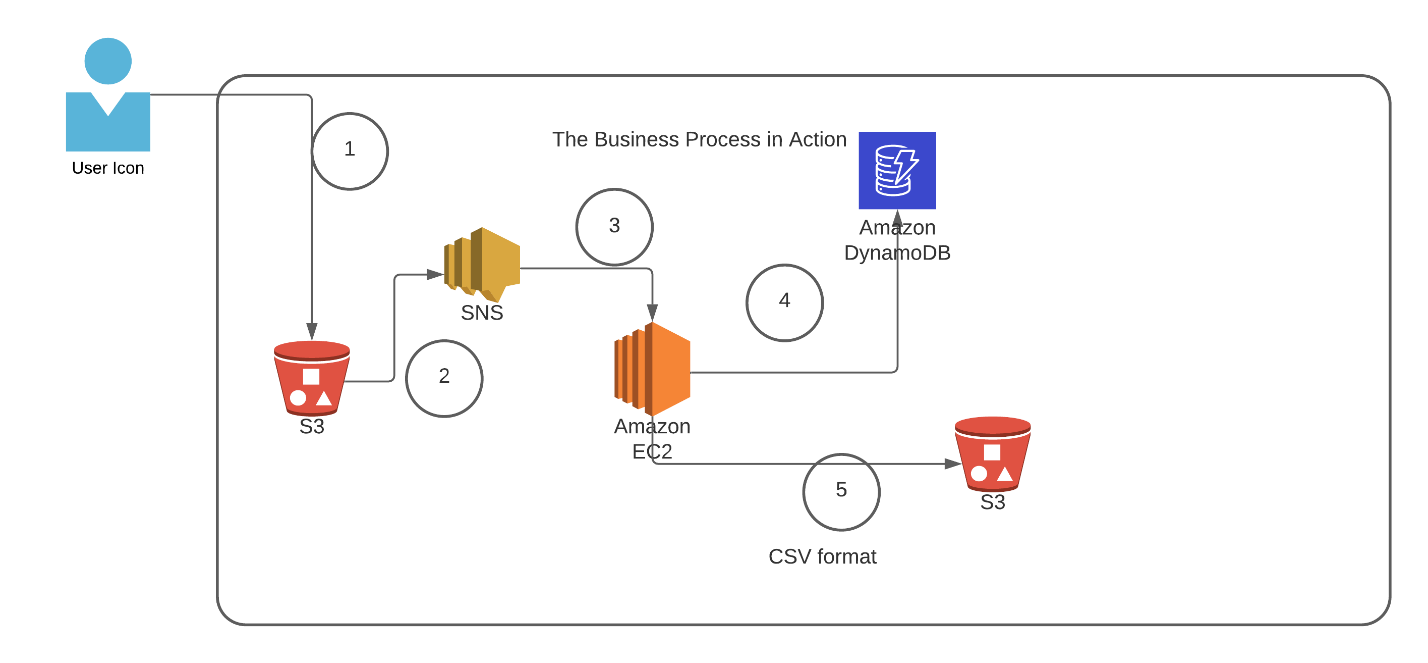
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Declaration** |  | | | | |
| Questions in this exercise are intentionally complex and could be convoluted or confusing. This is by design and to simulate real life situations where customers seldom give crystal clear requirements and ask unambiguous questions. | | | | | |
|
|
| I have read the above statement and agree to these conditions | | | | | |
| I AGREE | Rik Kisnah | | | | |
| <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 points | | | | | |
| 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\_PROJECT2.  For example: PGPCCMAY18\_JOHN\_DOE\_PROJECT2.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. | | | | | |

Scenario

In the connected world, it is imperative that the organizations be interlinked with the customers and vendors. This process has been very sluggish, manual, batch-based and prone to failures. Such integration design has lead to impaired decision-making and delay in detection of fraudulent actions.

The objective is to create an automated, event-based real-time process that does not have these limitations. Data should flow rapidly from the source to the destination in addition to maintaining a data lake of structured and unstructured data.

**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 along with saving the original record in DynamoDB |
| 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 invoicehave been shared along with this workbook on the LMS. |

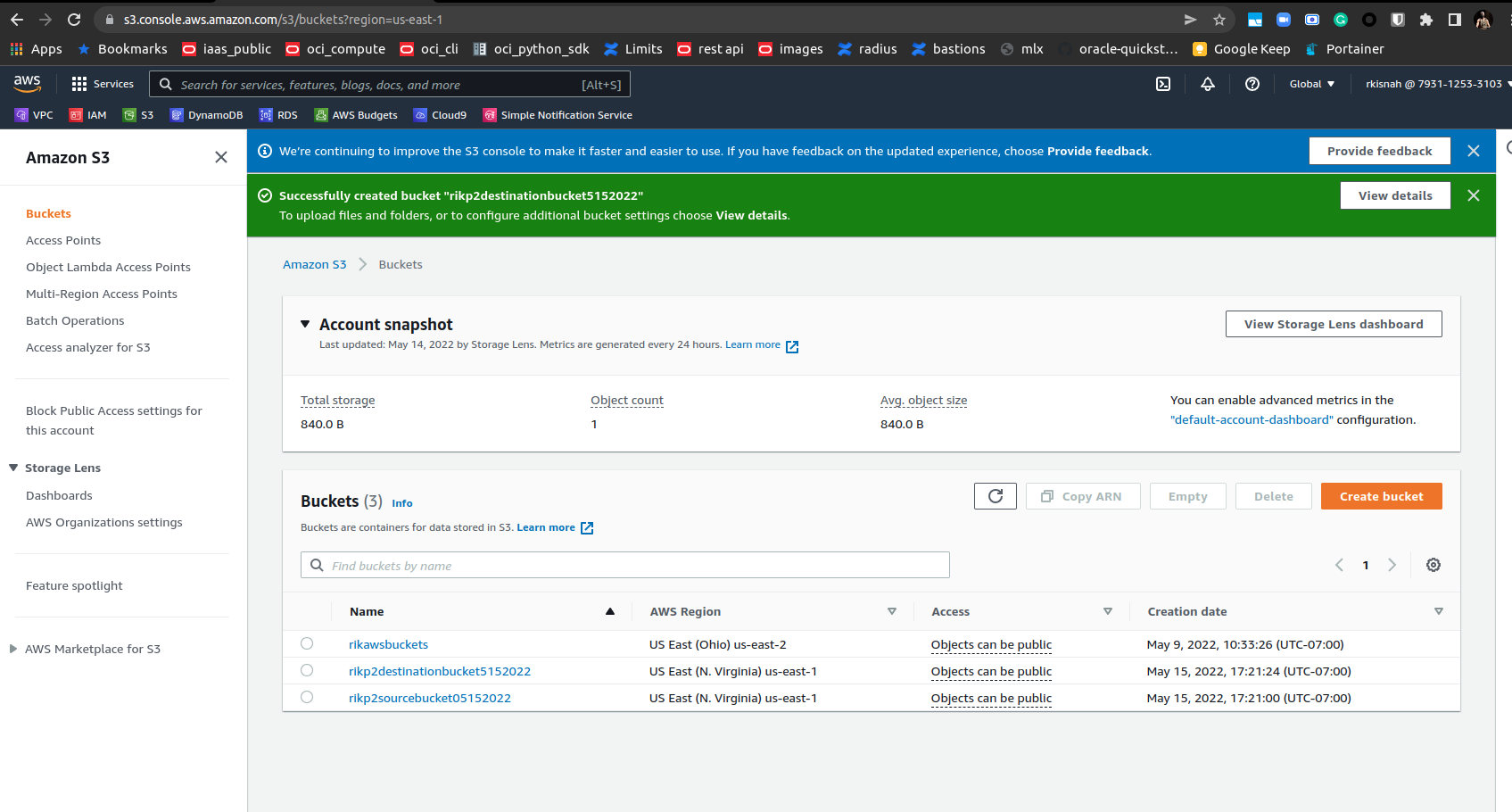
**Step 1: SNS and S3 topic creation**

|  |  |
| --- | --- |
| 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 |

**<Insert screenshot for a(1) here>**

**source bucket:** rikp2sourcebucket05152022 arn:aws:s3:::rikp2sourcebucket05152022

**target bucket:** rikp2destinationbucket5152022 arn:aws:s3:::rikp2destinationbucket5152022

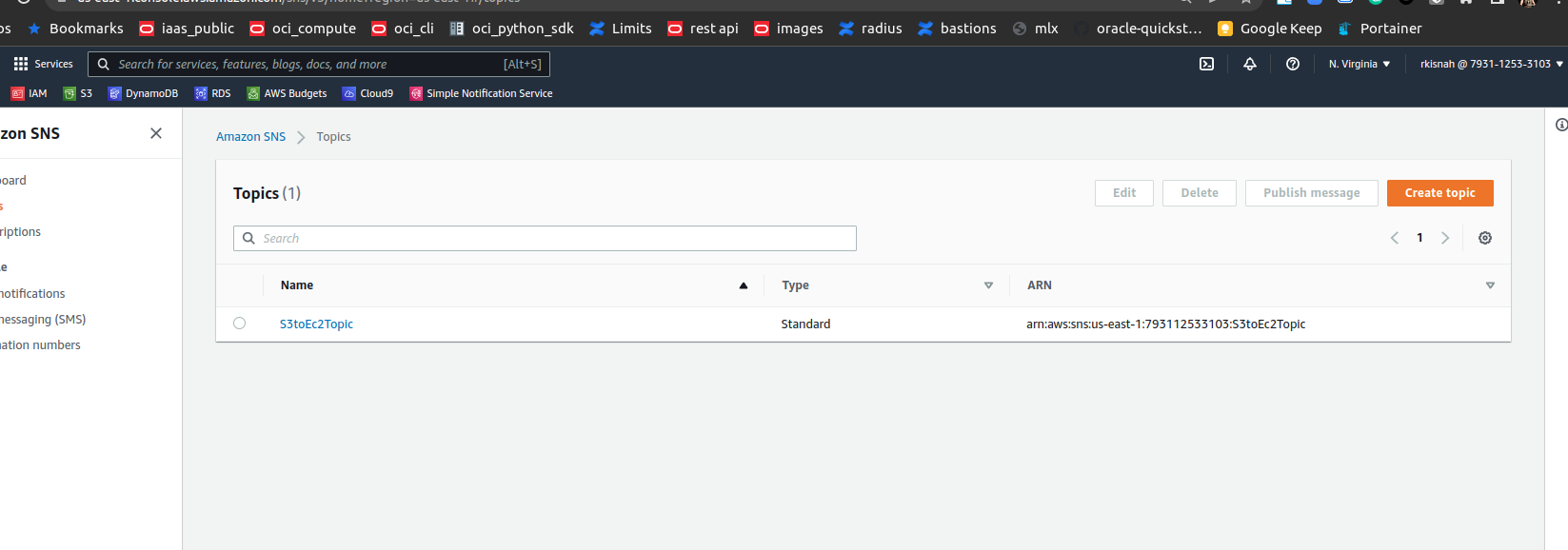


|  |  |  |
| --- | --- | --- |
| Step number | b |  |
| Step name | Creation of SNS subscription |  |
| Instructions | 1) Navigate to SNS -> Topics  2) Click on "Create Topic"  3) Enter the following fields  Name : S3toEC2Topic  Type : Standard  The other options can be ignored for now  4) Click on Create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

**<Insert screenshot for b(1) here>**

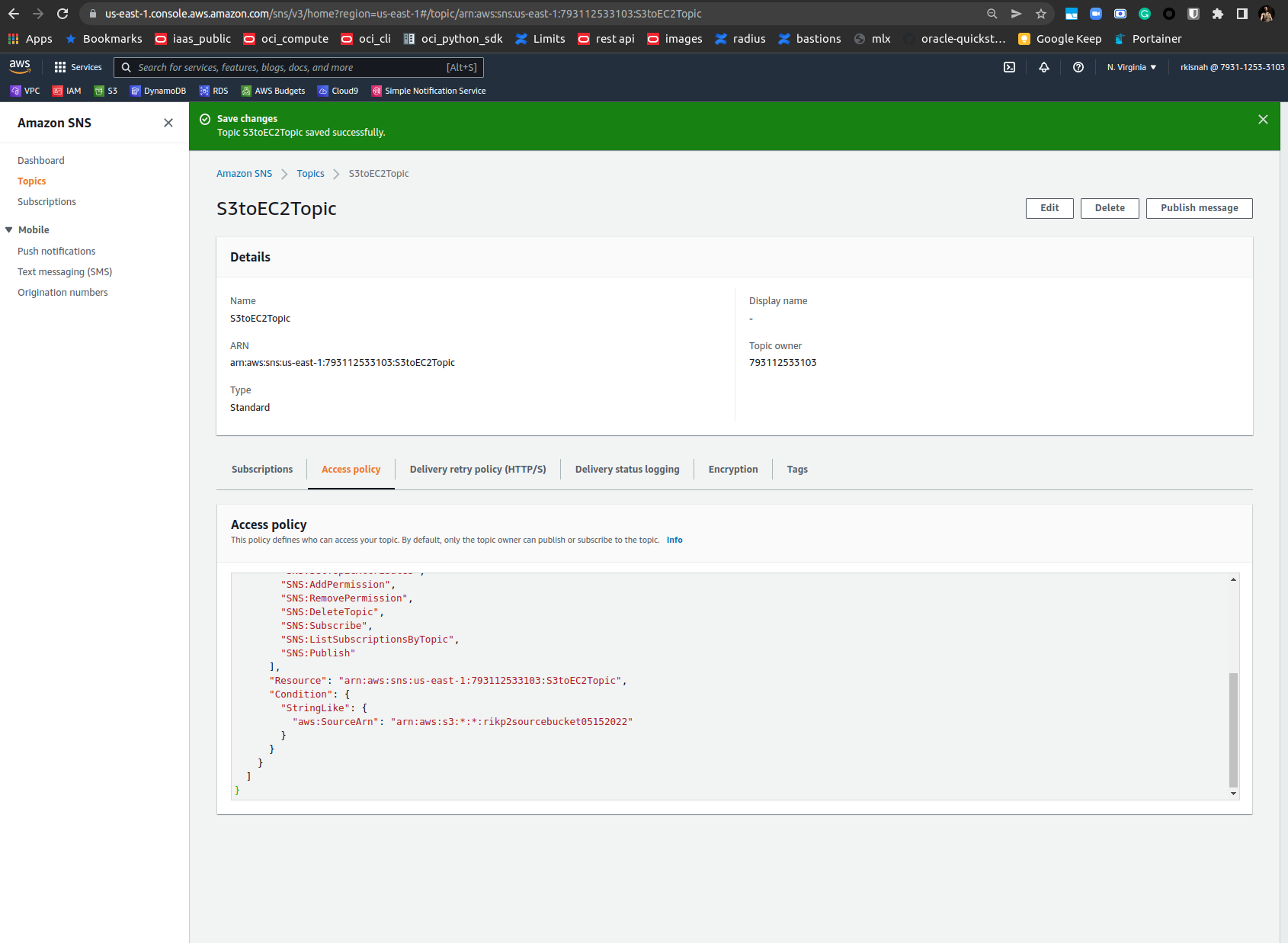
Name: S3toEc2Topic

ARN: arn:aws:sns:us-east-1:793112533103:S3toEc2Topic



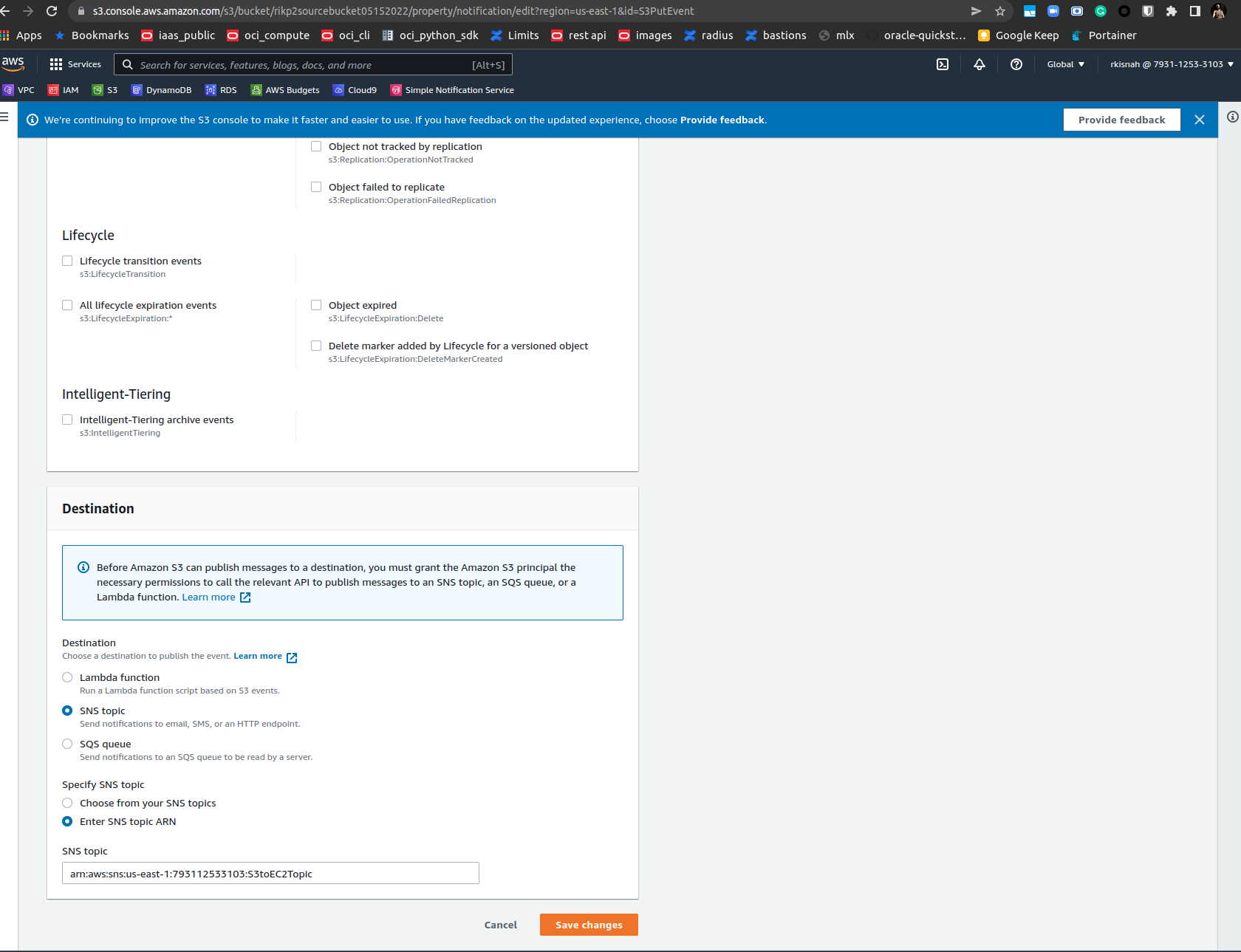
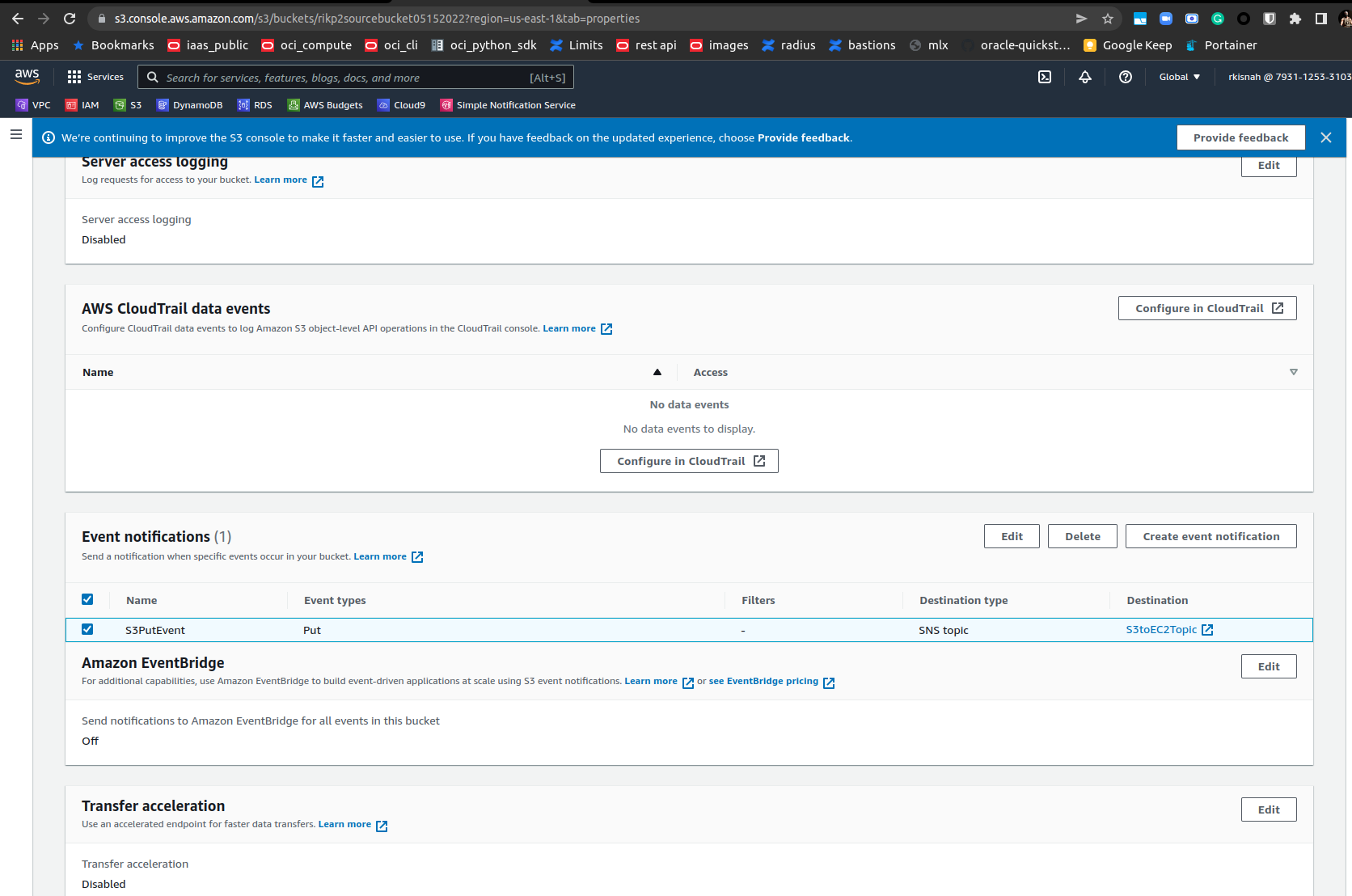
|  |  |
| --- | --- |
| Step number | c |
| Step name | Modification of SNS Access Policy |
| Instructions | 1) Navigate to SNS -> Topics and select the topic created in the previous step  2) Note down the ARN shown in the topic details  2) Click on Edit and select "Access Policy".  3) Replace the text 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 |

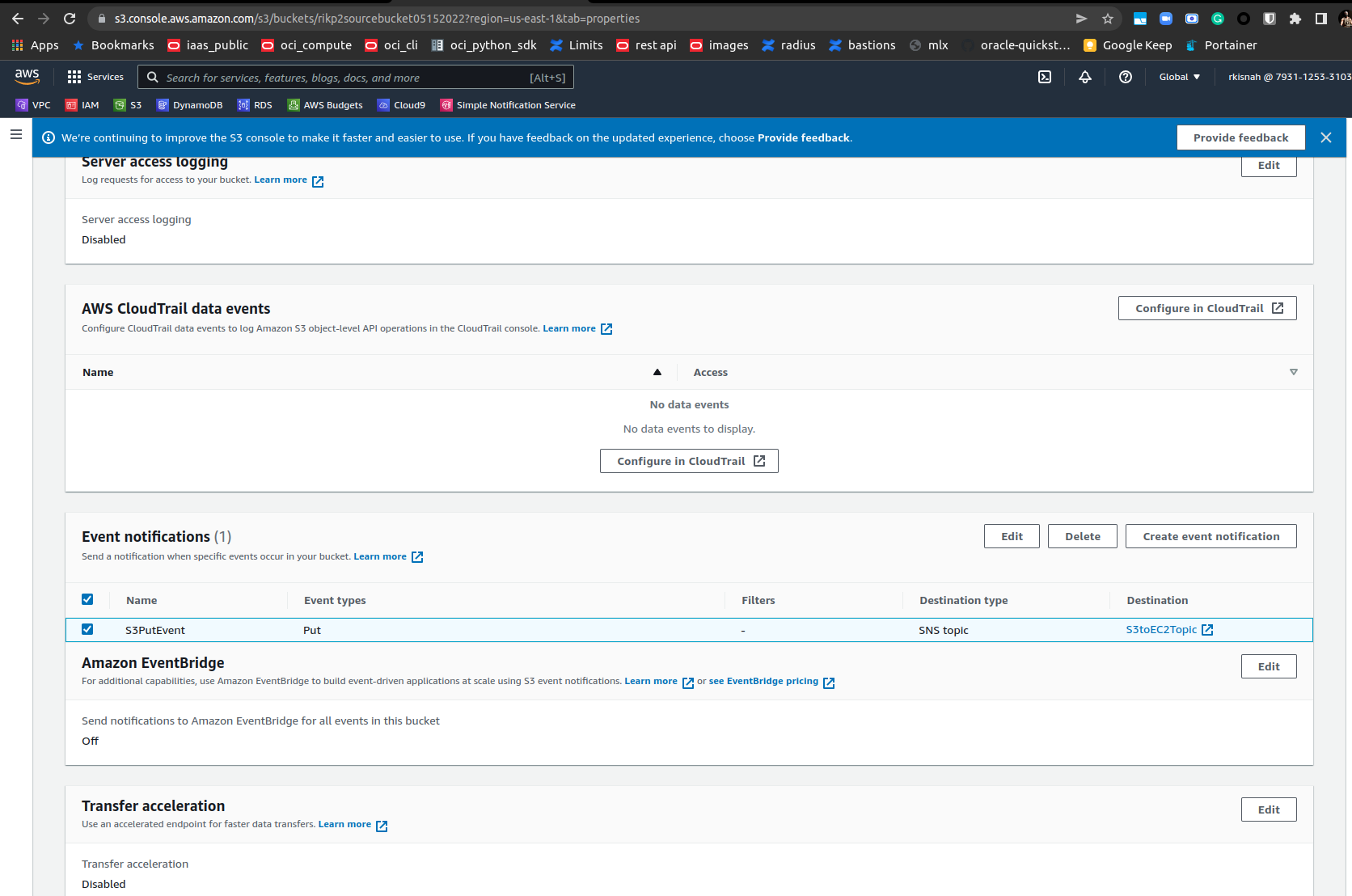
**<Insert screenshot for c(1) here>**



|  |  |
| --- | --- |
| 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 |

**<Insert screenshot for d(1) here>**

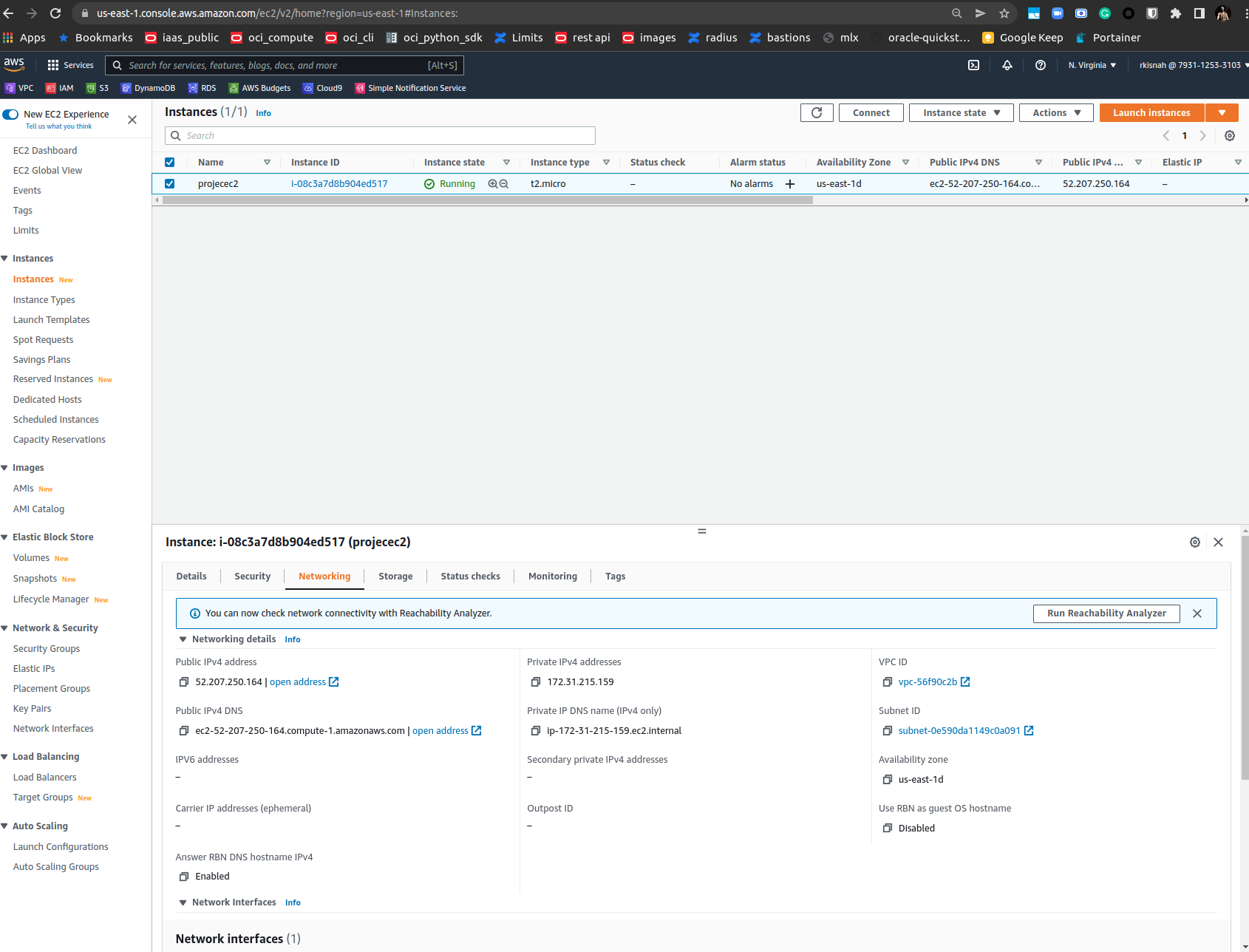




Step 2: Run the custom program in the EC2 instance

|  |  |  |
| --- | --- | --- |
| Step number | a |  |
| Step name | Creation of the EC2 instance |  |
| Instructions | 1) Navigate to EC2 -> Instances  2) Create an EC2 instance with the following parameters  AMI : Amazon Linux 2 AMI  VPC : Default  Security group : Ports 22 and 8080 should be opened |  |
| Expected screenshots | 1) List of instances after creation of EC2 instance |  |

**<Insert screenshot for a(1) here>**



#Connecting to the EC2

rkisnah@machine1-system76 ~/.aws:()$ eval $(ssh-agent)

Agent pid 191320

rkisnah@machine1-system76 ~/.aws:()$ ssh -i "rik\_key\_pair\_aws.pem" ec2-user@ec2-52-207-250-164.compute-1.amazonaws.com

Last login: Tue May 17 04:27:22 2022 from 160.34.93.163

\_\_| \_\_|\_ )

\_| ( / Amazon Linux 2 AMI

\_\_\_|\\_\_\_|\_\_\_|

https://aws.amazon.com/amazon-linux-2/

2 package(s) needed for security, out of 5 available

Run "sudo yum update" to apply all updates.

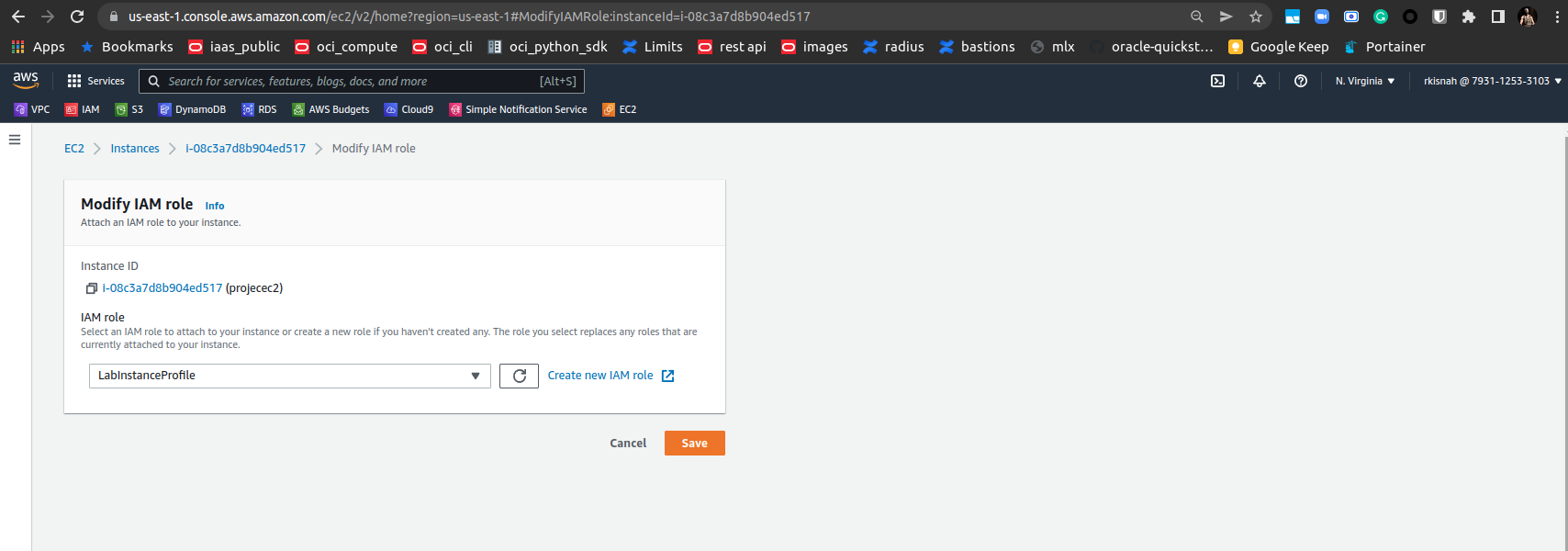
[ec2-user@ip-172-31-215-159 ~]$

[ec2-user@ip-172-31-215-159 .ssh]$ cat authorized\_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCU2c1NwuLkai6kRhzsoOPUdZJzO4brqMwT/zUVAUqDavLBuyUO09yjAIFVJXmADxVmyWDmDbvuLZzO4LUz3Crw8ROFhT/PkYnLLwhfl8eSn4neWolnlLKL3m/bkB3+ykWFoW72D/0xOuJ/tgRD3ZSloEO6gTAz7Br66U+9nlN6E6+A2w7m3qNRT0EUag2hQWuACHdrAXGcCDPvm3Auj2lqbiY+1X29F8YCXMtOUygkzl9twLjflyYCpJ2vFEf+1hRKSI+iFRObNQ5Fh9DOUDBbNv1sj9VEbeGBMoRlUcZittBBAebuKtV0DD6J7y3cQkQuJitCgsOokF4sdKfjQem9 rik\_key\_pair\_aws

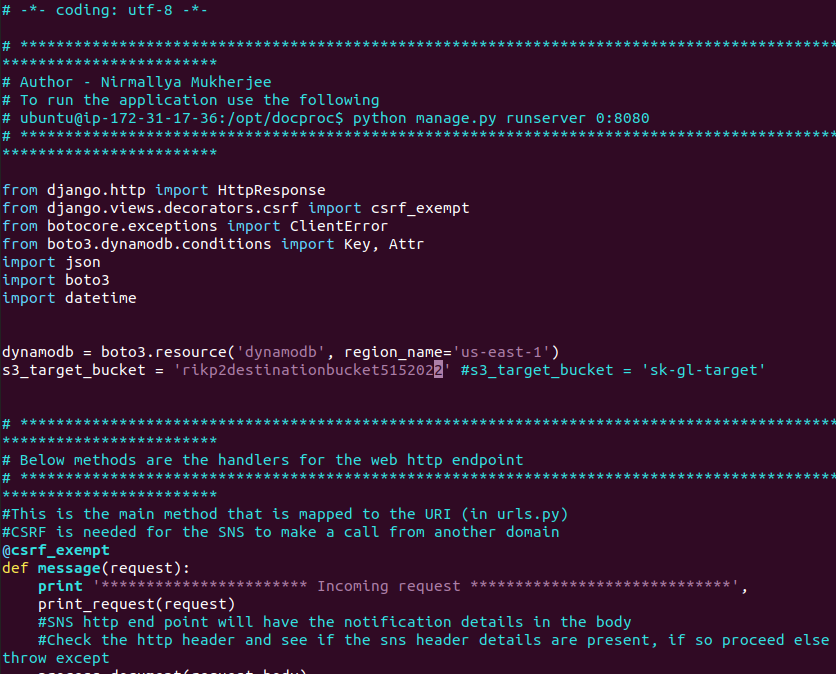
|  |  |  |
| --- | --- | --- |
| Step number | b |  |
| Step name | Creation of IAM role for EC2 instance |  |
| Instructions | 1) Navigate back to EC2  Instances  2) Select the EC2 instance created in the previous  step and select Actions  Security  Modify IAM role  3) Select the role LabInstanceProfile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

**<Insert screenshot for b(1) here>**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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](http://views.py/) in the API folder using a text editor  4) In line number 19, modify the target bucket name to the one created in Step 2 (a) 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](http://views.py/) file to point to the target bucket | 2)Copying the folder to the EC2 instance |  |  |

**<Insert screenshot for c(1) here>**



import json

import boto3

import datetime

dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')

s3\_target\_bucket = 'rikp2destinationbucket5152022' #s3\_target\_bucket = 'sk-gl-target'

**<Insert screenshot for c(2) here>**

rkisnah@machine1-system76 ~/src/sirhomersimpson/pgp-cc/project2/docproc-new:(main)$ eval $(ssh-agent)

Agent pid 192361

rkisnah@machine1-system76 ~/src/sirhomersimpson/pgp-cc/project2/docproc-new:(main)$ scp -i /home/rkisnah/.aws/rik\_key\_pair\_aws.pem -r /home/rkisnah/src/sirhomersimpson/pgp-cc/project2/docproc-new ec2-user@ec2-52-207-250-164.compute-1.amazonaws.com:/home/ec2-user

tests.py 100% 125 1.5KB/s 00:00

models.py 100% 122 1.2KB/s 00:00

views.py 100% 7948 86.9KB/s 00:00

apps.py 100% 146 1.7KB/s 00:00

admin.py 100% 128 1.5KB/s 00:00

\_\_init\_\_.py 100% 0 0.0KB/s 00:00

\_\_init\_\_.py 100% 0 0.0KB/s 00:00

settings.py 100% 3129 36.6KB/s 00:00

urls.py 100% 798 9.7KB/s 00:00

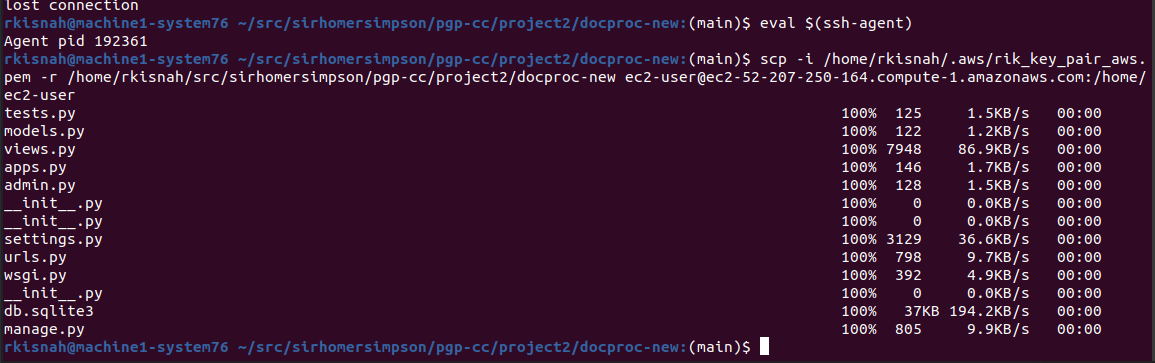
wsgi.py 100% 392 4.9KB/s 00:00

\_\_init\_\_.py 100% 0 0.0KB/s 00:00

db.sqlite3 100% 37KB 194.2KB/s 00:00

manage.py 100% 805 9.9KB/s 00:00

rkisnah@machine1-system76 ~/src/sirhomersimpson/pgp-cc/project2/docproc-new:(main)$



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 -W ignore manage.py runserver 0:8080  **Keep this terminal window open throughout the rest of the exercise** | | | |
| Expected screenshots | 1) Server in waiting state |  |  |  |

#Step 1 go to node

ssh -i "rik\_key\_pair\_aws.pem" ec2-user@ec2-52-207-250-164.compute-1.amazonaws.com

# Step 2 On node launch the server

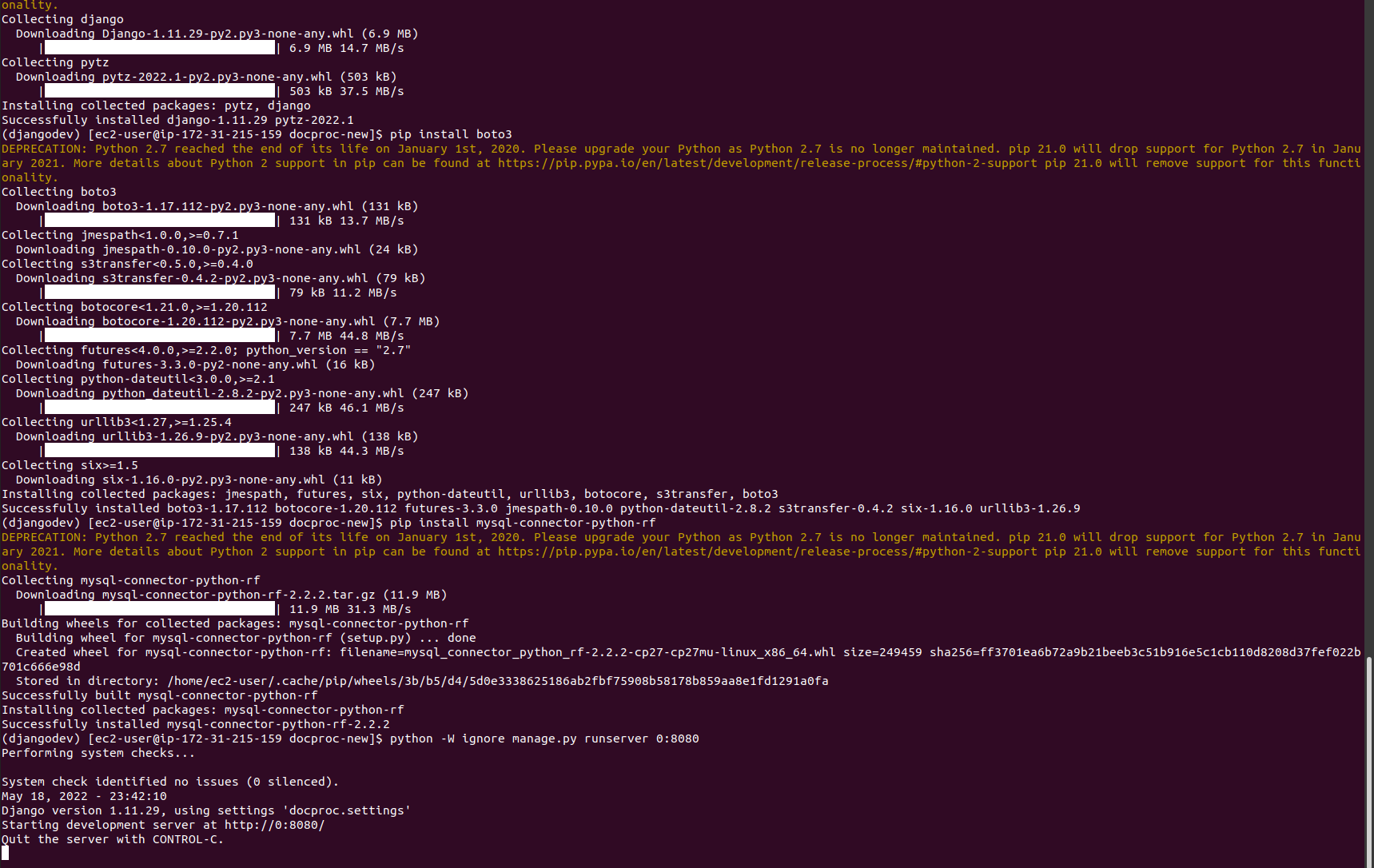
source ~/.virtualenvs/djangodev/bin/activate

python -W ignore /opt/docproc-new/manage.py runserver 0:8080

# Step 3 check it has access from outside the VPN from a local dev

curl http://ec2-52-207-250-164.compute-1.amazonaws.com:8080/

**<Insert screenshot for a(1) here>**

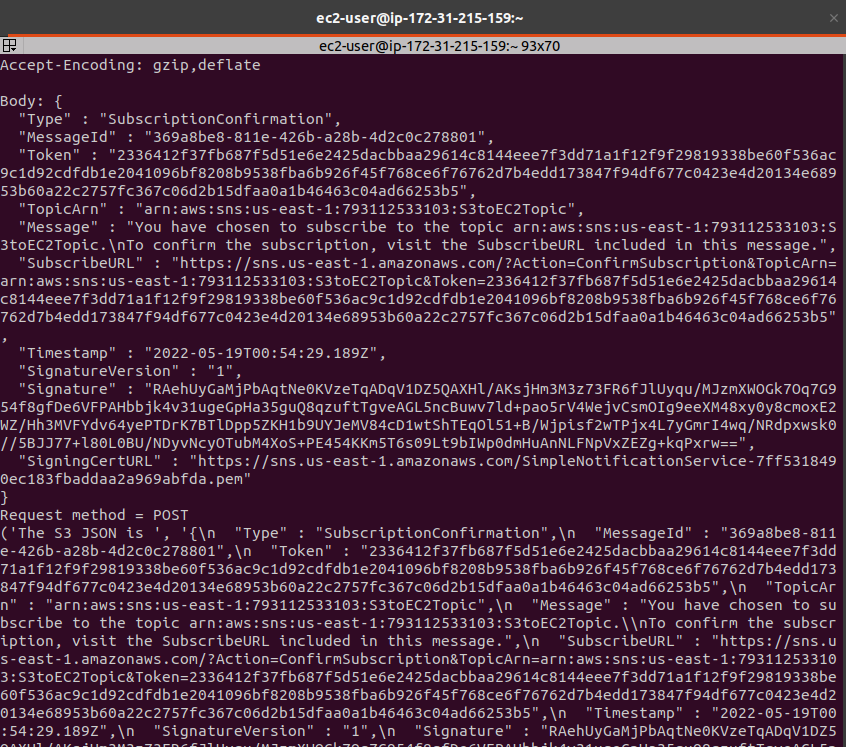


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | b |  |  |  |
| Step name | Creation of SNS subscription |  |  |  |
| Instructions | 1) Navigate to SNS in the AWS Console and select the topic S3ToEC2Topic  2) Click on Create Subscription  3) Enter the following details  Protocol : HTTP  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 |  |  |  |

http://52.207.250.164:8080/sns

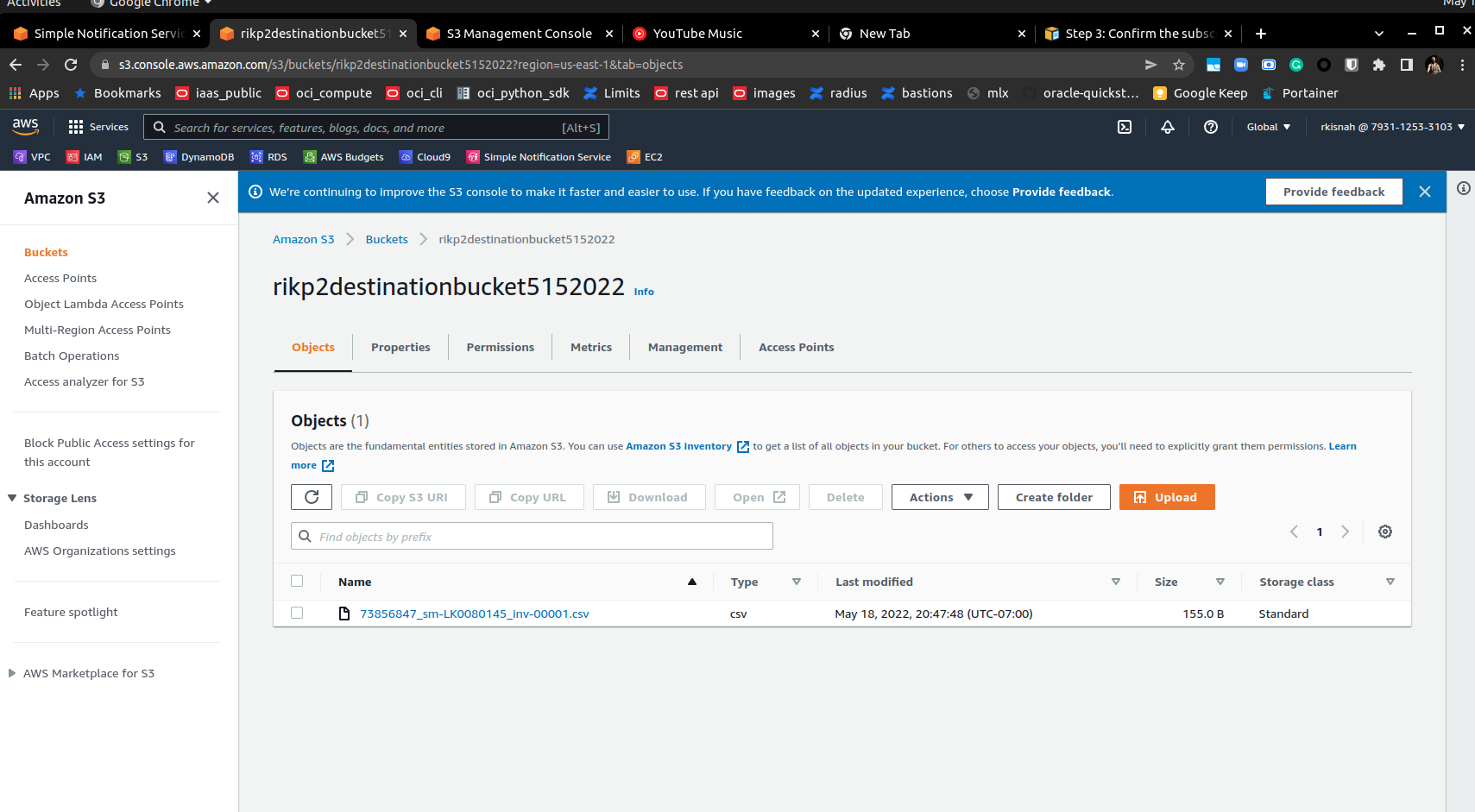
"SubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-1:793112533103:S3toEC2Topic&Token=2336412f37fb687f5d51e6e2425dacbbaa29614c8144eee7f3dd71a1f12f9f29819338be60f536ac9c1d92cdfdb1e2041096bf8208b9538fba6b926f45f768ce6f76762d7b4edd173847f94df677c0423e4d20134e68953b60a22c2757fc367c06d2b15dfaa0a1b46463c04ad66253b5",

**<Insert screenshot for b(1) here>**

****

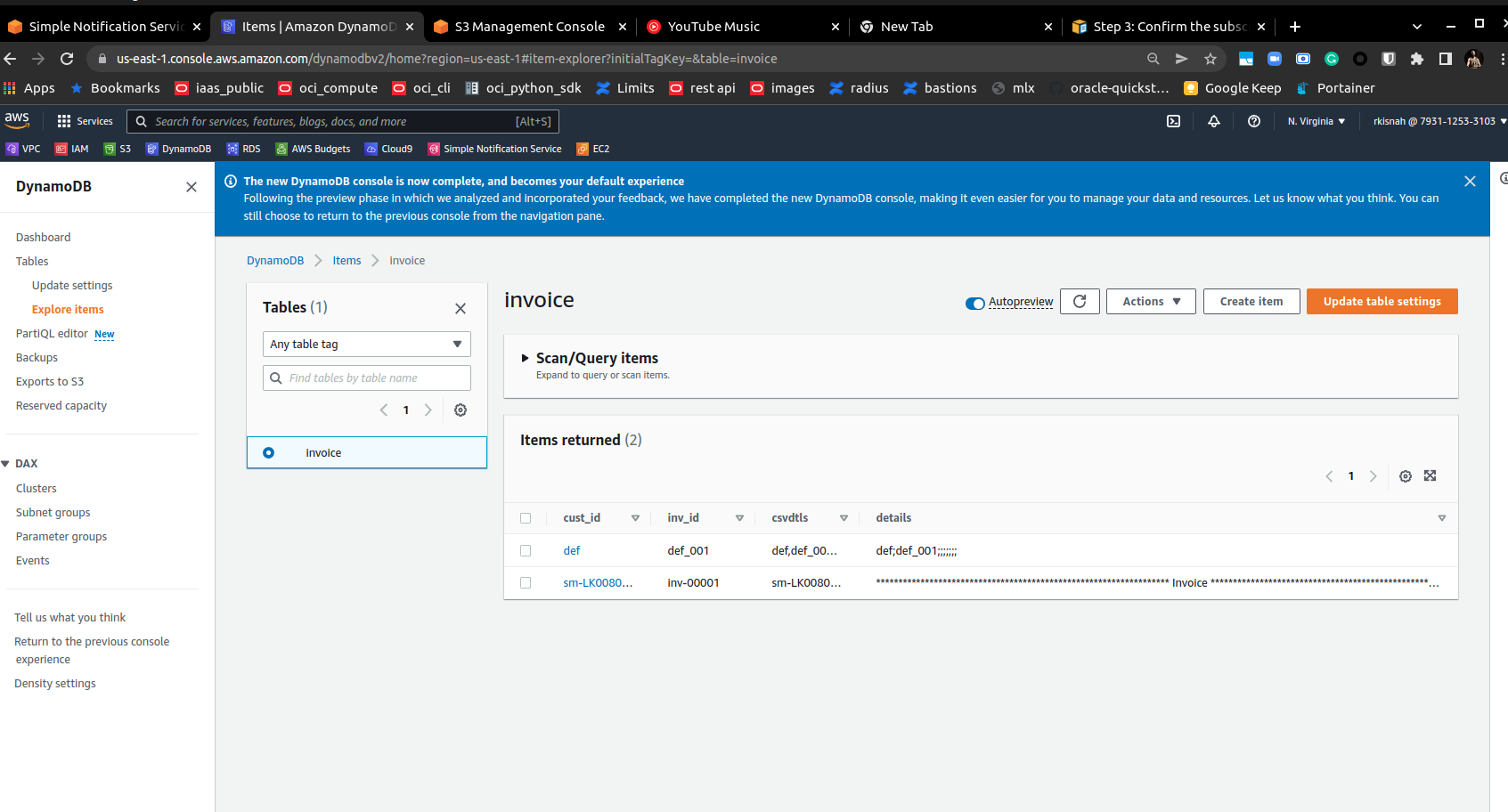
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | c |  |  |  |
| Step name | Generation of CSV file |  |  |  |
| Instructions | 1) Download the file **docproc-invoice.txt** provided with this workbook  2) Navigate to S3 in the AWS Console  3) Upload the sample invoice file to the source 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 | | | |
| Expected screenshots | 1) Generated CSV file in the target S3 bucket | |  |  |

**<Insert screenshot c(1) here>**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step number | d |  |  |  |
| Step name | Table creation in DynamoDB |  |  |  |
| Instructions | 1) Navigate to DynamoDB using the Services Menu  2) Click on tables on the left side  3) Select the table "invoice"  4) Click on the "Items" tab and verify that a record has been created in the table with the contents of the invoice file. | | | |
| Expected screenshots | 1) Items tab showing the table records | |  |  |

**<Insert screenshot for d(1) here>**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Answer the following questions** | | | | **Points** |
| Q1 | 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 |  |  |
|  |  |  |  |  |
| Q2 | Which of the following cannot be an subscriber for SNS topics? | | |  |
|  | a) VPC |  |  |  |
|  | b) Lambda |  |  |  |
|  | c) Email |  |  |  |
|  | d) None of these |  |  |  |
|  | Enter your answer here | d |  |  |
|  |  |  |  |  |
| Q3 | 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 | c |  |  |
|  |  |  |  |  |
| Q4 | Which port is being used by SNS to send the notification to the custom program? | | |  |
|  | a) 8081 |  |  |  |
|  | b) 80 |  |  |  |
|  | c) 8080 |  |  |  |
|  | d) 8065 |  |  |  |
|  | Enter your answer here | c |  |  |
|  |  |  |  |  |
| Q5 | How many IAM roles can be attached to an EC2 instance at a time? | | |  |
|  | a) 2 |  |  |  |
|  | b) 3 |  |  |  |
|  | c) 1 |  |  |  |
|  | d) Depends on the policies required |  |  |  |
|  | Enter your answer here | c |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Q6 | As a product manager, how would you describe the benefits of this architecture to an client, as compared to an equivalent on-premises architecture? | | | 6 |
|  |  | | |  |
|  | (1) cost saving – you pay on usage i.e SNS by traffic – no permanent CAPEX  (2) scaleable – S3 99.999% SLA, DynamoDB can automatically scale for traffic, EC2 can be configured via autoscaling | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  | **Max points** | **16** |

|  |  |
| --- | --- |
| **Grades distribution** |  |
| MCQs | 10 (2 point each) |
| Subjective questions | 6 points |
| Implementation screenshots | 24 points (2 points each) |
| Total | 40 points |