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

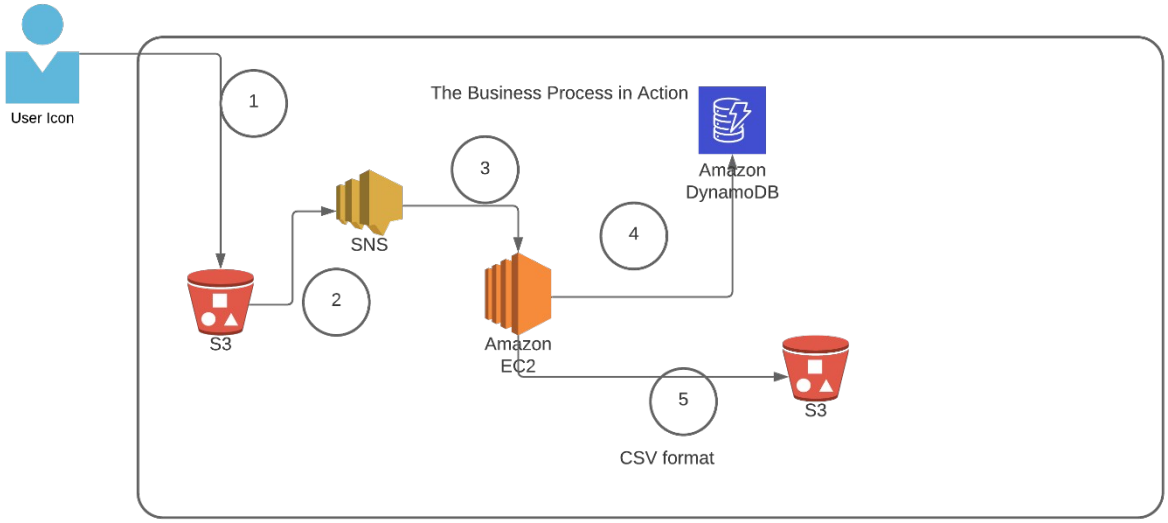
Resource Clean Up
Cloud is always pay per use model and all resources/services that we consume are chargeable. Cleaning up when you’ve completed your lab or project is always necessary. This is true whether you’re doing a lab or implementing a project at your workplace.
After completing the lab, make 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 invoice have been shared along with this workbook on the LMS.

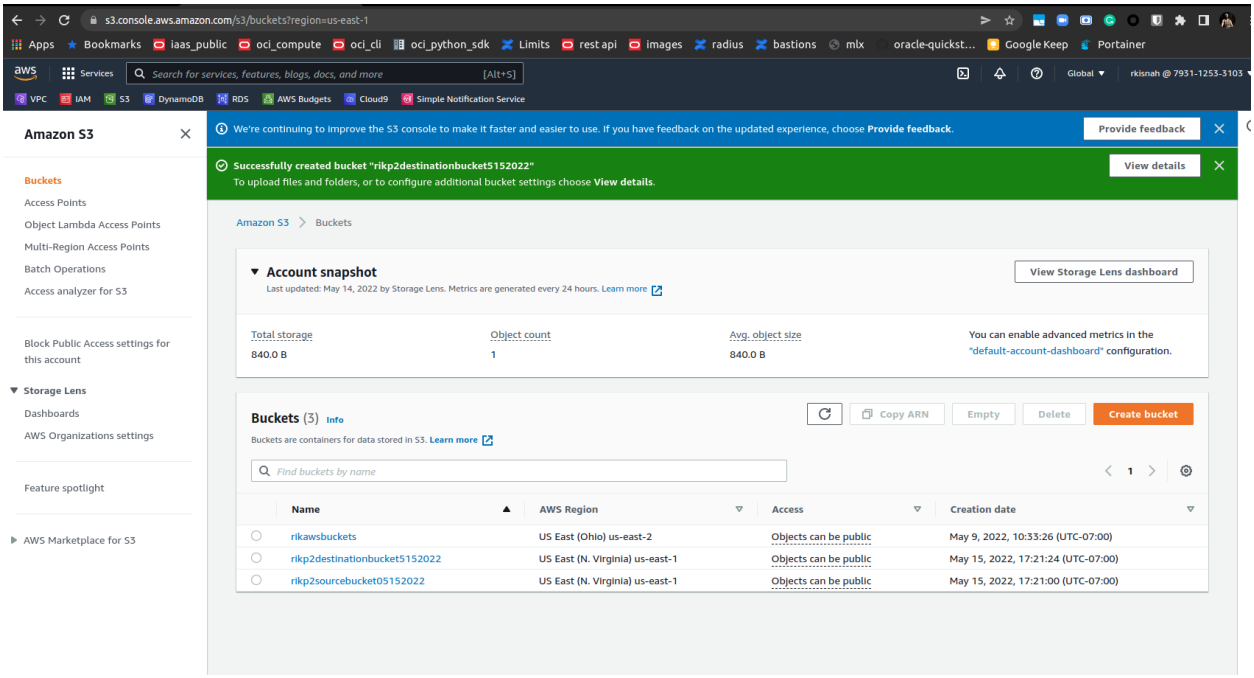
Step 1: SNS and S3 topic creation

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

<Insert screenshot for a(1) here>

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

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

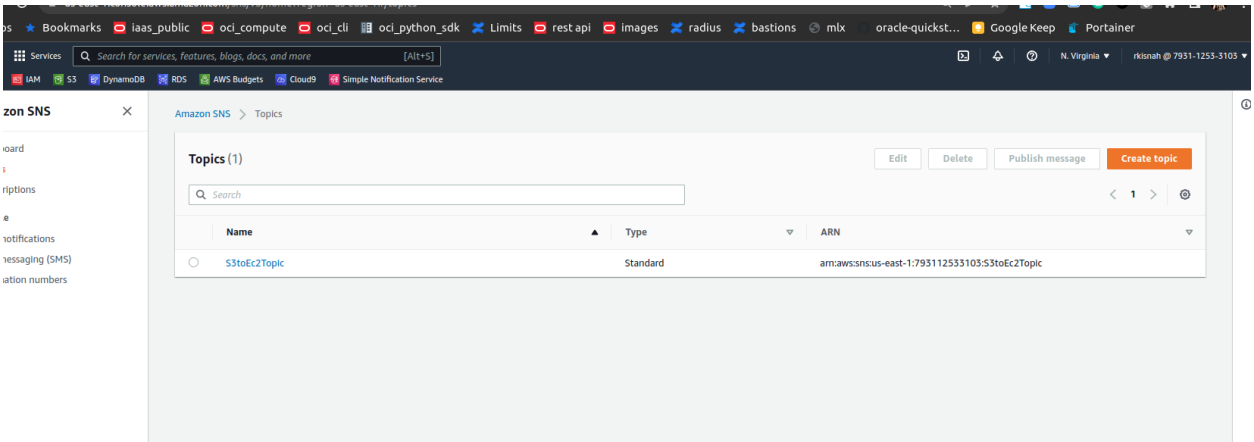


Step number	b
Step name	Creation of SNS subscription
Instructions	1) Navigate to SNS -> Topics 2) Click on "Create Topic" 3) Enter the following fields Name : S3toEC2Topic Type : Standard The other options can be ignored for now 4) Click on Create Topic
Expected screenshots	1) Creation of SNS topic

<Insert screenshot for b(1) here>

Name: S3toEc2Topic

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

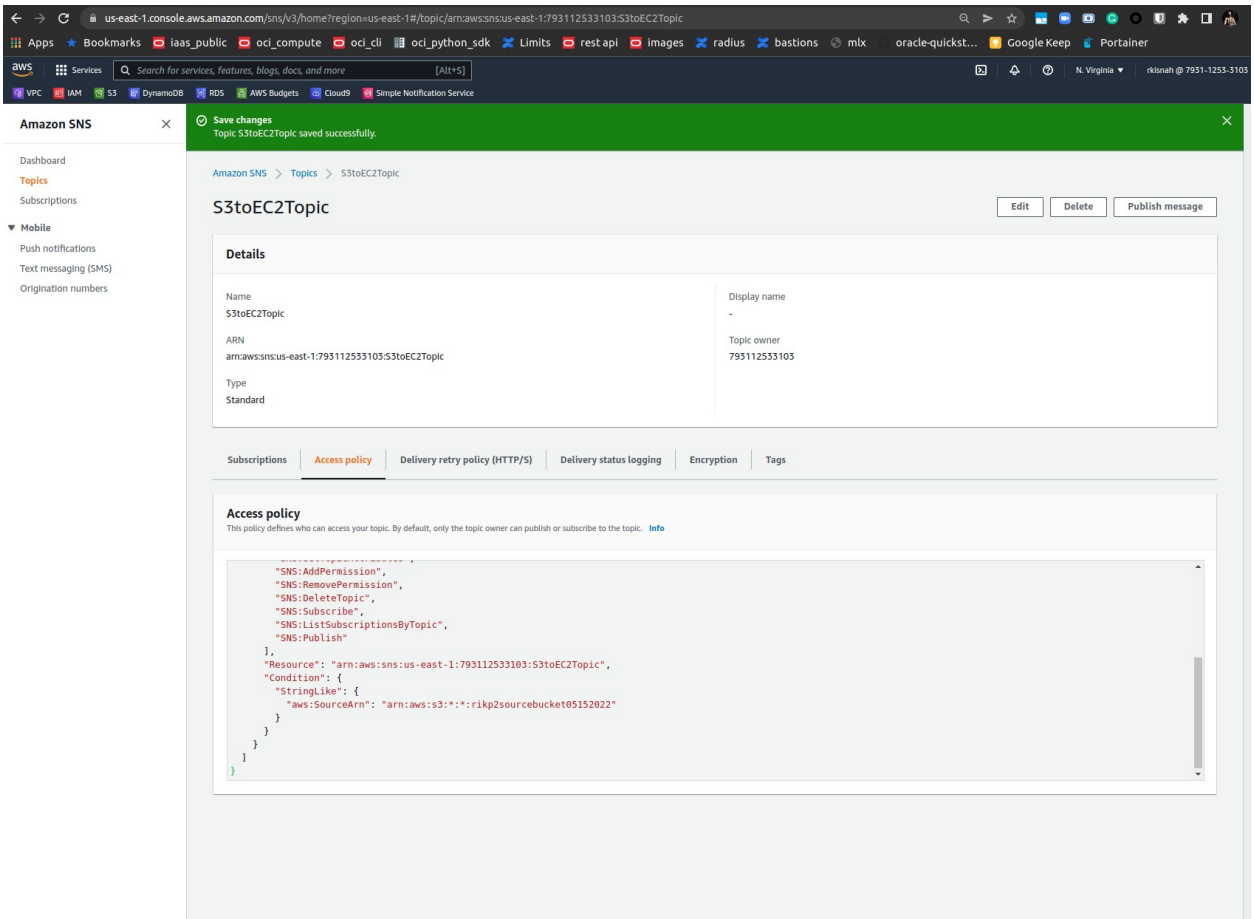


Step number	c
Step name	Modification of SNS Access Policy

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>

s3.console.aws.amazon.com/s3/bucket/rikp2sourcebucket05152022/property/notification/edit?region=us-east-1&id=S3PutEvent

Apps | Bookmarks | iaas_public | oci_compute | oci_cli | oci_python_sdk | Limits | rest api | images | radius | bastions | mlx | oracle-quickst... | Google Keep | Portainer

aws | Services | Search for services, features, blogs, docs, and more [Alt+S] | Global | rkisnah @ 7931-1253-3103

VPC | IAM | S3 | DynamoDB | RDS | AWS Budgets | Cloud9 | Simple Notification Service

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose **Provide feedback**. Provide feedback

- [ ] Object not tracked by replication
  s3:Replication:OperationNotTracked
- [ ] Object failed to replicate
  s3:Replication:OperationFailedReplication

Lifecycle

- [ ] Lifecycle transition events
  s3:LifecycleTransition
- [ ] All lifecycle expiration events
  s3:LifecycleExpiration:*
- [ ] Object expired
  s3:LifecycleExpiration:Delete
- [ ] Delete marker added by Lifecycle for a versioned object
  s3:LifecycleExpiration:DeleteMarkerCreated

Intelligent-Tiering

- [ ] Intelligent-Tiering archive events
  s3:IntelligentTiering

Destination

Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. Learn more

Destination
Choose a destination to publish the event. Learn more

- ( ) Lambda function
  Run a Lambda function script based on S3 events.
- (•) SNS topic
  Send notifications to email, SMS, or an HTTP endpoint.
- ( ) SQS queue
  Send notifications to an SQS queue to be read by a server.

Specify SNS topic

- ( ) Choose from your SNS topics
- (•) Enter SNS topic ARN

SNS topic

arn:aws:sns:us-east-1:793112533103:S3toEC2Topic

Cancel | Save changes

s3.console.aws.amazon.com/s3/buckets/rikp2sourcebucket05152022?region=us-east-1&tab=properties

Apps | Bookmarks | iaas_public | oci_compute | oci_cli | oci_python_sdk | Limits | rest api | images | radius | bastions | mlx | oracle-quickst... | Google Keep | Portainer

aws | Services | Search for services, features, blogs, docs, and more [Alt+S] | Global | rkisnah @ 7931-1253-3103

VPC | IAM | S3 | DynamoDB | RDS | AWS Budgets | Cloud9 | Simple Notification Service

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose **Provide feedback**. [Provide feedback]

Server access logging

Log requests for access to your bucket. Learn more

[Edit]

Server access logging

Disabled

AWS CloudTrail data events

Configure CloudTrail data events to log Amazon S3 object-level API operations in the CloudTrail console. Learn more

[Configure in CloudTrail]

Name	Access

No data events

No data events to display.

[Configure in CloudTrail]

Event notifications (1)

Send a notification when specific events occur in your bucket. Learn more

[Edit] [Delete] [Create event notification]

Name	Event types	Filters	Destination type	Destination
S3PutEvent	Put	-	SNS topic	S3toEC2Topic

Amazon EventBridge

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

[Edit]

Send notifications to Amazon EventBridge for all events in this bucket

Off

Transfer acceleration

Use an accelerated endpoint for faster data transfers. Learn more

[Edit]

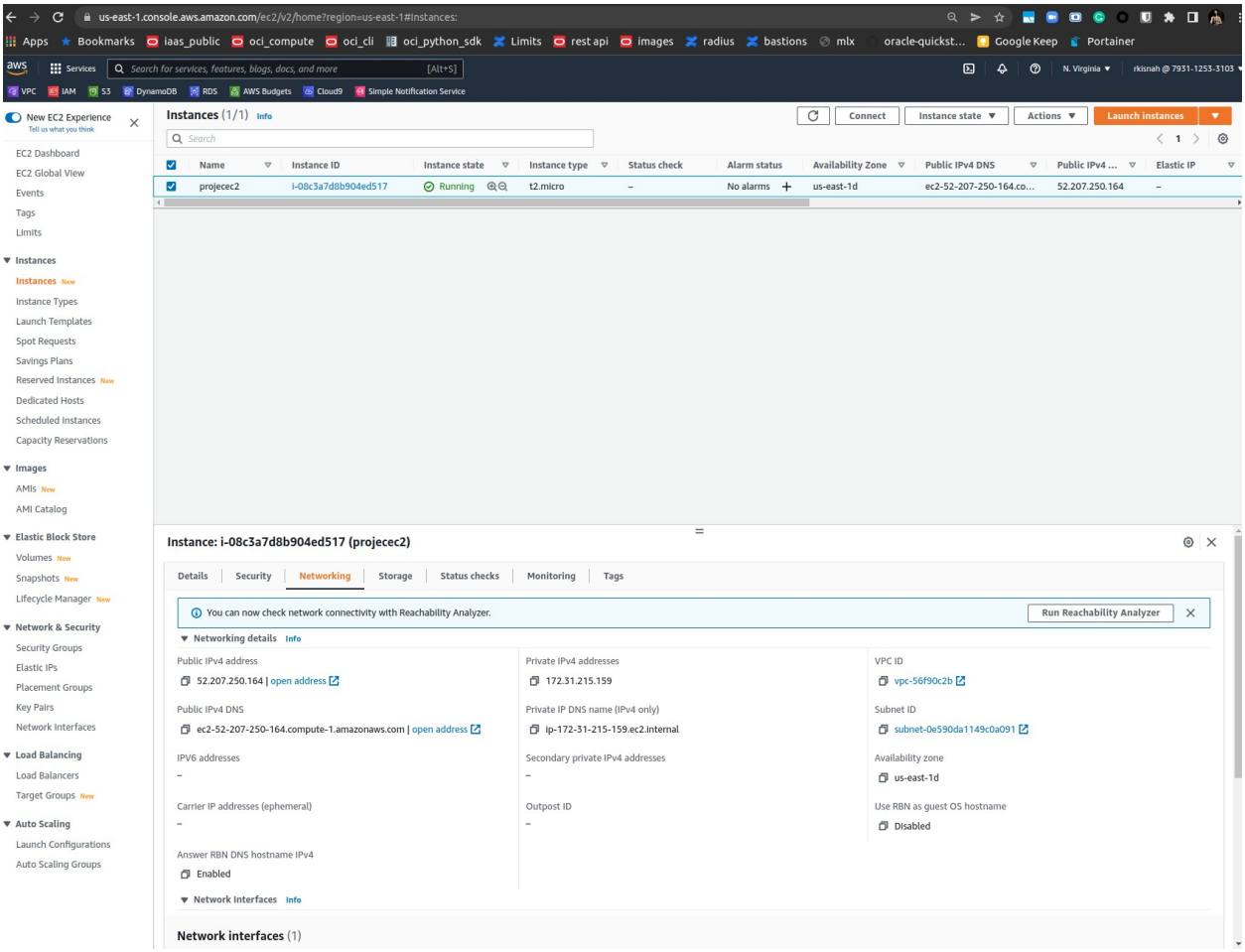
Transfer acceleration

Disabled

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

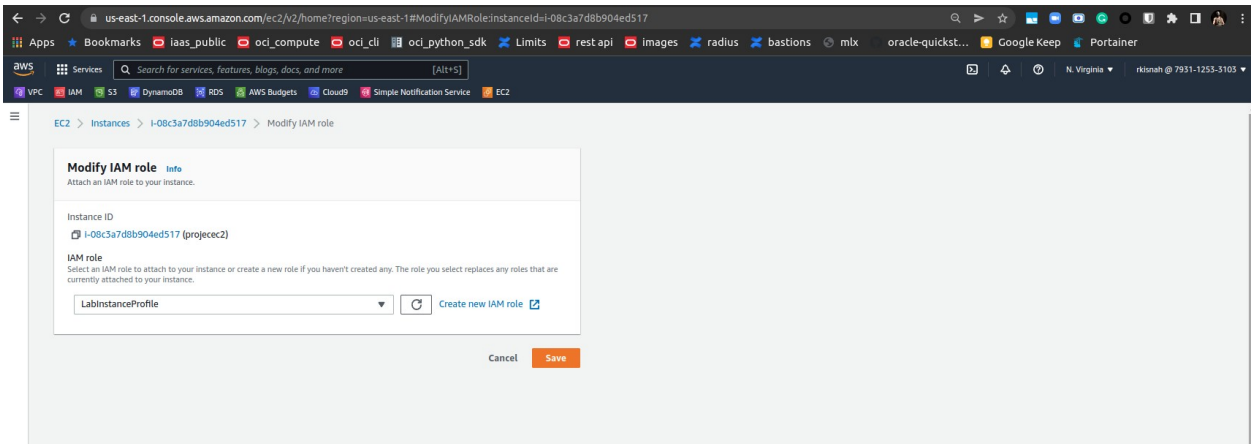
<Insert screenshot for a(1) here>



```
#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/z
UVAUqDavLBuyUO09yjAIFVJXmADxVmyWDmDbvuLZzO4LUz3Crw8ROFhT/
PkYnLLwhfl8eSn4neWolnlLKL3m/bkB3+ykWFoW72D/0xOuJ/
tgRD3ZSloEO6gTAz7Br66U+9nlN6E6+A2w7m3qNRT0EUag2hQWuACHdrAXGcCDPvm3
Auj2lqbiY+1X29F8YCXMtOUygkzl9twLjflyYCpJ2vFEf+1hRKSI+iFRObNQ5Fh9DOUDBbNv
1sj9VEbeGBMoRlUcZittBBAebuKtV0DD6J7y3cQkQuJitCgsOokF4sdKfjQem9
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 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 file to point to the target bucket	2)Copying the folder to the EC2 instance

```
# -*- coding: utf-8 -*-

# ****************************************************************************************
************************
# Author - Nirmallya Mukherjee
# To run the application use the following
# ubuntu@ip-172-31-17-36:/opt/docproc$ python manage.py runserver 0:8080
# ****************************************************************************************
************************

from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt
from botocore.exceptions import ClientError
from boto3.dynamodb.conditions import Key, Attr
import json
import boto3
import datetime


dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
s3_target_bucket = 'rikp2destinationbucket515202 2' #s3_target_bucket = 'sk-gl-target'


# ****************************************************************************************
************************
# Below methods are the handlers for the web http endpoint
# ****************************************************************************************
************************
#This is the main method that is mapped to the URI (in urls.py)
#CSRF is needed for the SNS to make a call from another domain
@csrf_exempt
def message(request):
    print '************************ Incoming request ****************************',
    print_request(request)
    #SNS http end point will have the notification details in the body
    #Check the http header and see if the sns header details are present, if so proceed else
throw except
```

```
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)$
```

```
lost connection
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>

```
onality.
Collecting django
  Downloading Django-1.11.29-py2.py3-none-any.whl (6.9 MB)
     |████████████████████████████████| 6.9 MB 14.7 MB/s
Collecting pytz
  Downloading pytz-2022.1-py2.py3-none-any.whl (503 kB)
     |████████████████████████████████| 503 kB 37.5 MB/s
Installing collected packages: pytz, django
Successfully installed django-1.11.29 pytz-2022.1
(djangodev) [ec2-user@ip-172-31-215-159 docproc-new]$ pip install boto3
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in Janu
ary 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functi
onality.
Collecting boto3
  Downloading boto3-1.17.112-py2.py3-none-any.whl (131 kB)
     |████████████████████████████████| 131 kB 13.7 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting s3transfer<0.5.0,>=0.4.0
  Downloading s3transfer-0.4.2-py2.py3-none-any.whl (79 kB)
     |████████████████████████████████| 79 kB 11.2 MB/s
Collecting botocore<1.21.0,>=1.20.112
  Downloading botocore-1.20.112-py2.py3-none-any.whl (7.7 MB)
     |████████████████████████████████| 7.7 MB 44.8 MB/s
Collecting futures<4.0.0,>=2.2.0; python_version == "2.7"
  Downloading futures-3.3.0-py2-none-any.whl (16 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB 46.1 MB/s
Collecting urllib3<1.27,>=1.25.4
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
     |████████████████████████████████| 138 kB 44.3 MB/s
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: jmespath, futures, six, python-dateutil, urllib3, botocore, s3transfer, boto3
Successfully installed boto3-1.17.112 botocore-1.20.112 futures-3.3.0 jmespath-0.10.0 python-dateutil-2.8.2 s3transfer-0.4.2 six-1.16.0 urllib3-1.26.9
(djangodev) [ec2-user@ip-172-31-215-159 docproc-new]$ pip install mysql-connector-python-rf
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in Janu
ary 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functi
onality.
Collecting mysql-connector-python-rf
  Downloading mysql-connector-python-rf-2.2.2.tar.gz (11.9 MB)
     |████████████████████████████████| 11.9 MB 31.3 MB/s
Building wheels for collected packages: mysql-connector-python-rf
  Building wheel for mysql-connector-python-rf (setup.py) ... done
  Created wheel for mysql-connector-python-rf: filename=mysql_connector_python_rf-2.2.2-cp27-cp27mu-linux_x86_64.whl size=249459 sha256=ff3701ea6b72a9b21beeb3c51b916e5c1cb110d8208d37fef022b
701c666e98d
  Stored in directory: /home/ec2-user/.cache/pip/wheels/3b/b5/d4/5d0e3338625186ab2fbf75908b58178b859aa8e1fd1291a0fa
Successfully built mysql-connector-python-rf
Installing collected packages: mysql-connector-python-rf
Successfully installed mysql-connector-python-rf-2.2.2
(djangodev) [ec2-user@ip-172-31-215-159 docproc-new]$ python -W ignore manage.py runserver 0:8080
Performing system checks...

System check identified no issues (0 silenced).
May 18, 2022 - 23:42:10
Django version 1.11.29, using settings 'docproc.settings'
Starting development server at http://0:8080/
Quit the server with CONTROL-C.
```

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

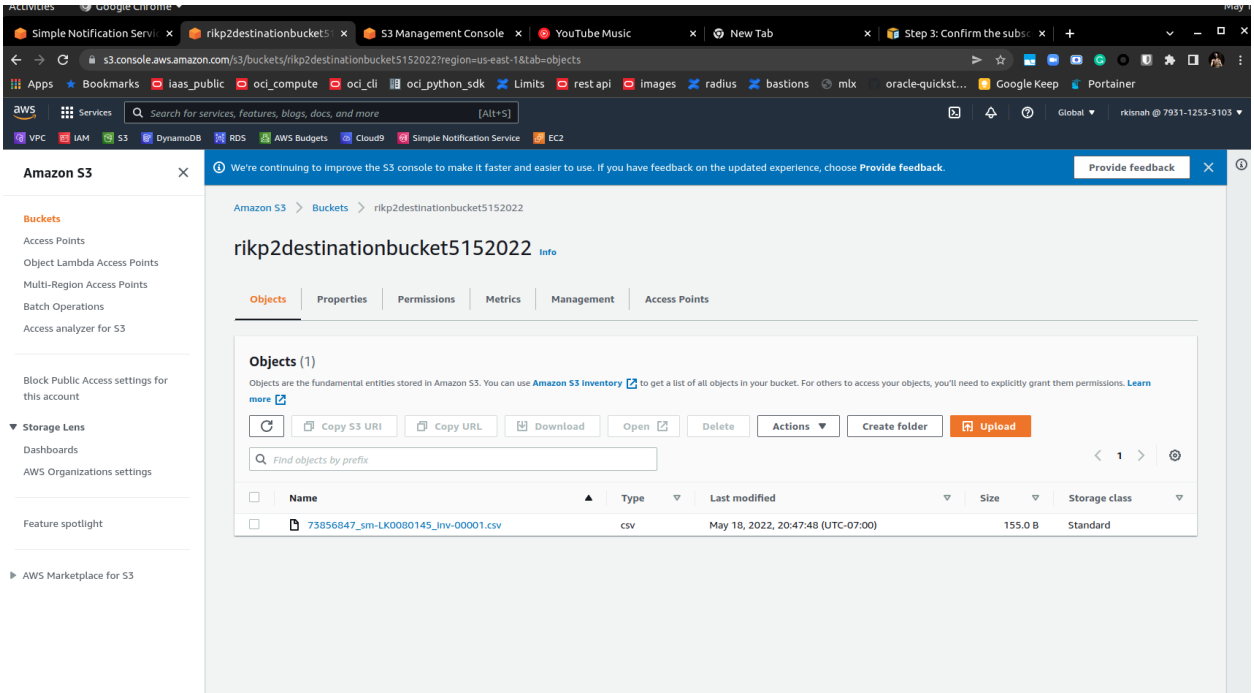
"SubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-1:793112533103:S3toEC2Topic&Token=2336412f37fb687f5d51e6e2425dacbbaa29614c8144eee7f3dd71a1f12f9f29819338be60f536ac9c1d92cdfdb1e2041096bf8208b9538fba6b926f45f768ce6f76762d7b4edd173847f94df677c0423e4d20134e68953b60a22c2757fc367c06d2b15dfaa0a1b46463c04ad66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

<Insert screenshot for b(1) here>

```
ec2-user@ip-172-31-215-159:~
ec2-user@ip-172-31-215-159:~ 93x70
Accept-Encoding: gzip,deflate

Body: {
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "369a8be8-811e-426b-a28b-4d2c0c278801",
  "Token" : "2336412f37fb687f5d51e6e2425dacbbaa29614c8144eee7f3dd71a1f12f9f29819338be60f536ac
9c1d92cdfdb1e2041096bf8208b9538fba6b926f45f768ce6f76762d7b4edd173847f94df677c0423e4d20134e689
53b60a22c2757fc367c06d2b15dfaa0a1b46463c04ad66253b5",
  "TopicArn" : "arn:aws:sns:us-east-1:793112533103:S3toEC2Topic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-east-1:793112533103:S
3toEC2Topic.\nTo confirm the subscription, visit the SubscribeURL included in this message.",
  "SubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscription&TopicArn=
arn:aws:sns:us-east-1:793112533103:S3toEC2Topic&Token=2336412f37fb687f5d51e6e2425dacbbaa29614
c8144eee7f3dd71a1f12f9f29819338be60f536ac9c1d92cdfdb1e2041096bf8208b9538fba6b926f45f768ce6f76
762d7b4edd173847f94df677c0423e4d20134e68953b60a22c2757fc367c06d2b15dfaa0a1b46463c04ad66253b5"
,
  "Timestamp" : "2022-05-19T00:54:29.189Z",
  "SignatureVersion" : "1",
  "Signature" : "RAehUyGaMjPbAqtNe0KVzeTqADqV1DZ5QAXHl/AKsjHm3M3z73FR6fJlUyqu/MJzmXWOGk7Oq7G9
54f8gfDe6VFPAHbbjk4v31ugeGpHa35guQ8qzuftTgveAGL5ncBuwv7ld+pao5rV4WejvCsmOIg9eeXM48xy0y8cmoxE2
WZ/Hh3MVFYdv64yePTDrK7BTlDpp5ZKH1b9UYJeMV84cD1wtShTEqOl51+B/Wjpisf2wTPjx4L7yGmrI4wq/NRdpxwsk0
//5BJJ77+l80L0BU/NDyvNcyOTubM4XoS+PE454KKm5T6s09Lt9bIWp0dmHuAnNLFNpVxZEZg+kqPxrw==",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-7ff531849
0ec183fbaddaa2a969abfda.pem"
}
Request method = POST
('The S3 JSON is ', '{\n  "Type" : "SubscriptionConfirmation",\n  "MessageId" : "369a8be8-811
e-426b-a28b-4d2c0c278801",\n  "Token" : "2336412f37fb687f5d51e6e2425dacbbaa29614c8144eee7f3dd
71a1f12f9f29819338be60f536ac9c1d92cdfdb1e2041096bf8208b9538fba6b926f45f768ce6f76762d7b4edd173
847f94df677c0423e4d20134e68953b60a22c2757fc367c06d2b15dfaa0a1b46463c04ad66253b5",\n  "TopicAr
n" : "arn:aws:sns:us-east-1:793112533103:S3toEC2Topic",\n  "Message" : "You have chosen to su
bscribe to the topic arn:aws:sns:us-east-1:793112533103:S3toEC2Topic.\\nTo confirm the subscr
iption, visit the SubscribeURL included in this message.",\n  "SubscribeURL" : "https://sns.u
s-east-1.amazonaws.com/?Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-1:79311253310
3:S3toEC2Topic&Token=2336412f37fb687f5d51e6e2425dacbbaa29614c8144eee7f3dd71a1f12f9f29819338be
60f536ac9c1d92cdfdb1e2041096bf8208b9538fba6b926f45f768ce6f76762d7b4edd173847f94df677c0423e4d2
0134e68953b60a22c2757fc367c06d2b15dfaa0a1b46463c04ad66253b5",\n  "Timestamp" : "2022-05-19T00
:54:29.189Z",\n  "SignatureVersion" : "1",\n  "Signature" : "RAehUyGaMjPbAqtNe0KVzeTqADqV1DZ5
```

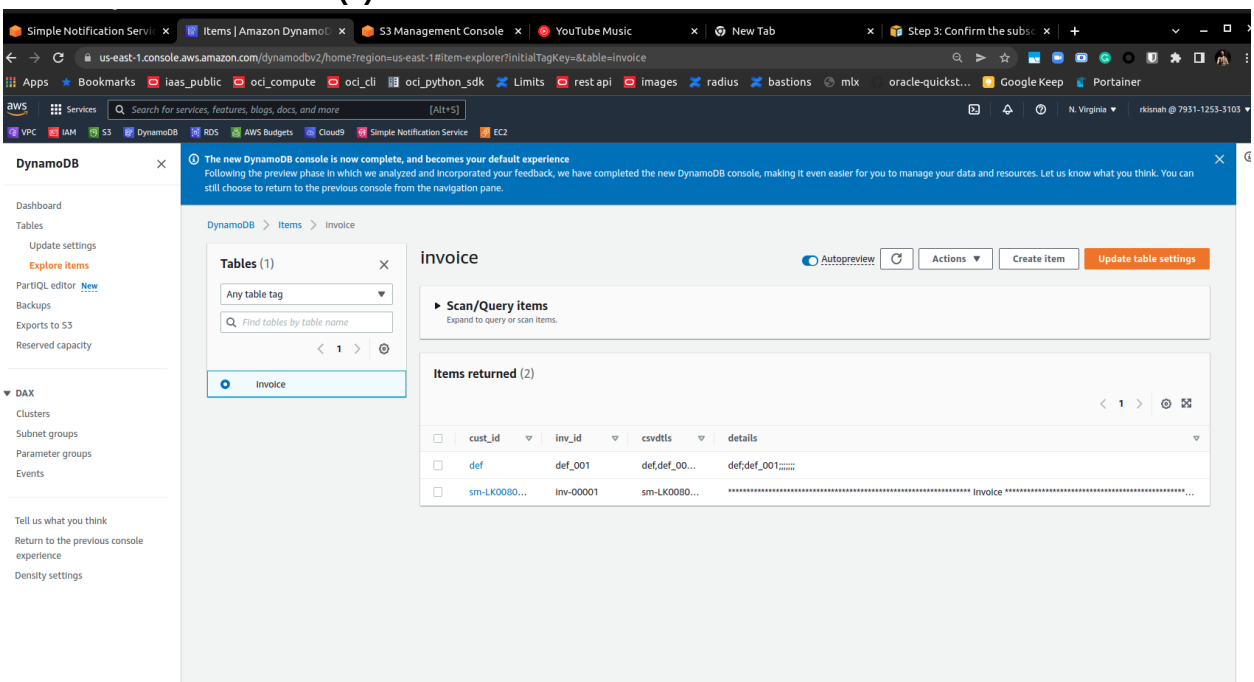
Step number	c
Step name	Generation of CSV file
Instructions	1) Download the file **docproc-invoice.txt** provided with this workbook 2) Navigate to S3 in the AWS Console 3) Upload the sample invoice file to the source S3 bucket using the default options 4) Verify that a CSV file is generated in the target 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