” and click on it.



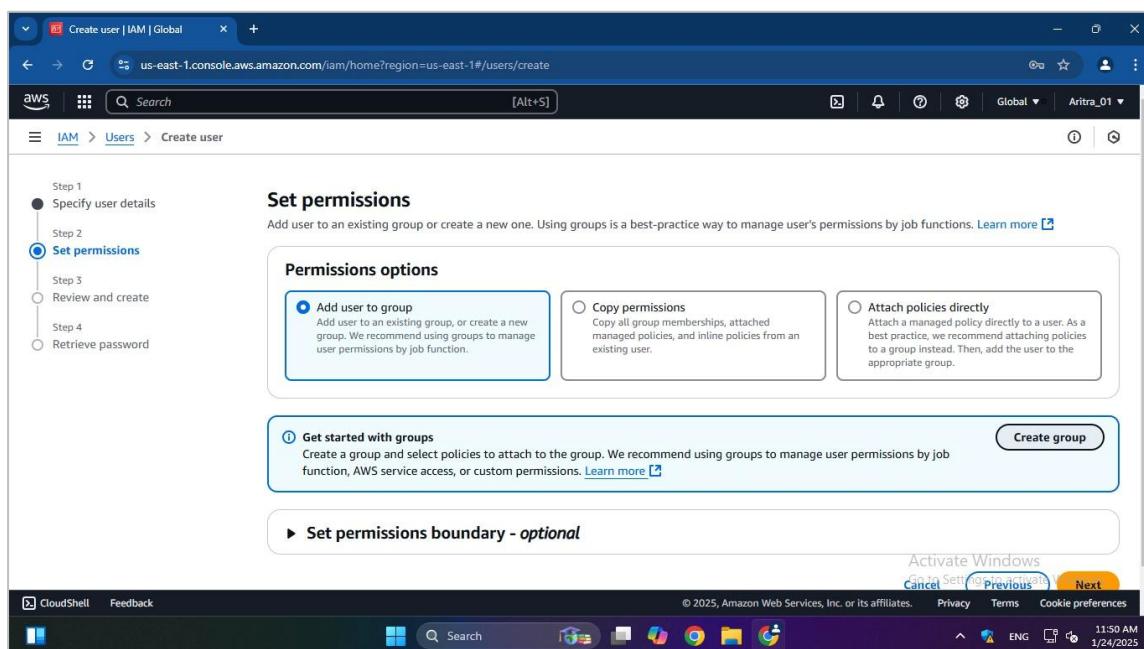
2. Under “**IAM resources**” click on “**Users**”.



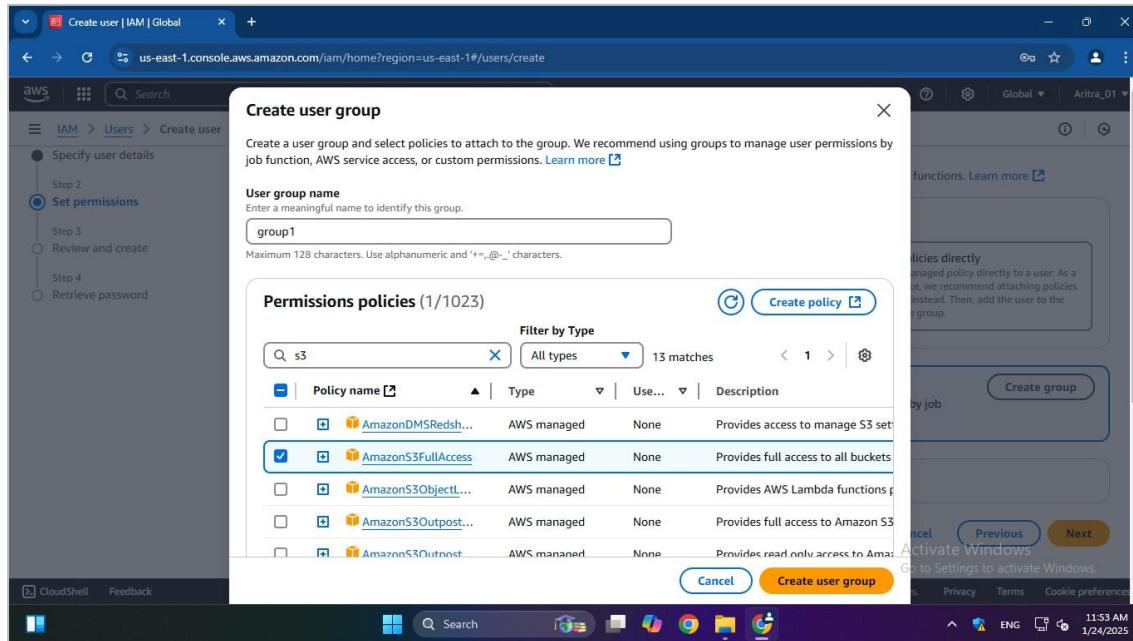
3. Click on “**Create User**”. Provide User details (name, password, etc.) and click on “**Next**” to proceed.



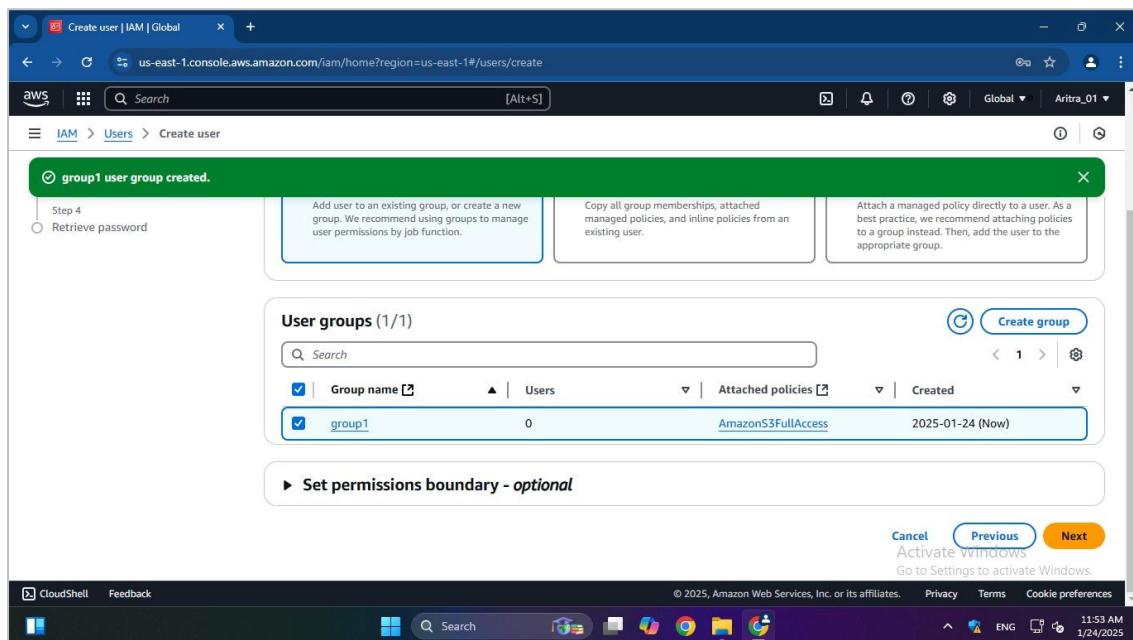
4. Under “**Permissions options**”, click on “**Add user to group**” and go to “**Create group**”.



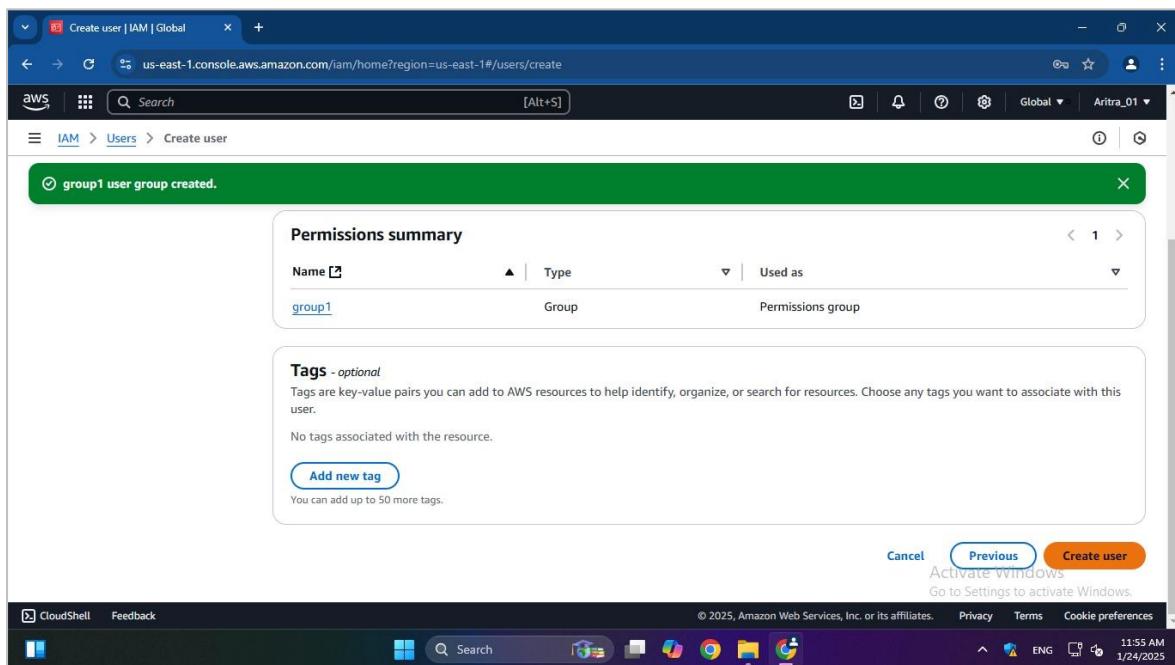
5. To create user group, first provide “**User group name**”, then from “**Permission policies**” search for “**S3**” and select full access of the particular permission we want to give to that group. Then click on “**Create User Group**”.



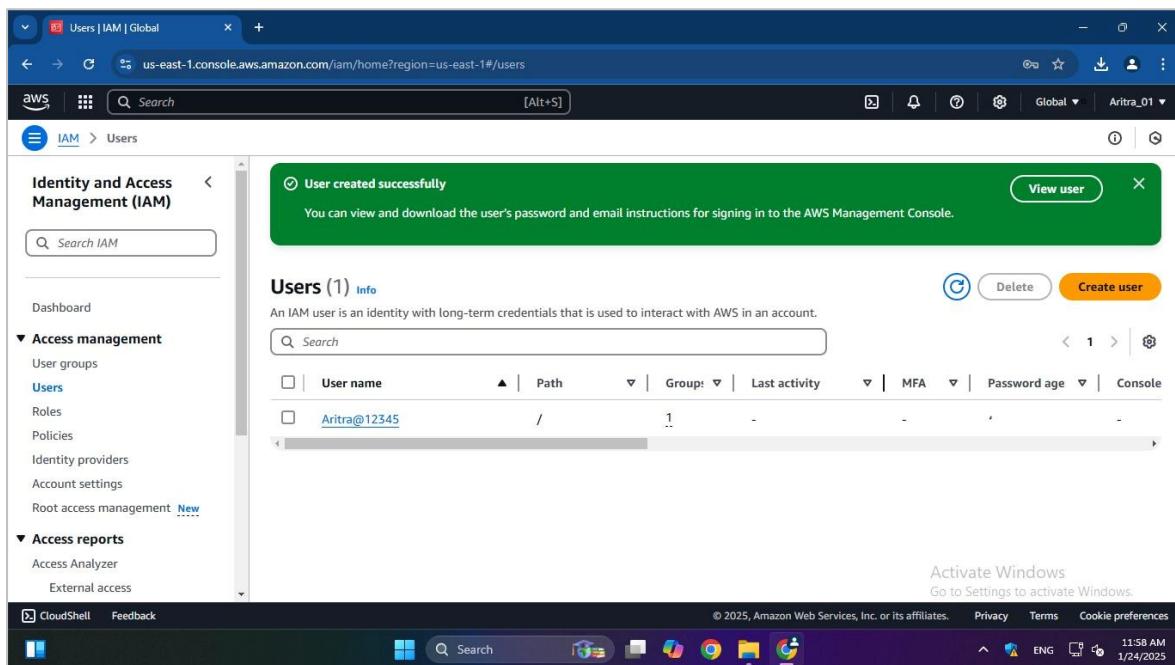
6. The group is successfully created. Tick the checkbox of the created group in “**User groups**”, so that our IAM user is connected to this group and gets the required permission. Then click on “**Next**”.



7.Under “**Review and create**”, click on “**Create user**” to create our IAM user.



8. IAM user is created successfully.

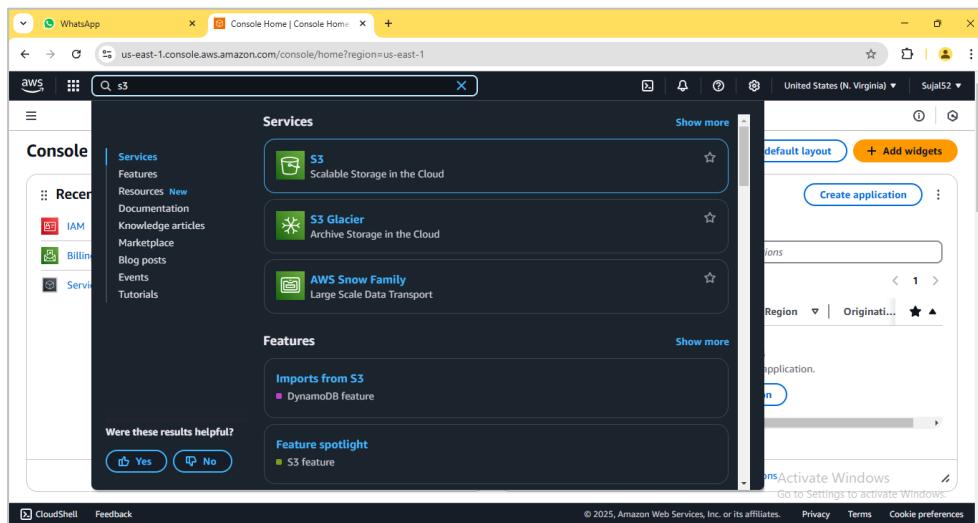


ASSIGNMENT 4:

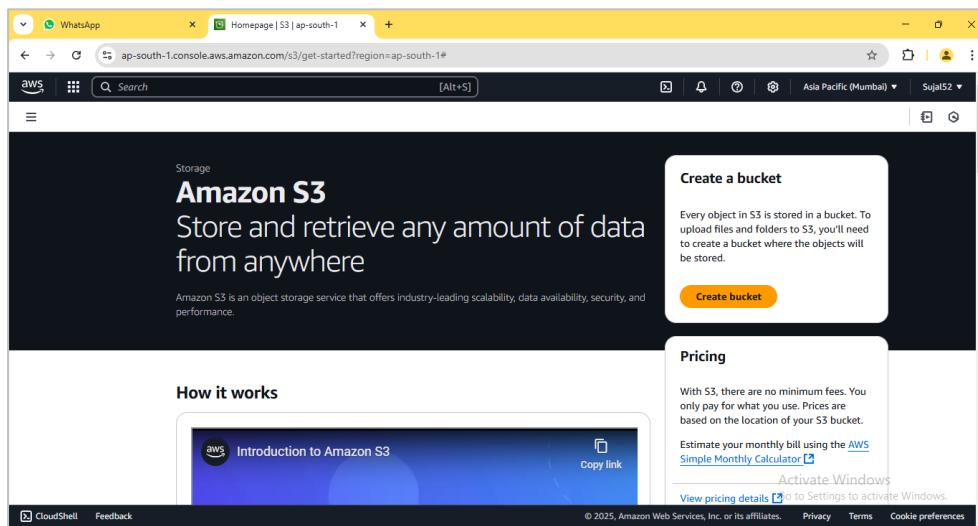
Problem Definition: Create a private bucket in AWS. Upload a file and check by reassigned URL whether you can access the file or not

Instructions:

1. Access the **AWS console**, search for **S3**, and select the top option from the search results.



2. Click on “**Create bucket**”.



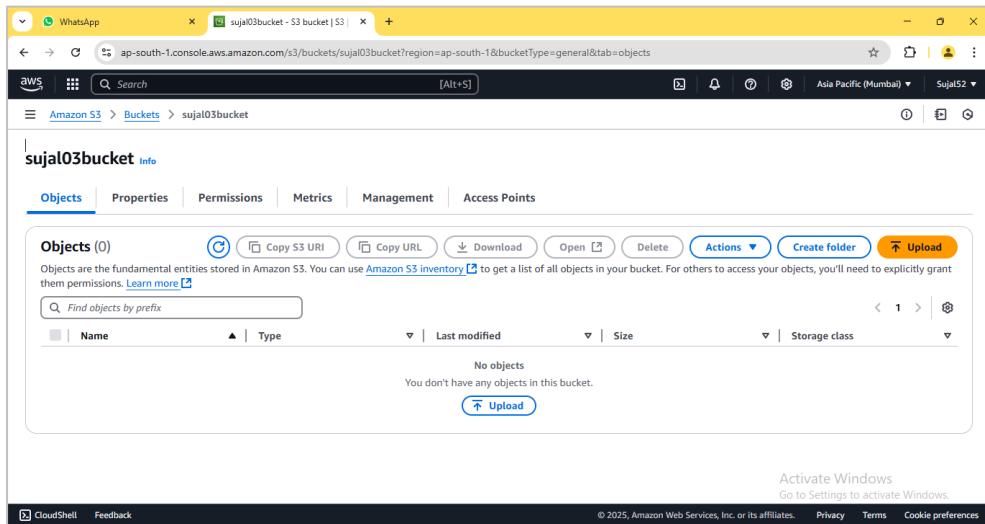
3. Enter bucket name. Leave the other options set to default as we are creating a private bucket. Then click on “**Create bucket**”.

The screenshots show the 'Create S3 bucket' wizard. In the first step, 'General configuration', the 'Bucket name' is set to 'sujal03bucket'. In the second step, 'Advanced settings', the 'Encryption type' is set to 'Server-side encryption with Amazon S3 managed keys (SSE-S3)' and 'Bucket Key' is set to 'Enable'. Both steps include a note about activating Windows and a 'Create bucket' button at the bottom right.

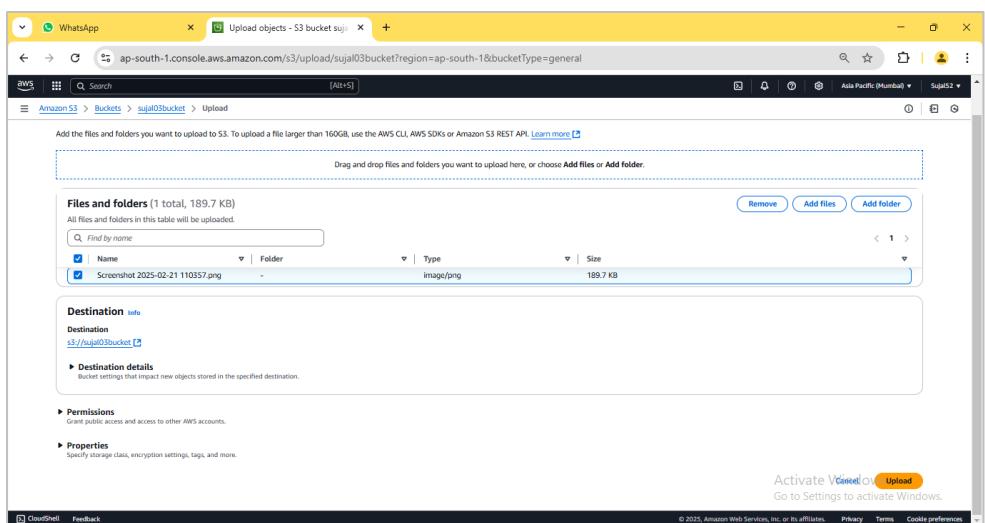
4. A green coloured pop-up dialogue box shall confirm the creation of our bucket. It shall also be displayed on the bucket list.

The screenshot shows the 'Buckets' page in the AWS S3 console. A green success message at the top states 'Successfully created bucket "sujal03bucket"'. Below it, the bucket 'sujal03bucket' is listed under 'General purpose buckets'. The bucket details show it was created on February 21, 2025, at 10:59:18 (UTC+05:30).

5. Click on the name of our newly created bucket. On the bucket detail page, click on “**Upload**”.



6. Click on “**Add files**” then select the file we want to upload to our bucket. Select our file from the list and click on “**Upload**” button.



7. The notification confirms that the file has been successfully uploaded. Click on “**Close**” to return to our bucket details page.

Upload succeeded
For more information, see the [Files and folders table](#).

Summary

Destination	Succeeded	Failed
s3://sujal03bucket	1 file, 189.7 KB (100.00%)	0 files, 0 B (0%)

Files and folders [Configuration](#)

Files and folders (1 total, 189.7 KB)

Name	Folder	Type	Size	Status
Screenshot 2025-02-21 110357.png	-	image/png	189.7 KB	Succeeded

8. Click on our previously uploaded file from the list. Copy the “**Object URL**” from the file details page that appears.

sujal03bucket [Info](#)

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
Screenshot 2025-02-21 110357.png	png	February 21, 2025, 11:05:19 (UTC+05:30)	189.7 KB	Standard

Screenshot 2025-02-21 110357.png [Info](#)

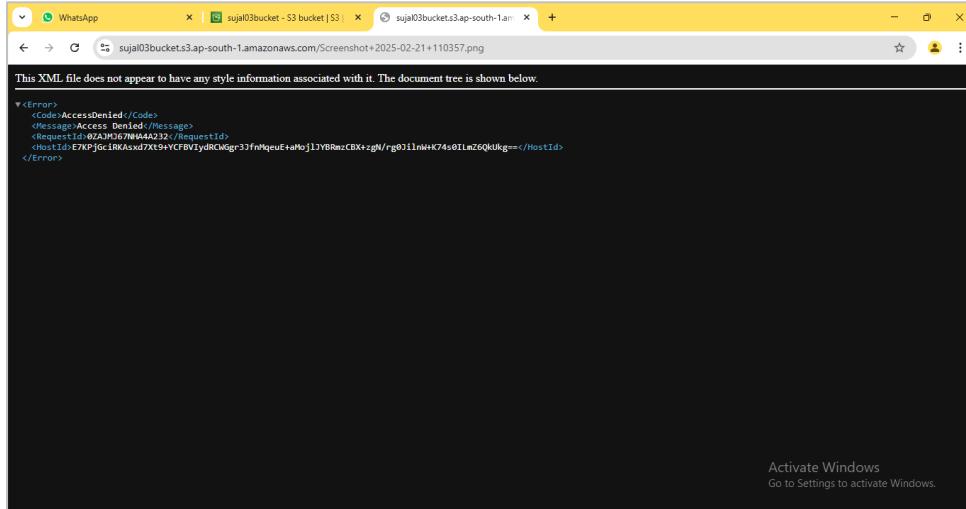
[Copy S3 URI](#) [Download](#) [Open](#) [Actions](#) [Create folder](#) [Upload](#)

Properties [Permissions](#) [Versions](#)

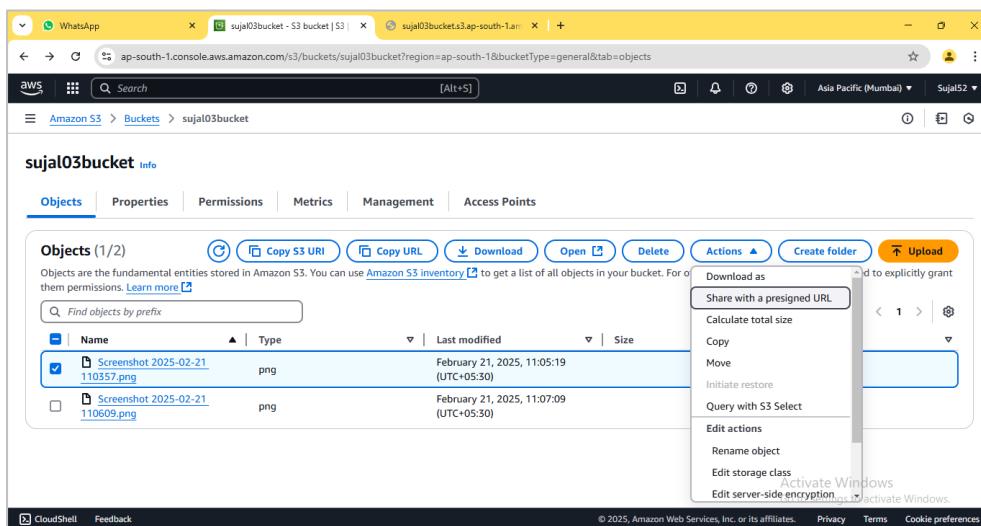
Object overview

Owner 87cf713c3bb9b38c14915ccabb85070c617e031db5ea4e48decc68fb69b0f00c	S3 URI https://s3.ap-south-1.amazonaws.com/sujal03bucket/Screenshot%202025-02-21%20110357.png
AWS Region Asia Pacific (Mumbai) ap-south-1	Amazon Resource Name (ARN) arn:aws:s3:::sujal03bucket/Screenshot%202025-02-21%20110357.png
Last modified February 21, 2025, 11:05:19 (UTC+05:30)	Entity tag (Etag) c75b38dd5d6cb5025146f35250e98306
Size 189.7 KB	Object URL https://s3.ap-south-1.amazonaws.com/sujal03bucket/Screenshot%202025-02-21%20110357.png
Type png	
Key	

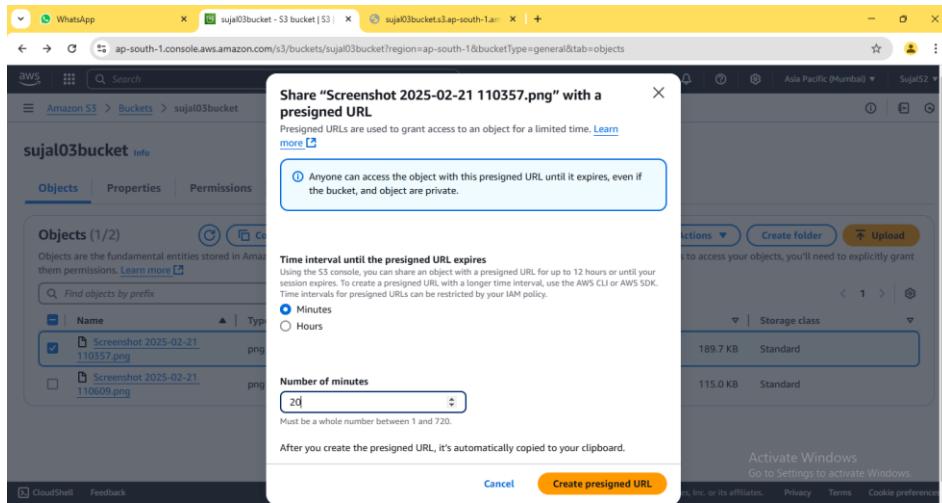
9. Open the copied URL in a new browser window. An error appears indicating that access to the contents is restricted since this is a private bucket.



10. Go back to bucket details page, select our file from the list, click on “**Actions**” and select “**Share with a presigned URL**” from the dropdown.



11. Select the desired duration until you want the presigned URL to expire then click on “**Create presigned URL**”.



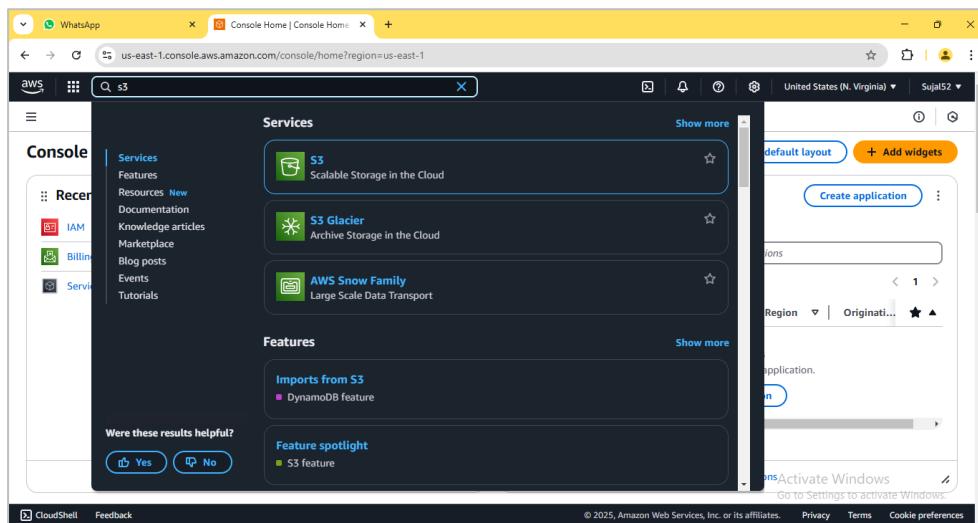
12. Copy the pre-assigned URL and paste it on a new window. Now, you can view the uploaded file.

ASSIGNMENT 5:

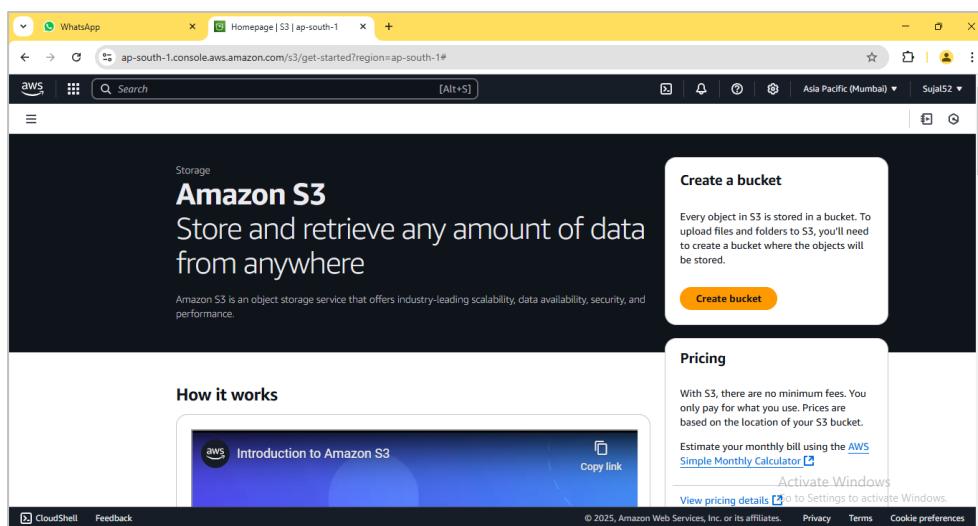
Problem Definition: Create a public Bucket in AWS. Upload a file and give the necessary permission to check whether file the URL is working.

Instructions:

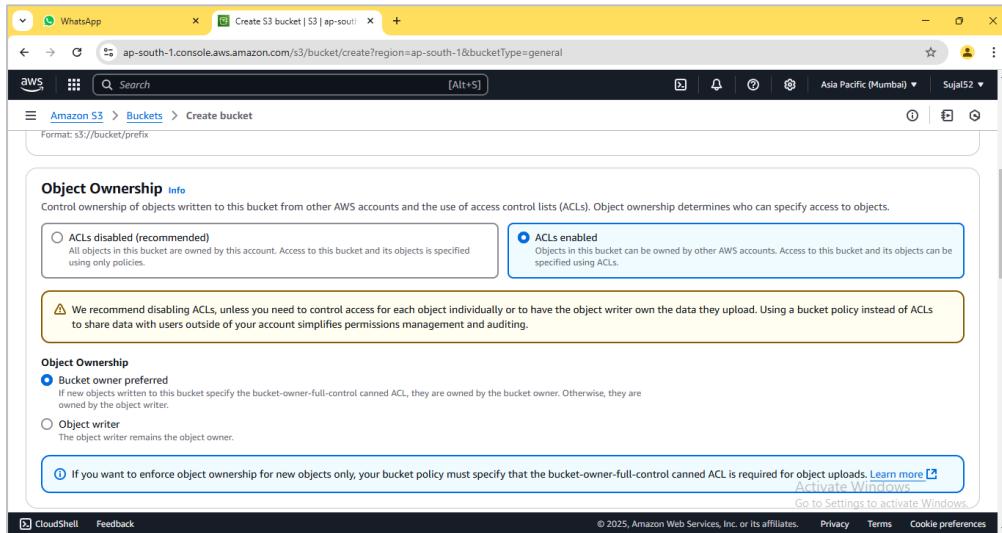
1. Access the **AWS console**, search for **S3**, and select the top option from the search results.



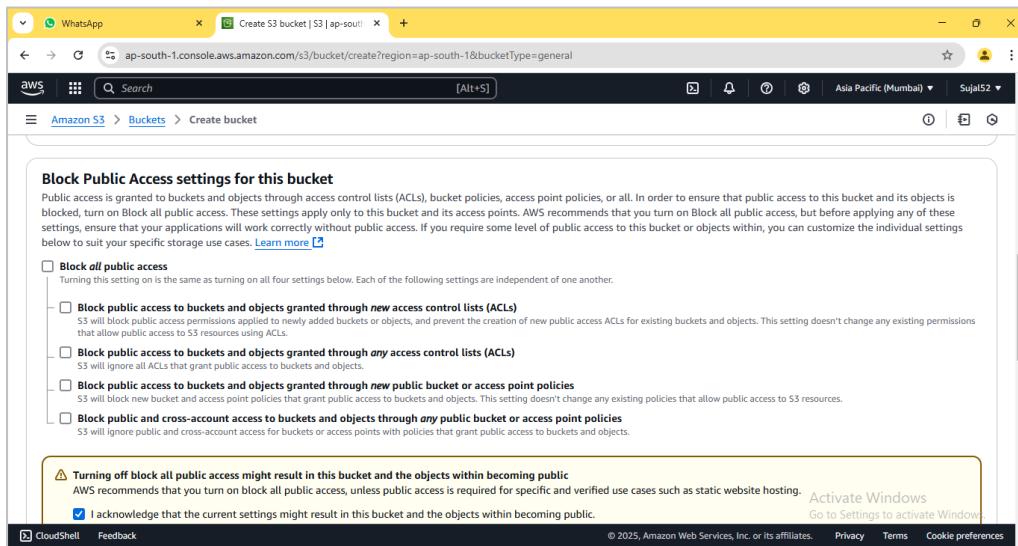
2. Click on “**Create bucket**”.



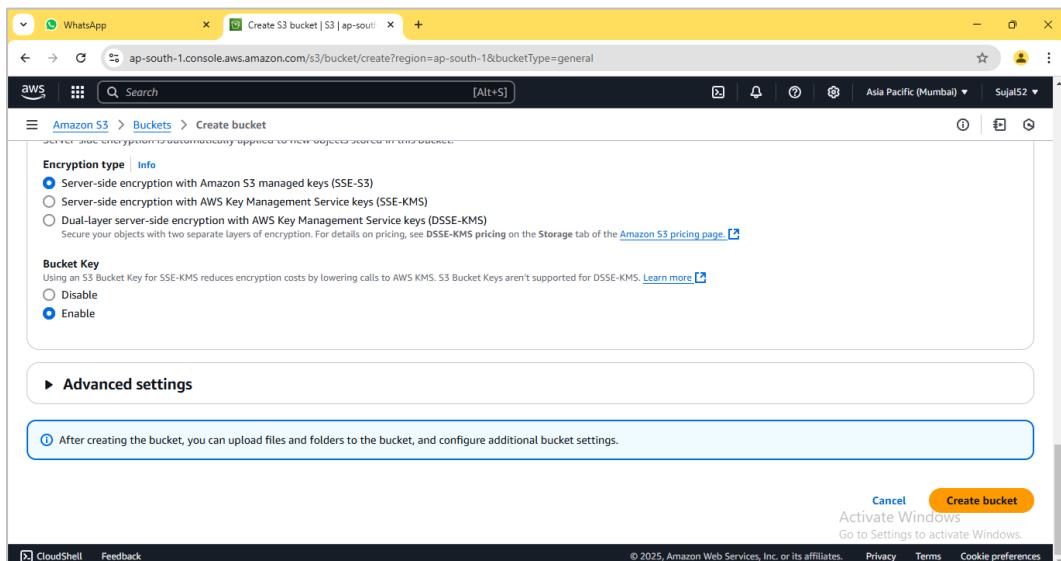
3. Enter bucket name. Select “**ACLs enabled**” since we are creating a public bucket.



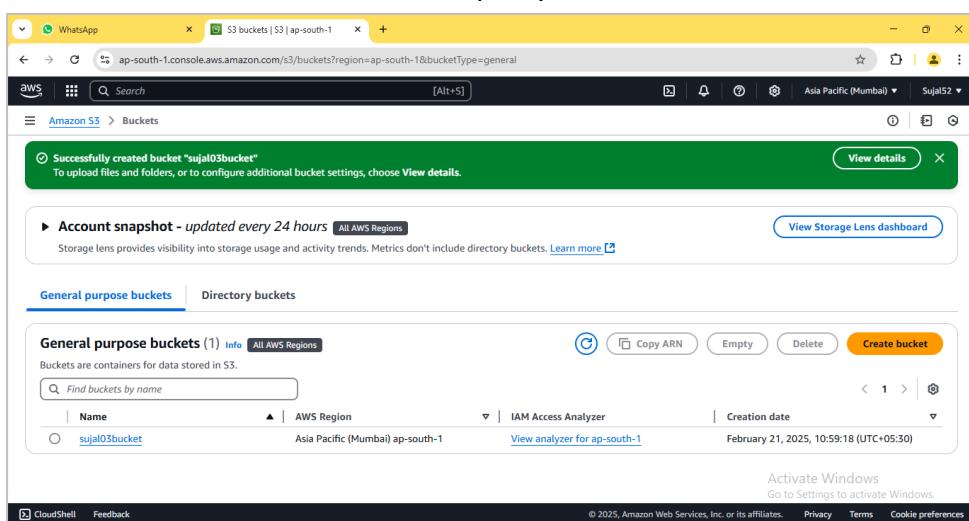
4. Now, uncheck the box for “**Block all public access**”. At the bottom of the window, check the box confirming that the current setting changes may result in a public bucket.



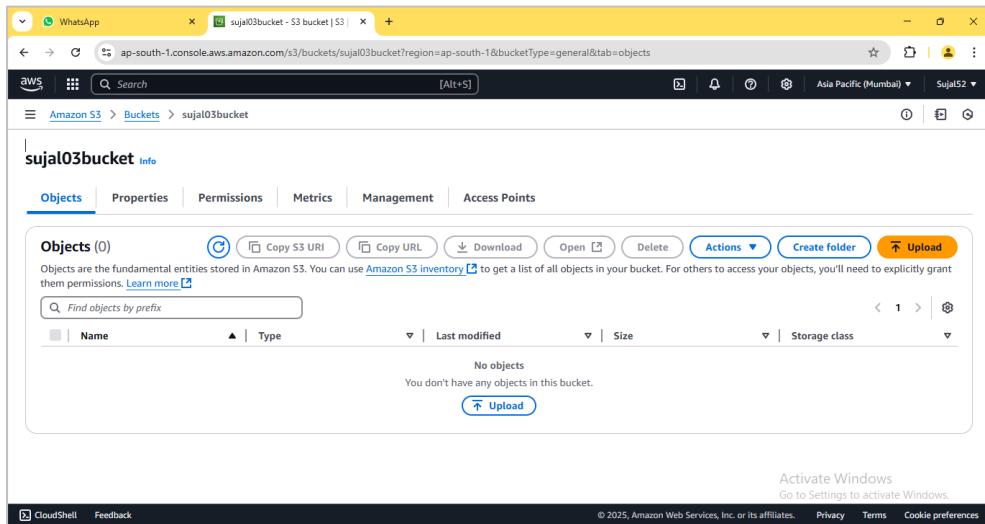
5. Scroll down without any further modifications and proceed to click on “**Create Bucket**”.



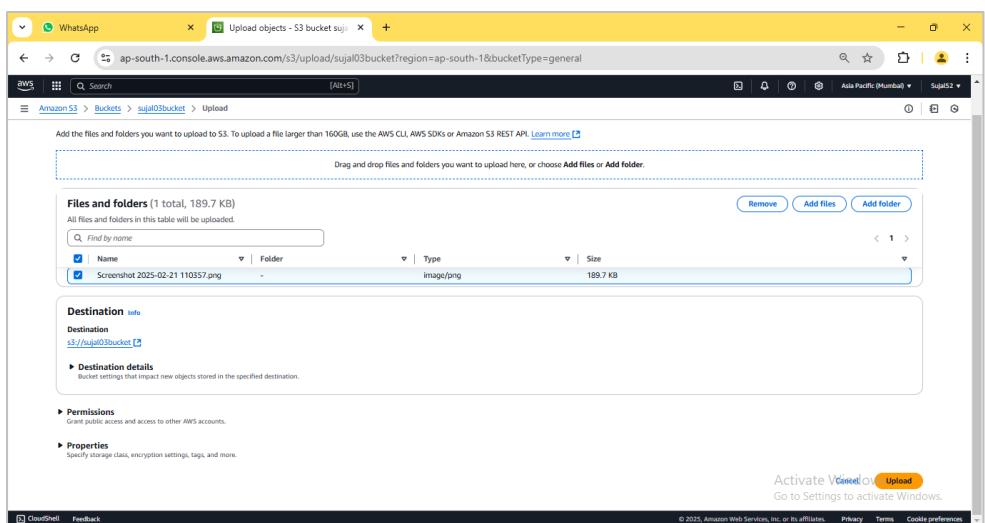
6. A green coloured pop-up dialogue box shall confirm the creation of our bucket. It shall also be displayed on the bucket list.



7. Click on the name of our newly created bucket. On the bucket detail page, click on “**Upload**”.



8. Click on “**Add files**” then select the file we want to upload to our bucket. Select our file from the list and click on “**Upload**” button.



9. The notification confirms that the file has been successfully uploaded. Click on “**Close**” to return to our bucket details page.

Upload succeeded
For more information, see the [Files and folders table](#).

Summary

Destination	Succeeded	Failed
s3://sujal03bucket	1 file, 189.7 KB (100.00%)	0 files, 0 B (0%)

Files and folders [Configuration](#)

Files and folders (1 total, 189.7 KB)

Name	Folder	Type	Size	Status
Screenshot 2025-02-21 110357.png	-	image/png	189.7 KB	Succeeded

10. Click on our previously uploaded file from the list. Copy the “**Object URL**” from the file details page that opens.

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
Screenshot 2025-02-21 110357.png	png	February 21, 2025, 11:05:19 (UTC+05:30)	189.7 KB	Standard

Screenshot 2025-02-21 110357.png [Info](#)

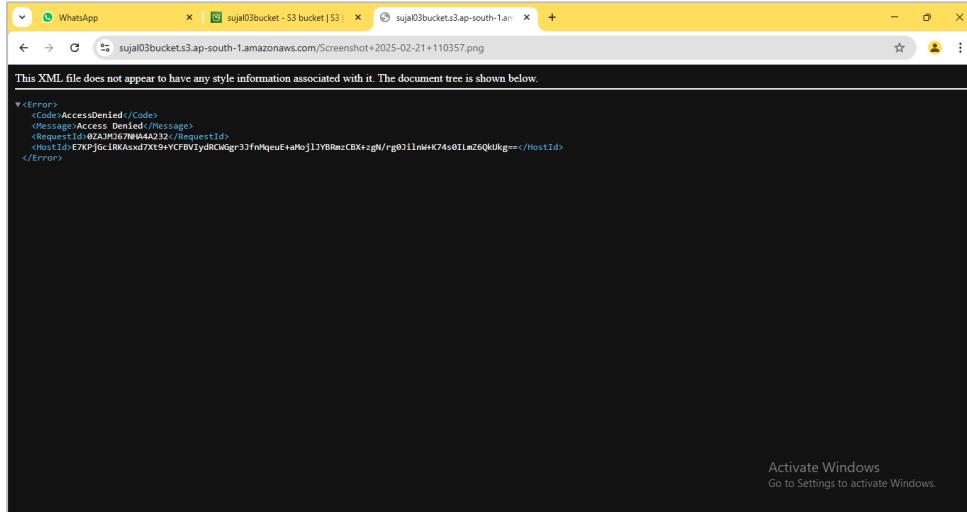
[Copy S3 URI](#) [Download](#) [Open](#) [Actions](#) [Create folder](#) [Upload](#)

Properties [Permissions](#) [Versions](#)

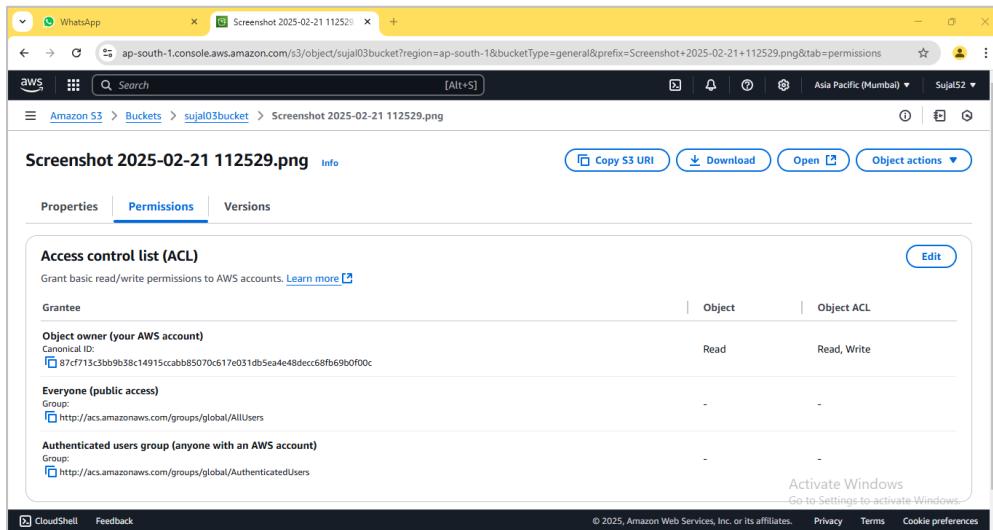
Object overview

Owner 87cf713c3bb9b38c14915ccabb85070c617e031db5ea4e48decc68fb69b0f00c	S3 URI https://s3.ap-south-1.amazonaws.com/sujal03bucket/Screenshot%202025-02-21%20110357.png
AWS Region Asia Pacific (Mumbai) ap-south-1	Amazon Resource Name (ARN) arn:aws:s3:::sujal03bucket/Screenshot%202025-02-21%20110357.png
Last modified February 21, 2025, 11:05:19 (UTC+05:30)	Entity tag (Etag) c75b38dd5d6cb5025146f35250e98306
Size 189.7 KB	Object URL https://sujal03bucket.s3.ap-south-1.amazonaws.com/Screenshot%202025-02-21%20110357.png
Type png	
Key	

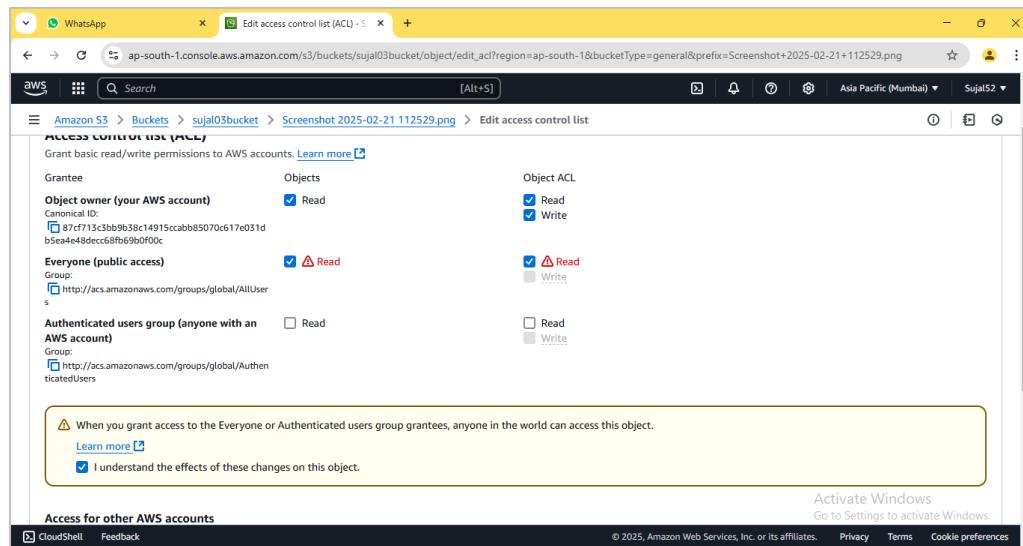
11. Open the copied URL in a new browser window. An error appears indicating that access to the contents is restricted since this is a private bucket.



12. Go back to file details page and click on “**Permissions**” tab. Then click on “**Edit**” button.



13. Edit the permissions as follow then click on “**Save changes**”.



The screenshot shows the 'Edit access control list (ACL)' page for an S3 object. The 'Object ACL' section shows 'Read' and 'Write' permissions are checked for 'Everyone (public access)'. A note below states: 'When you grant access to the Everyone or Authenticated users group grantees, anyone in the world can access this object.' A checkbox for 'I understand the effects of these changes on this object.' is checked. The bottom navigation bar includes CloudShell, Feedback, and links for Activate Windows, Go to Settings, Privacy, Terms, and Cookie preferences.

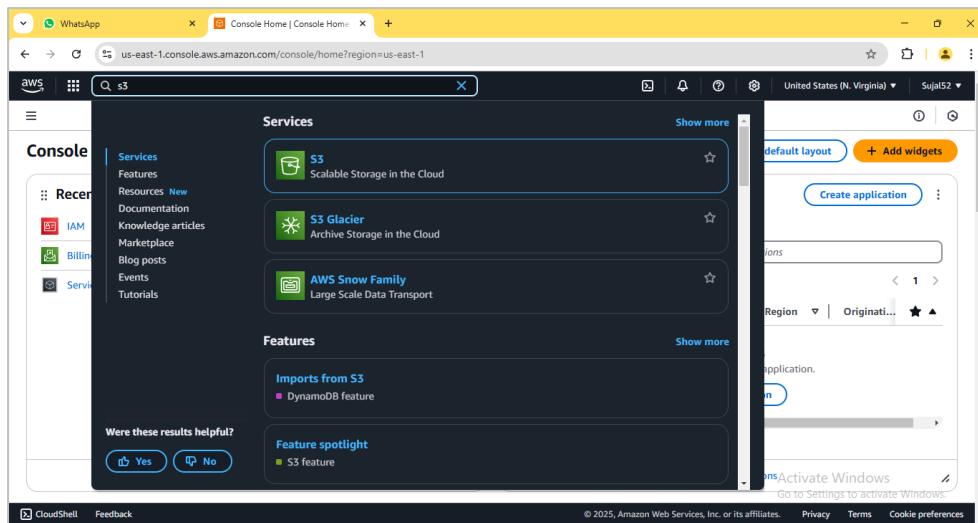
14. Copy the URL as mentioned previously and open it on new window.

ASSIGNMENT 6:

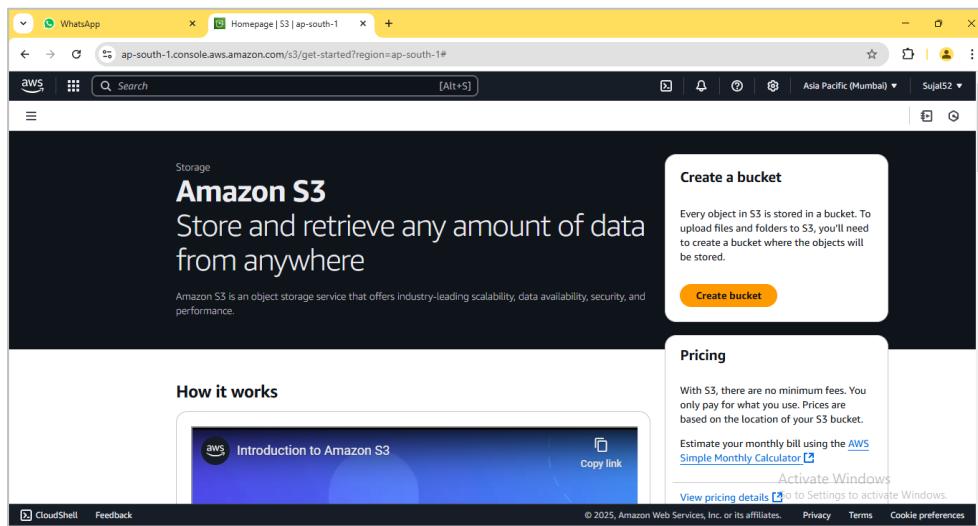
Problem Definition: Upload a static website on S3.

Instructions:

1. Access the **AWS console**, search for **S3**, and select the top option from the search results.



2. Click on “**Create bucket**”.



3. Enter bucket name. Select “**ACLs enabled**” since we are creating a public bucket. Uncheck the box for “**Block all public access**”. Leave other options as default and click on “**Create bucket**”.

The screenshots show the 'Create S3 bucket' wizard in the AWS Management Console. The first step, 'Object Ownership', shows the 'ACLs enabled' option selected. The second step, 'Block Public Access settings for this bucket', shows the 'Block all public access' checkbox unselected. The third step, 'Encryption type', shows 'Server-side encryption with Amazon S3 managed keys (SSE-S3)' selected. All steps include a note about activating Windows.

Object Ownership info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

- Bucket owner preferred**
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.
- Object writer**
The object writer remains the object owner.

If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. [Learn more](#) [Activate Windows](#) [Go to Settings to activate Windows](#)

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting. [Activate Windows](#) [Go to Settings to activate Windows](#)

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Encryption type [Info](#)

- Server-side encryption with Amazon S3 managed keys (SSE-S3)**
- Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- Dual-layer server-side encryption with AWS Key Management Service keys (DSS-E-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see [DSS-E-KMS pricing](#) on the Storage tab of the [Amazon S3 pricing page](#).

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSS-E-KMS. [Learn more](#)

- Disable**
- Enable**

Advanced settings

Note: After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Create bucket [Cancel](#) [Activate Windows](#) [Go to Settings to activate Windows](#)

4. A green coloured pop-up dialogue box shall confirm the creation of our bucket. It shall also be displayed on the bucket list.

The screenshot shows the AWS S3 console in the Asia Pacific (Mumbai) region. A green success message at the top states "Successfully created bucket 'sujal03bucket'". Below it, the "General purpose buckets" section lists one bucket: "sujal03bucket" (Info, All AWS Regions). The bucket details show it was created on February 21, 2025, at 10:59:18 UTC+05:30. The "Create bucket" button is visible. The interface includes standard AWS navigation elements like CloudShell and Feedback.

5. Click on the name of our newly created bucket. On the bucket detail page, click on “**Upload**”.

The screenshot shows the AWS S3 bucket detail page for "sujal03bucket". The "Objects" tab is selected, showing 0 objects. The "Actions" menu has an "Upload" option highlighted. The page includes sections for Properties, Permissions, Metrics, Management, and Access Points. The "Upload" button is located in the center of the object list area. The interface includes standard AWS navigation elements like CloudShell and Feedback.

6. Click on “**Add files**”. Select our html files from the list. Expand “**Permissions**” tab, select “**Grant public read access**” and check the confirmation popup. Finally, click on “**Upload**” button.

The screenshot shows the AWS S3 console interface for uploading files to a bucket named "staticwebsitesujal03".

Top Navigation: ap-south-1.console.aws.amazon.com/s3/upload/staticwebsitesujal03?region=ap-south-1&bucketType=general

Breadcrumbs: Amazon S3 > Buckets > staticwebsitesujal03 > Upload

Permissions Tab (Visible):

- Access control list (ACL):
 - Choose from predefined ACLs (selected)
 - Specify individual ACL permissions
- Predefined ACLs:
 - Private (recommended) Only the object owner will have read and write access.
 - Grant public-read access Anyone in the world will be able to access the specified objects. The object owner will have read and write access. (Selected)
- Granting public-read access is not recommended: Anyone in the world will be able to access the specified objects. (Info link)
- I understand the risk of granting public-read access to the specified objects. (checkbox checked)

Properties Tab (Collapsed): Specify storage class, encryption settings, tags, and more.

Upload Tab (Visible):

- Activate Windows: Go to Settings to activate Windows.
- Upload button
- Drag and drop files and folders you want to upload here, or choose Add files or Add folder.
- Files and folders (2 total, 652.0 B):

Name	Type	Size
about.html	text/html	326.0 B
index.html	text/html	326.0 B
- Destination:
 - Destination: s3://staticwebsitesujal03 (Info link)
 - Destination details: Bucket settings that impact new objects stored in the specified destination.
- Permissions:
 - Grant public access and access to other AWS accounts.

7. The notification confirms that the file has been successfully uploaded. Click on “**Close**” to return to our bucket details page.

Upload objects - S3 bucket stat

ap-south-1.console.aws.amazon.com/s3/upload/staticwebsitesujal03?region=ap-south-1&bucketType=general

Upload: status

After you navigate away from this page, the following information is no longer available.

Summary

Destination	Succeeded	Failed
s3://staticwebsitesujal03	2 files, 652.0 B (100.00%)	0 files, 0 B (0%)

Files and folders Configuration

Files and folders (2 total, 652.0 B)

Name	Folder	Type	Size	Status	Error
about.html	-	text/html	326.0 B	SUCCEEDED	-
index.html	-	text/html	326.0 B	SUCCEEDED	-

Activate Windows
Go to Settings to activate Windows.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

8. From the bucket details page, click on “**Properties**” tab, scroll down and click on “**Edit**” next to “**Static website hosting**”.

staticwebsitesujal03 - S3 bucket

ap-south-1.console.aws.amazon.com/s3/buckets/staticwebsitesujal03?region=ap-south-1&tab=properties

Object Lock
Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. [Learn more](#)

Object Lock
Disabled

Requester pays
When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. [Learn more](#)

Requester pays
Disabled

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#)

We recommend using AWS Amplify Hosting for static website hosting
Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about [Amplify Hosting](#) or [View your existing Amplify apps](#)

Create Amplify app

S3 static website hosting
Disabled

Activate Windows
Go to Settings to activate Windows.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

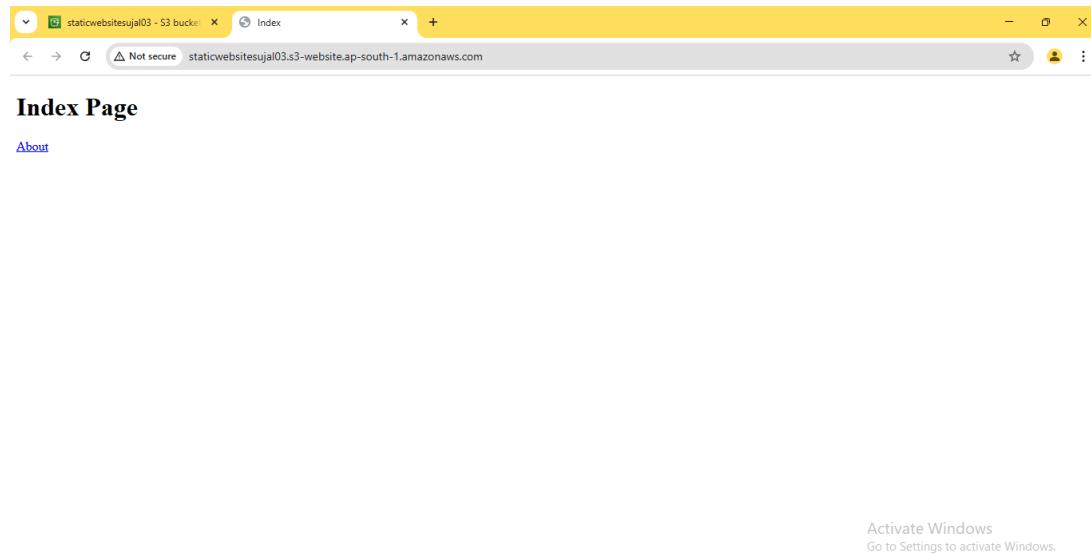
9. Under “**Static website hosting**”, click on “**Enable**”, then fill in the name of index document and click on “**Save changes**”.

The screenshot shows the 'Edit static website hosting' page for the 'staticwebsitelujal03' bucket. The 'Static website hosting' section is open, showing the 'Enable' radio button selected. The 'Index document' field contains 'index.html'. A note at the bottom states: 'For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access.' The page also includes sections for 'Error document - optional' (set to 'error.html') and 'Redirection rules - optional' (empty). At the bottom right, there's an 'Activate Windows' message with a link to Settings.

10. Once again, scroll down to “**Static website hosting**” and copy the URL under “**Bucket website endpoint**”.

The screenshot shows the 'Properties' tab of the 'staticwebsitelujal03' bucket. In the 'Static website hosting' section, the 'Bucket website endpoint' is listed as 'http://staticwebsitelujal03.s3-website.ap-south-1.amazonaws.com'. There is a 'Create Amplify app' button next to it. Other sections visible include 'Object Lock' (Disabled), 'Requester pays' (Disabled), and 'Static website hosting' (Enabled, with a note about AWS Amplify Hosting).

11. Open the copied URL in a new browser window. Our website should be accessible now.

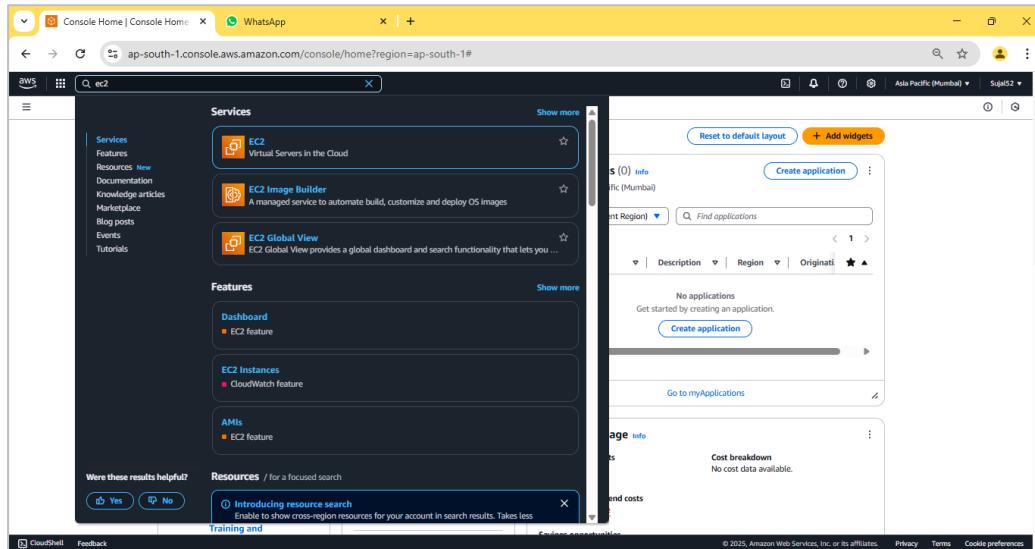


ASSIGNMENT 7:

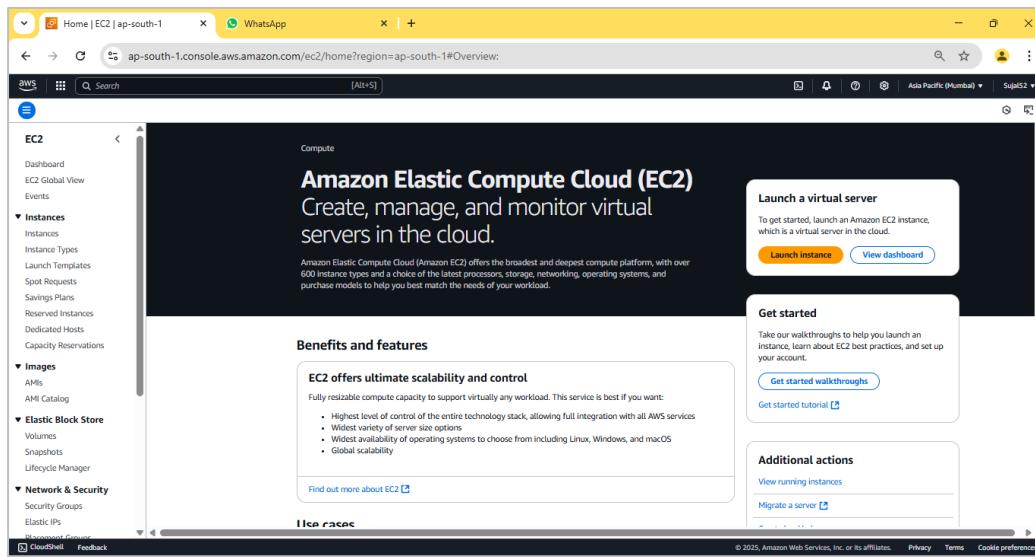
Problem Definition: Hosting a website on EC2.

Instructions:

1. Access the **AWS console**, search for **EC2**, and select the top option from the search results.



2. Click on "**Launch instance**".



3. Enter instance name and select **Ubuntu** under “**Application and OS Images**”. Under “**Key Pair (login)**” click on “**Create new key pair**”, enter key pair name then click on “**Create key pair**” button, a *.pem file will be downloaded. Then click on “**Launch instance**” button

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Name and tags' section, the instance is named 'SujatEC2'. In the 'Application and OS Images (Amazon Machine Image)' section, 'Ubuntu Server 24.04 LTS (HVM, SSD Volume Type)' is selected. Under 'Key pair (login)', a new key pair is being created with the name 'mykey123'. The 'Create key pair' button is highlighted. The 'Launch instance' button is also visible at the bottom right.

4. Our instance is created successfully.

The screenshot shows the AWS EC2 Instances 'Launch an instance' page. A green success banner at the top reads 'Successfully initiated launch of instance i-0c4e4674909306b65'. Below the banner, there's a 'Launch log' section with a link to 'View log'. Under 'Next Steps', there are several cards: 'Create billing and free tier usage alerts', 'Connect to your instance', 'Connect an RDS database', 'Create EBS snapshot policy', 'Manage detailed monitoring', 'Create Load Balancer', 'Create AWS budget', and 'Manage CloudWatch alarms'. At the bottom of the page, there are links for 'CloudShell' and 'Feedback'.

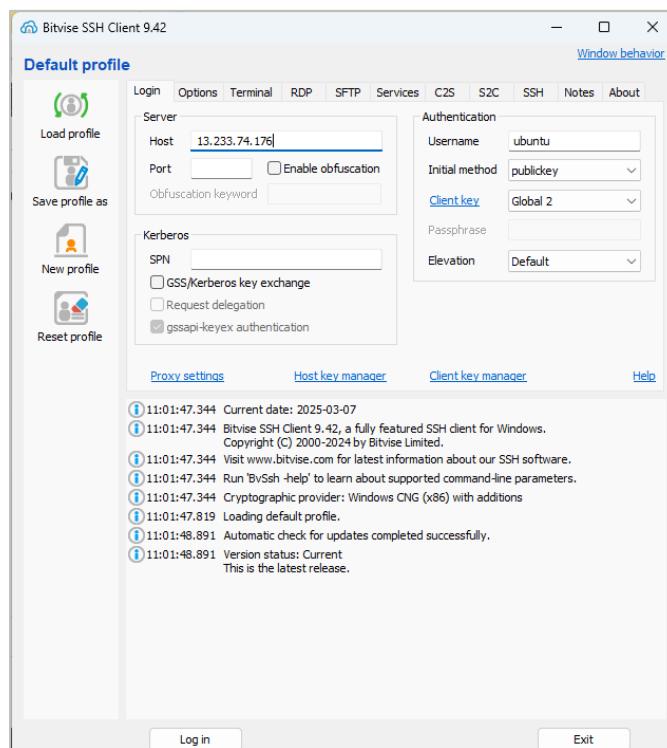
5. Go to “**Instance**” from the sidebar, then click on **Instance ID** of our newly created instance.

The screenshot shows the AWS EC2 Instances page. The main table lists one instance: 'SujalEC2' with Instance ID 'i-0c4e4674909306b65', State 'Running', and Type 't2.micro'. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 address, Elastic IP, and IPv6 IPs. A 'View alarms +' button is also present. Below the table, there's a 'Select an instance' dropdown menu.

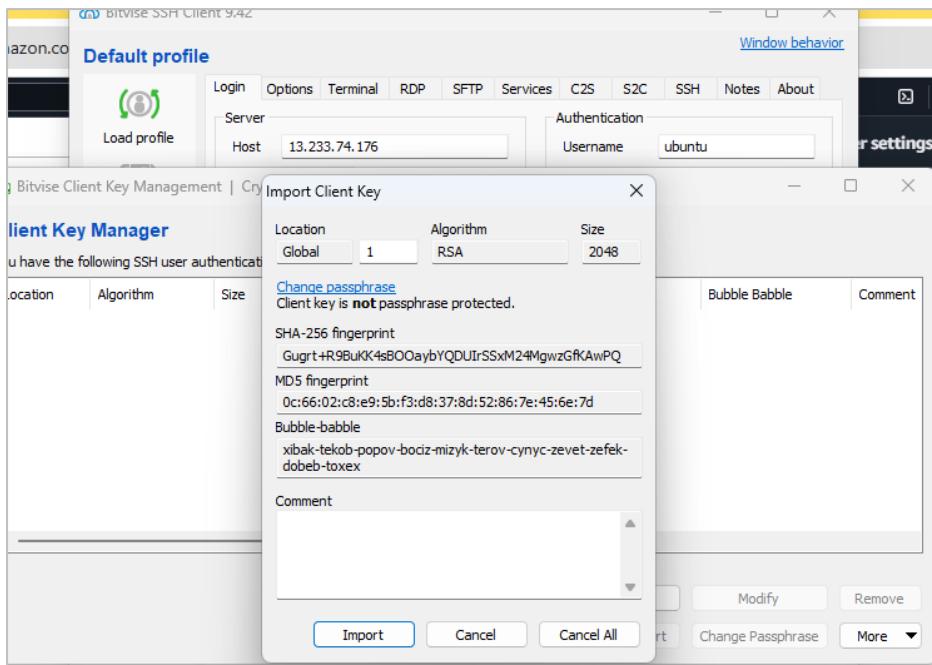
6. Copy “**Public IPV4 address**”.

The screenshot shows the AWS EC2 Instance details page for instance 'i-0c4e4674909306b65'. The left sidebar shows various instance details like Instance ID, IPv6 address, Hostname type, Auto-assigned IP address, IAM Role, IMDSv2 Required, Operator, and AMI ID. The right panel displays the 'Instance summary' for the instance, including its Public IPv4 address (13.233.74.176), Private IP DNS name (ip-172-31-9-56.ap-south-1.compute.internal), Instance type (t2.micro), VPC ID (vpc-0135b70f94ff1776), Subnet ID (subnet-0c35ba680cc9a16ea), and Instance ARN (arn:aws:ec2:ap-south-1:577638390776:instance/i-0c4e4674909306b65). It also shows Private IPv4 addresses (172.31.9.56) and Public IPv4 DNS (ec2-13-233-74-176.ap-south-1.compute.amazonaws.com). The 'Details' tab is selected at the bottom.

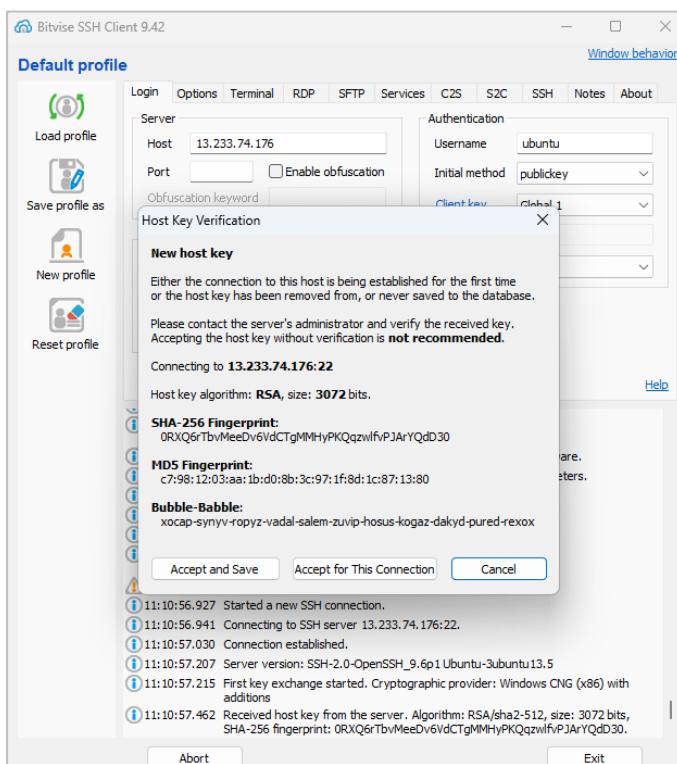
7. Open **Bitvise SSH Client**, paste the previously copied address in host field.



8. Click "**Client key manager**", click on "**Import**" and select our previously downloaded key pair file. Click on "**Import**" button. Our key pair should be imported successfully.

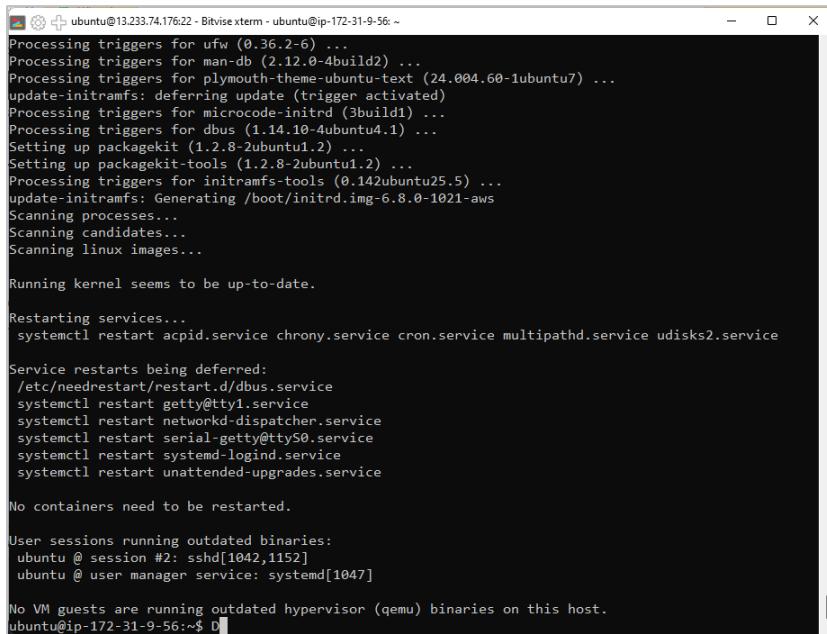


9. In “**Bitvise SSH Client**”, under “**Authentication**” give the username as “**ubuntu**”, select “**Global 1**” under “**Client Key**” then click on “**Log in**” & “**Accept and save**”.



10. Click on “**New terminal console**” from the sidebar. In the terminal enter the following commands:
 sudo apt-get update && sudo apt-get upgrade

```
sudo apt-get install nginx
```



```
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for plymouth-theme-ubuntu-text (24.004.60-lubuntu7) ...
update-initramfs: deferring update (trigger activated)
Processing triggers for microcode-initrd (3build1) ...
Processing triggers for dbus (1.14.10-4ubuntu4.1) ...
Setting up packagekit (1.2.8-2ubuntu1.2) ...
Setting up packagekit-tools (1.2.8-2ubuntu1.2) ...
Processing triggers for initramfs-tools (0.142ubuntu25.5) ...
update-initramfs: Generating /boot/initrd.img-6.8.0-1021-aws
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
systemctl restart acpid.service chrony.service cron.service multipathd.service udisks2.service

Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart serial-getty@ttyS0.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

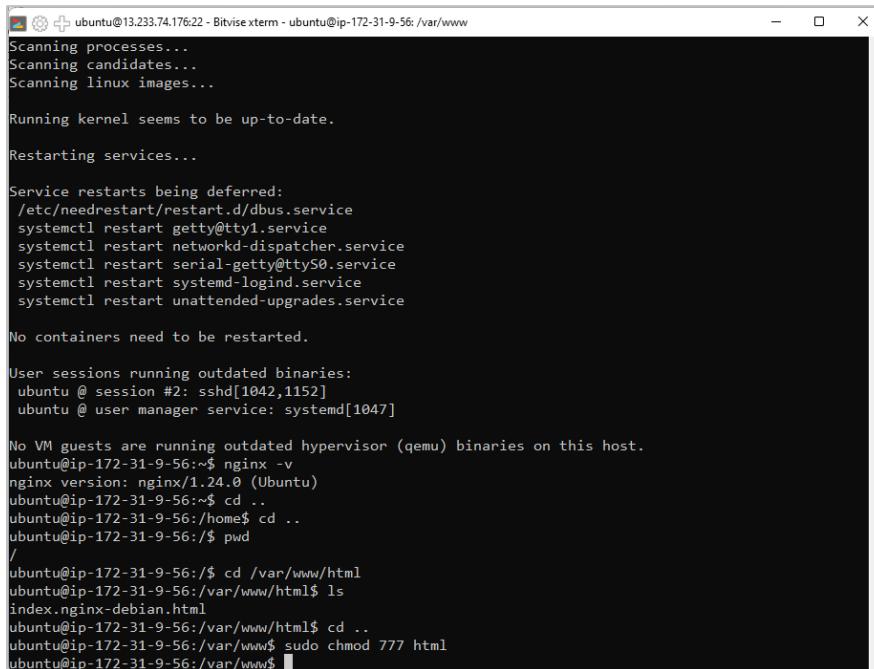
User sessions running outdated binaries:
ubuntu @ session #2: sshd[1042,1152]
ubuntu @ user manager service: systemd[1047]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-9-56:~$ D
```

11. We need to move our html files into var/www/html directory to host it, but first we need to give write permission to do so. To change file permission enter the following command:

```
cd var/www/
```

```
sudo chmod 777 html
```



```
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...

Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart serial-getty@ttyS0.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service

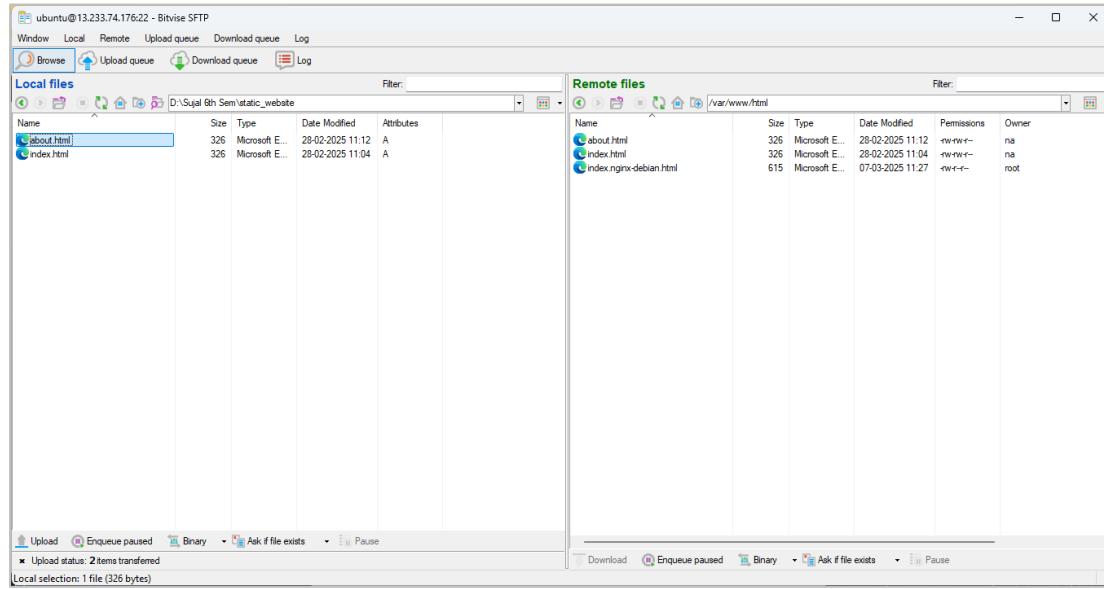
No containers need to be restarted.

User sessions running outdated binaries:
ubuntu @ session #2: sshd[1042,1152]
ubuntu @ user manager service: systemd[1047]

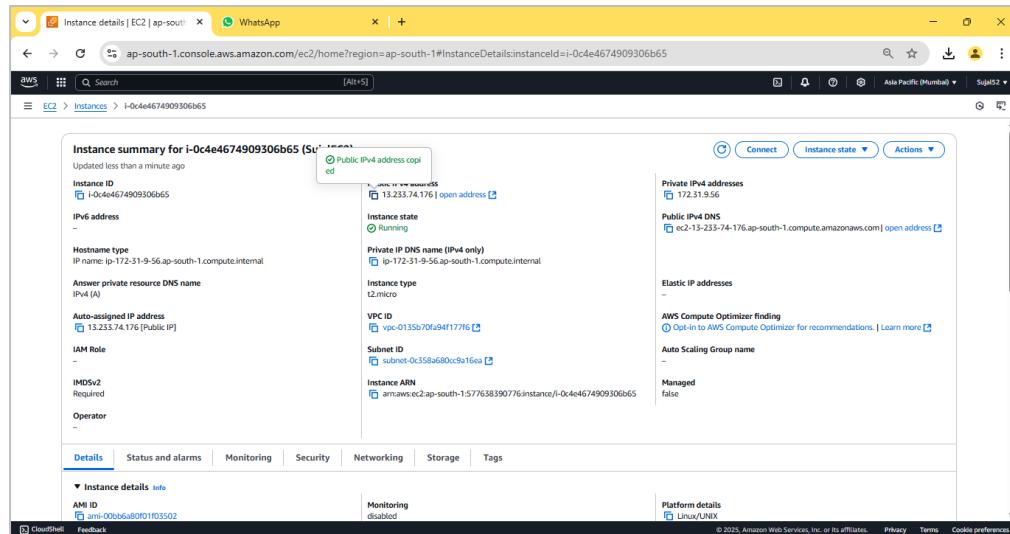
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-9-56:~$ nginx -v
nginx version: nginx/1.24.0 (Ubuntu)
ubuntu@ip-172-31-9-56:~$ cd ..
ubuntu@ip-172-31-9-56:~/home$ cd ..
ubuntu@ip-172-31-9-56:/$ pwd
/
ubuntu@ip-172-31-9-56:/$ cd ./var/www/html
ubuntu@ip-172-31-9-56:~/var/www/html$ ls
index.nginx-debian.html
ubuntu@ip-172-31-9-56:~/var/www/html$ cd ..
ubuntu@ip-172-31-9-56:~/var/www$ sudo chmod 777 html
ubuntu@ip-172-31-9-56:~/var/www$
```

12. Go to SFTP window from the sidebar, under **Local file** open the

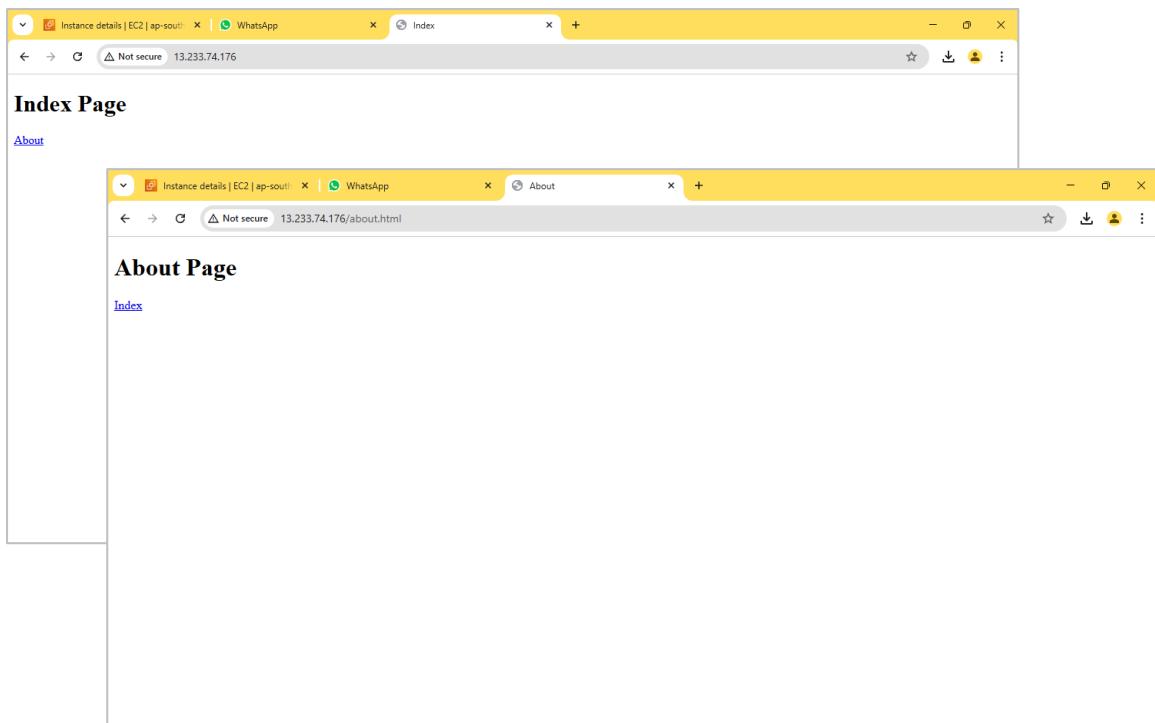
folder containing html files and under **Remote file** open /var/www/html. Drag and drop the HTML files from local to remote.



13. Now go back to the AWS window and copy the “**Public IPV4 address**”.



14. Open the copied address in a new browser window. Our website should be accessible now.



ASSIGNMENT 8:

Problem Definition: Deploy a project from a local machine to GitHub and vice versa.

Instructions:

1. Create a new public Github repository. Click on "**Create repository**" to proceed.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner * Repository name *

SujalGupta52 / SDITO_Lab SDITO_Lab is available.

Great repository names are short and memorable. Need inspiration? How about [scaling-eureka](#) ?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.
 Private You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore .gitignore template: None Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license License: None A license tells others what they can and can't do with your code. [Learn more about licenses](#).

You are creating a public repository in your personal account.

Create repository

2. Go to **Settings** then **Developer settings**. Now click on “**Personal access tokens**” dropdown and then click on “**Tokens(classic)**”.

The first screenshot shows the GitHub user profile sidebar with options like Set status, Your profile, Your repositories, and Settings.

The second screenshot shows the developer settings sidebar with options like Codespaces, Packages, Copilot, Pages, Saved replies, Security, Code security, Integrations, Applications, Scheduled reminders, Archives, Security log, Sponsorship log, and Developer settings.

The third screenshot shows the tokens page under Developer settings. It lists GitHub Apps, OAuth Apps, and Personal access tokens. The Personal access tokens section is expanded, showing a dropdown for "Tokens you have generated". One token, "mytoken — admin:enterprise", is listed with the scope "admin:gpg_key, admin:org".

3. Now click on “**Generate new token**” and then “**Generate new token**”

The screenshot shows the tokens page with the "Personal access tokens (classic)" section open. A "Generate new token" button is visible. A modal window titled "Generate new token" is open, showing the scope "Fine-grained, repo-scoped". Another modal window titled "Generate new token (classic)" is also visible, showing the scope "For general use".

4. Set expiration date for as long as possible and select all scopes.
Then click on “**Generate token**”.

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note
mytoken

What's this token for?

Expiration

The token will expire on the selected date

Select scopes
Scopes define the access for personal tokens. [Read more about OAuth scopes](#):

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo:deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runnersorg	Manage org runners and runner groups
<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks

<input checked="" type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_runners:enterprise	Manage enterprise runners and runner groups
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input type="checkbox"/> scim:enterprise	Provisioning of users and groups via SCIM
<input checked="" type="checkbox"/> audit_log	Full control of audit log
<input type="checkbox"/> read:audit_log	Read access of audit log
<input checked="" type="checkbox"/> codespace	Full control of codespaces
<input type="checkbox"/> codespace:secrets	Ability to create, read, update, and delete codespace secrets
<input checked="" type="checkbox"/> copilot	Full control of GitHub Copilot settings and seat assignments
<input type="checkbox"/> manage_billing:copilot	View and edit Copilot Business seat assignments
<input checked="" type="checkbox"/> write:network_configurations	Write org hosted compute network configurations
<input type="checkbox"/> read:network_configurations	Read org hosted compute network configurations
<input checked="" type="checkbox"/> project	Full control of projects
<input type="checkbox"/> read:project	Read access of projects
<input checked="" type="checkbox"/> admin:gpg_key	Full control of public user GPG keys
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys
<input checked="" type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys

5. After generating the token, click on copy icon and copy the token somewhere secure.

Personal access tokens (classic)

[Generate new token](#)

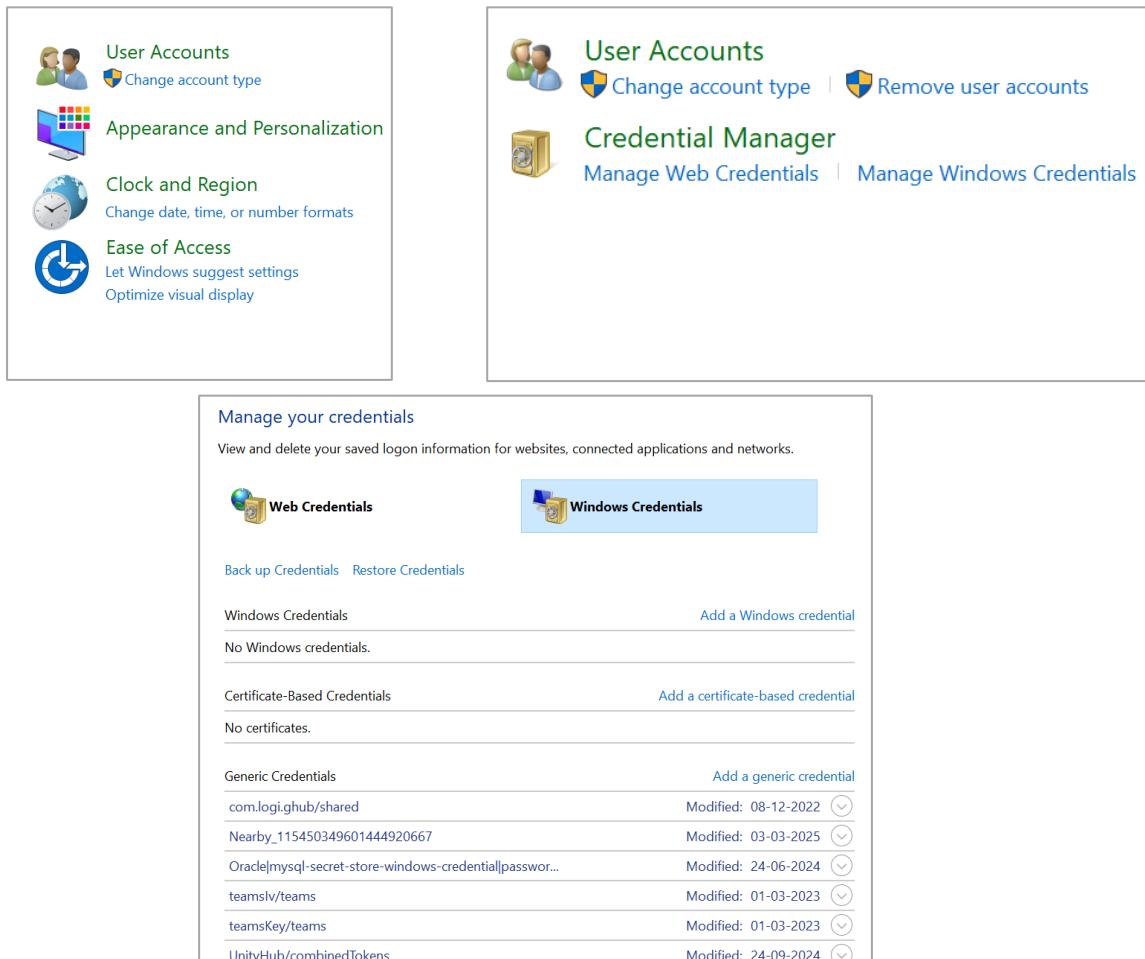
Tokens you have generated that can be used to access the [GitHub API](#).

mytoken — **admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:network_configurations, write:packages**
Last used within the last week

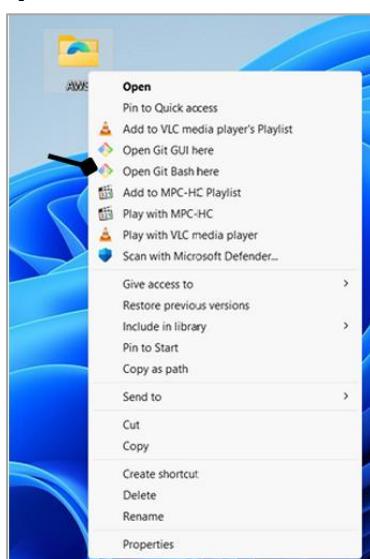
Expires on **Tue, May 20 2025**.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

6. Now go to **Control Panel**, click **User's account**, then go to **Credential Manager** and click on **Window Credentials**. From **Generic Credentials** remove any existing GitHub account credentials.



7. Go to the folder containing your project files. Right Click, and select “**Open Git Bash Here**” from the pop-up menu.

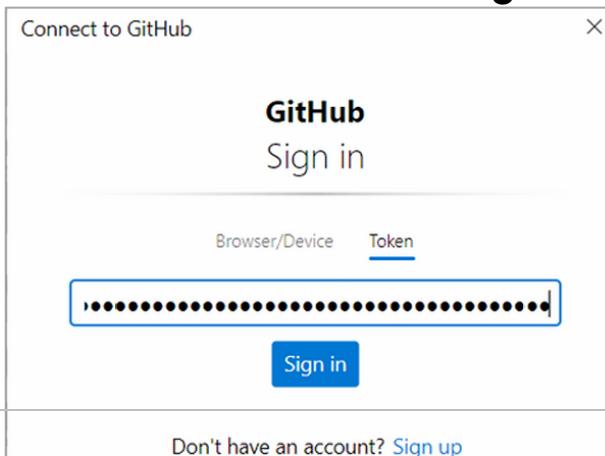


8. Run the following commands:

```
git init  
git add .  
git status  
git config --global user.email <user email address>  
git config --global user.name <username>  
git commit -m "<message>"  
git push -u origin master
```

```
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS  
$ git init  
Initialized empty Git repository in C:/Users/Meghana/Desktop/AWS/.git/  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ dir  
index.html index1.html index2.html  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ git add .  
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it  
warning: in the working copy of 'index1.html', LF will be replaced by CRLF the next time Git touches it  
warning: in the working copy of 'index2.html', LF will be replaced by CRLF the next time Git touches it  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ git status  
On branch master  
No commits yet  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file: index.html  
    new file: index1.html  
    new file: index2.html  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ git config --global user.email meghanachoudhary1504@gmail.com  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ git config --global user.name meghana-choudhary  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ git commit -m "Done"  
[master (root-commit) 31d1c75] Done  
 3 files changed, 40 insertions(+)  
  create mode 100644 index.html  
  create mode 100644 index1.html  
  create mode 100644 index2.html  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ git remote add origin https://github.com/meghana-choudhary/MyAws.git  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ git push -u origin master  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (5/5), done.  
Writing objects: 100% (5/5) 63 bytes | 582.00 KiB/s, done.  
Total 5 (delta 0), reused 0 (from 0)  
remote: Resolving deltas: 100% (1/1) done.  
To https://github.com/meghana-choudhary/MyAws.git  
 * [new branch] master -> master  
branch 'master' set up to track 'origin/master'.  
Meghana@DESKTOP-FBKGRS MINGW64 ~/Desktop/AWS (master)  
$ |
```

9. After the last command, a new “**Connect to GitHub**” window should open. Go to “**Token**” tab and paste the previously generated token into the text field. Click on “**Sign in**” button.



10. Open the repository in **Github**. Our project files should now be visible. Now click on “**Code**” button and copy the HTTPS URL.

The screenshot shows the GitHub repository page for 'SDITO-Lab'. The 'Code' tab is active. A modal window is open in the center, titled 'Clone', showing three options: 'HTTPS', 'SSH', and 'GitHub CLI'. The 'HTTPS' option is selected, displaying the URL 'https://github.com/SujalGupta52/SDITO-Lab.git'. Below the URL, there are links to 'Clone using the web URL.', 'Open with GitHub Desktop', 'Open with Visual Studio', and 'Download ZIP'.

11. Now make a new folder and open the newly created folder in git bash terminal. Run the following command to clone our repo:
`git clone <Repository address>`

```
PS C:\Users\sujal\OneDrive\Desktop\AWS> git clone git@github.com:SujalGupta52/SDITO-Lab.git
Cloning into 'SDITO-Lab'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (7/7), 48.22 KiB | 137.00 KiB/s, done.
```

12. Our repo files is successfully cloned locally.

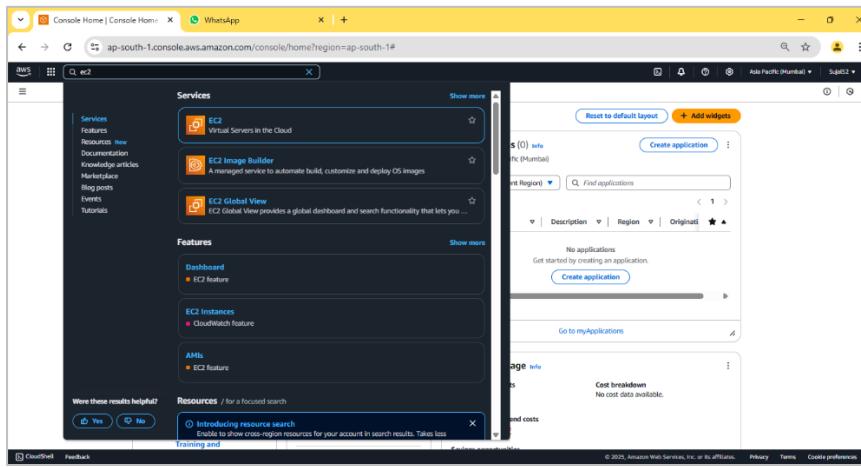
Name	Date modified	Type	Size
.git	26-03-2025 22:09	File folder	
.gitignore	26-03-2025 22:09	Text Document	1 KB
index.js	26-03-2025 22:09	JavaScript Source File	1 KB
New Text Document.txt	26-03-2025 22:09	Text Document	1 KB
package.json	26-03-2025 22:09	JSON Source File	1 KB
package-lock.json	26-03-2025 22:09	JSON Source File	179 KB

ASSIGNMENT 9:

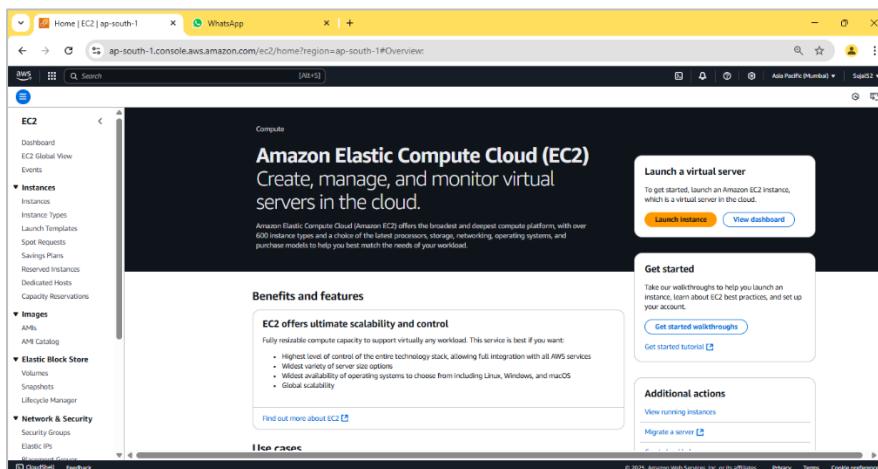
Problem Definition: Deploy a project from GitHub to EC2.

Instructions:

1. Access the **AWS console**, search for **EC2**, and select the top option from the search results.



2. Click on “**Launch instance**”.



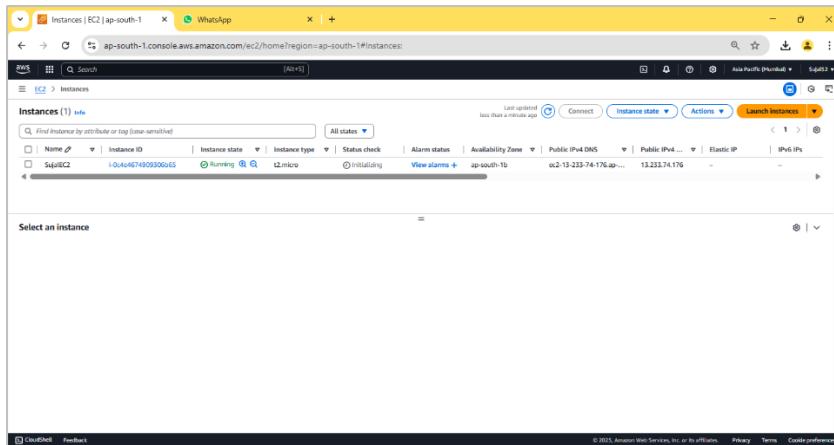
3. Enter instance name and select **Ubuntu** under “**Application and OS Images**”. Under “**Key Pair (login)**” click on “**Create new key pair**”, enter key pair name then click on “**Create key pair**” button, a *.pem file will be downloaded. Then click on “**Launch instance**” button

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Application and OS Images (Amazon Machine Image)' section, 'Ubuntu' is selected. A tooltip for the 'Create key pair' step states: 'Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro when t2.micro isn't available) when used with free tier AMIs. This includes 100 hours per month of t2.micro instance usage, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.' Below this, the 'Launch instance' button is visible.

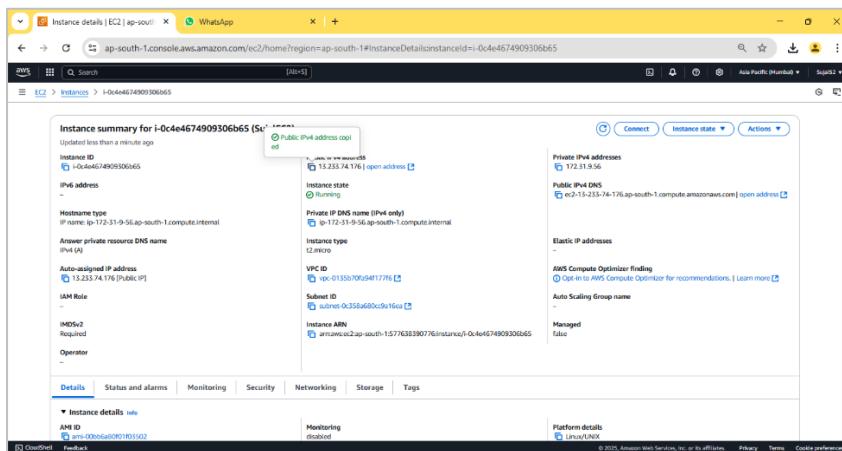
4. Our instance is created successfully.

The screenshot shows the AWS EC2 'Launch an instance' wizard. A green success message at the top reads: 'Success Successfully initiated launch of instance i-0c4a674999306565'. Below this, a 'Next Steps' section lists several options: 'Create billing and free tier usage alerts', 'Connect to your instance', 'Connect an RDS database', 'Create EBS snapshot policy', 'Manage detailed monitoring', 'Create Load Balancer', 'Create AWS budget', and 'Manage CloudWatch alarms'. Each option has a corresponding 'Learn more' link.

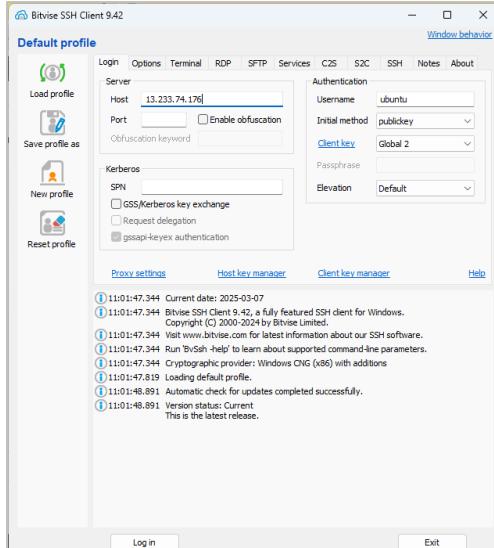
5. Go to “**Instance**” from the sidebar, then click on **Instance ID** of our newly created instance.



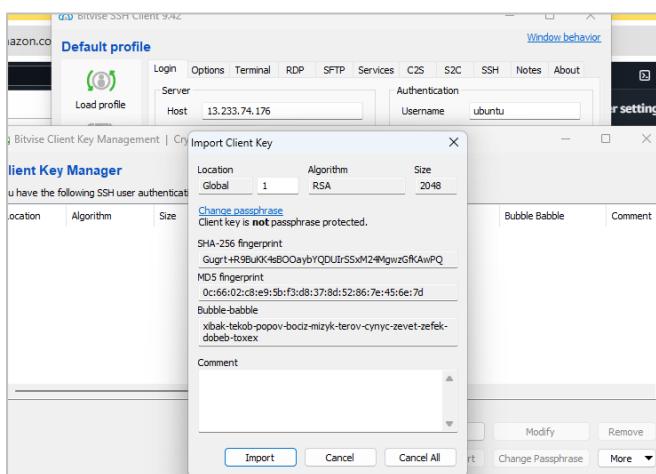
6. Copy “**Public IPV4 address**”.



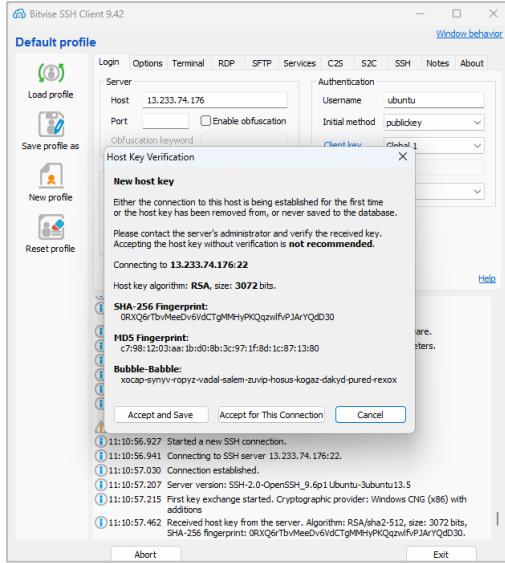
7. Open **Bitvise SSH Client**, paste the previously copied address in host field.



8. Click “**Client key manager**”, click on “**Import**” and select our previously downloaded key pair file. Click on “**Import**” button. Our key pair should be imported successfully.



9. In “**Bitvise SSH Client**”, under “**Authentication**” give the username as “**ubuntu**”, select “**Global 1**” under “**Client Key**” then click on “**Log in**” & “**Accept and save**”.



10. Click on “**New terminal console**” from the sidebar. In the terminal enter the following commands to clone our repo and run our Node server:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo apt-get install nginx
```

```
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

```
sudo apt install nodejs
```

```
git clone <Github repo link>
```

```
cd <Github repo name>
```

```
npm install
```

```
node index.js
```

```

ubuntu@ip-172-31-13-131:~$ ls
Aritra_01
ubuntu@ip-172-31-13-131:~$ cd Aritra_01
ubuntu@ip-172-31-13-131:~/Aritra_01$ ls
index.js  package-lock.json  package.json
ubuntu@ip-172-31-13-131:~/Aritra_01$ npm install
npm warn deprecated uid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 258 packages, and audited 259 packages in 8s

18 packages are looking for funding
  run 'npm fund' for details

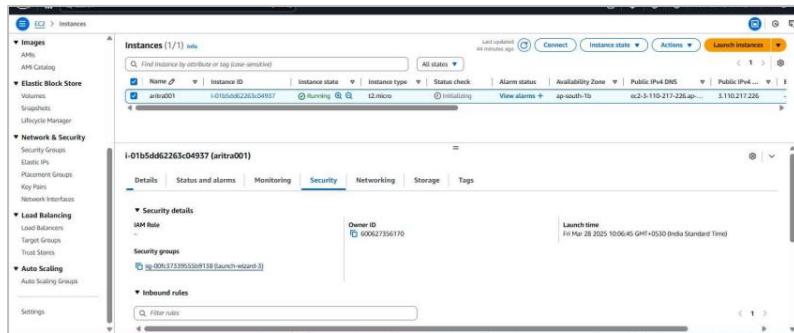
23 vulnerabilities (3 low, 2 moderate, 16 high, 2 critical)

To address all issues, run:
  npm audit fix

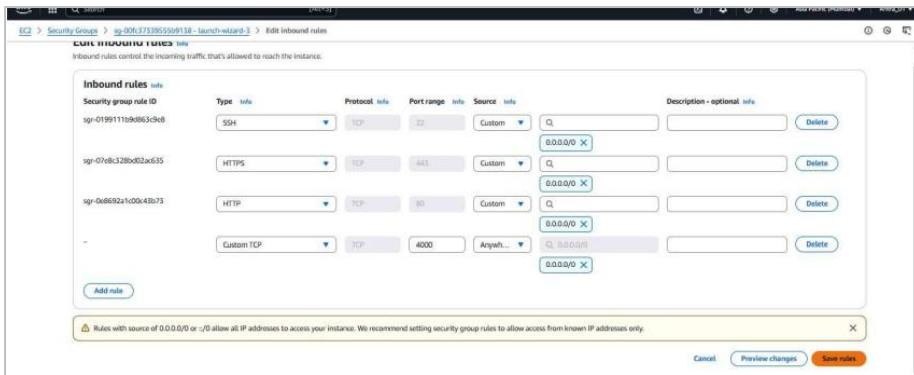
Run 'npm audit' for details.
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.2.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.2.0
npm notice To update run: npm install -g npm@11.2.0
npm notice
ubuntu@ip-172-31-13-131:~/Aritra_01$ npm -v
10.8.2
ubuntu@ip-172-31-13-131:~/Aritra_01$ node index.js
Started server

```

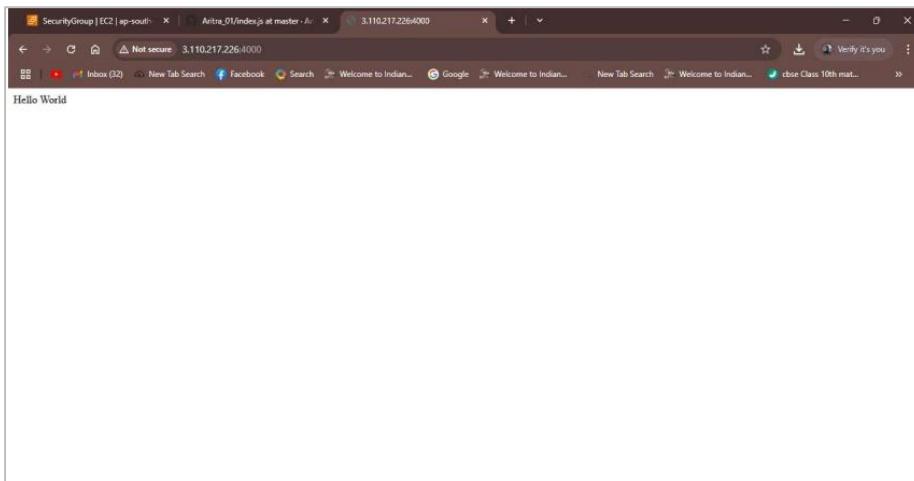
11. Go to “**Instance Id**”, and click on our instance. Then go to ‘Security’ and click on link under ‘Security groups’ to get the edit option.



12. Then in “**Inbound rules**”, click on “**Edit inbound rules**”. Click on “**Add rule**” and set the “**Port range**” to “**4000**”, in “**Source**” set “**0.0.0.0/0**” & click on “**Save rules**”. A confirmation is shown stating our rules are saved.



13. Now copy the “**Public IPv4 address**” & paste it on a new tab. Add “**:4000**” at the end and then press enter. The project has been successfully deployed from GitHub to EC2.

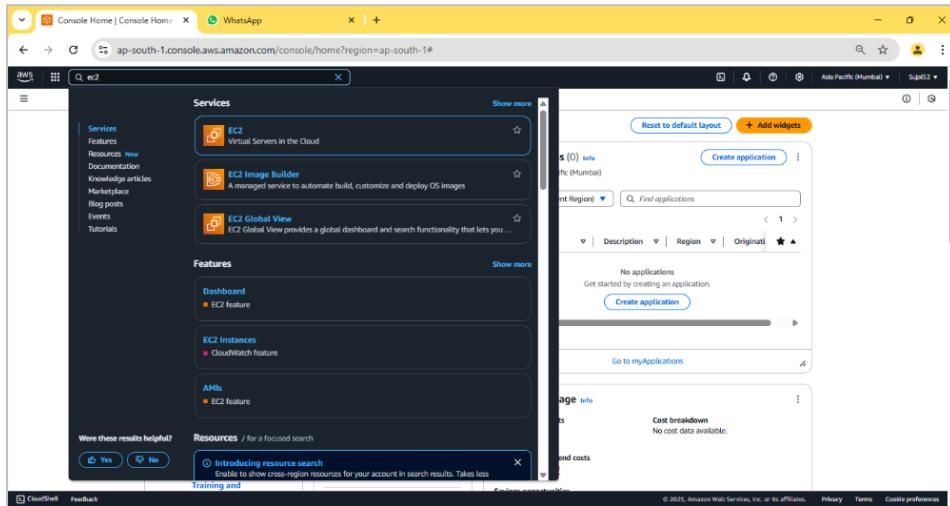


ASSIGNMENT 10:

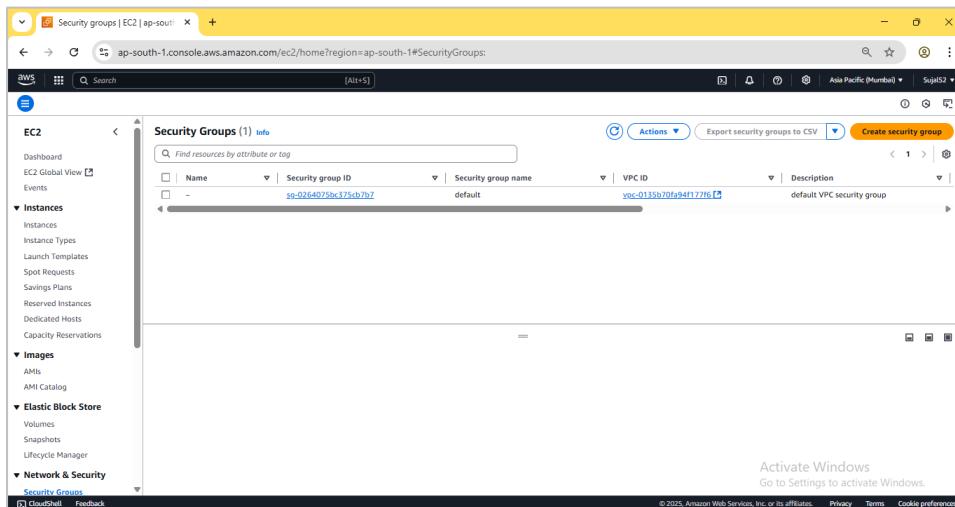
Problem Definition: Deploy a project from GitHub to EC2 by creating a new security group and user data.

Instructions:

1. Access the **AWS console**, search for **EC2**, and select the top option from the search results.



2. Select **Security Groups** from the side bar and click on “**Create security group**”.



3. Add name and description. Under “**Inbound Rules**”, add the “**Add rule**” to add the following rules: SSH, HTTP, HTTPS and Custom TCP with port 4000. Select 0.0.0.0/0 under source for all rules. Finally click

on “**Create security group**”. Pop up is shown stating our rules are created successfully.

The screenshots show the 'Create security group' wizard in the AWS Management Console. The first screen shows basic details like security group name ('MySecurityGroup'), description ('Security Group'), and VPC ('vpc-0135b70fa94f177f6'). The second screen shows inbound rules being added, including custom TCP (port 4000), SSH (port 22), HTTP (port 80), and HTTPS (port 443). The third screen shows outbound rules being added, including all traffic (Custom, port range 0.0.0.0/-). Each screenshot includes a success message at the bottom: 'Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' and 'Rules with destination of 0.0.0.0/0 or ::/0 allow your instances to send traffic to any IPv4 or IPv6 address. We recommend setting security group rules to be more restrictive and to only allow traffic to specific known IP addresses.'

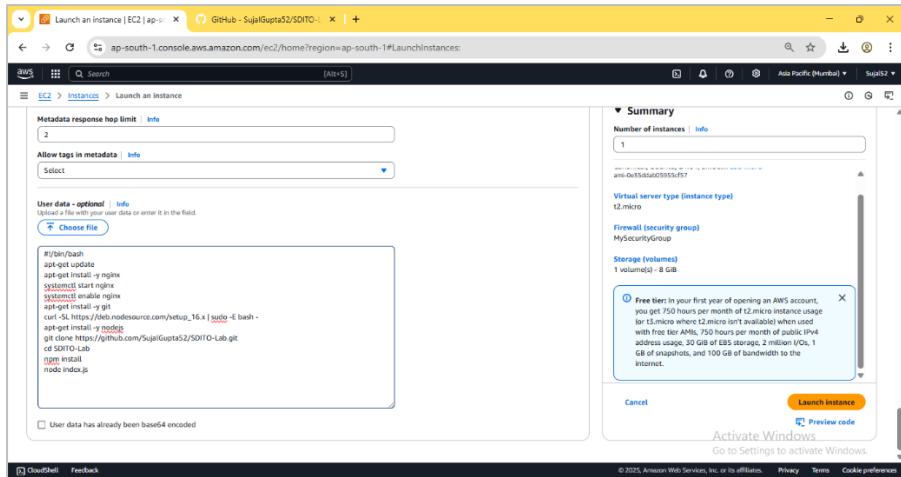
4. Go to **Instances** from the sidebar and click on “**Launch Instance**”. Enter instance name, select Ubuntu as operating system, select an existing key pair or create a new one. Under “**Network**

settings", click on "**Select existing security group**" and select our previously created security group.

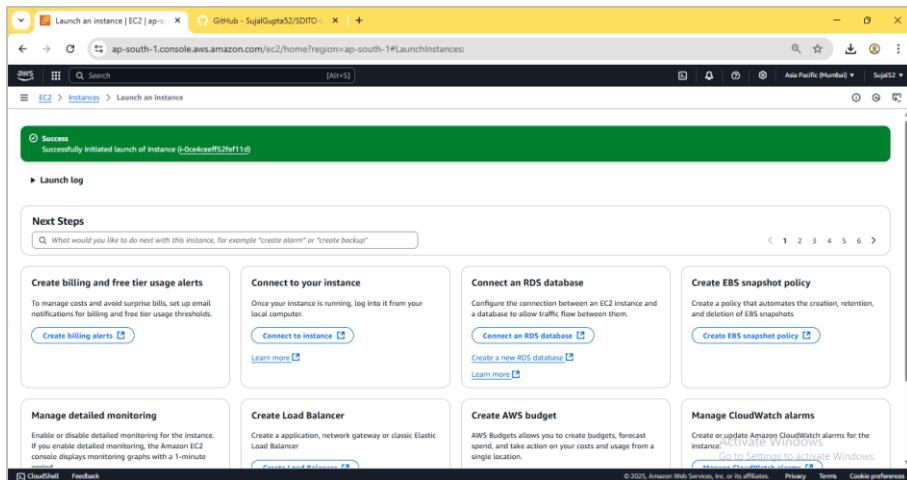
The image consists of three vertically stacked screenshots of the AWS EC2 "Launch an instance" wizard, showing the process of creating a new Amazon Linux 2 instance.

- Screenshot 1:** Shows the initial configuration screen where the user has selected the "Amazon Linux" AMI and chosen the "t2.micro" instance type. A tooltip for the "Free tier" is visible, stating: "In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free-tier AMIs, 750 hours per month of public IPv4 address usage, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet."
- Screenshot 2:** Shows the "Key pair" configuration step. The user has created a key pair named "mykey123". A tooltip for the "Free tier" is visible, stating: "In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free-tier AMIs, 750 hours per month of public IPv4 address usage, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet."
- Screenshot 3:** Shows the "Network settings" configuration step. The user has selected a VPC and subnet, and chosen a security group named "MySecurityGroup". A tooltip for the "Free tier" is visible, stating: "In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free-tier AMIs, 750 hours per month of public IPv4 address usage, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet."

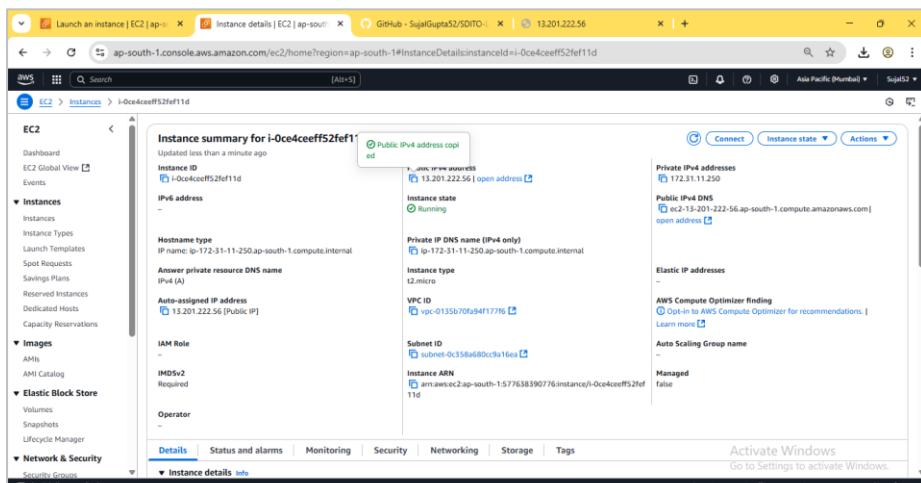
5. Expand "**Advanced details**", scroll down to "**User data**" and put the following bash script. Then click on "**Launch Instance**".



6. Our instance is created successfully

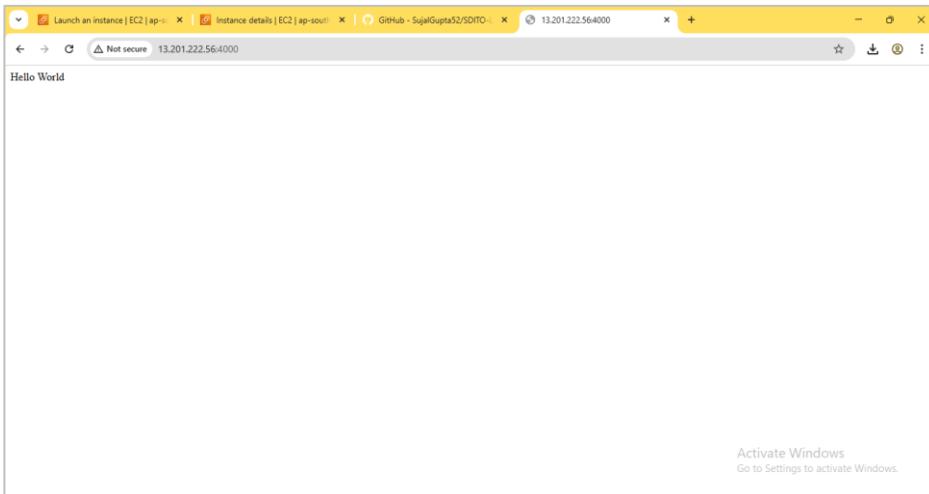


7. Go to **Instance**, click on “**Instance ID**” of our newly created instance. Copy the **IPV4 address**.



8. Paste the previously copied address into a new browser tab and suffix “:4000” to indicate our port. Our NodeJS server is successfully

deployed.

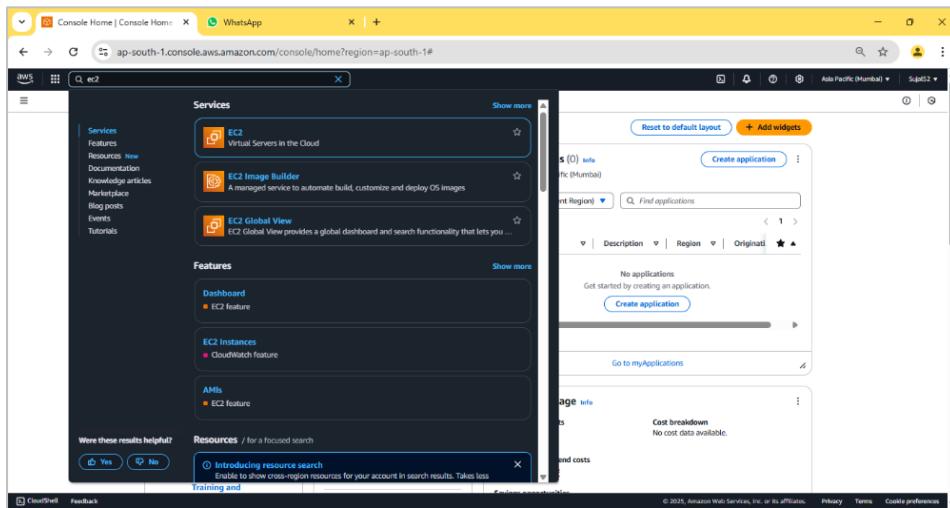


ASSIGNMENT 11:

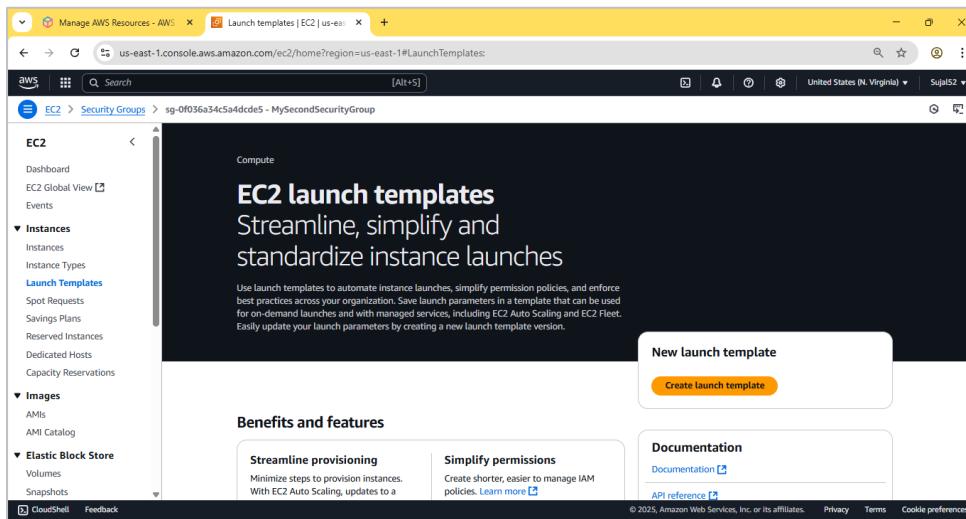
Problem Definition: Build scaling plans in AWS that balance load on different EC2 instances.

Instructions:

1. Access the **AWS console**, search for **EC2**, and select the top option from the search results.

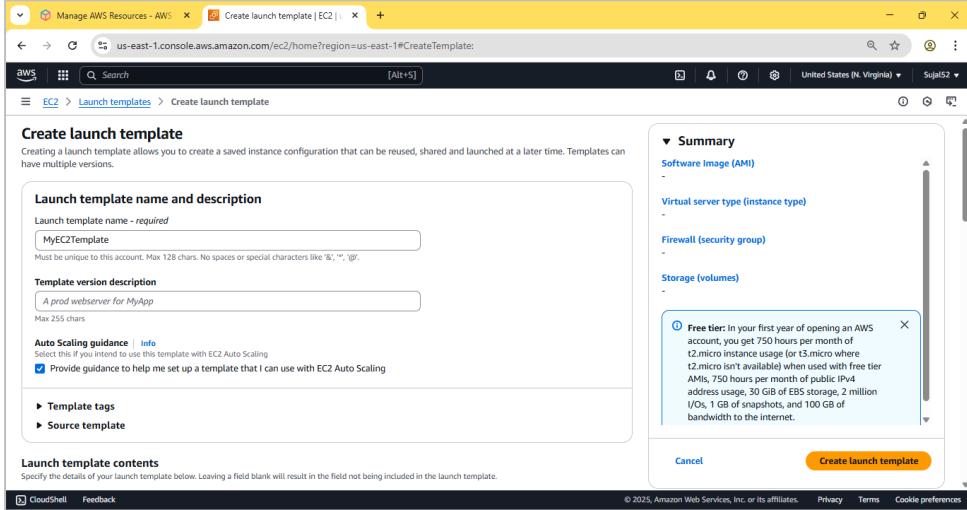


2. Click on “**Launch Templates**” from the sidebar then click on “**Create launch template**”.

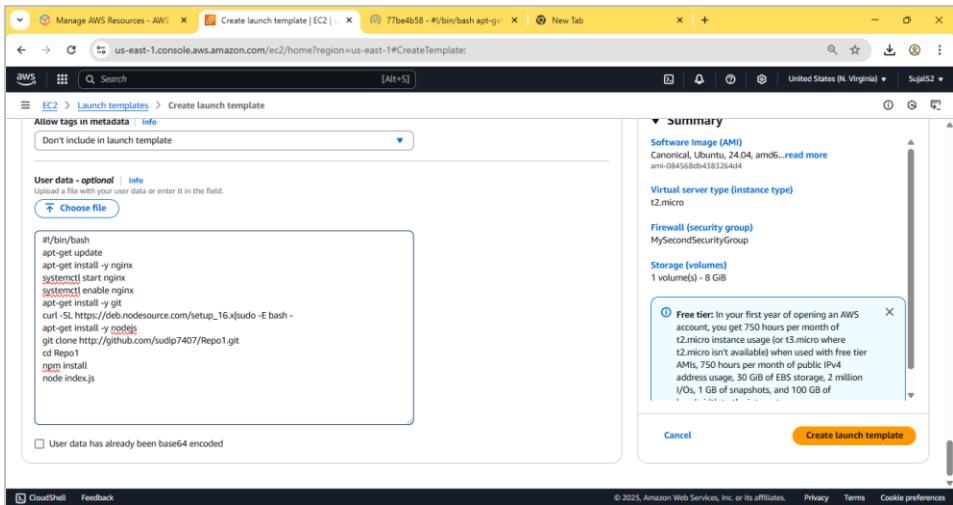


3. Enter name and description. Check the box under “**Auto Scaling**”

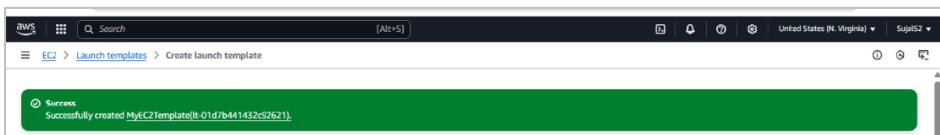
guidance". Select *Ubuntu* as OS Image, instance type as *t2.micro*. Create keypair and select an existing security group where port 4000 is open for incoming traffic.



Under “**Advance details**”, scroll down to “**User data**” and add the following bash script and then click on create instance

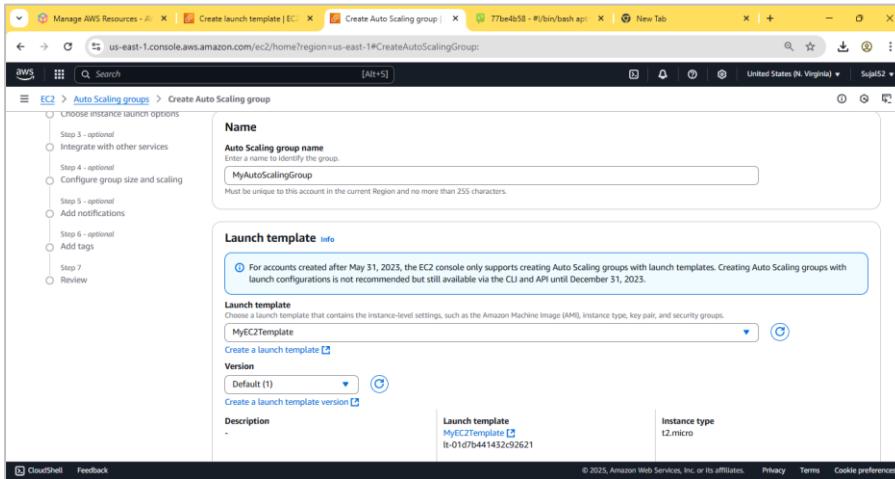


Our launch template is created successfully.

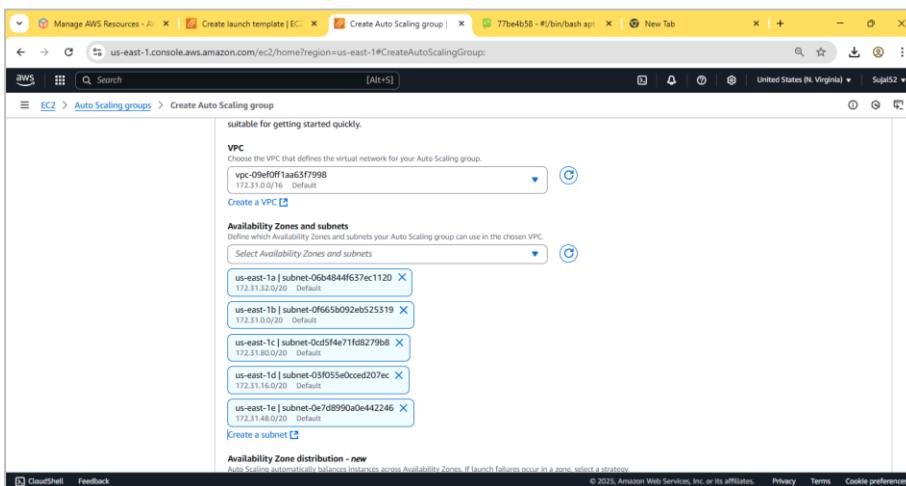


4. From sidebar, scroll down and select “**Auto Scaling Groups**”. Enter

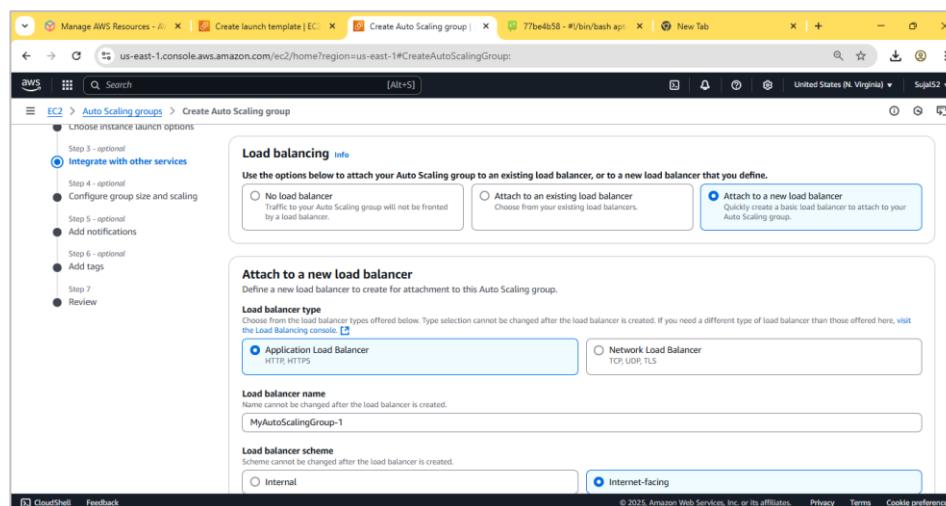
name and select our previously created launch template from the dropdown. Then click on next.



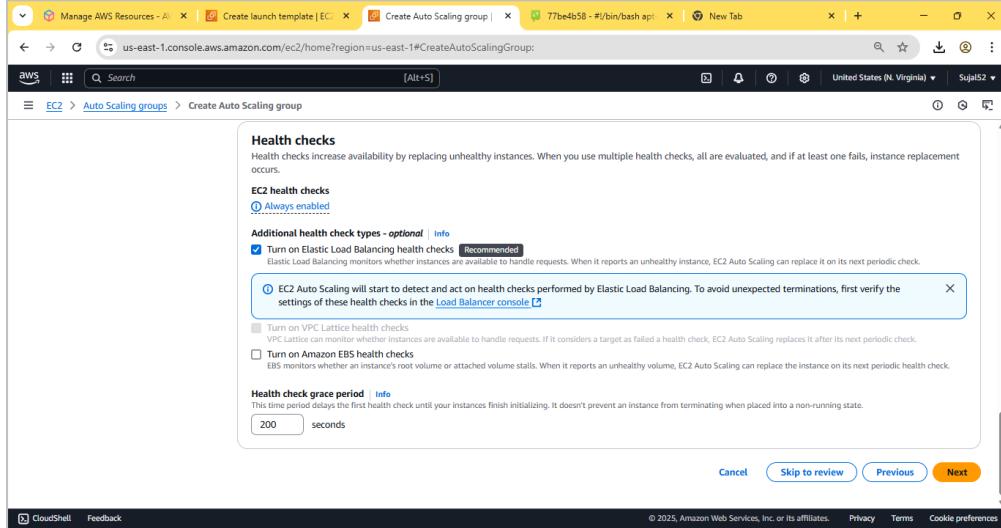
Choose all “**Availability Zones**” available then click on next.



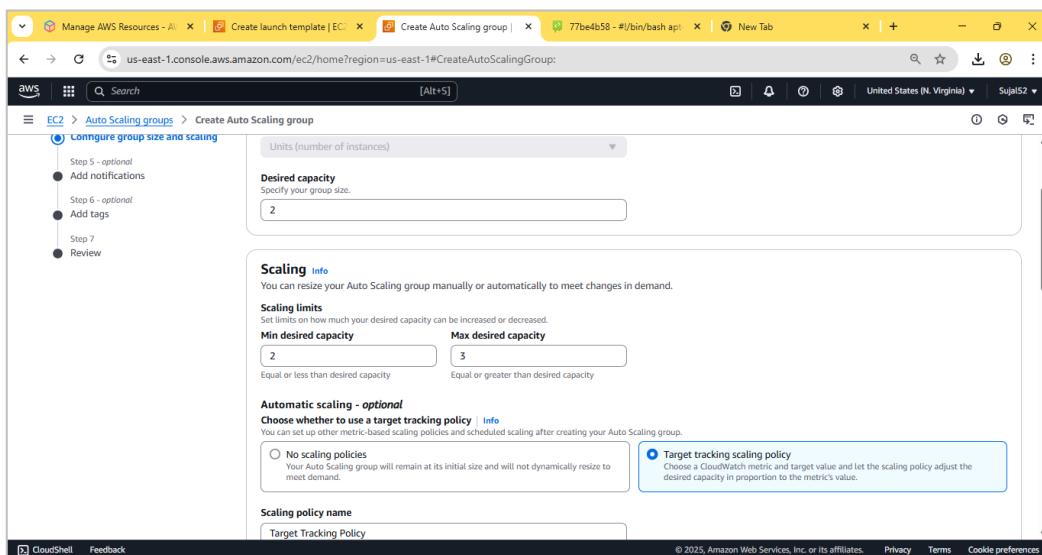
Under “**Load balancing**”, select “**Attach a new load balancer**”, under type select “**Application load balancer**” and “**Internet-facing**”



Under “**Health checks**” tick the box on “**Turn on Elastic Load Balancing Health Check**” and enter the “**Health check grace period**” as 240 seconds. Click on next.



In **Group size** set **Desired capacity** as 2. In **Scaling** set **Min desired capacity** as 2 and **Max desired capacity** as 3. In **Automatic scaling** choose **target scaling policy** and set the **instance warmup** as 240 second. Keeping everything else as default, click on Next.



Skip last two steps and click on **Create Auto Scaling group**. Our auto scaling group is successfully created.

Auto Scaling groups (1) <small>Info</small>		<small>C</small> Launch configurations	Launch templates	Actions	<small>C</small> Create Auto Scaling group
<small>Search your Auto Scaling groups</small>					
<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity
<input type="checkbox"/>	MyAutoScalingGroup	MyEC2Template Version Default	0	Updating capacity...	2
					2 3
					us-east-1a, us-east-1d...

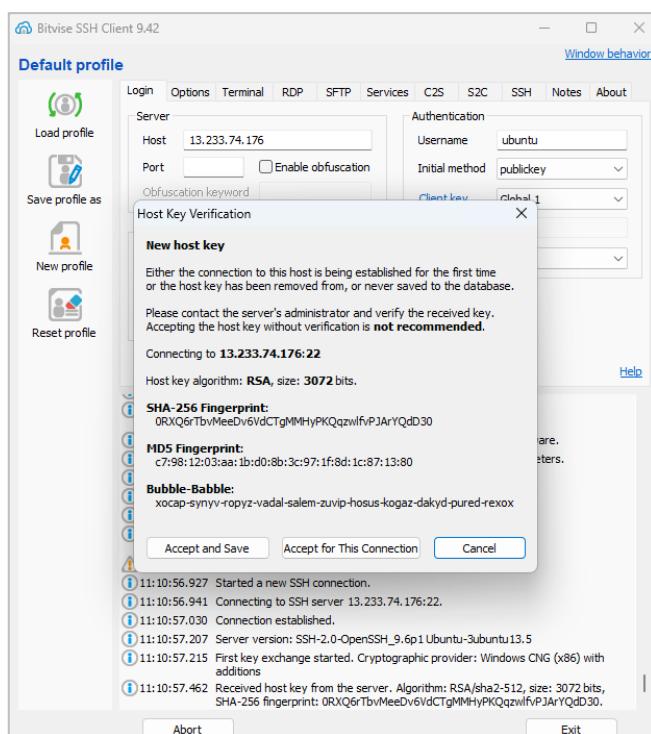
5. Now go to **Instances** from the sidebar. We can see two instances are created. Copy the **IPv4 address** of any of the two instance and log into **Bitwise SSH Client** using key pair.

Instances (2) Info		Last updated less than a minute ago	Connect	Instance state	Actions	Launch instances
			All states			
<input type="checkbox"/>	Name					
<input type="checkbox"/>	i-05a8ca1c17e0d6bed	Running	View details	t2.micro	Initializing	View alarms
<input type="checkbox"/>	i-08f2e75400ee1b3da	Running	View details	t2.micro	Initializing	View alarms

Instance summary for i-05a8ca1c17e0d6bed [Info](#)

Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-05a8ca1c17e0d6bed	54.89.81.166 [open address]	172.31.35.150



6. Open terminal console and enter the following commands:

```
vim file.sh
```

Now paste the following code:

```
#!/bin/bash

while(true)

do

echo "Inside Loop"
```

done

Now make the script executable and run it enter the following commands:

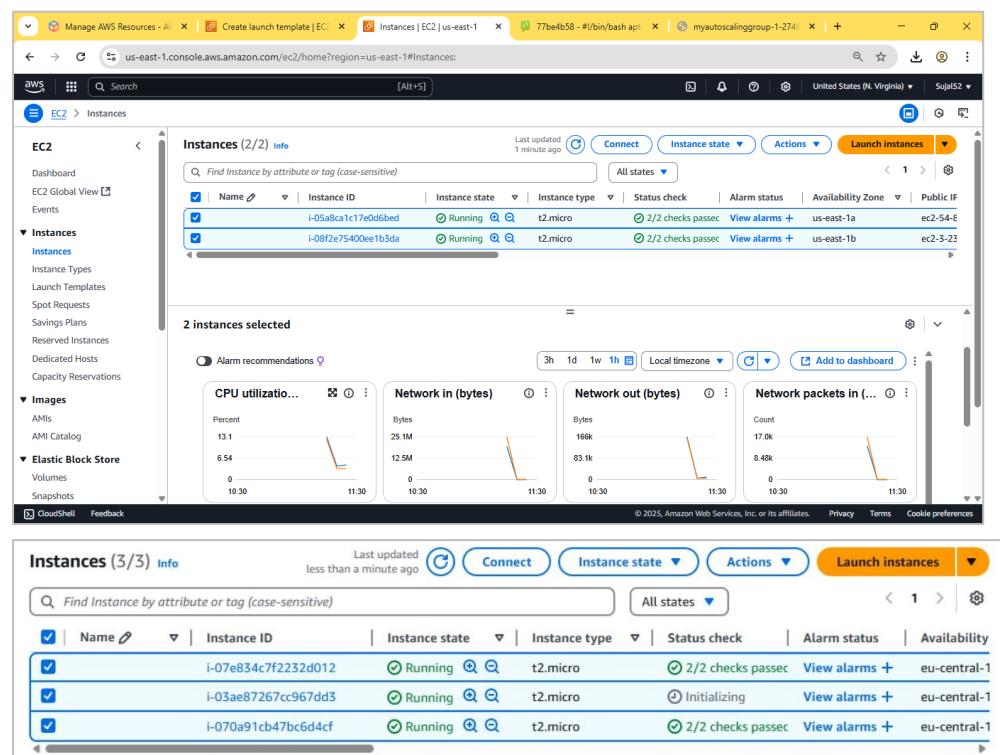
```
sudo chmod 777 file.sh
```

sh file.sh

```
ubuntu@ip-172-31-35-150:~$ sudo nano file.sh  
ubuntu@ip-172-31-35-150:~$ sudo chmod +x file.sh  
ubuntu@ip-172-31-35-150:~$
```

An infinite loop will start to run, stressing the cpu of the instance.

7. Select the two instances to see their utilization graph. After a while, when it surpasses 50% threshold, we can see that auto scaler spins up another instance to balance the load.

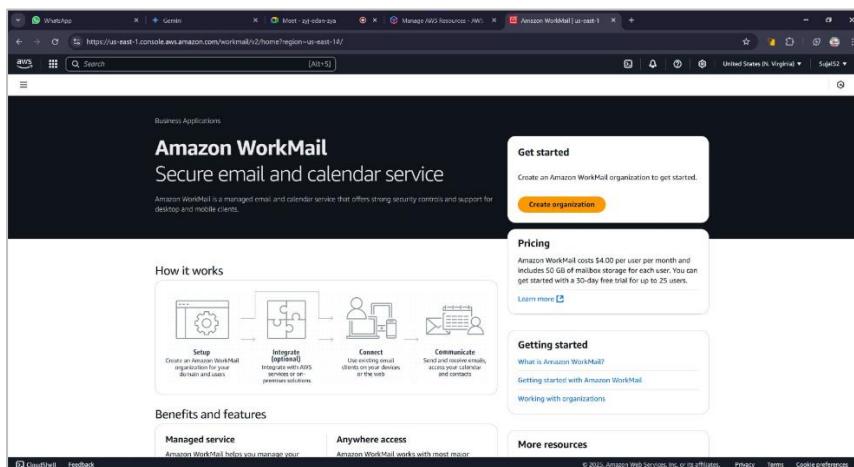


ASSIGNMENT 13:

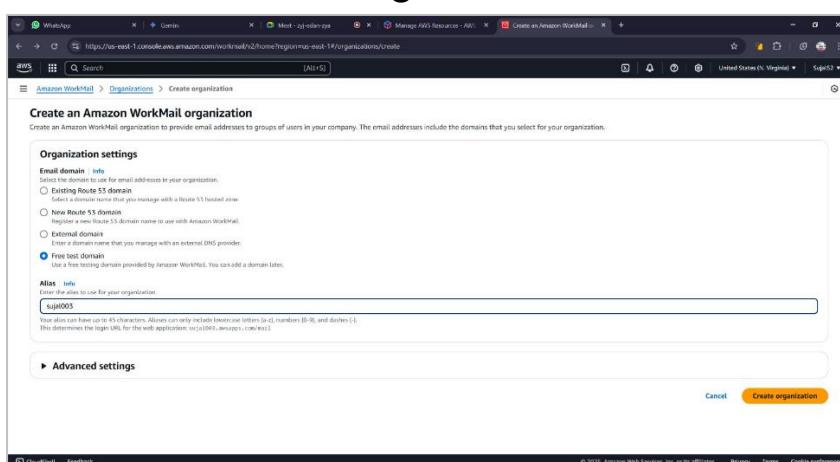
Problem Definition: Create a workmail for your organization.

Instructions:

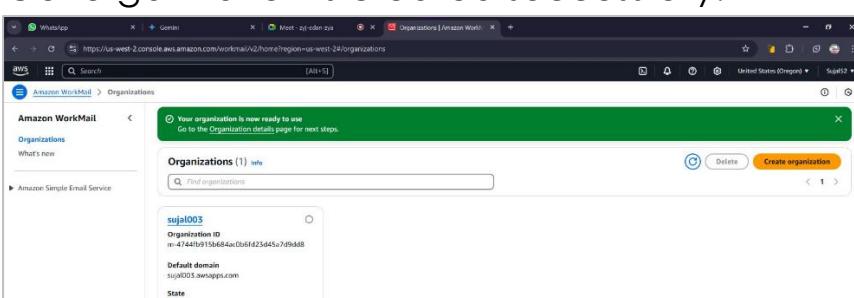
1. Access the **AWS console**, search for **Workmail**, and select the top option from the search results. Click on **Create Organization**.



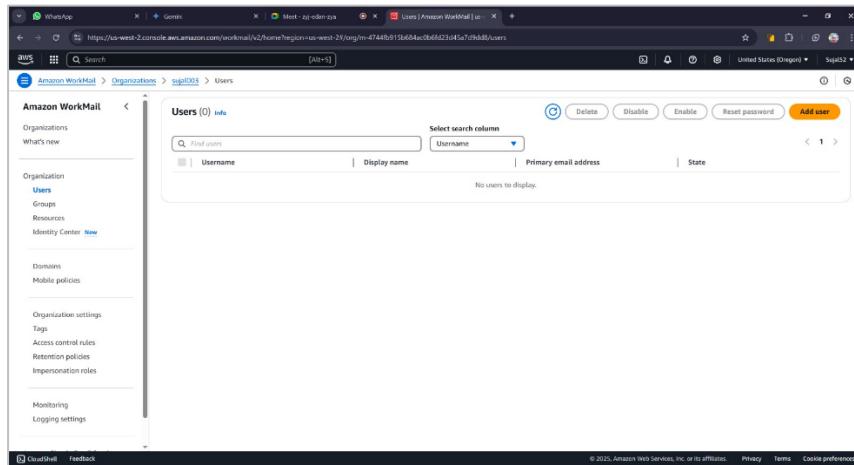
2. Select **Free Test Domain** under **Organization settings**. Enter an alias then click on **Create Organization**.



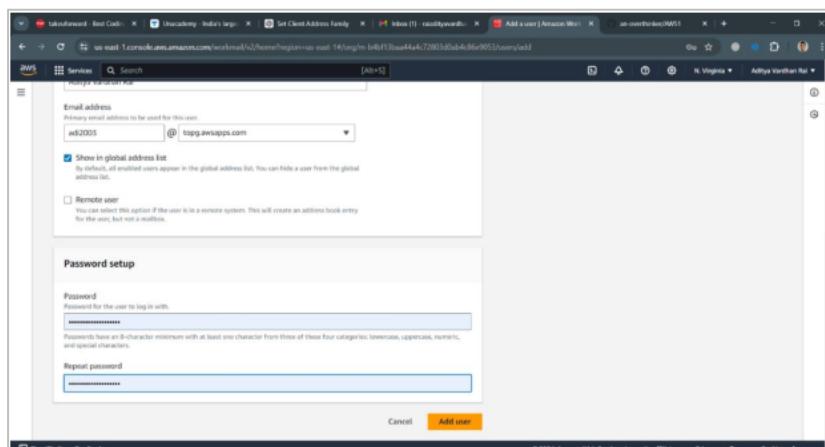
Our organization is created successfully.



3. Click on Organisation name. Go to **Users** and click on **Add User** button.



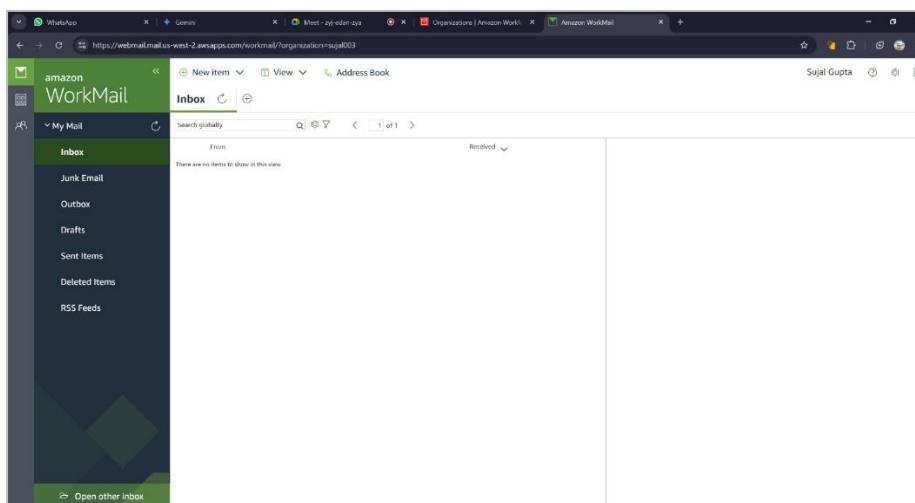
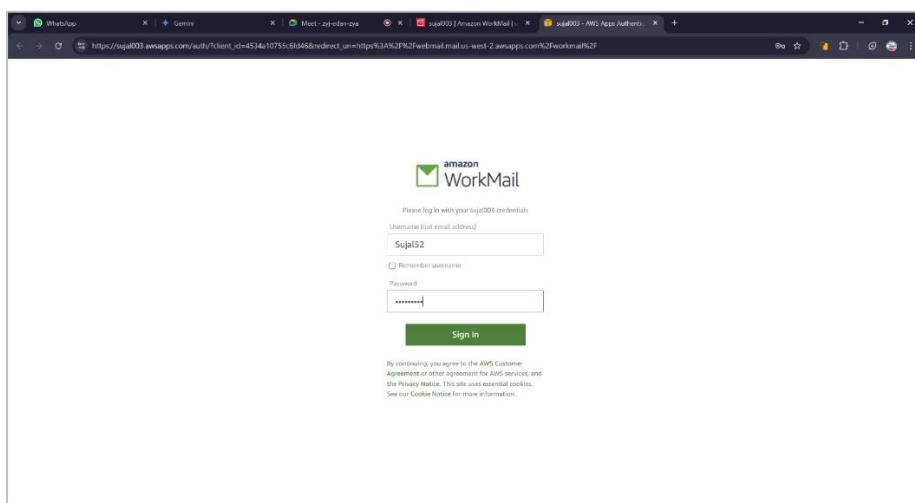
4. Fill username, display name and password. Then click on **Add User**.
A green banner will be displayed indicating our user is created successfully.



5. Go to Organisation Details page again and click on the **Amazon WorkMail web application**.

The screenshot shows the AWS Amazon WorkMail organization creation interface. On the left, a sidebar lists various organization settings like Users, Groups, Resources, Identity Center, Domains, and Monitoring. The main area displays the 'User setup guide' and 'Organization details'. The 'Organization details' section includes fields for Organization ID (m-4744b9151684ac061c23485779d8B), State (Active), Date created (April 19, 2025 at 11:27 (UTC+5:30)), Directory type (WorkMail directory), and Directory ID (d-24f7b9a452). A note at the top says, "You are using the test domain as your default domain. We recommend that you add a custom domain and set it as the default domain." Buttons for 'Find email address' and 'Delete organization' are also present.

6. Enter our previously created username and password. The inbox should open indicating our workmail is created successfully.

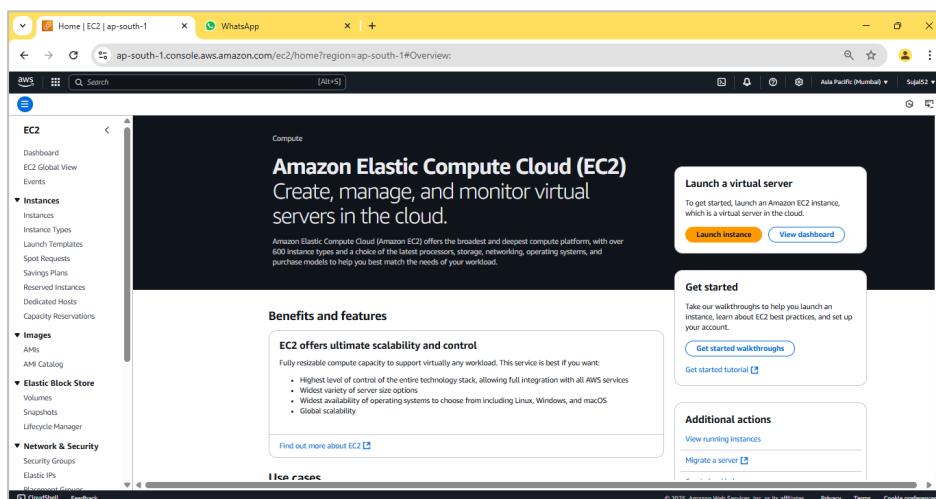


ASSIGNMENT 14:

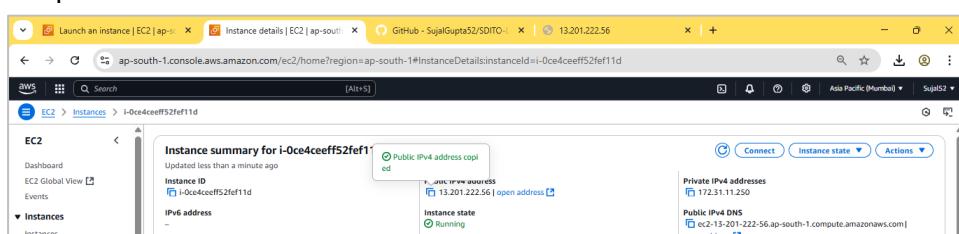
Problem Definition: Create an Elastic IP for an Instance.

Instructions:

1. Access the **AWS console**, search for **EC2**, and select the top option from the search results. Then go to **Instances** from the sidebar.



2. Launch a EC2 instance and copy its Public IPv4 address for future comparison.

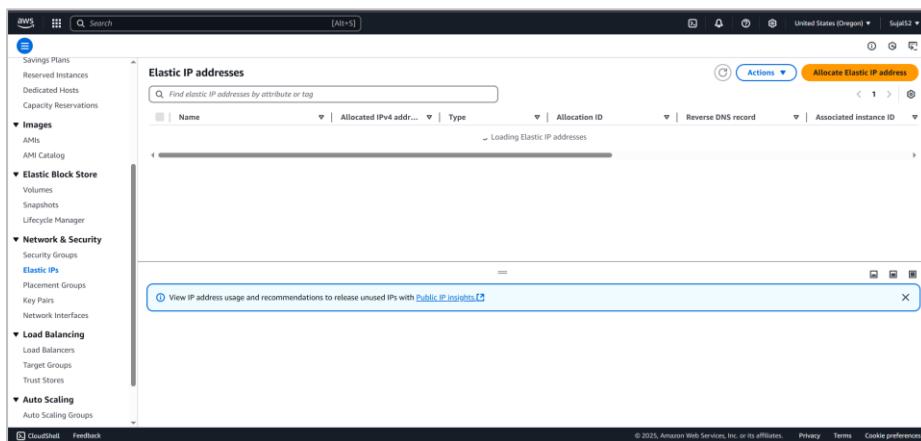


3. Now restart the instance and compare its Public IPv4 address with

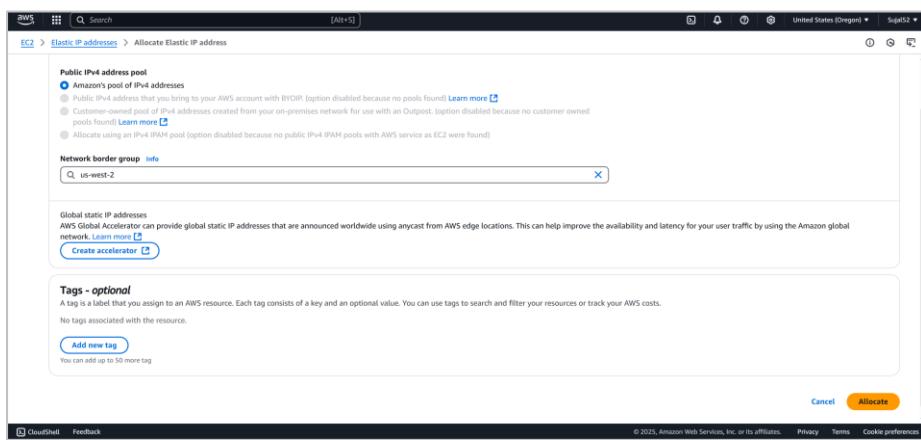
the one we previously saved. We can see it changes every time a instance is launched.



4. From the **EC2 Dashboard**, go to **Elastic IPs** and click on **Allocate Elastic IP**



5. Select **Amazon's pool of IPv4 Addresses** and click on **Allocate**. A green banner will be shown indicating our Elastic IP has been created.



6. Click on **Associate Elastic IP address**. Select resource type as **Instance**, select name of our instance, its private IP address and

then click on **Associate**.

The screenshot shows two windows side-by-side. On the left is the AWS Elastic IP address association dialog box. It displays the message "Elastic IP address: 52.205.18.204" and asks for the "Resource type". The "Instance" option is selected. A warning message states: "If you associate an Elastic IP address with an instance that already has an Elastic IP address associated, the previously associated Elastic IP address will be disassociated, but the address will still be allocated to your account. Learn more." Below this, it says "If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address." The "Instance" field contains "i-0f9bc45320b17c05b" and the "Private IP address" field contains "172.31.22.249". Under "Reassociation", there is a checked checkbox "Allow this Elastic IP address to be reassigned". At the bottom are "Cancel" and "Associate" buttons. On the right is a CloudShell terminal window titled "CloudShell - No saved". It shows the command "aws ec2 associate-address --instance-id i-0f9bc45320b17c05b --public-ip 52.205.18.204" being run. The output shows the addresses before and after assignment:

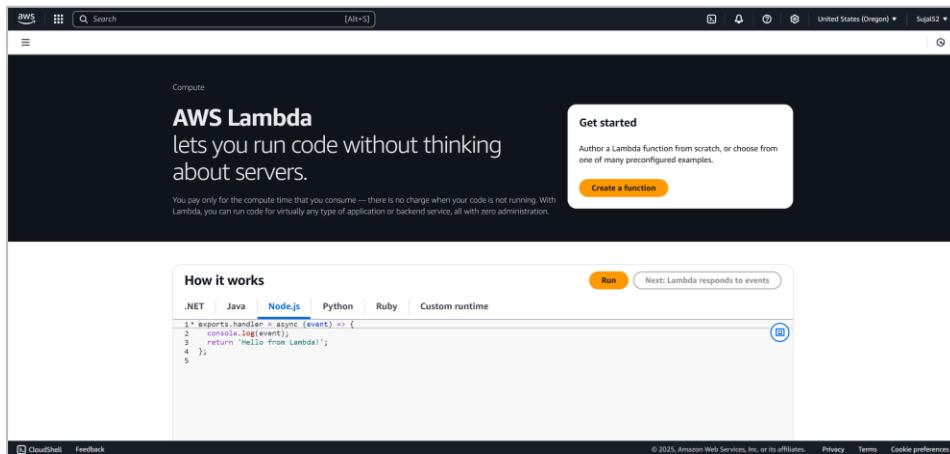
```
File Edit Format View Help
Before assigning Elastic IP:-  
3.89.245.135  
34.228.15.254  
  
After assigning Elastic IP:-  
52.205.18.204  
52.205.18.204
```

ASSIGNMENT 15:

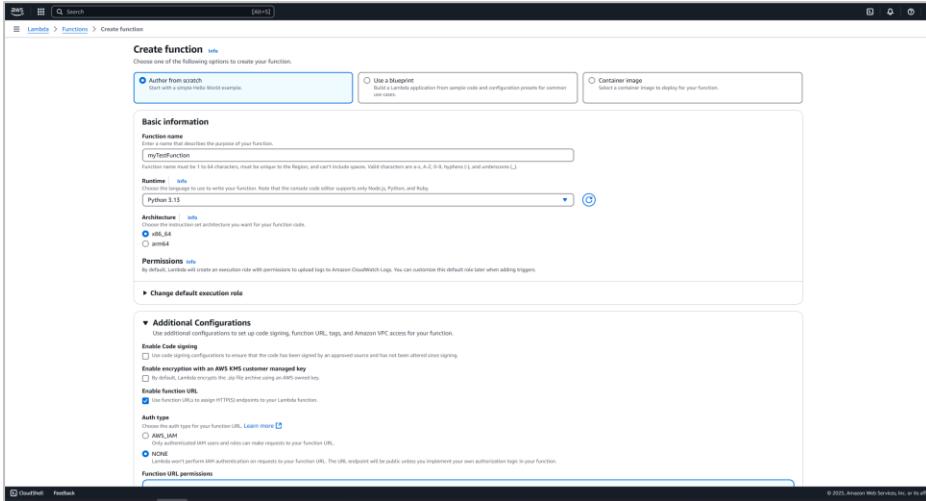
Problem Definition: Create a serverless computing service

Instructions:

1. Access the **AWS console**, search for **Lambda**, and select the top option from the search results.



2. Click on **Create a function**. Enter name and select **Python** as **Runtime**. Under **Advanced Configuration**, check **Enable function URL** and select **Auth type** as **None**. Click on **Create Function**.



3. Our function is created successfully. Now copy the function URL and open it in a new tab.

Successfully created the function myTestFunction. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

myTestFunction

Function overview

Description

Last modified 31 seconds ago

Function ARN arn:aws:lambda:us-west-2:577638390776:function:myTestFunction

Function URL https://cy6i7ae6ownrhymquikawmg0aheeo.lambda-url.us-west-2.on.aws

Code

```
myTestFunction
Hello from Lambda!
```

Code source

Upload from

CloudShell Feedback

https://cy6i7ae6ownrhymquikawmg0aheeo.lambda-url.us-west-2.on.aws

MPMC Job Search ML YouTube Tomato Timer - On!... Exceldium | Hand... Dashboard | The Od... Using Git in the Re... MEGA | 0%