MANAGED SERVICES ON AWS-PROJECT OPTION 2

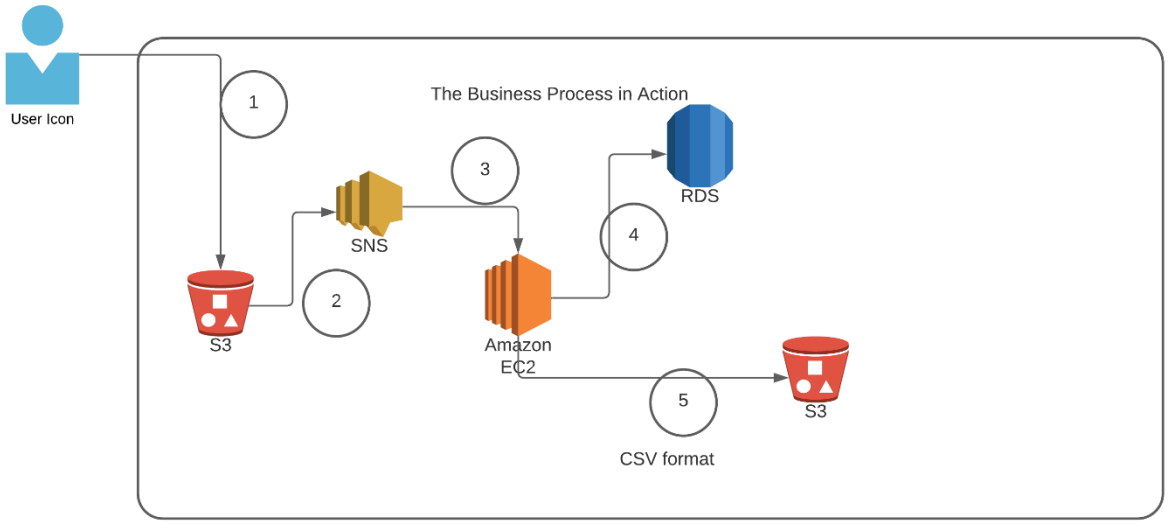
Declaration	
Questions in this exercise are intentionally complex and could be convoluted or confusing. This is by design and to simulate real life situations where customers seldom give crystal clear requirements and ask unambiguous questions.	

I have read the above statement and agree to these conditions	
I AGREE	**ADEKUNLE ADEYINKA ADEGBIE**
	<Enter your name above this line to indicate that you are in agreement>

Instructions
Every screenshot requested in this workbook is compulsory and carries 1 marks
Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed.
All screenshots must be in the order mentioned under "Expected Screenshots" for every step
DO NOT WAIT UNTIL THE LAST MINUTE. The program office will not extend the project submission deadline under any circumstances.
The file should be renamed in the format BATCH_FIRSTNAME_LASTNAME_PROJECT1. For example: PGPCCMAY18_VIJAY_DWIVEDI_PROJECT1.pdf

Resource Clean Up
Cloud is always pay per use model and all resources/services that we consume are chargeable. Cleaning up when you've completed your lab or project is always necessary. This is true whether you're doing a lab or implementing a project at your workplace.
After completing the lab, make sure to delete each resource created in reverse chronological order. Each AWS Academy session lasts for 4 hours by default, although you can extend a session to run longer by pressing the start button to reset your session timer. At the end of each session, any resources you created in the account will be preserved. Some AWS resources, such as EC2 instances, may be automatically shut down, while other resources, such as RDS instances will be left running.

Architecture diagram

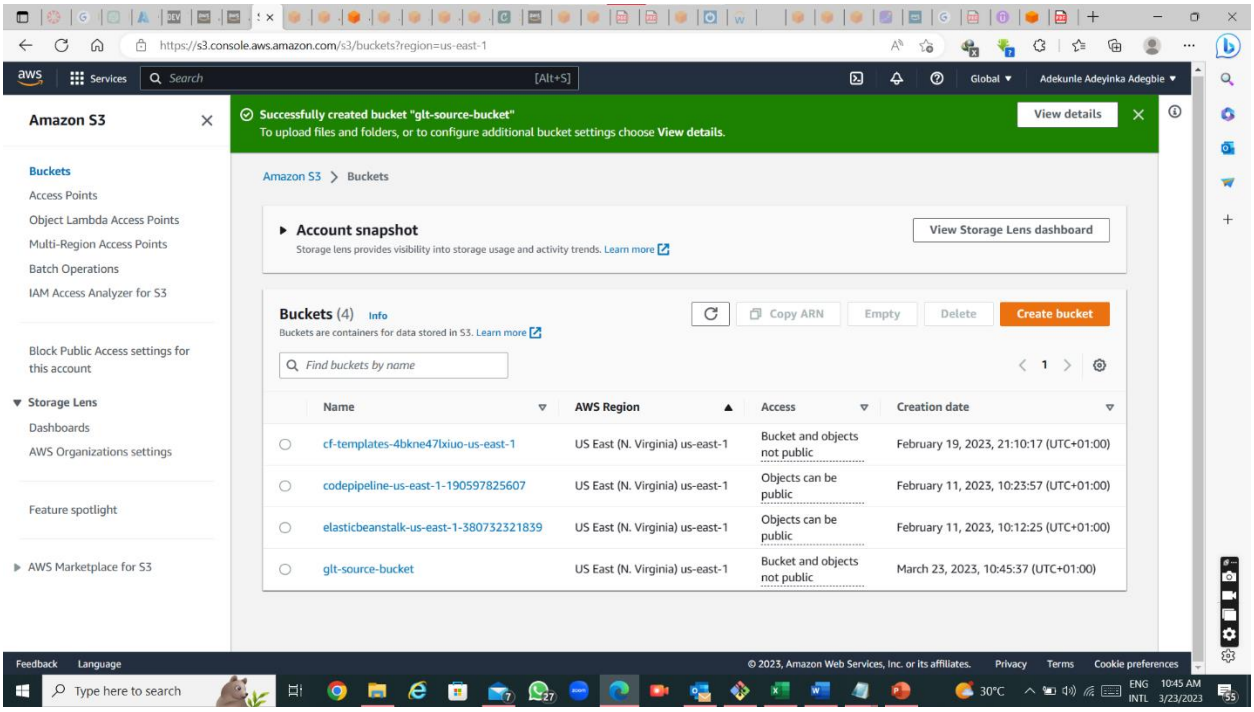


Architecture Implementation	
1	The customer uploads the invoice data to S3 bucket in a text format as per their guidelines and policies. This bucket will have a policy to auto delete any content that is more than 1 day old (24 hours).
2	An event will trigger in the bucket that will place a message in SNS topic
3	A custom program running in EC2 will subscribe to the SNS topic and get the message placed by S3 event
4	The program will use S3 API to read from the bucket, parse the content of the file and create a CSV record and save the details in an RDS database
5	The program will use S3 API to write CSV record to destination S3 bucket as new S3 object.
Note	The custom program codebase and sample invoice have been shared along with this workbook on the LMS.

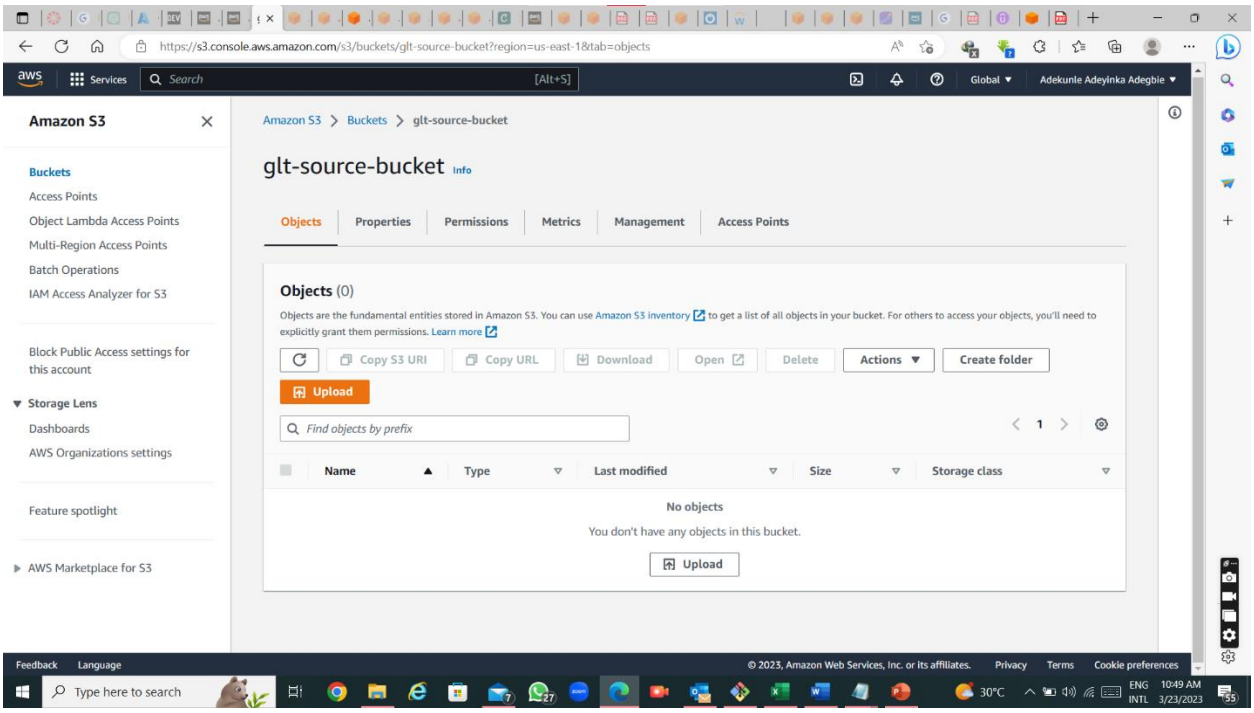
Step 1: SNS and S3 topic creation

Step number	a
Step name	Creation of Source and target buckets
Instructions	1) Navigate to S3 using the Services button at the top of the screen 2) Select "Create Bucket" 3) Enter a source bucket name and use the default options for the rest of the fields 4) Click on "Create Bucket' 5) Repeat the above steps to create a target bucket
Expected screenshots	1) Screen showing created S3 source and target buckets

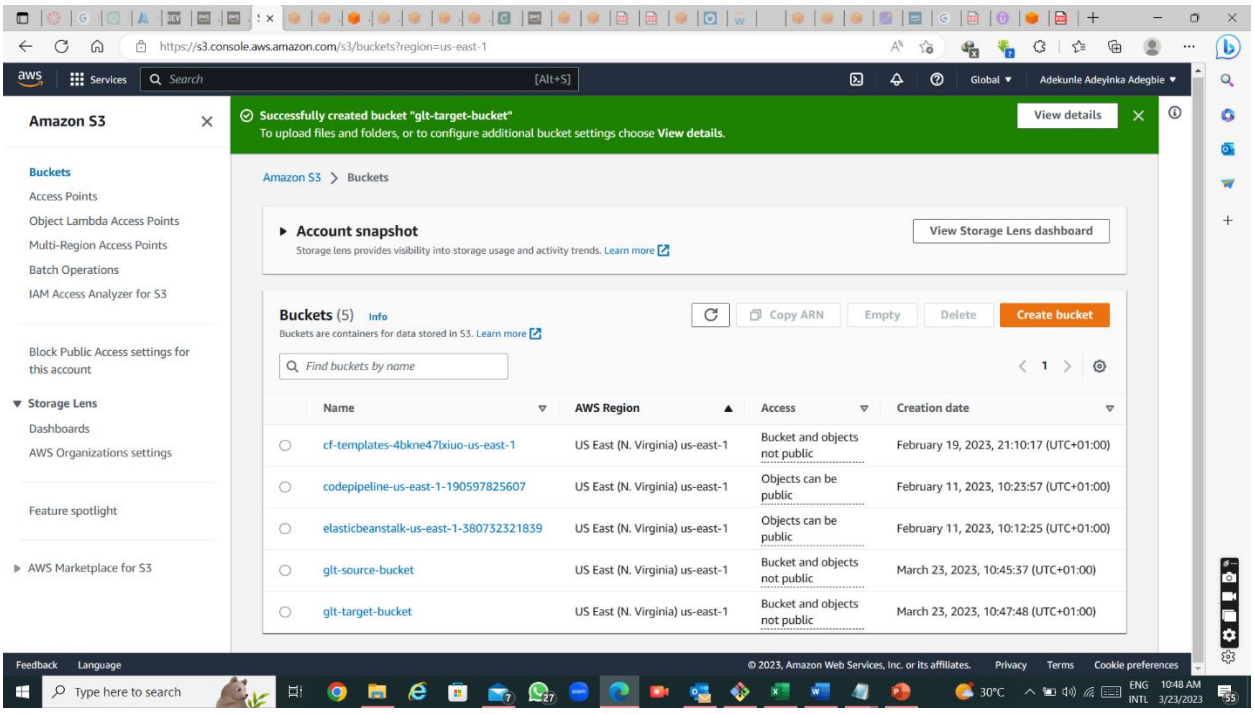
1. Creation of Source bucket



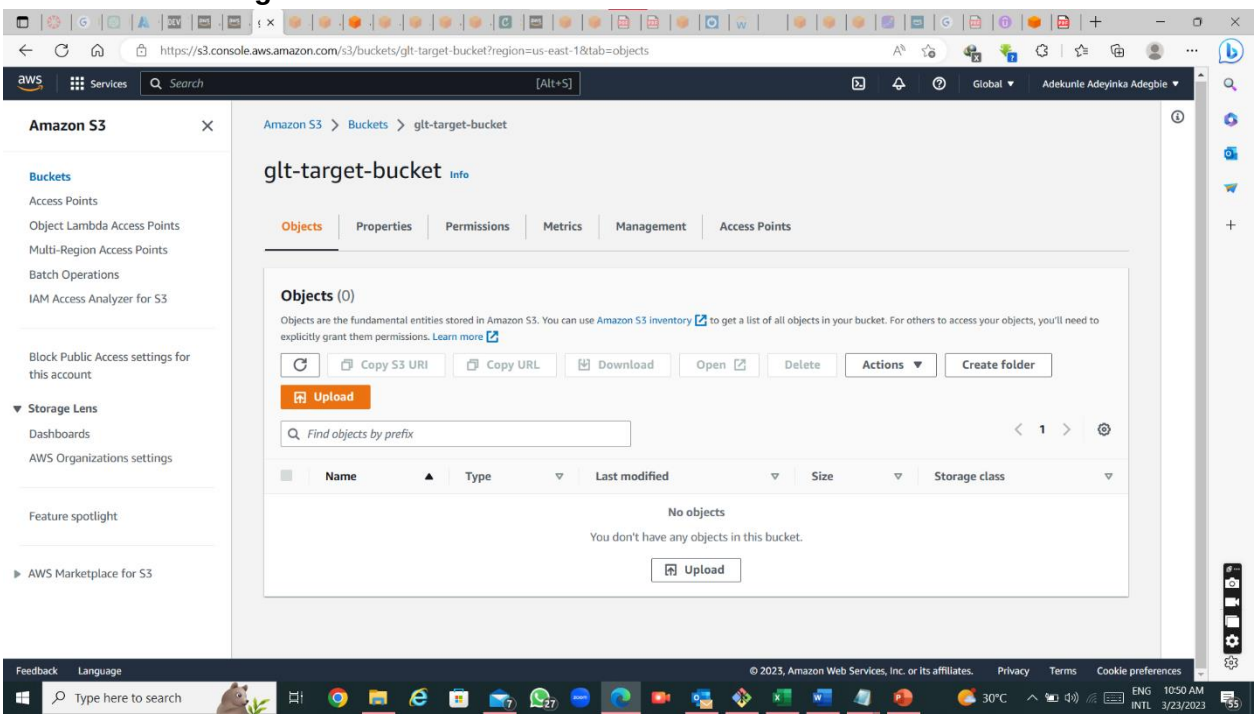
2. Created Source bucket



3. Creation of Target bucket

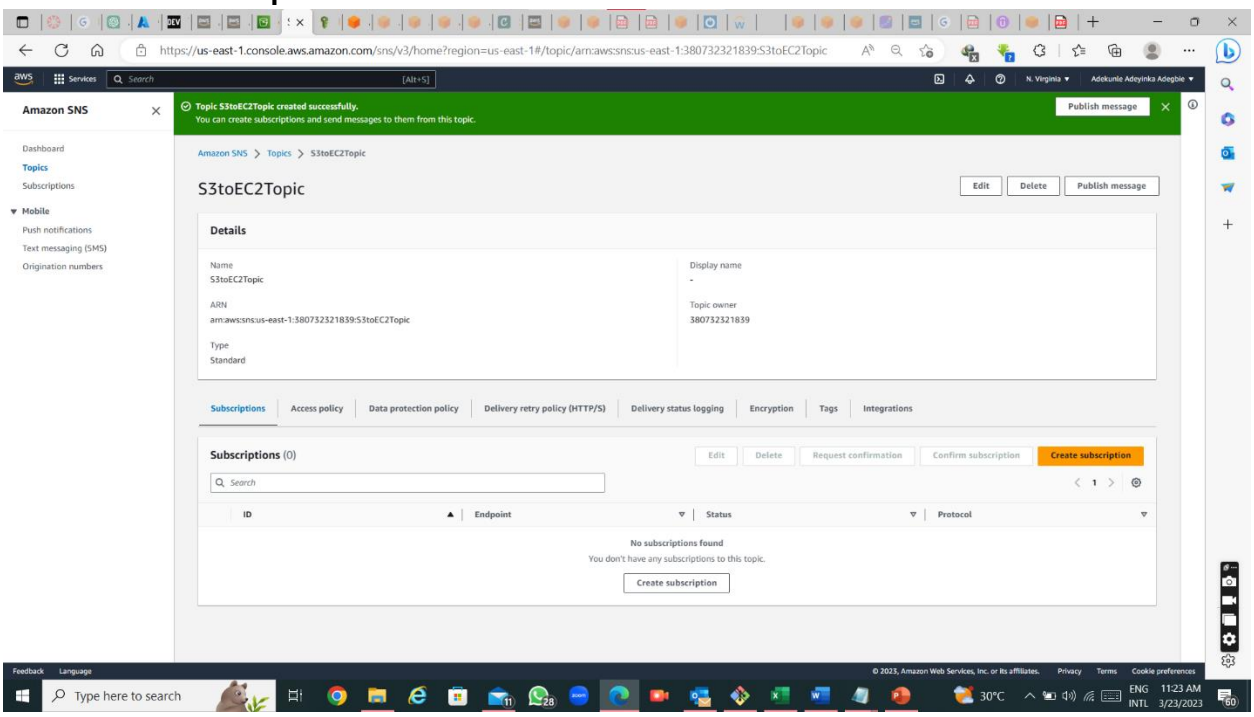


4. **Created target bucket**



Step number	B
Step name	Creation of SNS subscription
Instructions	1) Navigate to SNS -> Topics 2) Click on "Create Topic" 3) Enter the following fields Name : S3toEC2Topic The other options can be ignored for now 4) Click on Create Topic
Expected screenshots	1) Creation of SNS topic

5. Creation of SNS Topic

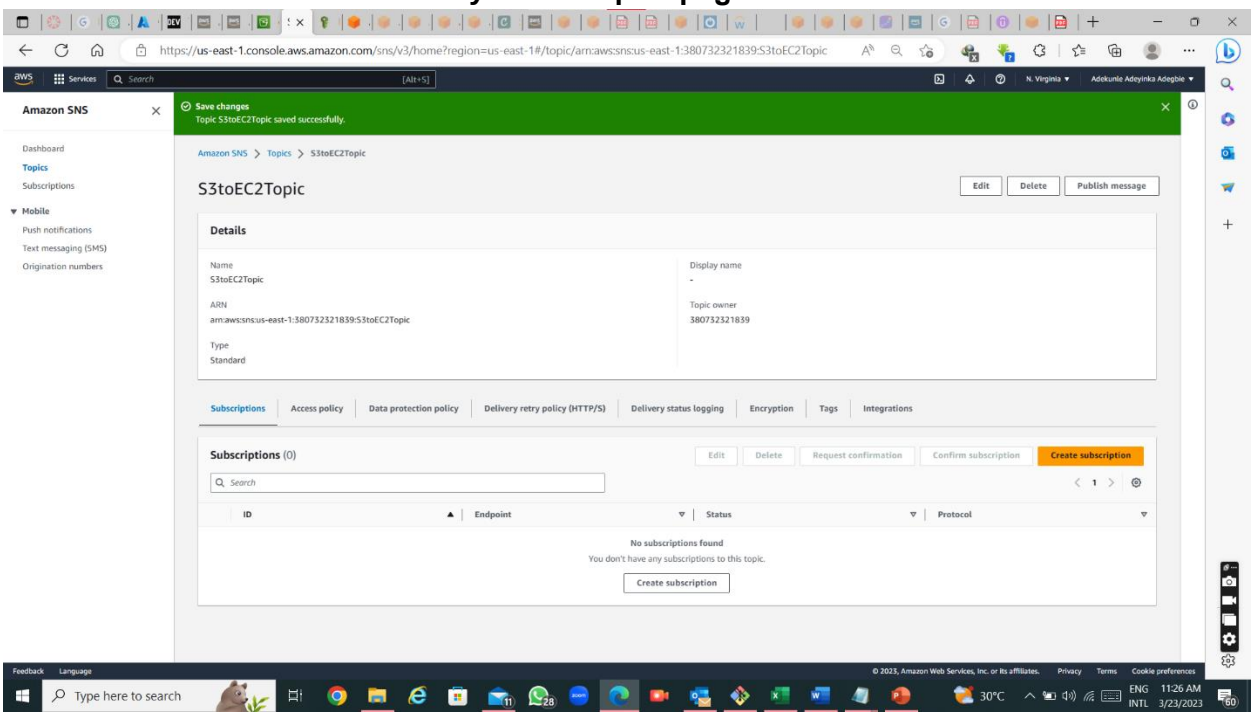


Step number	C
Step name	Modification of SNS Access Policy

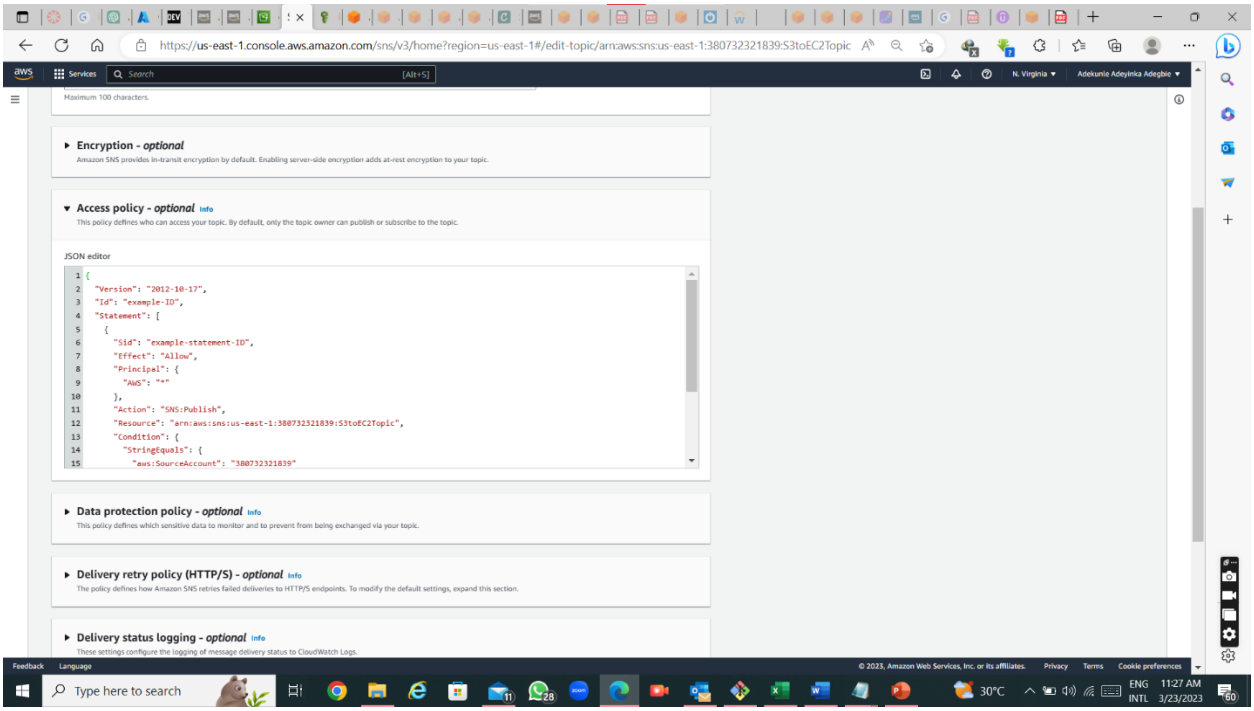
Instructions	1) Navigate to SNS -> Topics and select the topic created in the previous step
	2) Note down the ARN shown in the topic details
	2) Click on Edit and select "Access Policy".
	3) Replace the text in the JSON editor with the following
	{
	"Version": "2012-10-17",
	"Id": "example-ID",
	"Statement": [
	{
	"Sid": "example-statement-ID",
	"Effect": "Allow",
	"Principal": {
	"AWS":"*"
	},
	"Action": [
	"SNS:Publish"
	],
	"Resource": "**SNS-topic-ARN**",
	"Condition": {
	"ArnLike": { "aws:SourceArn": "arn:aws:s3:*:*:**bucket-name**" },
	"StringEquals": { "aws:SourceAccount": "**bucket-owner-account-id**" }
	}
	}
	]
	}
	
	4) Replace the bold text with the SNS topic ARN, source bucket name and your AWS account ID respectively.
	5) Click on Save Changes

Expected screenshots	1) JSON Editor screen

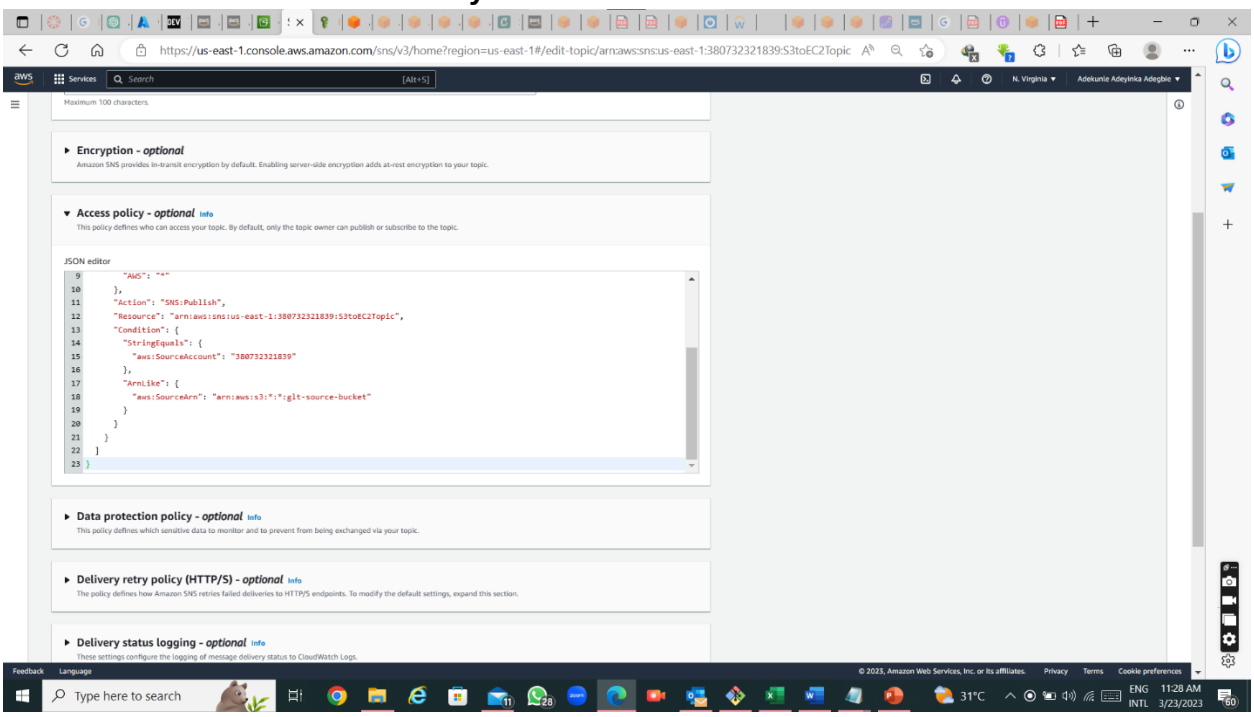
6. Modification of SNS Access Policy-“SNS Topic” page



7. Modification of SNS Access Policy-“JSON Editor screen 1”

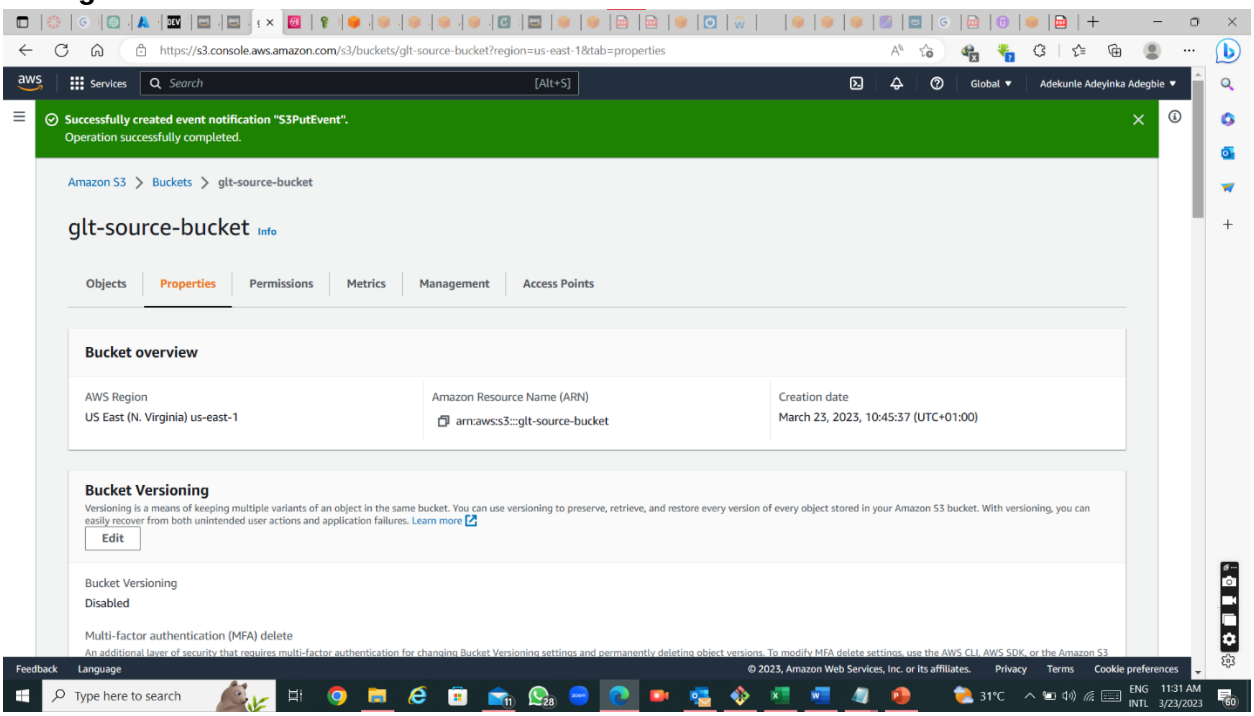


8. Modification of SNS Access Policy-“JSON Editor screen 2”



Step number	D
Step name	Configuring SNS notifications for S3
Instructions	1) Navigate to S3 and select the source bucket created in Step 1 (a) 2) Select Properties and scroll down to Event Notifications and select it 3) Select "Create Event Notification" 4) Fillup the details as follows Name : S3PutEvent Select PUT from the list of radio buttons Destination : Select SNS Topic SNS : Select S3ToEC2Topic 5) Save Changes
Expected screenshots	1) Event Configuration Screen

9. **Configuration of SNS notification for S3-“creation of event notification”**



Step 2: Run the custom program in the EC2 instance

Step number	A
Step name	Creation of the EC2 instance and RDS instance
Instructions	1) Navigate to EC2 -> Instances 2) Create an EC2 instance with the following parameters AMI : Amazon Linux 2 VPC : Default Security group : Ports 22 and 8080 should be opened 3) Navigate to RDS 4) Create an RDS instance with the following parameters: Engine type : MySql Template : Dev/Test Set the username and password as required DB Instance class : Burstable Instance type : t3.micro Public Access : Yes

VPC Security group : Create New ()

Under Additional Configuration, add an initial database name. Take note of this name as it will be required later.

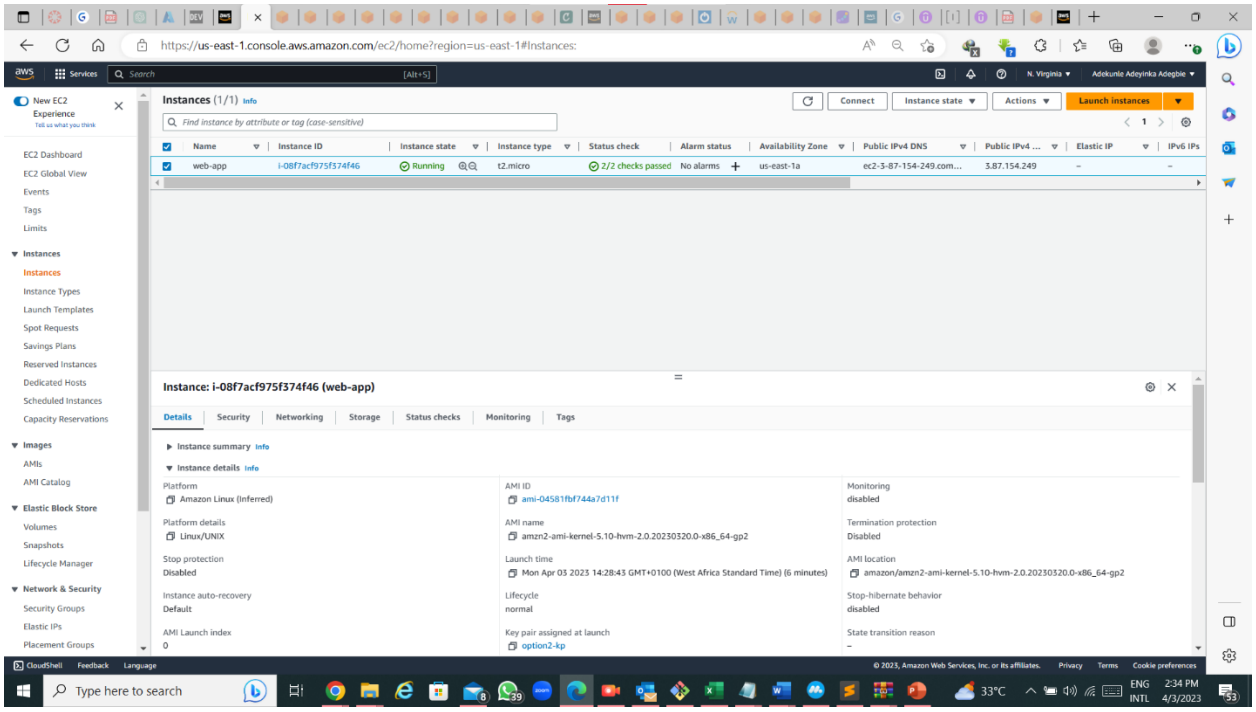
Uncheck “Enable Enhanced Monitoring”

Ensure that the security group created by the RDS deployment has port 3306 open for all incoming connections from all sources.

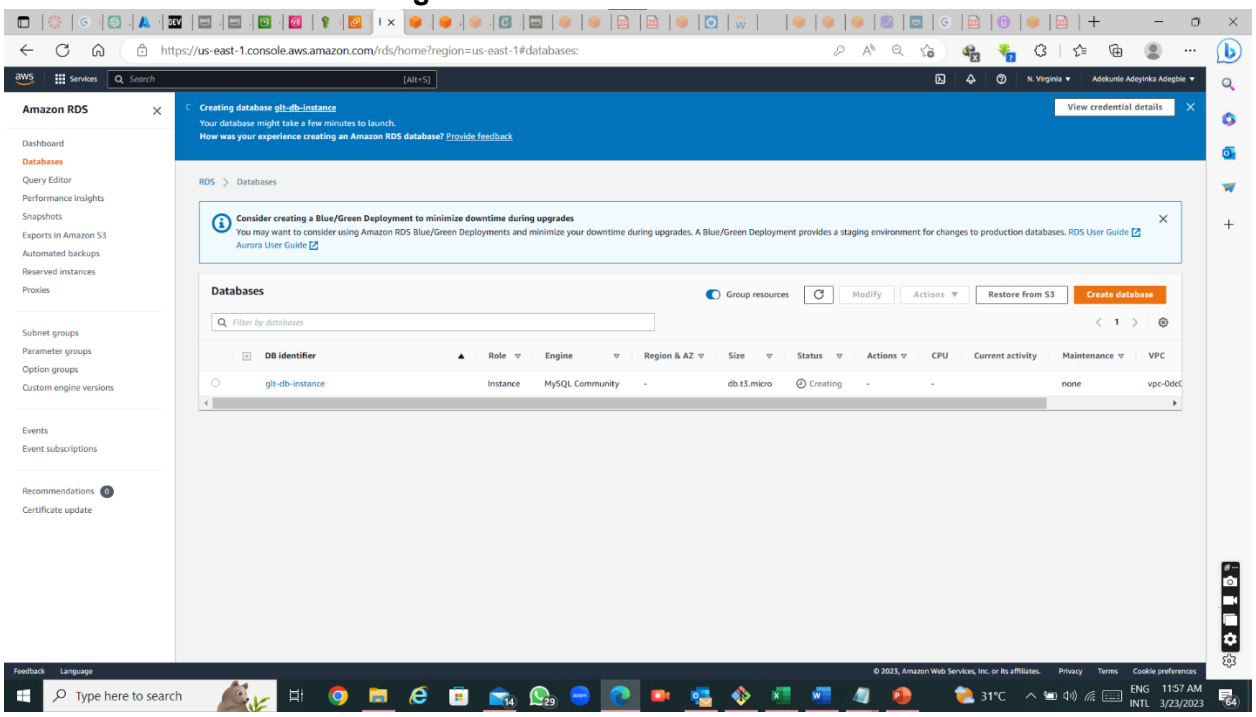
Expected screenshots

1) List of instances after creation of EC2 instance
2) List of RDS instances

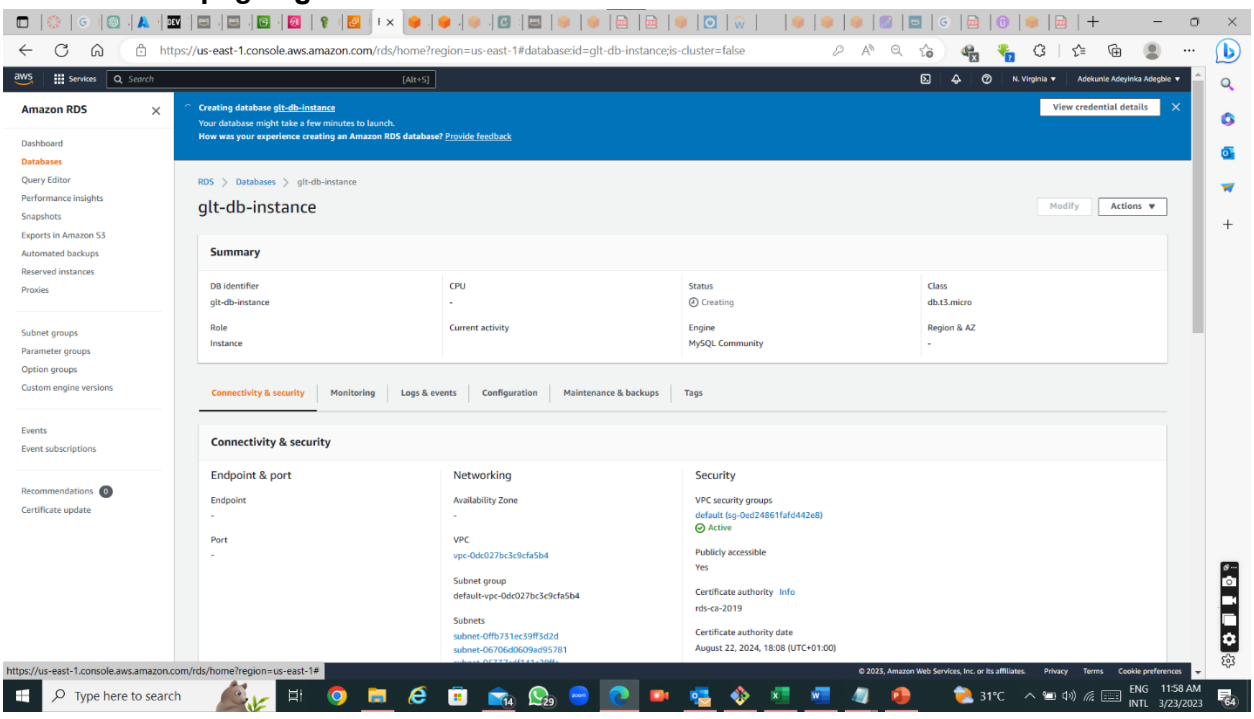
10. Lunched EC2 Instance-“web-page”



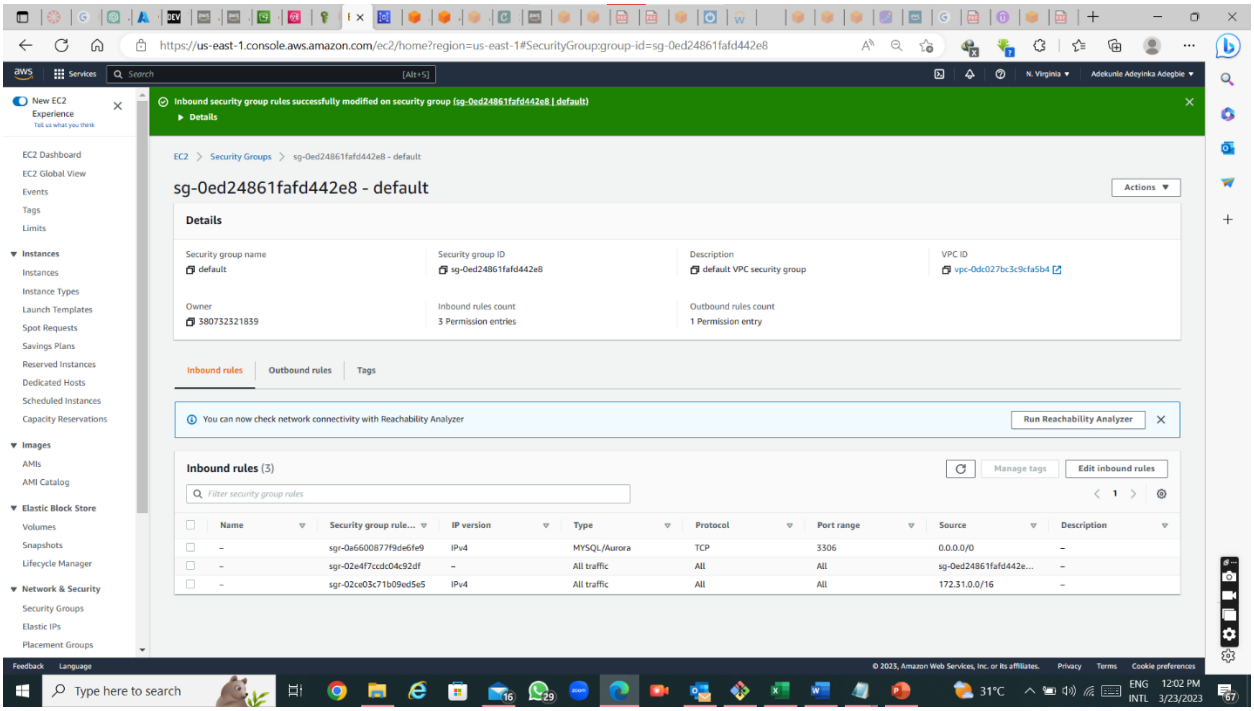
11. Launched RDS Instance-“glt-db-instance”



12. RDS Instance page-“glt-db-instance”

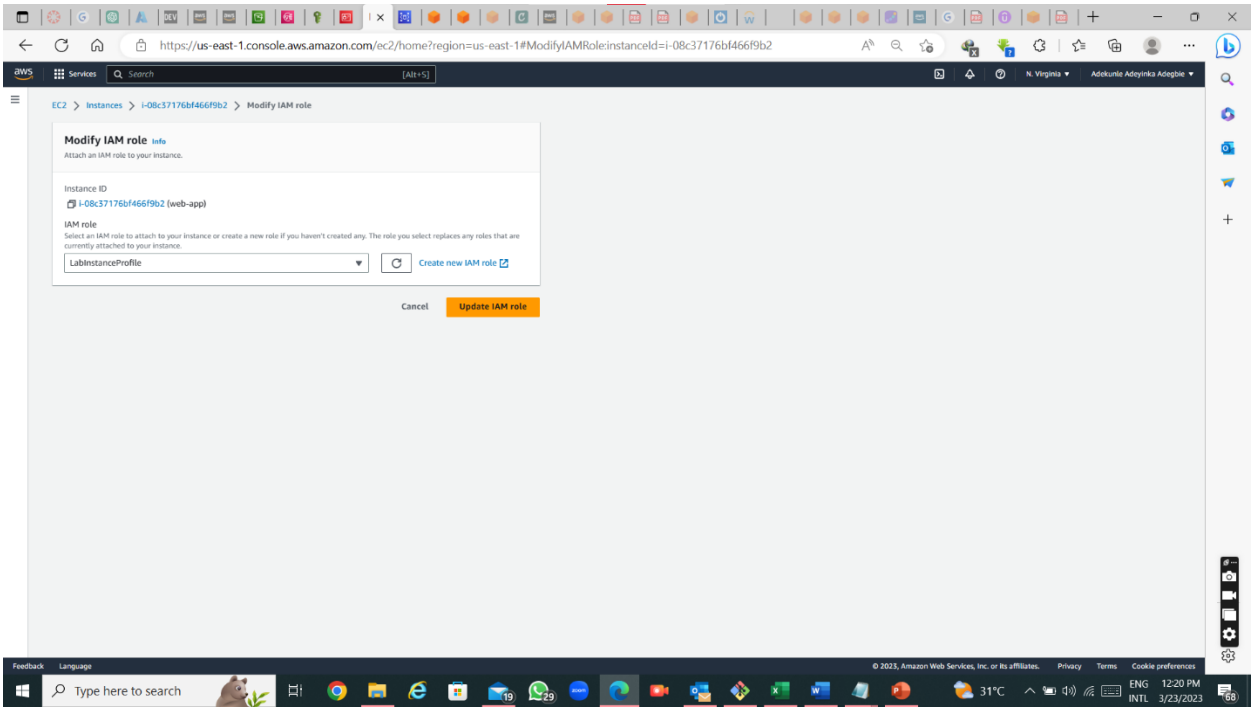


13. RDS security group has port 3306 opened for all incoming connections



Step number	B
Step name	Assignment of IAM role for EC2 instance
Instructions	1) Navigate back to EC2- > Instances 2) Select the EC2 instance created in the previous step and select Actions-> Security -> Modify IAM role 3) Select the role LabInstanceProfile from the dropdown and click on Save
Expected screenshots	1) Modify IAM role screen

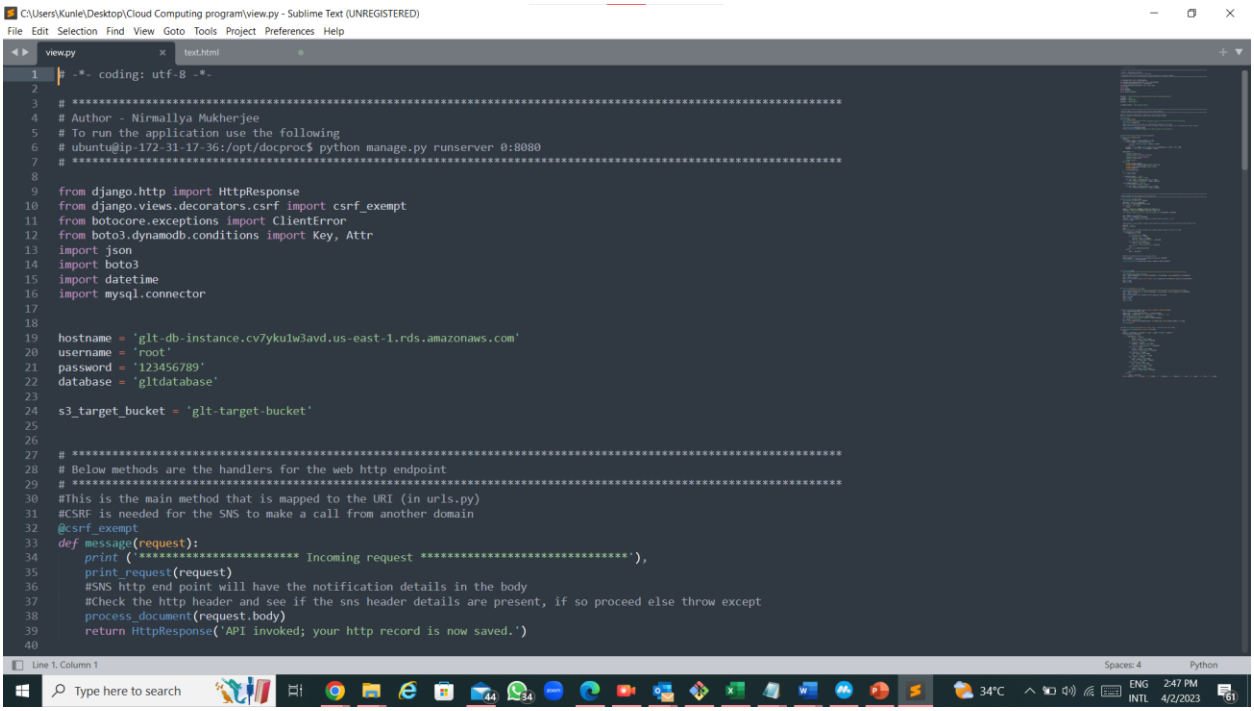
14 Modify IAM role screen



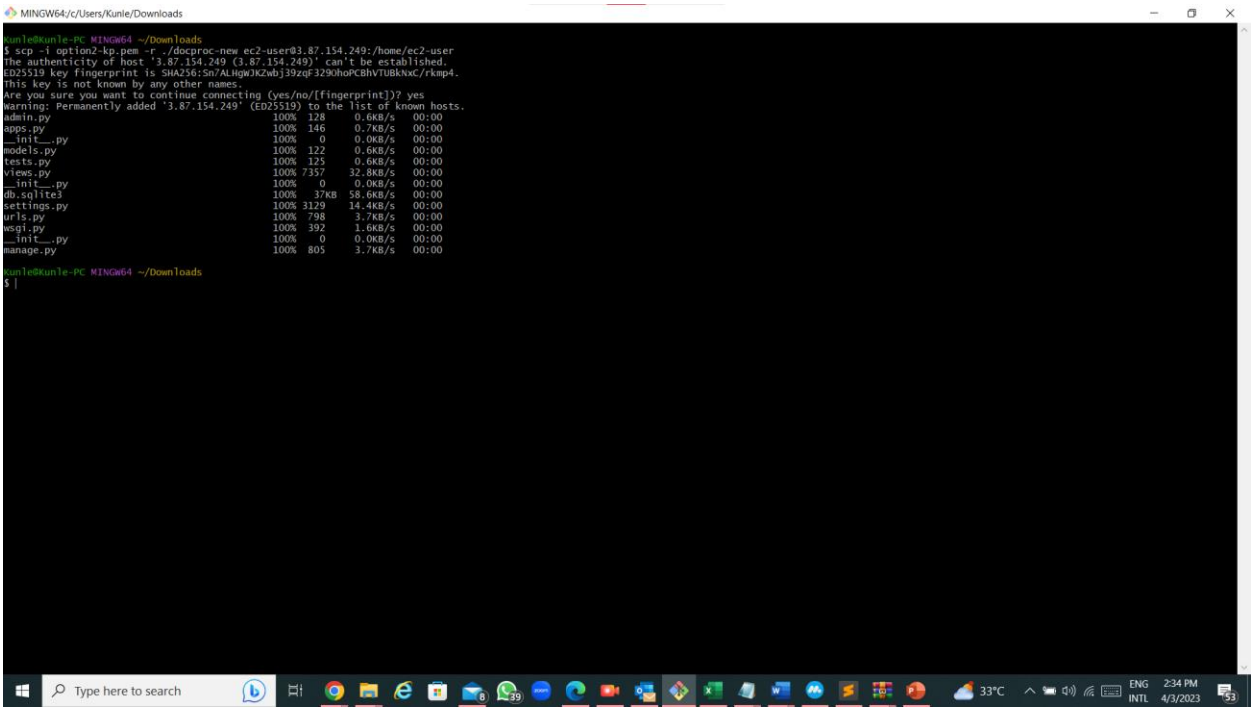
Step number	C
Step name	Configuration and Uploading of custom program
Instructions	1) Download the file **docproc-new.zip** on your machine 2) Unzip the downloaded file 3) Enter the unzipped folder and open the file views.py in the API folder using a text editor 4) In line number 19-24, modify the target bucket name to the one created in Step 2 (a) and modify the hostname, username, password and database variables to the values set while creating the RDS database and save the file 5) Copy the folder docproc-new to the home folder of the EC2 instance created in Step 3(a) using scp. Use the command given below *scp -i <pem> -r ./docproc-new ec2-user@<ip>:/home/ec2-user*

Expected screenshots	1) Modifying of the views.py file to point to the target bucket	2)Copying the folder to the EC2 instance

15 Modifying of the views.py file to point to the target bucket



16 Copying the folder to the EC2 Instance



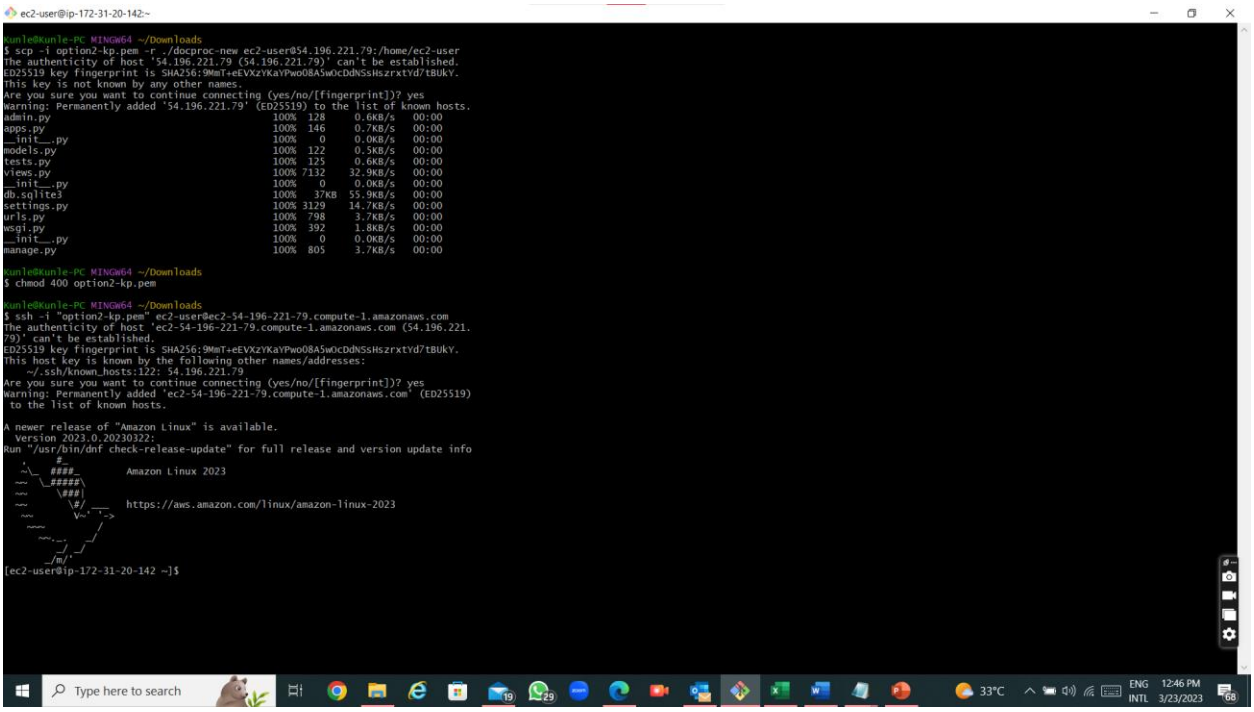
Step 3: Creation and Verification of SNS subscription and Generation of CSV file

Step number	a
Step name	Starting the EC2 custom program
Instructions	1) Log into the EC2 instance using SSH 2) Run the followng commands after successful SSH to start the server sudo cp -r docproc-new /opt sudo chown ec2-user:ec2-user -R /opt cd /opt/docproc-new sudo yum update sudo yum install python-pip -y python -m pip install --upgrade pip setuptools sudo pip install virtualenv virtualenv ~/.virtualenvs/djangodev source ~/.virtualenvs/djangodev/bin/activate pip install django pip install boto3 pip install mysql-connector-python-rf python manage.py runserver 0:8080

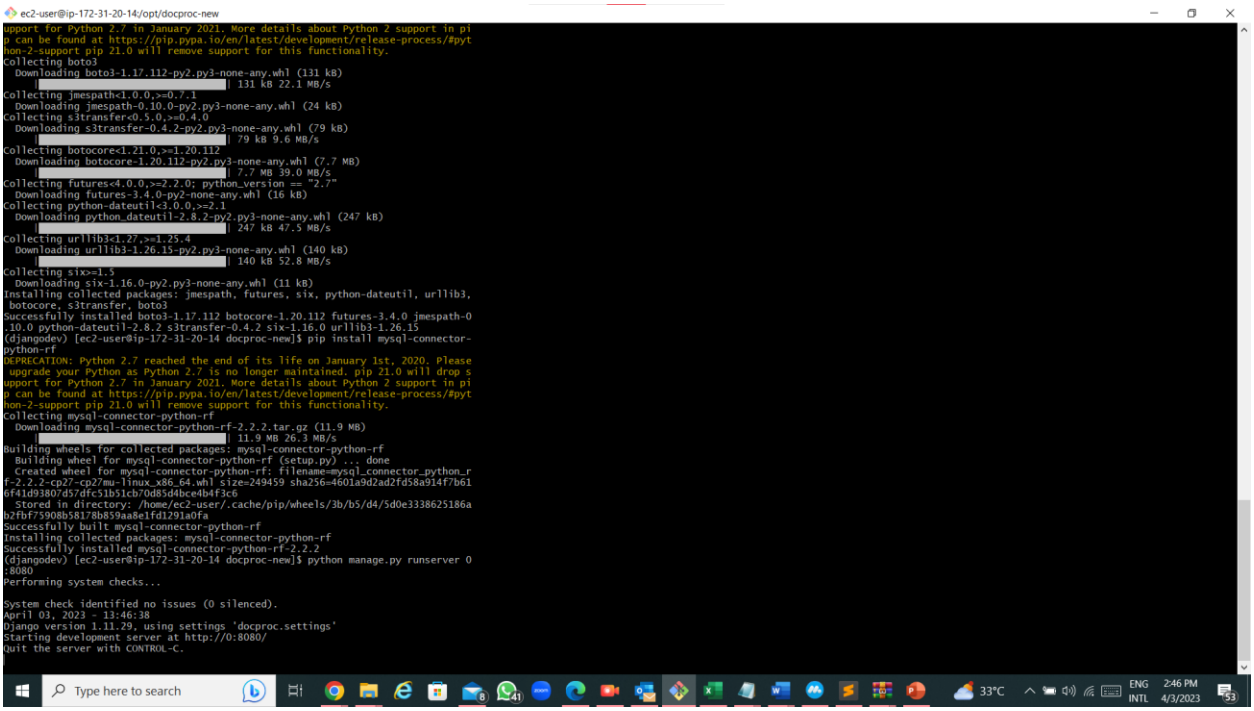
Keep this terminal window open throughout the rest of the exercise

Expected screenshots	1) Server in waiting state

17 Log into EC2 instance using SSH

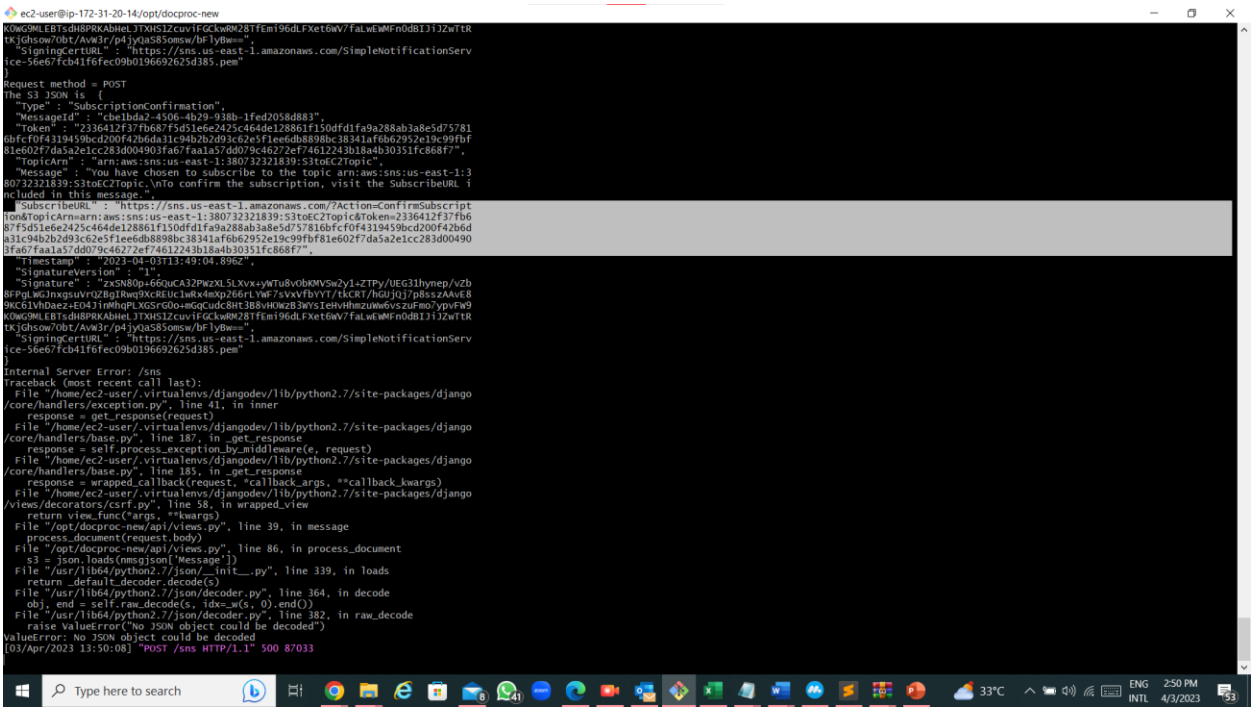


18 Server in waiting state

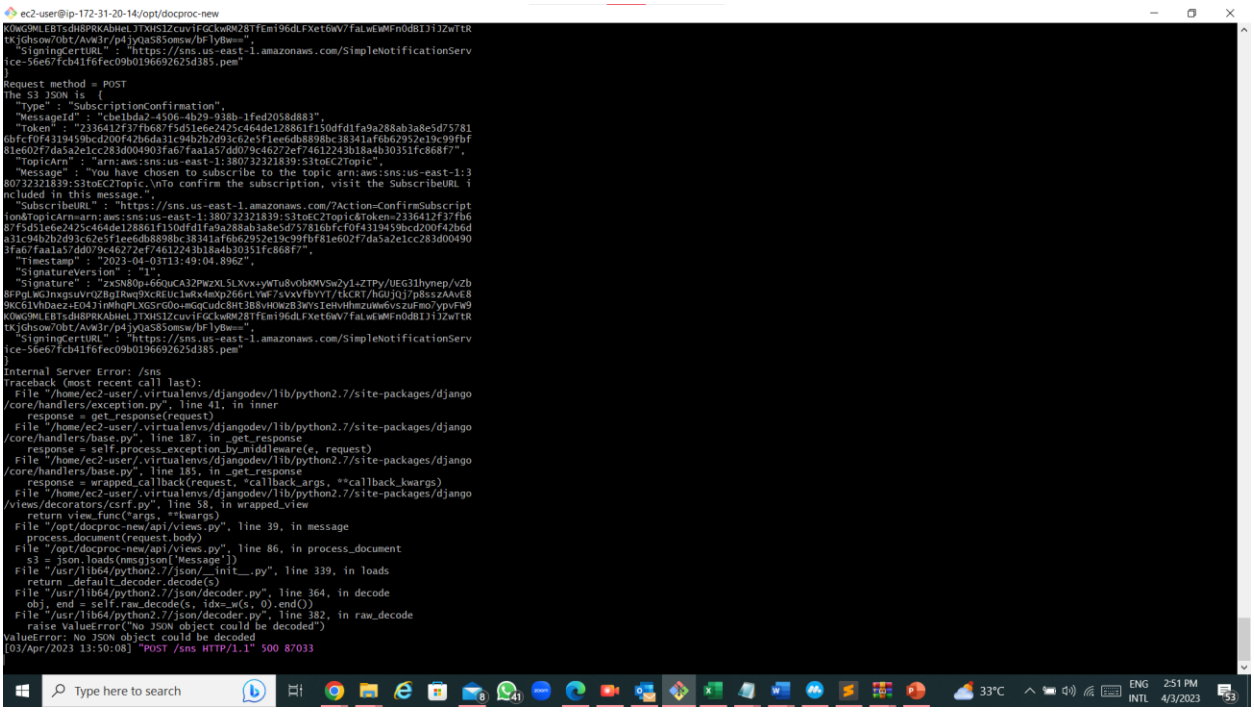


Step number	b
Step name	Creation of SNS subscription
Instructions	1) Navigate to SNS in the AWS Console and select the topic S3ToEC2Topic 2) Click on Create Subscription 3) Enter the following details Protocol : HTTP Endpoint : http://<host>:8080/sns where <host> in the public IP of the EC2 instance Click on Create Subscription 4) In the EC2 terminal window, look for the field "SubscribeURL" and copy the entire link given **Note: If a message is seen "ValueError: No JSON object could be decoded", it can be safely ignored** 5) Paste that link into a browser window to verify the SNS subscription (Ignore any messages received in the web browser)
Expected screenshots	1) Subscription URL in EC2 terminal Window

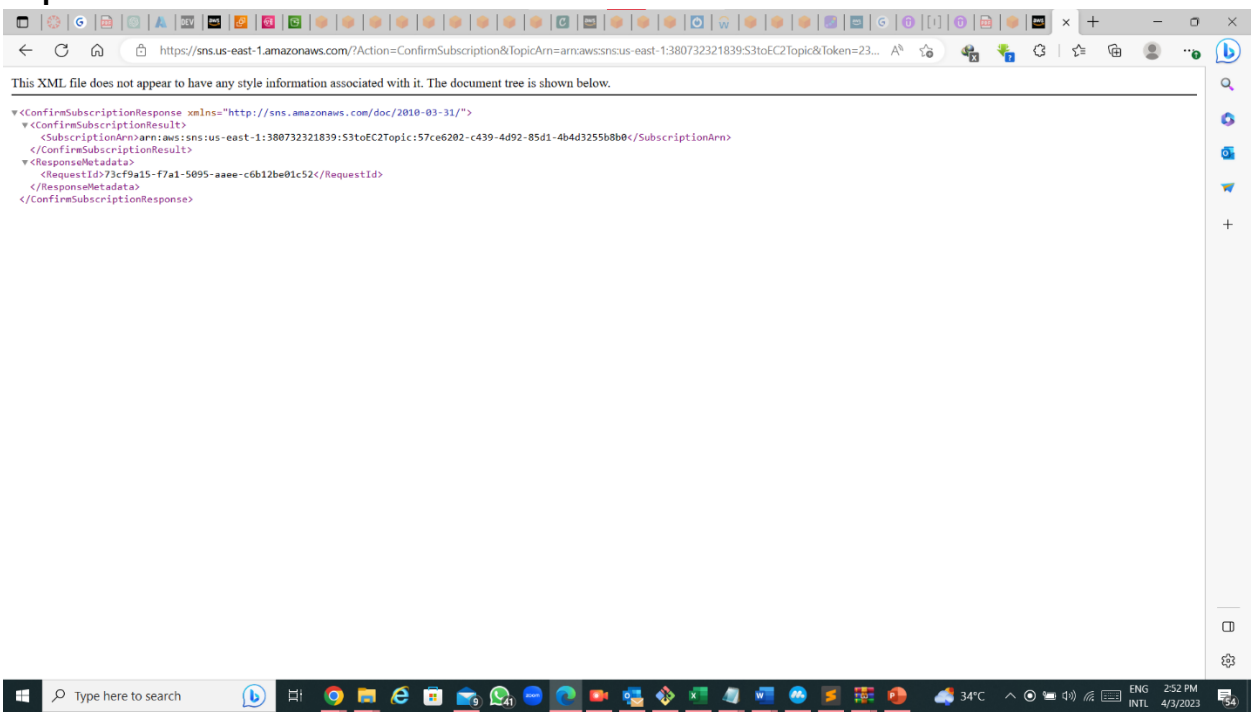
19. Subscription URL in EC2 terminal window (i)



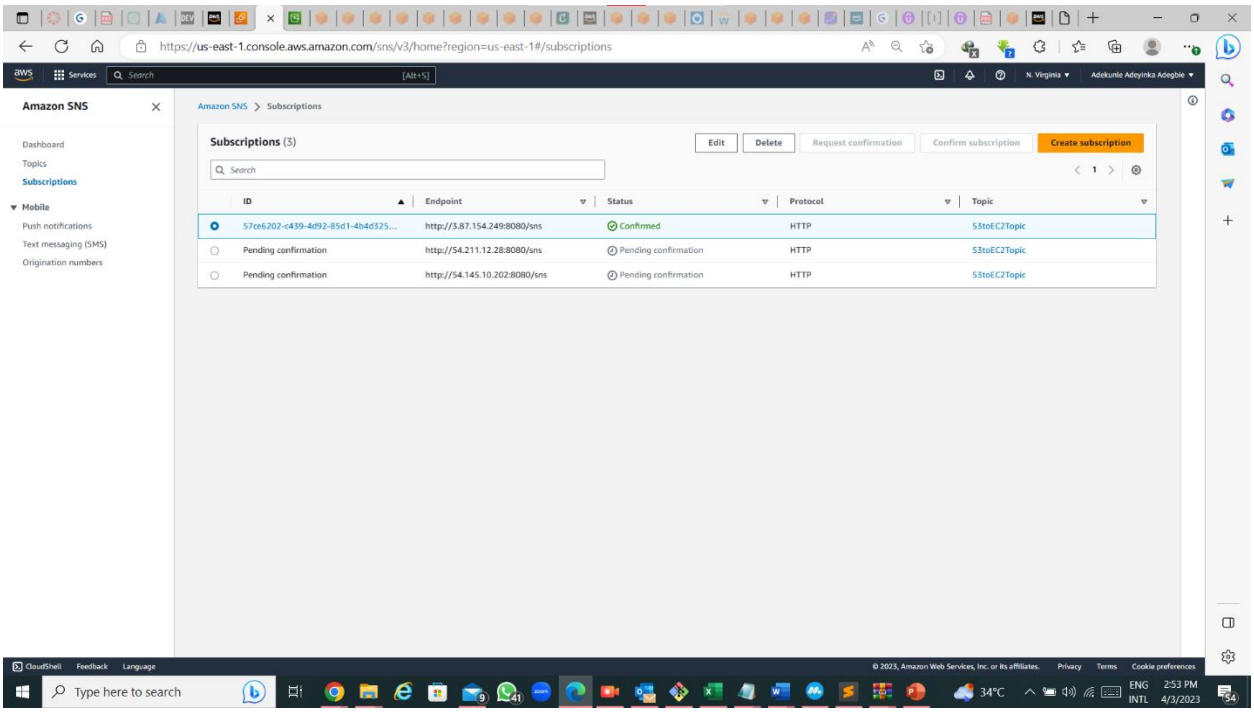
20. Subscription URL in EC2 terminal window (ii)



21. Copied SubscribeURL link in a browser

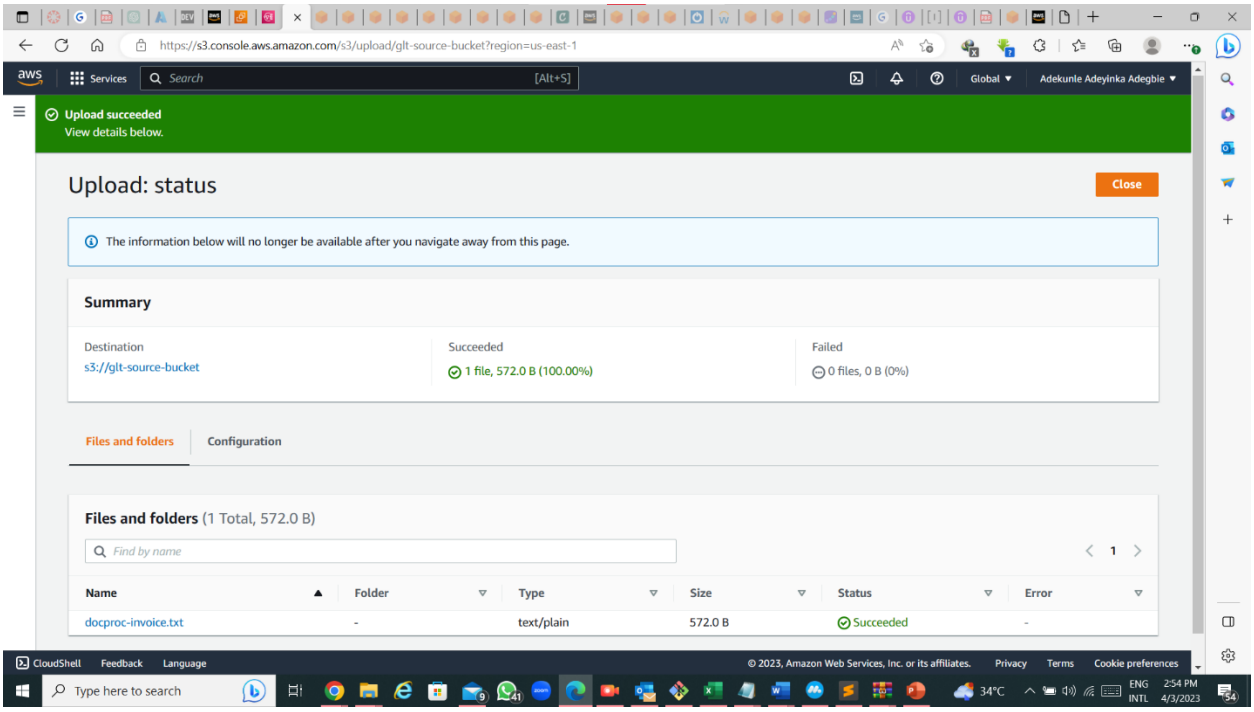


22. SNS Subscription showing confirmed status

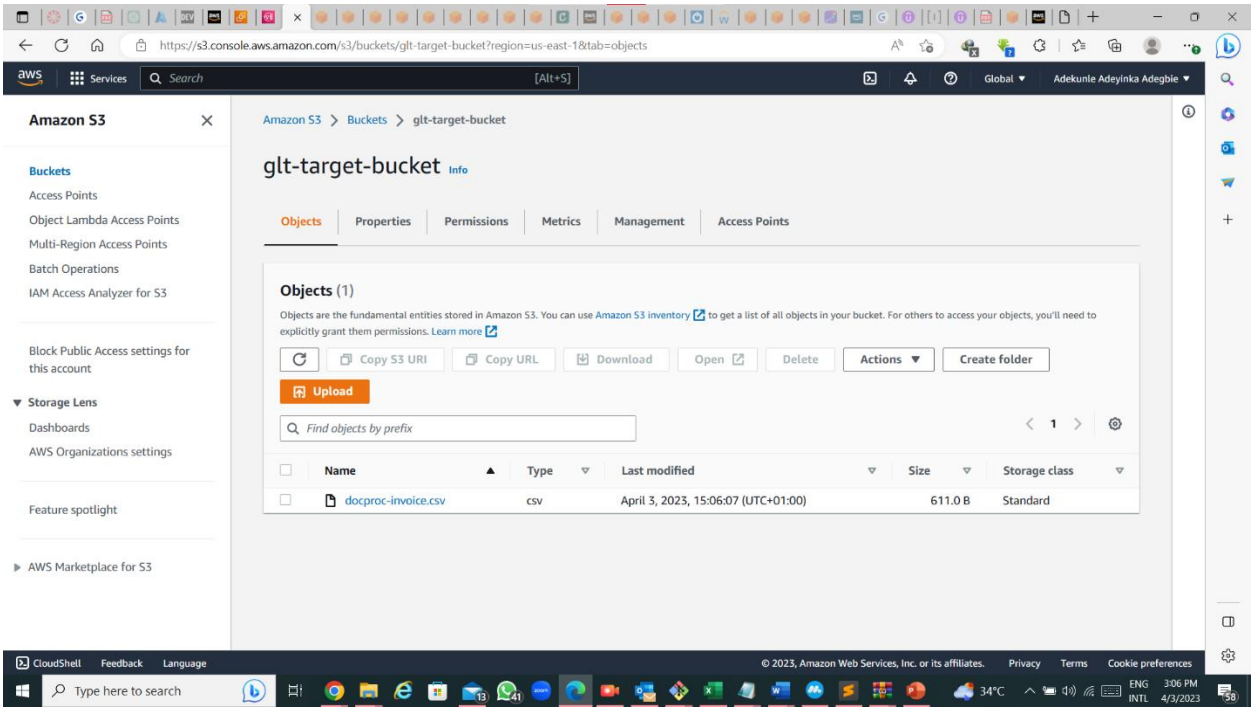


Step number	c
Step name	Generation of CSV file
Instructions	1) Download the file **docproc-invoice.txt** provided with this workbook 2) Navigate to S3 in the AWS Console 3) Upload the sample invoice file to the source S3 bucket using the default options 4) Verify that a CSV file is generated in the target S3 bucket. This may take a few minutes 5) (Optional) Login to the RDS instance using your preferred MySQL client and check the table created inside the specified database.
Expected screenshots	1) Generated CSV file in the target S3 bucket

23. Docproc-invoice.txt file uploaded into the “glt-source-bucket”



24. Generated CSV file in “glt-target-bucket”



Answer the following questions

Q 1 Which of the following properties of an AWS resource is sufficient and necessary to uniquely identify it across all of AWS?

a) ARN

b) Region and ARN

c) ARN and Account number

d) Depends on the resource used

Enter your answer here | A |

Q 2 Which of the following step numbers in Step 1 allowed S3 to publish to the SNS topic created?

a) 1(a)

b) 1(c)

c) 1(d)

d) 1(b)

Enter your answer here | D |

Q Which port is being used by SNS to send the notification to the custom program?
3

a) 8081

b) 80

c) 8080

d) 8065

Enter your answer here | B

Q How many IAM roles can be attached to an EC2 instance at a time?
4

a) 2

b) 3

c) 1

d) Depends on the policies required

Enter your answer here | C

Q As a product manager, how would you describe the benefits of this architecture to an client, as compared to
5 an equivalent on-premises architecture?

As a product manager, the following are some benefits that I would highlight to a client about the benefits of using AWS SNS, RDS, and S3 over an on-premises architecture. This can enable such clients better meet both the needs of business and customers and stay ahead of competition:

1. **Scalability:** With AWS SNS, RDS, and S3, client can easily scale up or down their infrastructure based on the demands of their business, without having to invest in additional hardware or worry about capacity planning. This means that they can respond quickly to changes in their business needs and avoid over-provisioning or under-provisioning your infrastructure.

2. **Cost-effectiveness:** By using AWS SNS, RDS, and S3, client can avoid the upfront costs associated with purchasing and maintaining hardware, and instead pay only for the resources they use. This means that client can reduce their capital expenditure and shift to an operational expenditure model, which can improve their cash flow and help them manage costs more effectively.

3. **Resilience and high availability:** AWS SNS, RDS, and S3 provide available and fault-tolerant infrastructure that is designed to minimize downtime and provide reliable access to data. With AWS SNS, client can ensure that they messages are delivered reliably to their subscribers, even in the event of a failure in one of their systems. With AWS RDS, they can leverage automatic failover and backup and recovery features to ensure that their databases are always available and protected. With AWS S3, they can replicate their data across multiple availability zones within a region, providing automatic failover in case of a failure in one zone.

4. **Security and compliance:** AWS SNS, RDS, and S3 provide a wide range of security and compliance features to help client protect their data, including built-in encryption, identity and access management, and compliance certifications. AWS also provides them with visibility and control over your infrastructure, which can help you meet your regulatory requirements and improve your overall security posture.

5. **Innovation and agility:** AWS SNS, RDS, and S3 provide a wide range of services and tools that enable client to innovate quickly and bring new products and features to market faster. With AWS, they can take advantage of the latest technologies and best practices without having to worry about managing the underlying infrastructure.

Grades distribution	
MCQs	10 (2.5 mark each)
Subjective questions	6 marks
Implementation screenshots	24 marks (2 marks each)
Total	40 marks