

Q4: Configure a Webserver on 'Regno_EC2_VM2' Instance and host your organizations website (Static Website) and provide access only to your machine.

Solution:

Step 1: Navigate to S3 services

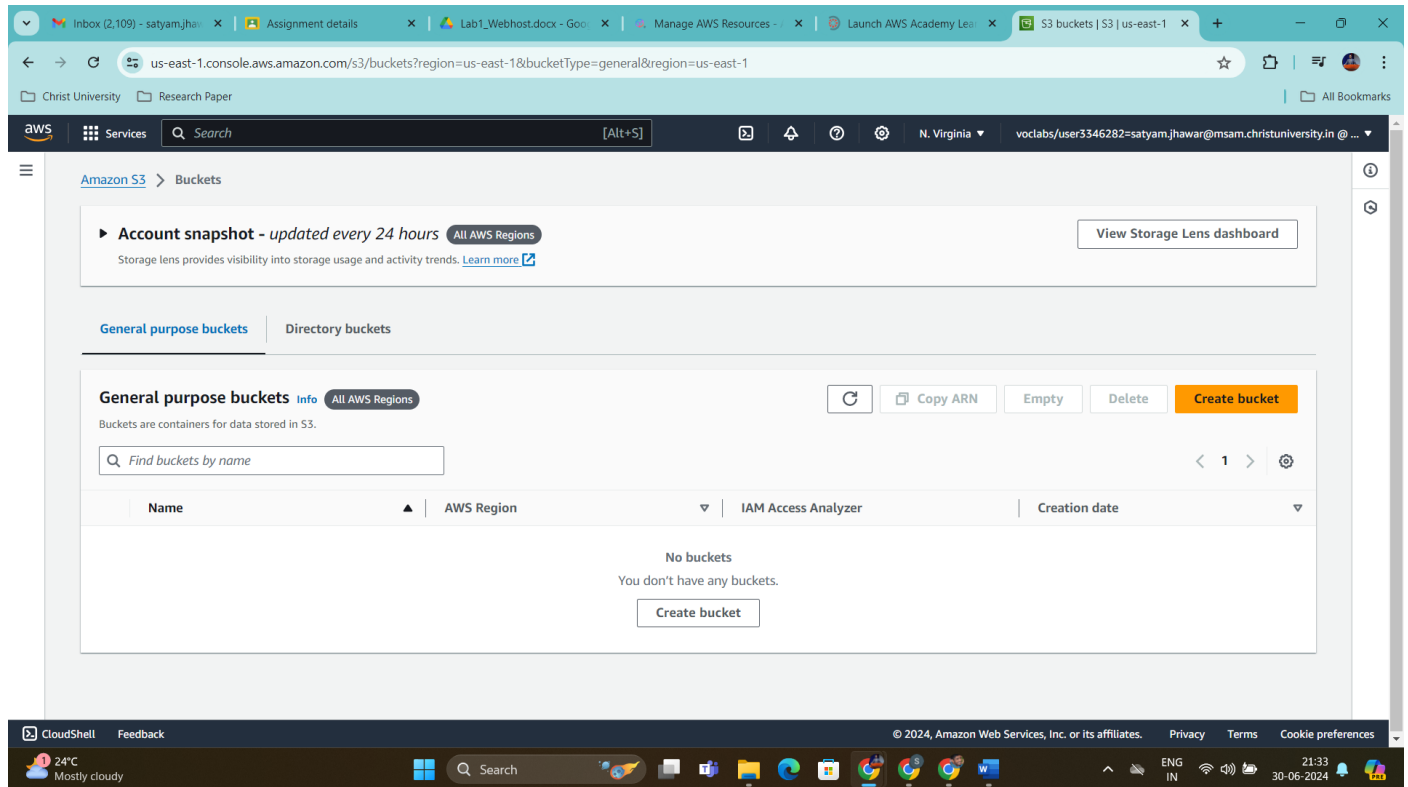


Figure 1: S3 Dashboard

Step 2: Click on Create Bucket

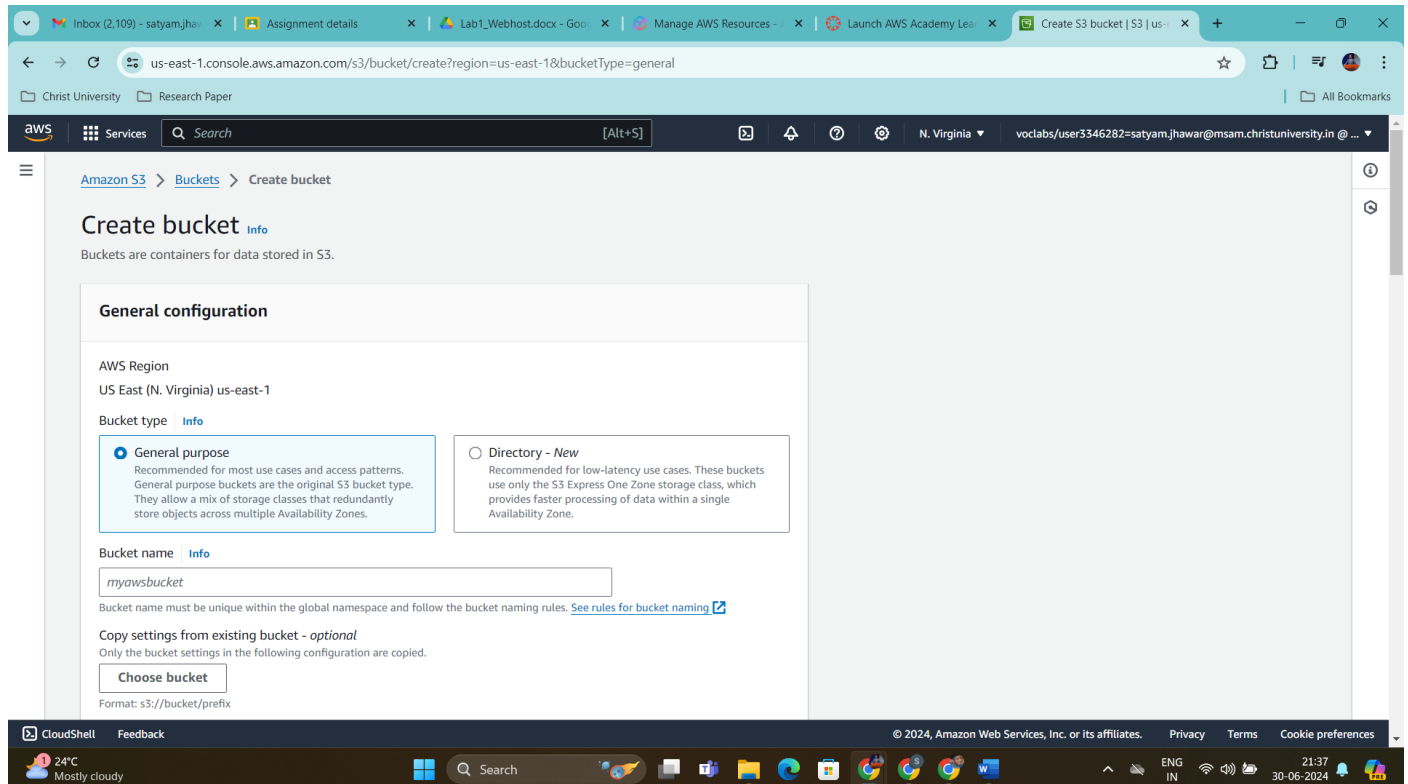


Figure 2: Create Bucket Dashboard Appears

Step 3: Configure the bucket

3.1) Add a name to the bucket (2348554s3)

Bucket name [Info](#)

2348554s3

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#) [↗](#)

Figure 3: Bucket Name

3.2) Object Ownership – Leave it as default only

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 enforced

Figure 4: Object Ownership

3.3) Disable Block all public access checkbox and acknowledge the warning message

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.

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

Figure 5: Block all public access

3.4) Disable Bucket Versioning

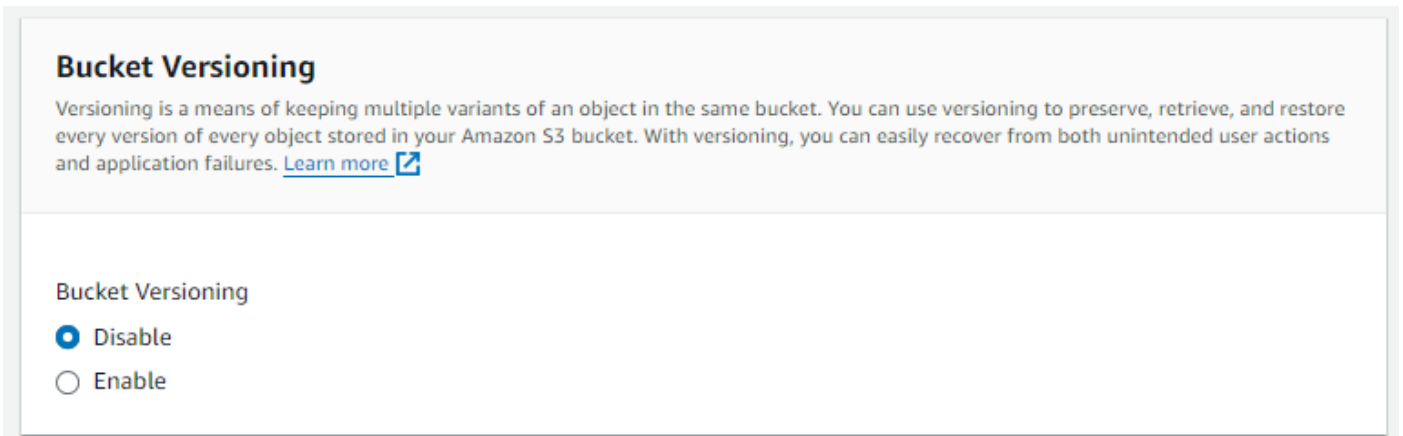


Figure 6: Bucket Versioning

3.5) Disable Default Encryption

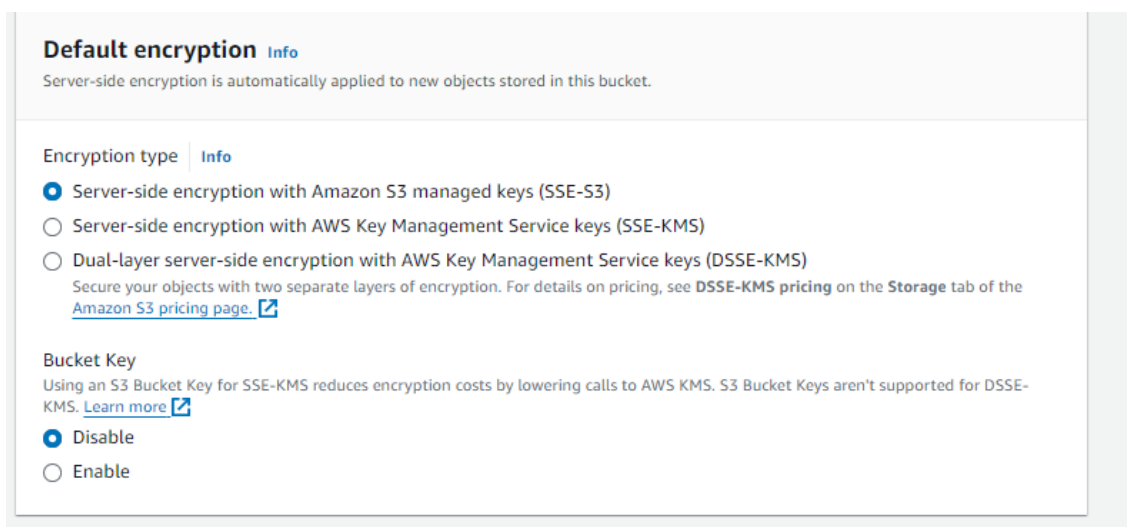


Figure 7: Bucket Encryption

3.6) Click on Create Bucket

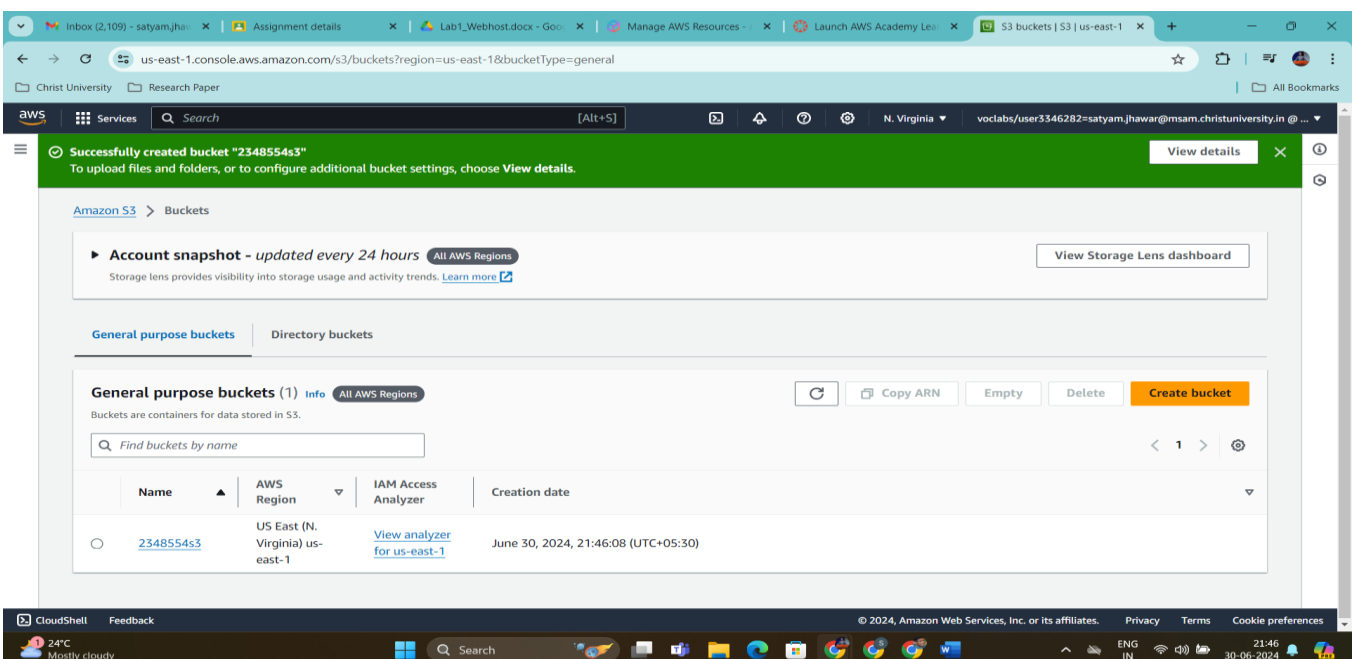


Figure 8: Bucket Creation Complete

Step 4: Click on the bucket name “2348554s3”

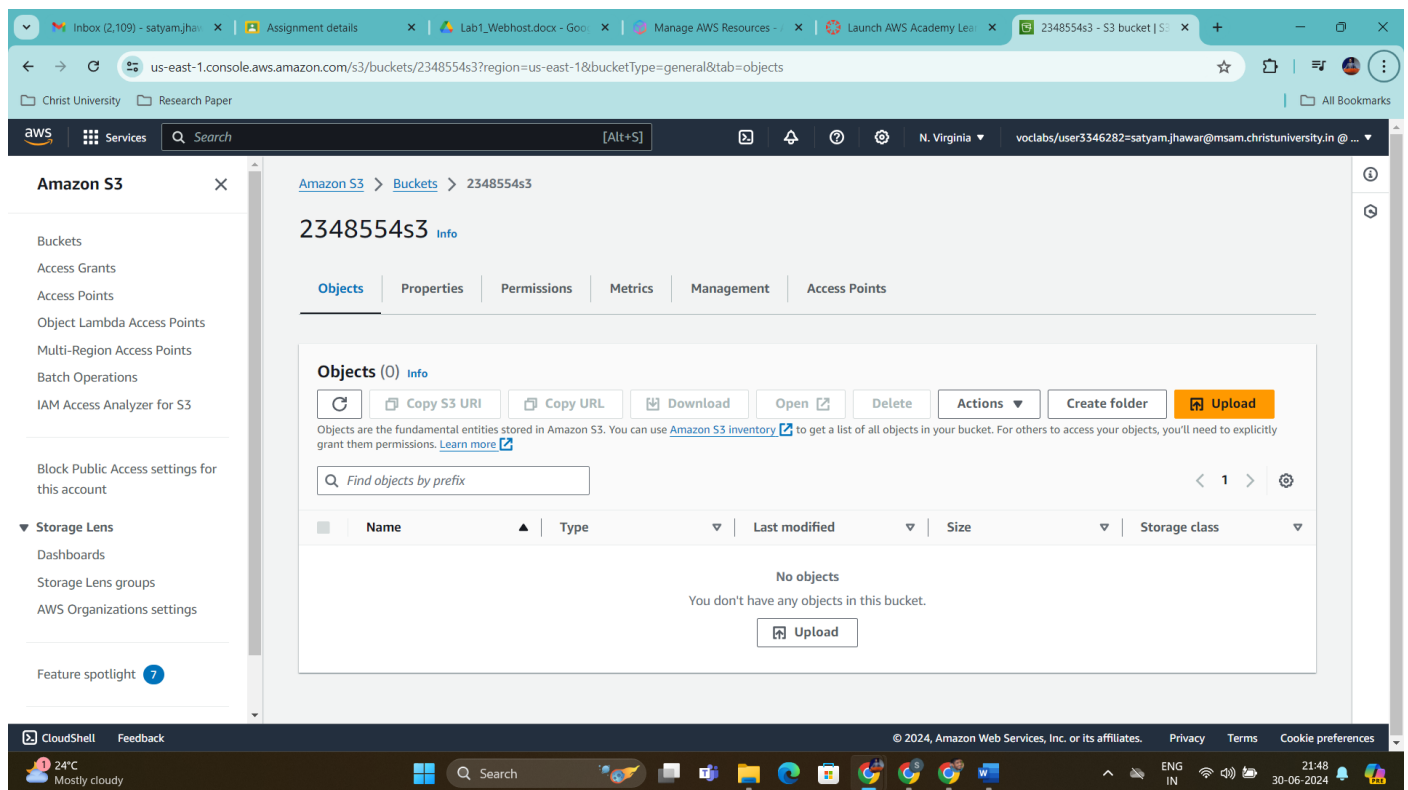


Figure 9: Within the bucket of 2348554s3

Step 5: Uploading the zip file of the static website

5.1) Click on Upload

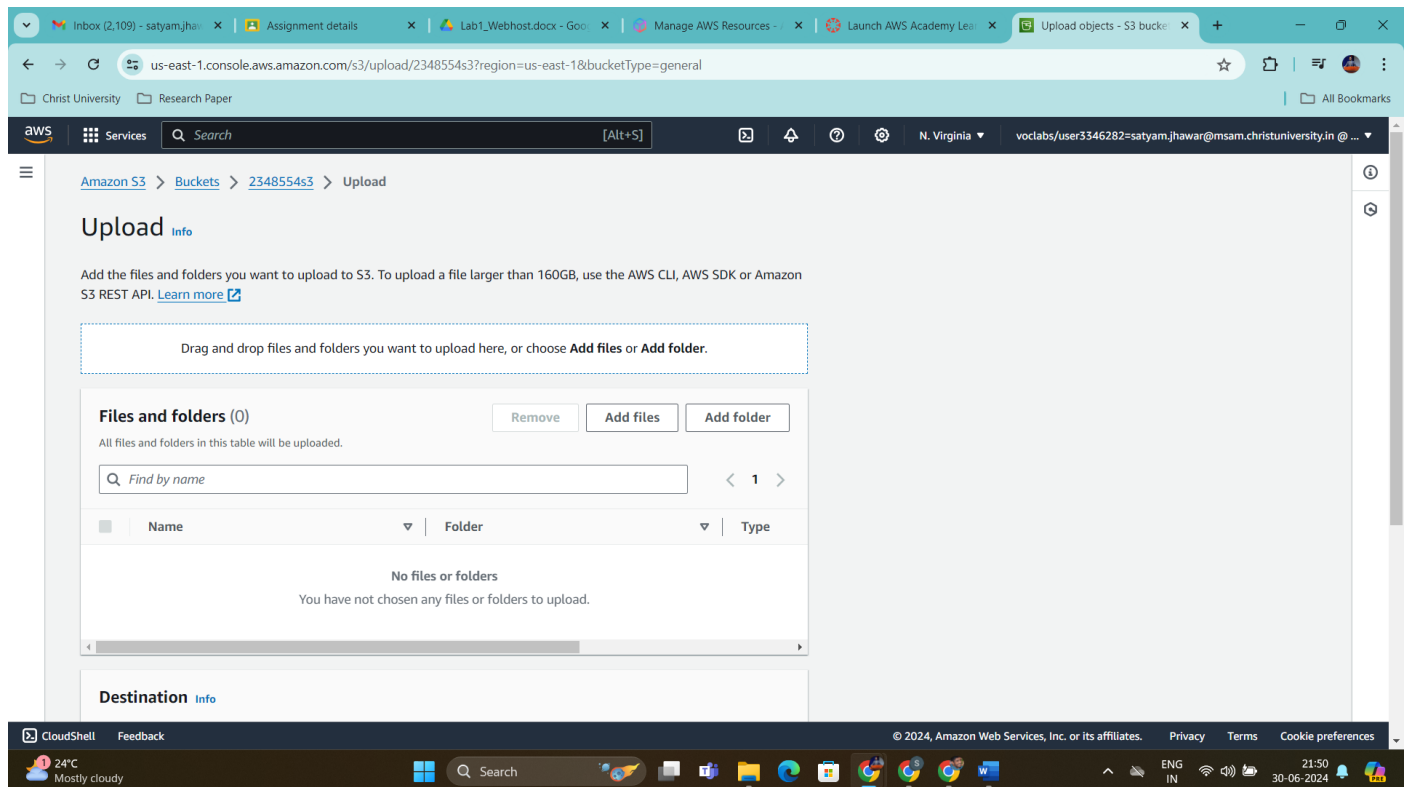
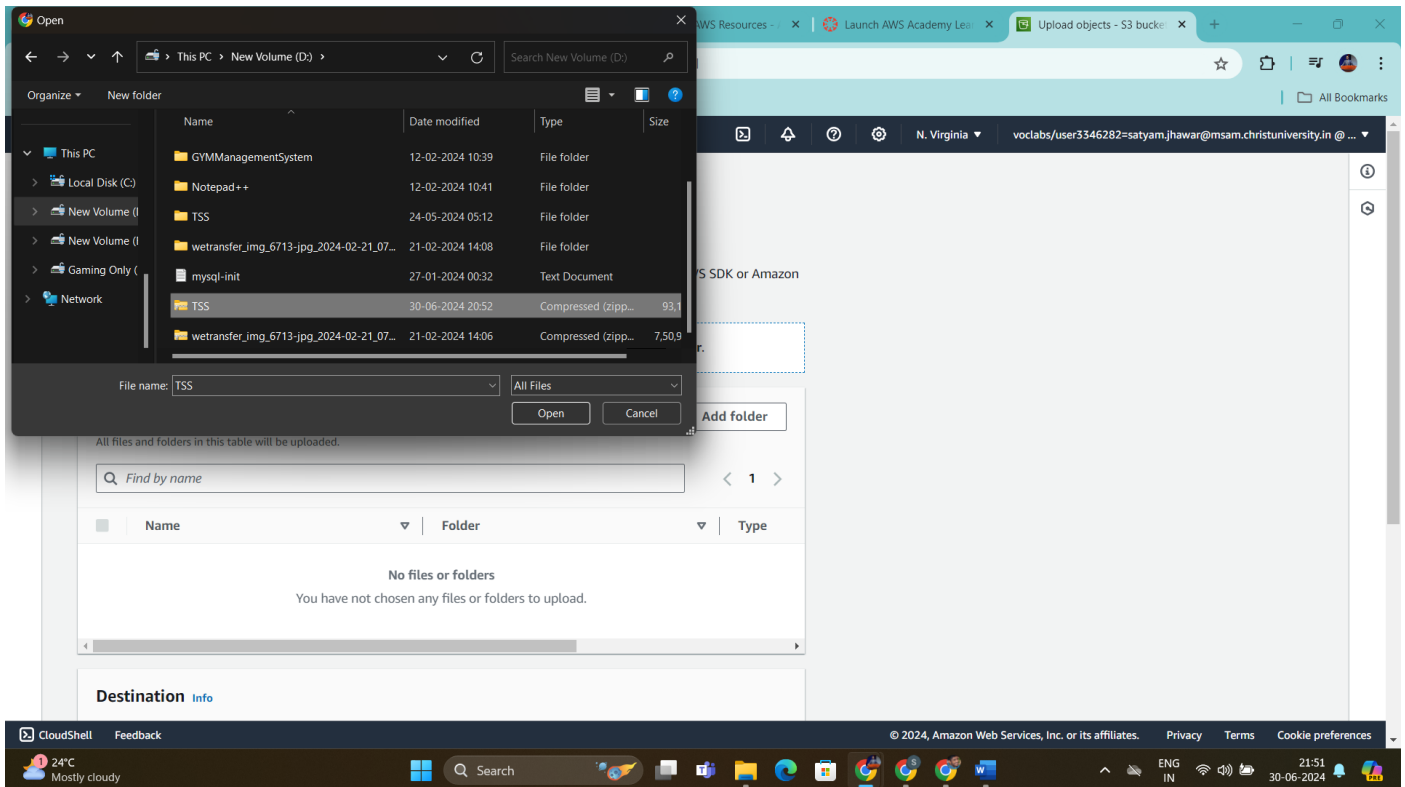


Figure 10: Upload Window Pane

5.2) Click on Add Files and navigate to the location of the static website and click on Open



5.3) Click on Upload

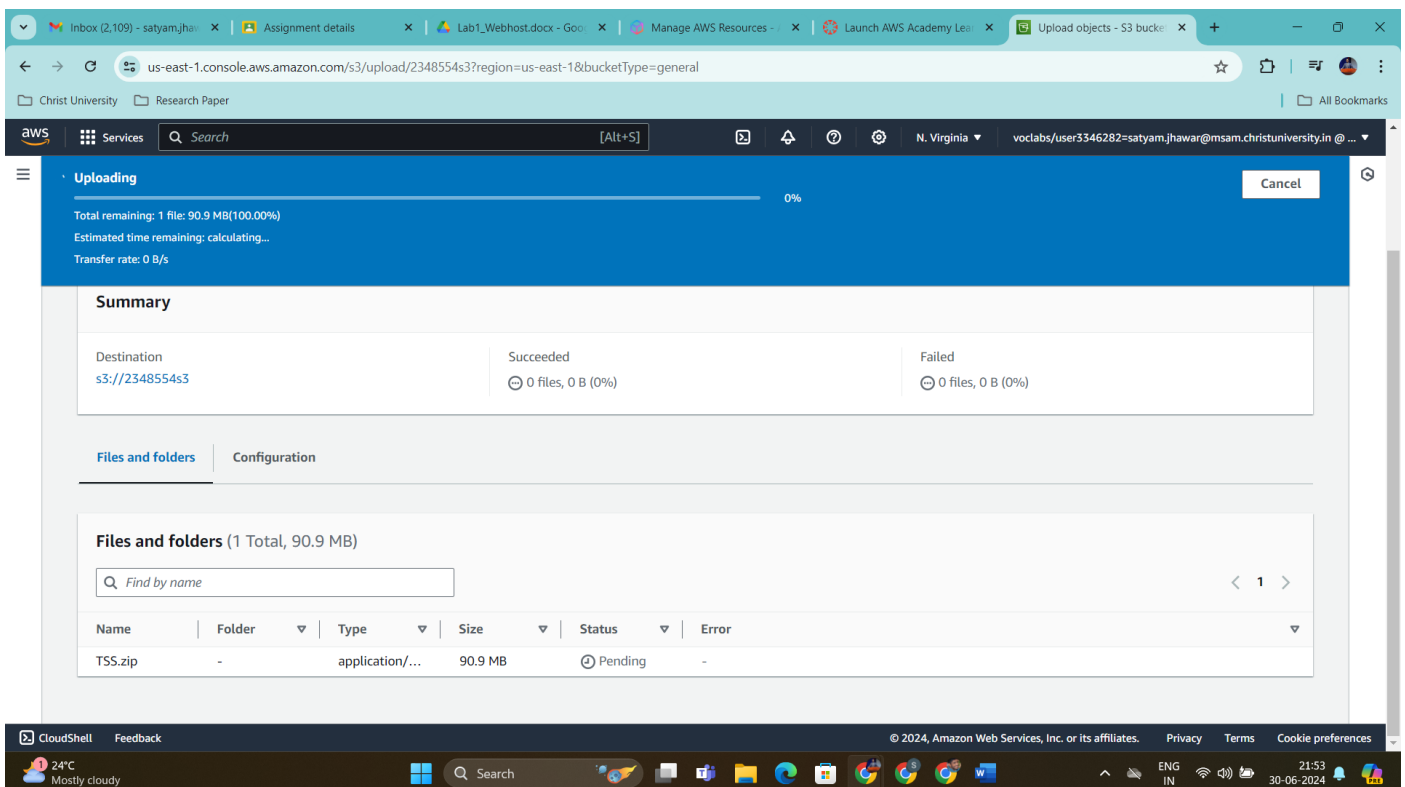
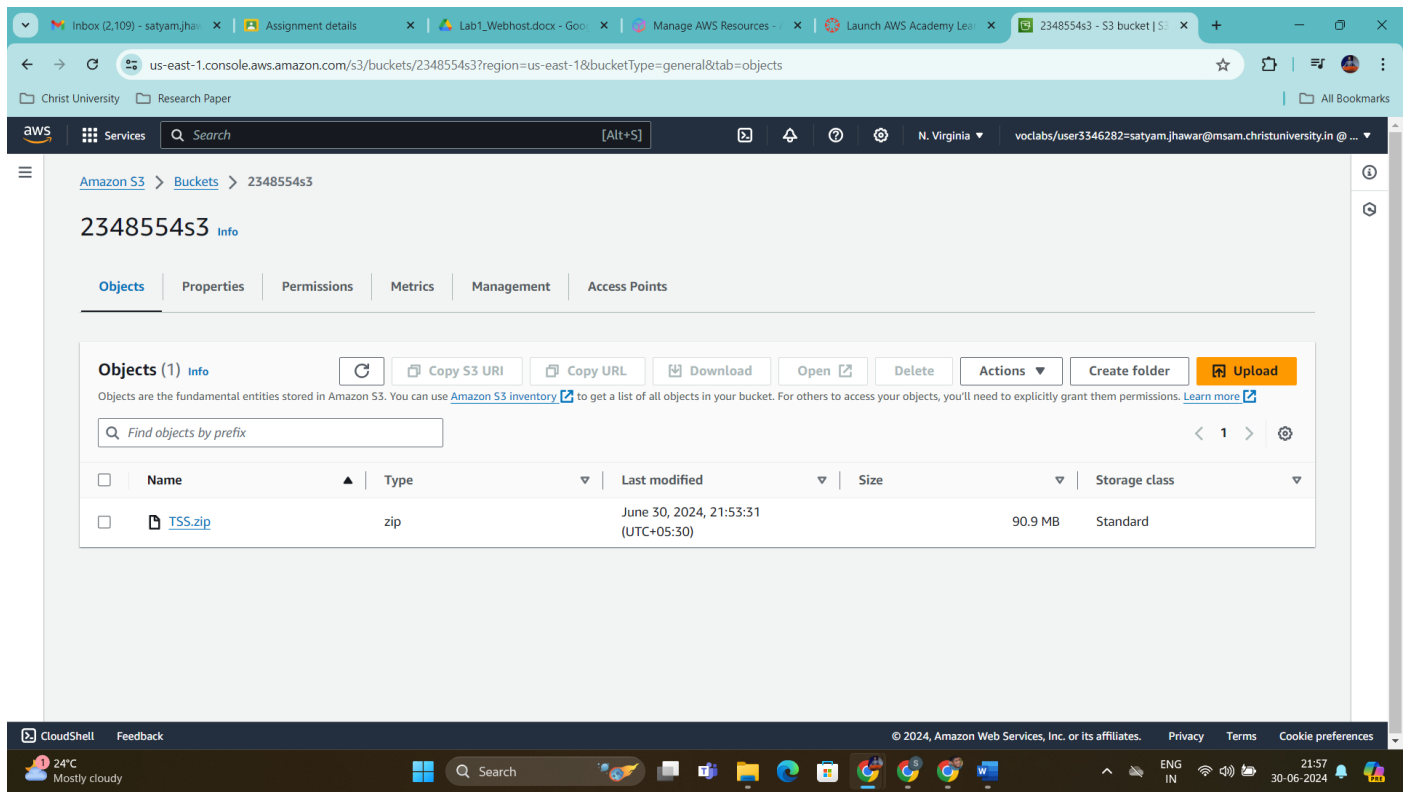


Figure 11: Confirmation of the Upload file awaited

Wait until the upload is completed

Once the Upload is completed. Click on Close and we will find the uploaded file within our bucket.



Step 6: Click on the Permissions Tab and navigate to edit bucket policy

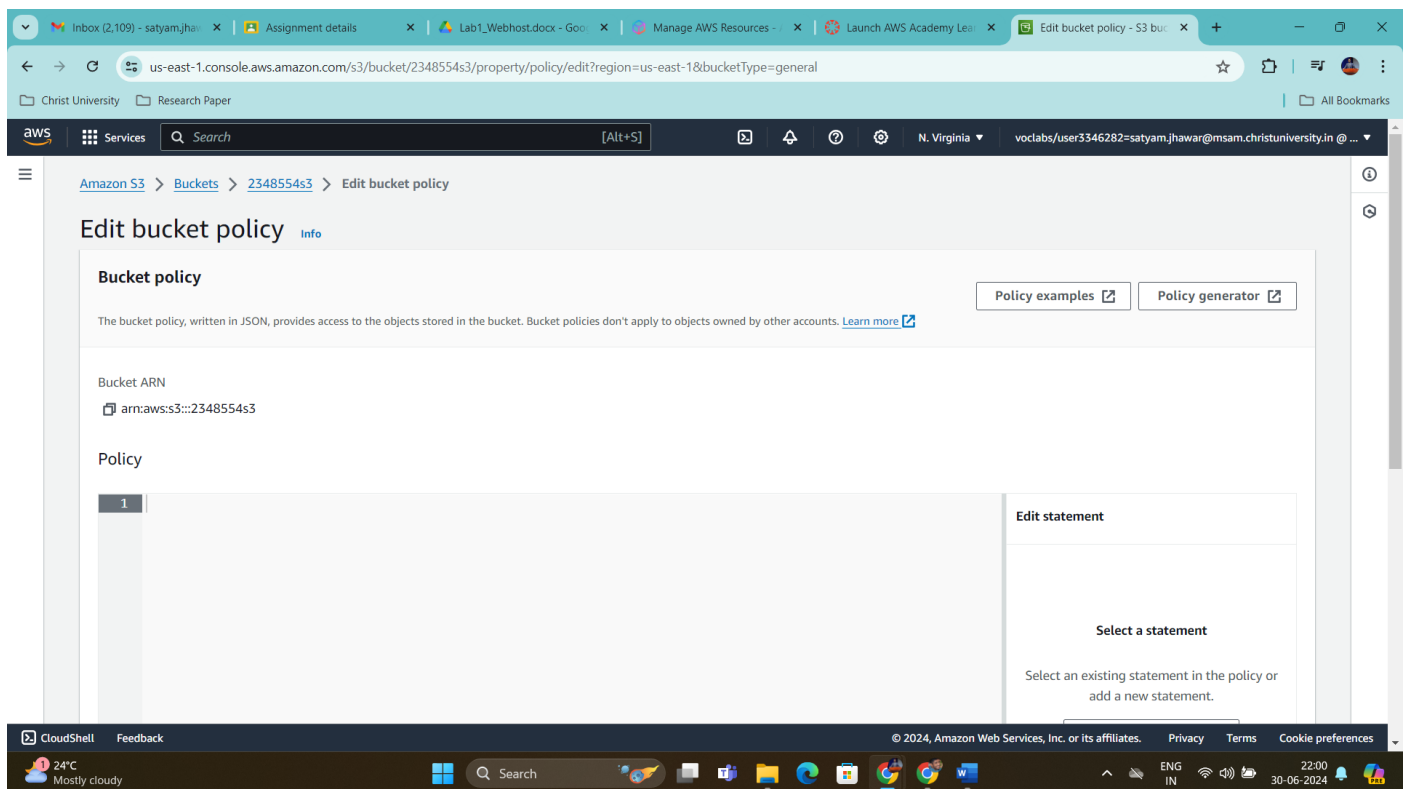
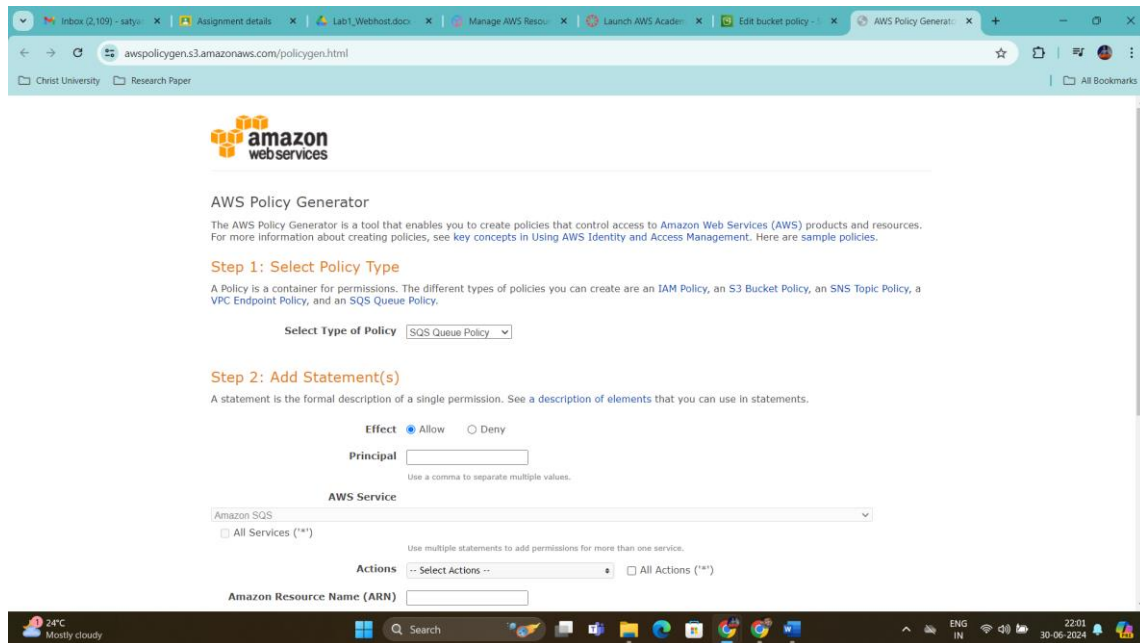


Figure 12: Bucket Policy Window

Step 7: Click on Policy Generator



The screenshot shows the AWS Policy Generator interface. At the top, the Amazon Web Services logo is displayed. Below it, the title 'AWS Policy Generator' is followed by a brief description. The main section is titled 'Step 1: Select Policy Type'. It explains that a policy is a container for permissions and lists the different types of policies that can be created. A dropdown menu labeled 'Select Type of Policy' is currently set to 'SQS Queue Policy'. Below this, 'Step 2: Add Statement(s)' is visible, showing fields for 'Effect' (set to 'Allow'), 'Principal', 'AWS Service' (set to 'Amazon SQS'), 'Actions' (set to 'Select Actions'), and 'Amazon Resource Name (ARN)'. The interface is clean and professional, with a light blue and white color scheme.

Figure 13: Policy Generator Window

Step 8: Configure Policy Generator

8.1) Select Policy Type as “S3 Bucket Policy” from the drop down menu

8.2) Enter Principal Text Field as “*”

8.3) Within Actions select “Get Object” from the drop down menu

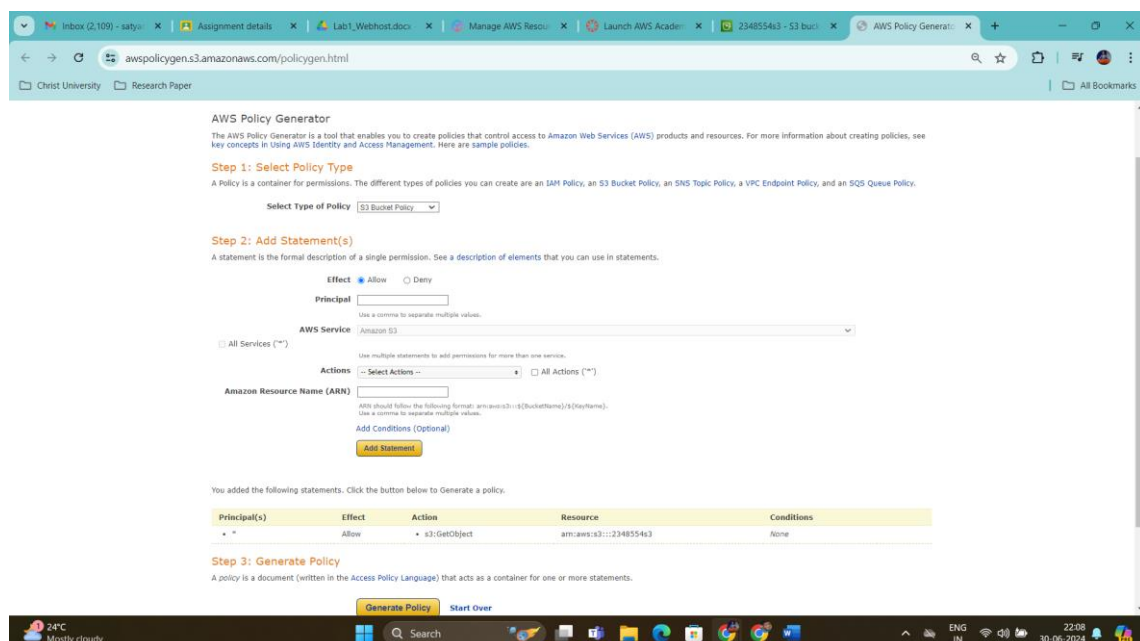
8.4) Enter the bucket ARN

8.4.1) Go back to the bucket

8.4.2) Click on Properties tab and copy the ARN of the bucket

8.4.3) Paste it in the text field

8.5) Click on Add Statement



The screenshot shows the AWS Policy Generator interface with 'Step 2: Add Statement(s)' selected. The 'Select Type of Policy' dropdown is now set to 'S3 Bucket Policy'. The 'Principal' field is set to '*'. The 'AWS Service' dropdown is set to 'Amazon S3'. The 'Actions' dropdown is set to 'Get Object'. The 'Amazon Resource Name (ARN)' field is set to 'arn:aws:s3:::23485543-bucket'. Below the form, there is a table showing the generated policy statement:

Principal(s)	Effect	Action	Resource	Conditions
*	Allow	s3:GetObject	arn:aws:s3:::23485543-bucket	None

Below the table, there is a 'Generate Policy' button and a 'Start Over' button. The interface is clean and professional, with a light blue and white color scheme.

Figure 14: Bucet Policy Configuration

Step 9: Click on Generate Policy

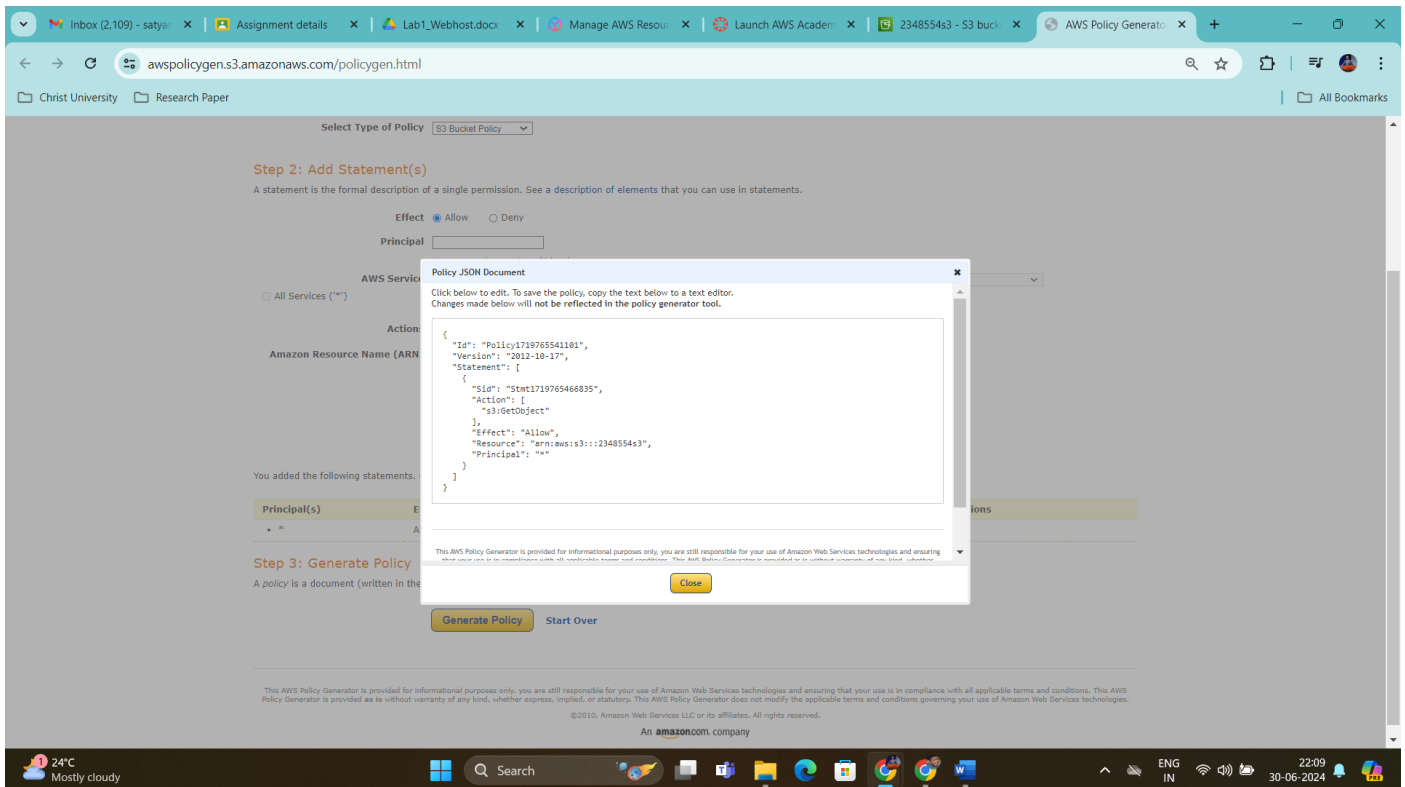


Figure 15: Policy Generated

Step 10: Copy the Policy and Paste it in the Edit Bucket Policy Page

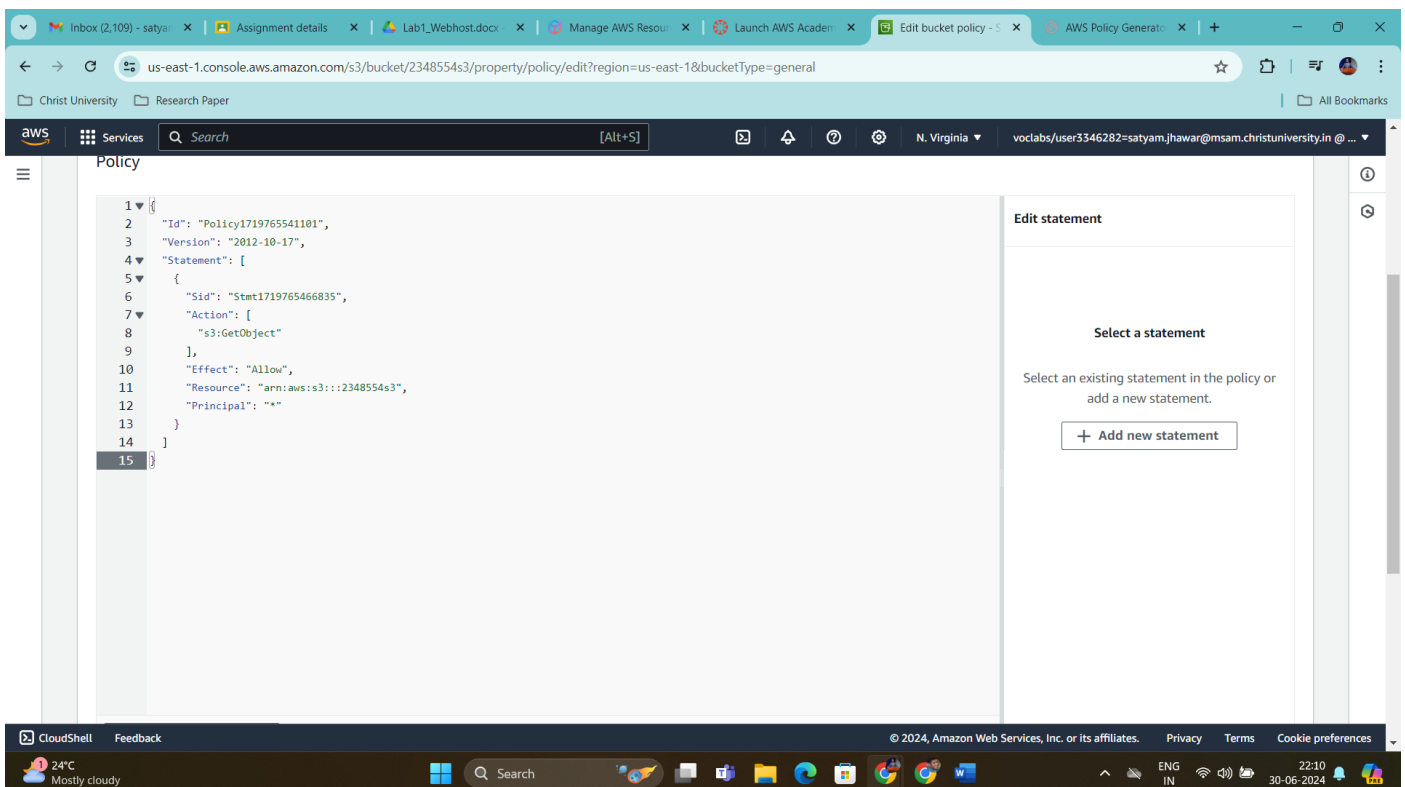


Figure 16: Policy Pasted

Step 11: Click on Save Changes

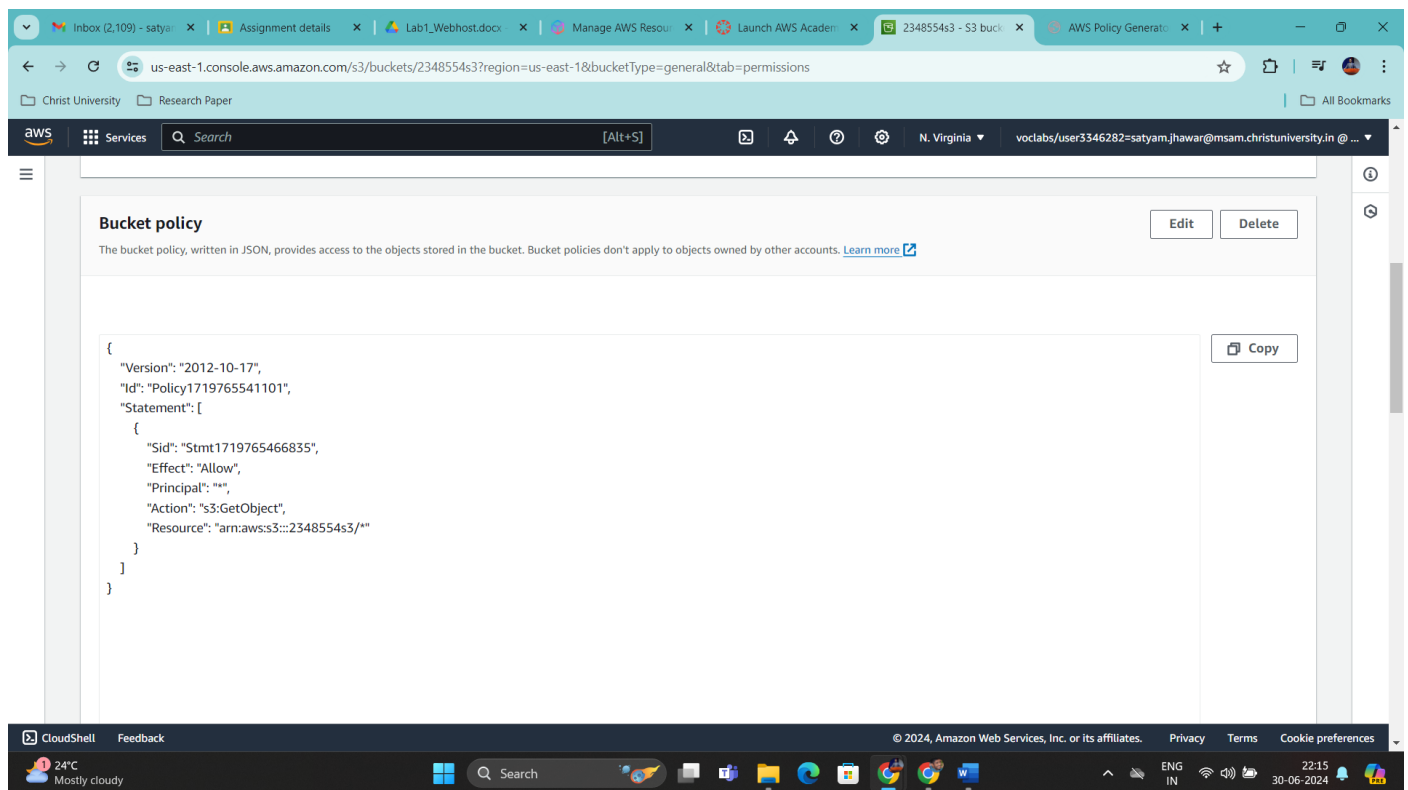


Figure 17: Bucket Policy Appears

Now the object within the bucket is publicly accessible

Step 12: Now go back to the EC2 Instance named 2348554_EC2_VM2 and connect it to the terminal

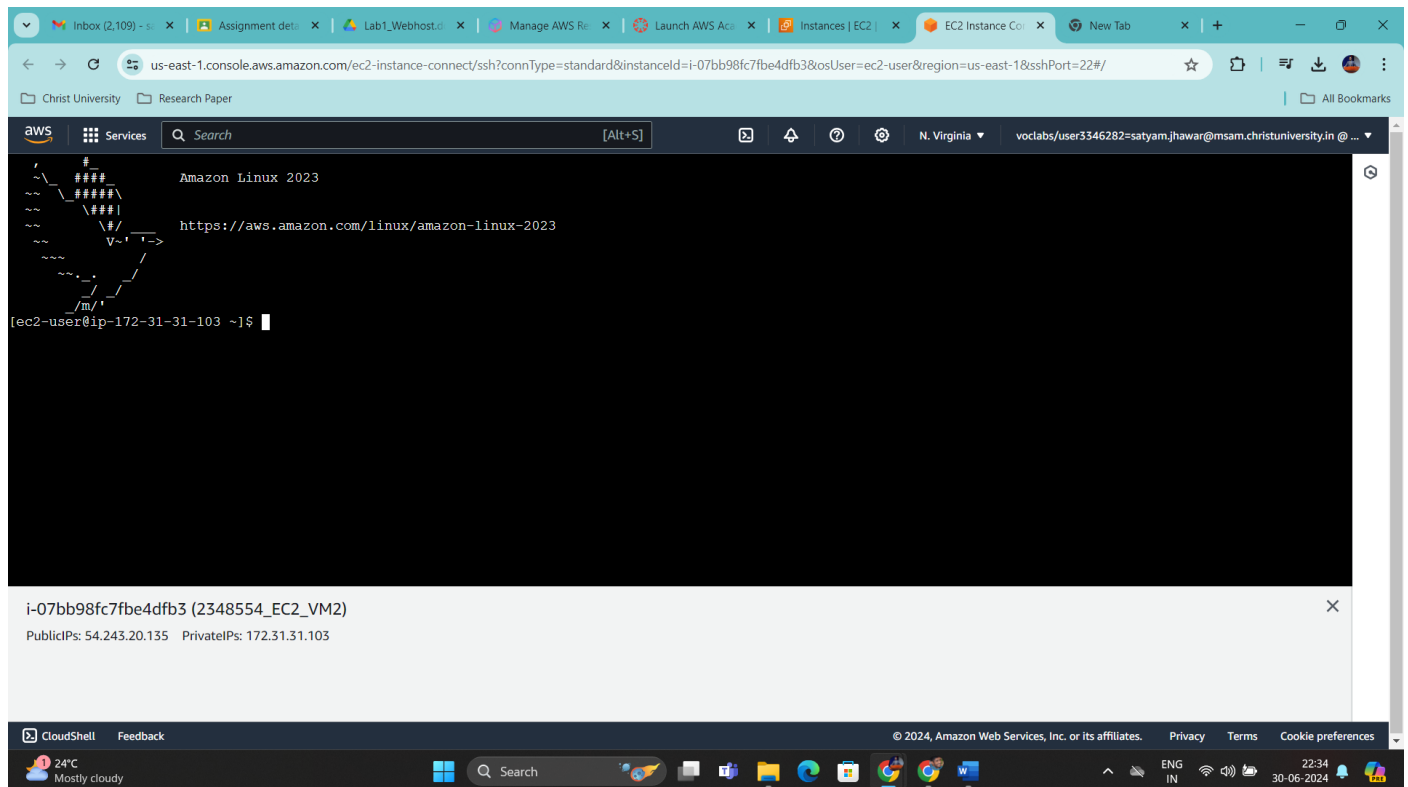


Figure 18: 2348554_EC2_VM2 Terminal

Step 13: Change to root user using the below command

Sudo su

```
[ec2-user@ip-172-31-31-103 ~]$ sudo su
[root@ip-172-31-31-103 ec2-user]#
```

Figure 19: Switched to Root User

Step 14: Update the instance

Yum update -y

```
[root@ip-172-31-31-103 ec2-user]# yum update -y
Last metadata expiration check: 0:06:46 ago on Sun Jun 30 16:59:16 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-31-103 ec2-user]#
```

Figure 20: Update command

Step 15: Install Apache Server using the below command

Sudo yum install httpd -y

```
[root@ip-172-31-31-103 ec2-user]# sudo yum install httpd
Last metadata expiration check: 0:08:28 ago on Sun Jun 30 16:59:16 2024.
Dependencies resolved.
=====
Package                                Architecture      Version           Repository        Size
=====
Installing:
httpd                                   x86_64            2.4.59-2.amzn2023  amazonlinux       47 k
Installing dependencies:
apr                                     x86_64            1.7.2-2.amzn2023.0.2  amazonlinux       129 k
apr-util                               x86_64            1.6.3-1.amzn2023.0.1  amazonlinux       98 k
generic-logos-httpd                   noarch            18.0.0-12.amzn2023.0.3  amazonlinux       19 k
httpd-core                             x86_64            2.4.59-2.amzn2023    amazonlinux       1.4 M
httpd-filesystem                       noarch            2.4.59-2.amzn2023    amazonlinux       14 k
httpd-tools                             x86_64            2.4.59-2.amzn2023    amazonlinux       81 k
libbrotli                               x86_64            1.0.9-4.amzn2023.0.2  amazonlinux       315 k
mailcap                                 noarch            2.1.49-3.amzn2023.0.3  amazonlinux       33 k
Installing weak dependencies:
apr-util-openssl                       x86_64            1.6.3-1.amzn2023.0.1  amazonlinux       17 k
mod_http2                              x86_64            2.0.27-1.amzn2023.0.2  amazonlinux       166 k
mod_lua                                x86_64            2.4.59-2.amzn2023    amazonlinux       61 k
=====
Installing      : httpd-core-2.4.59-2.amzn2023.x86_64                                8/12
Installing      : mod_http2-2.0.27-1.amzn2023.0.2.x86_64                             9/12
Installing      : mod_lua-2.4.59-2.amzn2023.x86_64                                  10/12
Installing      : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                  11/12
Installing      : httpd-2.4.59-2.amzn2023.x86_64                                   12/12
Running scriptlet: httpd-2.4.59-2.amzn2023.x86_64                                   12/12
Verifying       : apr-1.7.2-2.amzn2023.0.2.x86_64                                  1/12
Verifying       : apr-util-1.6.3-1.amzn2023.0.1.x86_64                             2/12
Verifying       : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64                     3/12
Verifying       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                 4/12
Verifying       : httpd-2.4.59-2.amzn2023.x86_64                                   5/12
Verifying       : httpd-core-2.4.59-2.amzn2023.x86_64                             6/12
Verifying       : httpd-filesystem-2.4.59-2.amzn2023.noarch                         7/12
Verifying       : httpd-tools-2.4.59-2.amzn2023.x86_64                             8/12
Verifying       : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                           9/12
Verifying       : mailcap-2.1.49-3.amzn2023.0.3.noarch                             10/12
Verifying       : mod_http2-2.0.27-1.amzn2023.0.2.x86_64                          11/12
Verifying       : mod_lua-2.4.59-2.amzn2023.x86_64                                12/12
Installed:
apr-1.7.2-2.amzn2023.0.2.x86_64      apr-util-1.6.3-1.amzn2023.0.1.x86_64      apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch  httpd-2.4.59-2.amzn2023.x86_64      httpd-core-2.4.59-2.amzn2023.x86_64
httpd-filesystem-2.4.59-2.amzn2023.noarch      httpd-tools-2.4.59-2.amzn2023.x86_64      libbrotli-1.0.9-4.amzn2023.0.2.x86_64
mailcap-2.1.49-3.amzn2023.0.3.noarch      mod_http2-2.0.27-1.amzn2023.0.2.x86_64      mod_lua-2.4.59-2.amzn2023.x86_64
Complete!
[root@ip-172-31-31-103 ec2-user]#
```

Figure 21: Apache Server Installed

Step 16: Start and enable apache server using the below commands

Systemctl start httpd

Systemctl enable httpd

Systemctl status httpd

```
[root@ip-172-31-31-103 ec2-user]# systemctl start httpd
[root@ip-172-31-31-103 ec2-user]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-31-103 ec2-user]# systemctl status httpd
• httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Active: active (running) since Sun 2024-06-30 17:11:44 UTC; 23s ago
     Docs: man:httpd.service(8)
  Main PID: 26203 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/sec"
    Tasks: 177 (limit: 1114)
  Memory: 12.9M
    CPU: 74ms
   CGroup: /system.slice/httpd.service
           └─26203 /usr/sbin/httpd -DFOREGROUND
           └─26226 /usr/sbin/httpd -DFOREGROUND
           └─26227 /usr/sbin/httpd -DFOREGROUND
           └─26228 /usr/sbin/httpd -DFOREGROUND
           └─26229 /usr/sbin/httpd -DFOREGROUND

Jun 30 17:11:44 ip-172-31-31-103.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jun 30 17:11:44 ip-172-31-31-103.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Jun 30 17:11:44 ip-172-31-31-103.ec2.internal httpd[26203]: Server configured, listening on: port 80
[root@ip-172-31-31-103 ec2-user]#
```

Figure 22: Apache server in running mode

Step 17: Change the current directory to var/www/html using the below command

Cd /var/www/html

```
[root@ip-172-31-31-103 ec2-user]# pwd
/home/ec2-user
[root@ip-172-31-31-103 ec2-user]# cd /var/www/html
[root@ip-172-31-31-103 html]# pwd
/var/www/html
[root@ip-172-31-31-103 html]#
```

Figure 23: Directory changed

Step 18: Navigate back to s3 in the AWS console and copy the uploaded file's Object URL

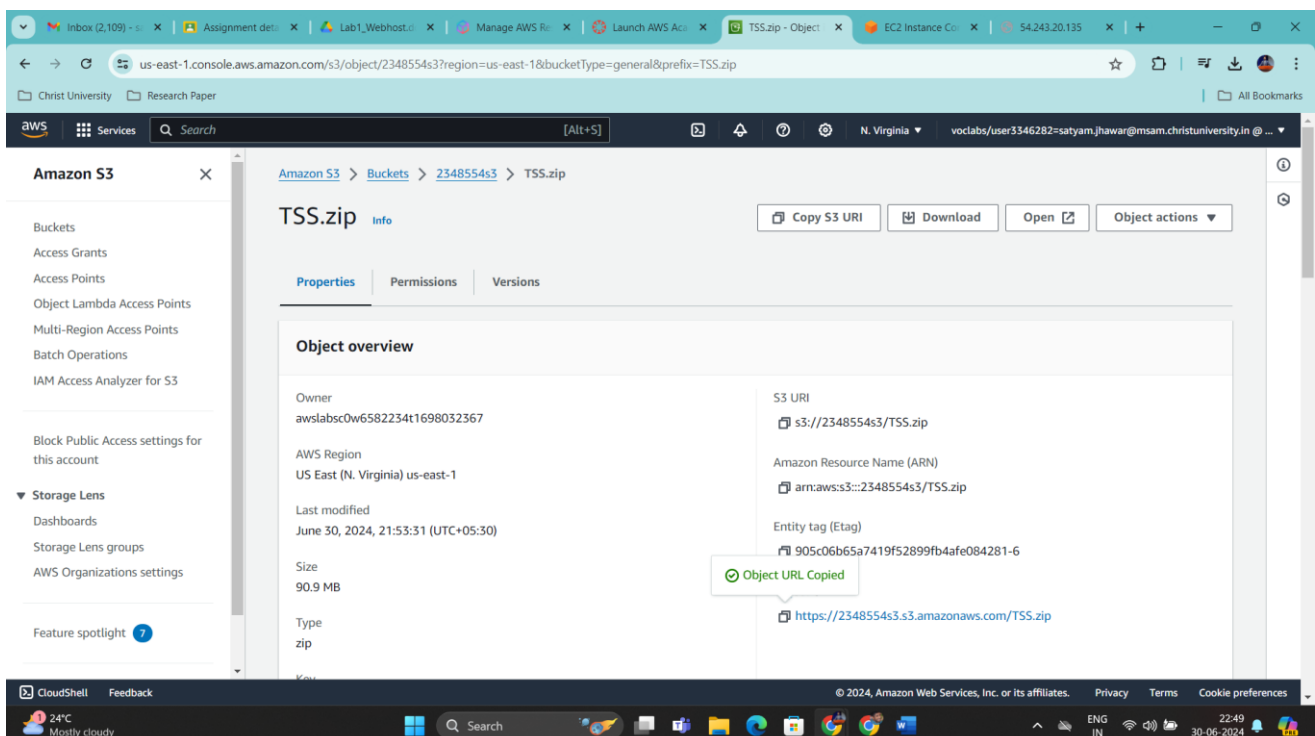


Figure 24: Object URL of the uploaded file

Step 19: Download the uploaded file in the s3 to our EC2 instance using the below command

Wget <https://2348554s3.s3.amazonaws.com/TSS.zip>

```
[root@ip-172-31-31-103 html]# wget https://2348554s3.s3.amazonaws.com/TSS.zip
--2024-06-30 17:22:38-- https://2348554s3.s3.amazonaws.com/TSS.zip
Resolving 2348554s3.s3.amazonaws.com (2348554s3.s3.amazonaws.com)... 54.231.130.225, 54.231.193.17, 54.231.230.49, ...
Connecting to 2348554s3.s3.amazonaws.com (2348554s3.s3.amazonaws.com)|54.231.130.225|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 95337809 (91M) [application/zip]
Saving to: 'TSS.zip'

TSS.zip
100%[=====>] 90.92M 77.5MB/s in 1.2s

2024-06-30 17:22:39 (77.5 MB/s) - 'TSS.zip' saved [95337809/95337809]

[root@ip-172-31-31-103 html]# ls
TSS.zip
```

Figure 25: Downloading the file

Step 20: Unzip the file using the below command

Unzip TSS.zip

```
[root@ip-172-31-31-103 html]# unzip TSS.zip
Archive:  TSS.zip
  creating: TSS/css/
  inflating: TSS/css/developerstyle.css
  inflating: TSS/css/mainstyle.css
  inflating: TSS/css/style.css
  inflating: TSS/developers.html
  creating: TSS/images/
  inflating: TSS/images/bg.jpg
  inflating: TSS/images/bg1.jpg
  inflating: TSS/images/christina.jpg
  extracting: TSS/images/models - checkpoint 50.zip
  inflating: TSS/images/reeve.jpg
  inflating: TSS/images/satyam.jpg
  inflating: TSS/index.html
[root@ip-172-31-31-103 html]# ls
TSS  TSS.zip
[root@ip-172-31-31-103 html]#
```

Figure 26: Folder unzipped

Step 21: Move all the files of the folder to the directory var/www/html using the below command

Mv TSS/*

```
[root@ip-172-31-31-103 html]# mv TSS/* .
[root@ip-172-31-31-103 html]# ls
TSS  TSS.zip  css  developers.html  images  index.html
[root@ip-172-31-31-103 html]#
```

Figure 27: Files have been moved

Step 22: Start the apache server again using the below command

Systemctl start httpd

Step 23: Now navigate back to EC2 and copy its public IP address and paste it in a browser, we will find the output of our Static website

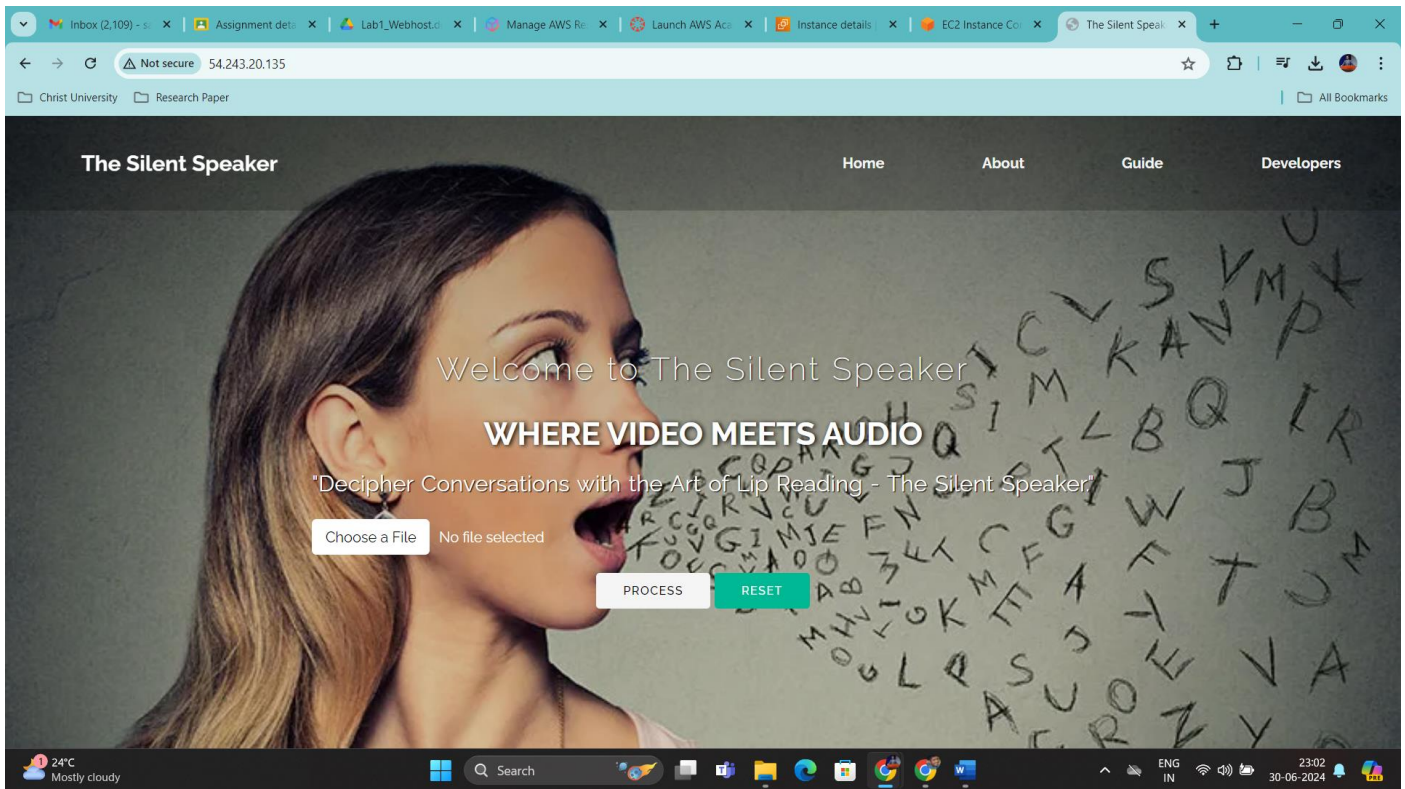


Figure 28: Static Website Deployed on EC2