AWS Three-Tier Image Recognition Application

Nishanth Murali

1. Problem statement

In this project, we aim to develop a scalable image recognition service on the cloud which has all the software components loosely coupled. Every component of AWS used is elastic and hence, the project can be scaled up and down as per the number of requests coming in.

- Our cloud application performs image recognition on user-provided images using the provided deep learning model. It then returns the recognition result as output to the users.
- Each image input through the website is a request to our application and our application can handle multiple requests concurrently. It automatically scales out when the request demand increases, and automatically scales in when the demand drops.
- All the inputs (images) and outputs (recognition results) are stored in S3 for persistency.
- The application should handle all the requests as fast as possible, and it should not miss any requests. The recognition requests should all be correct.

2. Design and implementation

2.1 Architecture

- The user accesses the web tier to upload the images. The web tier always runs on an AWS instance. This web-tier instance then uploads the image to an S3 bucket ('input' bucket) and gets the 'presigned URL' from it. The URLs are sent to the app instance through the Request queue (FIFO SQS), where each app instance polls for a message from it and processes the same to get the classification results. Each of these results is then sent back to the web tier using a Response queue (FIFO SQS) and these results are then merged in order to write to the output file that stores all the outputs in the S3 bucket, in order to achieve persistence.

- The SQS is responsible for maintaining two communication channels between the web tier instance and application tier instances. The FIFO queues will result in each image URL being processed exactly once by the app instances so that there is no duplication of the classification results.

- The bash script configured in each of the app instances will be used to run the image classification code automatically when an instance starts. This will avoid human intervention while processing the images.

  The high-level architecture of the cloud-application is shown in Fig. 1.

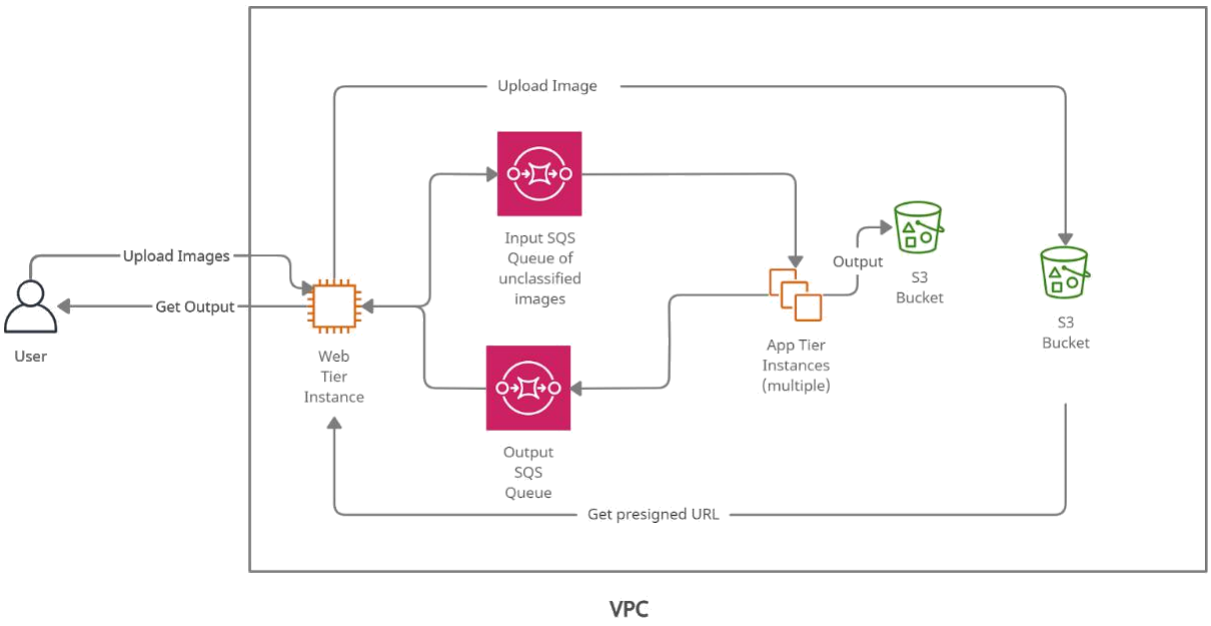

Fig. 1

2.2 Autoscaling

Autoscaling is one of the main reasons for hosting applications on the cloud. This project also utilizes the Autoscaling property of the cloud, so that the application resources are used efficiently. The web tier instance takes care of scaling up of the instances, based on the number of image-classification requests made. We will be using at most 19 app instances in this project and each instance processes the images to get the classification result. If the number of requests is more than 19 at a particular time, the request-messages are held in the Request queue until there is an app instance available to pick the request. The code embedded in each of the app instances will long poll the SQS request queue, in order to check if there are messages (image URLs) in the Request queue. If there are no messages left in the Request queue, the app instance will stop itself automatically, so that the resources are not wasted and thus the application is scaled down by the app instance themselves.

3. Testing and evaluation

Testing is an important part of building a cloud application, where you check if the application handles the requests concurrently and also if it auto-scales, so that minimum resources are used to maximize the efficiency of the final product.

In this project, we tested our application for stress and also efficient auto-scaling of the EC2 instances. The images below show that the application works as expected and the deployed product is efficient and fast.

Firstly, we uploaded 100 images through our web-tier instance (fig. 2), We then obtained the images' object URLs from the 'input' S3 bucket, where the images were stored. These URLs were then sent to the SQS 'Request' queue ('send_queue.fifo' in fig. 3). Initially there was just one EC2 instance running,

which was the web-tier instance (fig. 4) and as per the length of the Request queue, the app instances got started (fig. 5). The output messages (image classification results) were transferred to the SQS Response queue ('receive_queue.fifo' in fig. 6) and the messages in the Request queue were deleted. Once all the messages were received by the Response queue, the messages (results) were fed to the 'outputfile.txt' file, which was then sent to the 'output' S3 bucket, and the messages in the Response queue were also deleted (fig. 7). Meanwhile, when the EC2 app-instance had no messages to receive, they automatically shut themselves down (fig. 8). The classification results were then shown to the user on the UI (fig. 9). The 'input' bucket (fig. 10) and the 'output' bucket (fig. 11) had all the input image files and the output file respectively. The output file is shown in fig. 12.

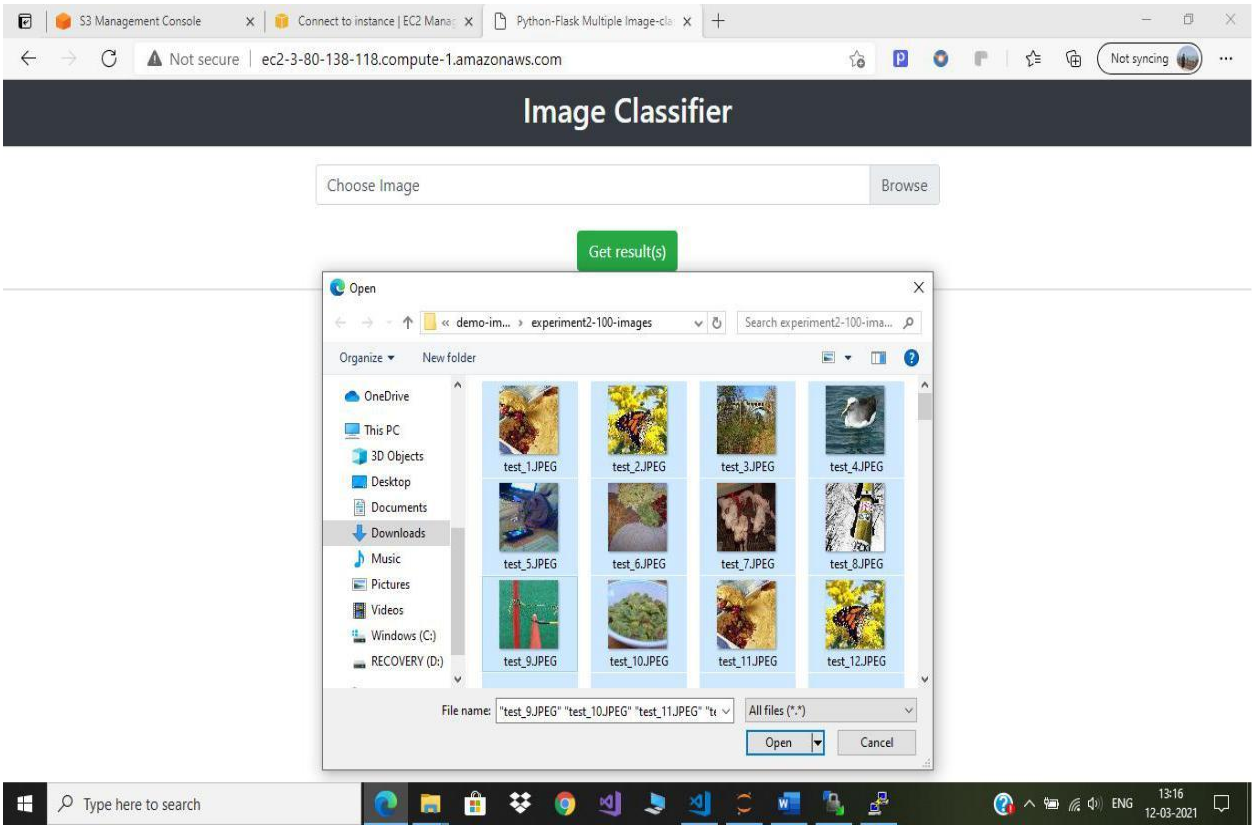

Fig. 2

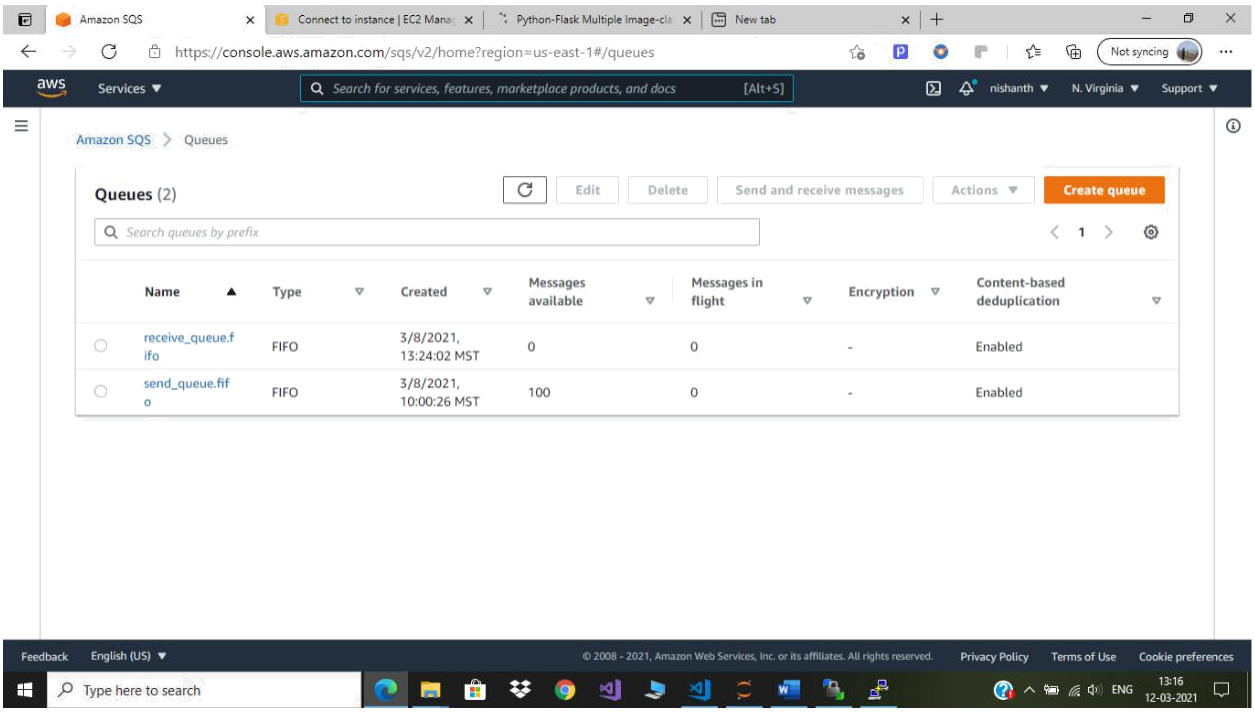
Amazon SQS
https://console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Services
Search for services, features, marketplace products, and docs [Alt+S]
nishanth
N. Virginia
Support
Amazon SQS > Queues
Queues (2)
Edit
Delete
Send and receive messages
Actions
Create queue
Search queues by prefix
Name
Type
Created
Messages available
Messages in flight
Encryption
Content-based deduplication
receive_queue.fifo
FIFO
3/8/2021, 13:24:02 MST
0
0
-
Enabled
send_queue.fifo
FIFO
3/8/2021, 10:00:26 MST
100
0
-
Enabled
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
13:16
12-03-2021

Fig. 3

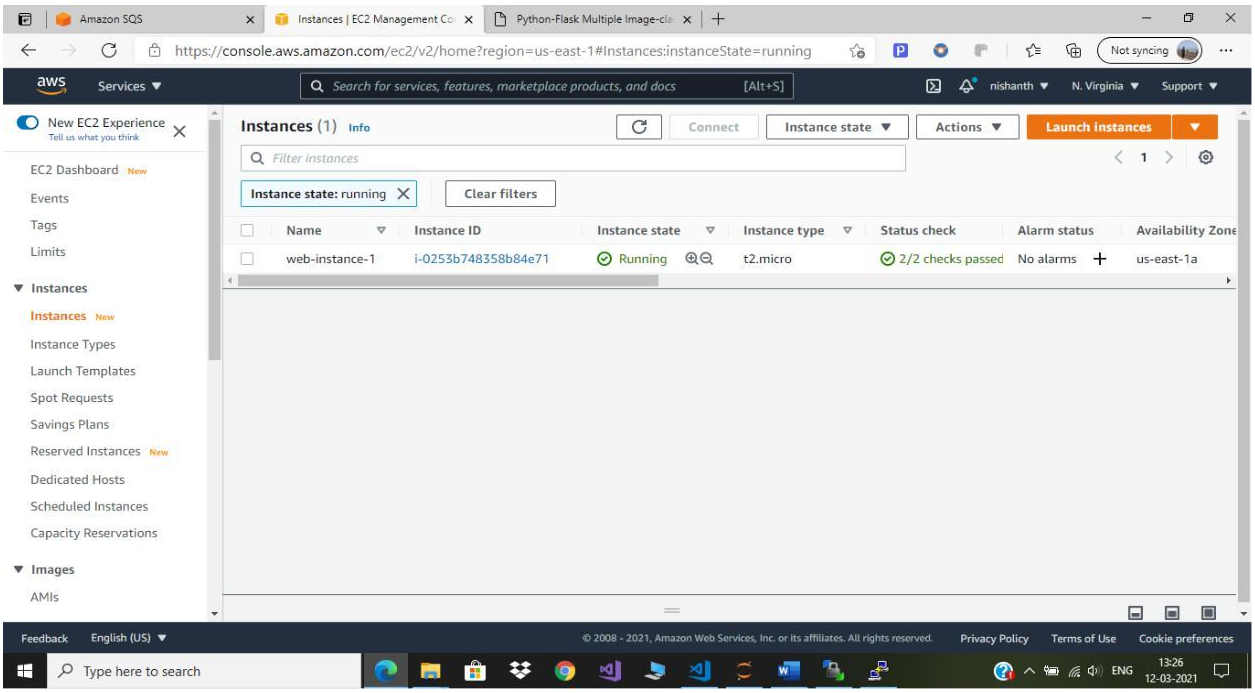
Amazon SQS
Instances | EC2 Management Co
Python-Flask Multiple Image-cla
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:instanceState=running
Services
Search for services, features, marketplace products, and docs [Alt+S]
nishanth
N. Virginia
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Capacity Reservations
Images
AMIs
Instances (1) Info
Connect
Instance state
Actions
Launch instances
Filter instances
Instance state: running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
web-instance-1
i-0253b748358b84e71
Running
t2.micro
2/2 checks passed
No alarms
us-east-1a
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
13:26
12-03-2021

Fig. 4

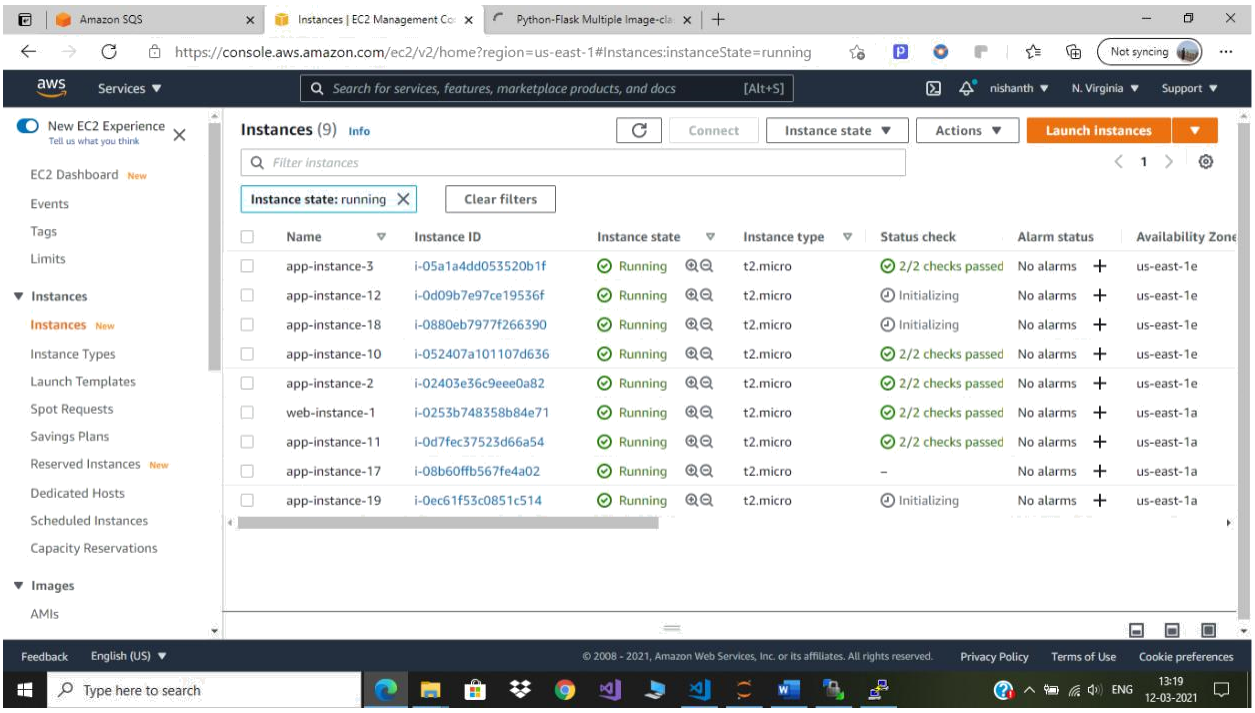
Amazon SQS
Instances | EC2 Management Co
Python-Flask Multiple Image-cla
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:instanceState=running
Not syncing
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
nishanth
N. Virginia
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Capacity Reservations
Images
AMIs
Instances (9) Info
Connect
Instance state
Actions
Launch instances
Filter instances
Instance state: running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
app-instance-3 i-05a1a4dd053520b1f Running t2.micro 2/2 checks passed No alarms us-east-1e
app-instance-12 i-0d09b7e97ce19536f Running t2.micro Initializing No alarms us-east-1e
app-instance-18 i-0880eb7977f266390 Running t2.micro Initializing No alarms us-east-1e
app-instance-10 i-052407a101107d636 Running t2.micro 2/2 checks passed No alarms us-east-1e
app-instance-2 i-02403e36c9eee0a82 Running t2.micro 2/2 checks passed No alarms us-east-1e
web-instance-1 i-0253b748358b84e71 Running t2.micro 2/2 checks passed No alarms us-east-1a
app-instance-11 i-0d7fec37523d66a54 Running t2.micro 2/2 checks passed No alarms us-east-1a
app-instance-17 i-08b60ffb567fe4a02 Running t2.micro - No alarms us-east-1a
app-instance-19 i-0ec61f53c0851c514 Running t2.micro Initializing No alarms us-east-1a
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
13:19
12-03-2021

Fig. 5

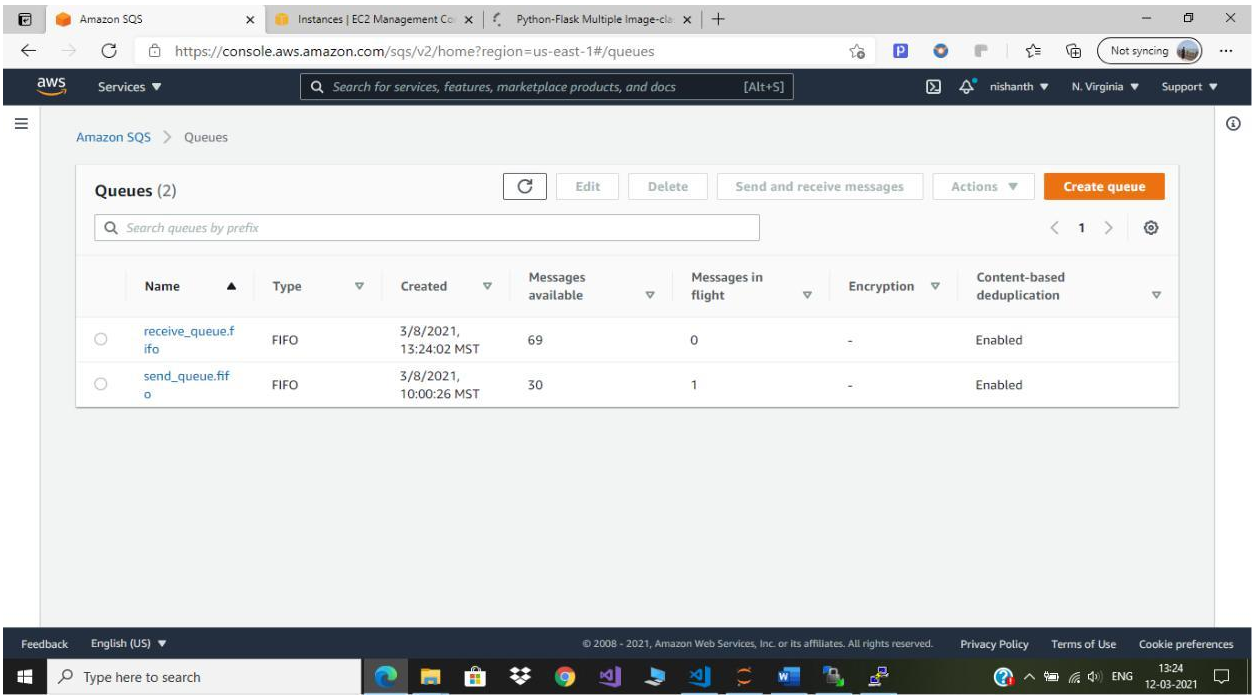
Amazon SQS
Instances | EC2 Management Co
Python-Flask Multiple Image-cla
https://console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Not syncing
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
nishanth
N. Virginia
Support
Amazon SQS > Queues
Queues (2)
Edit
Delete
Send and receive messages
Actions
Create queue
Search queues by prefix
Name
Type
Created
Messages available
Messages in flight
Encryption
Content-based deduplication
receive_queue.fifo FIFO 3/8/2021, 13:24:02 MST 69 0 - Enabled
send_queue.fifo FIFO 3/8/2021, 10:00:26 MST 30 1 - Enabled
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
13:24
12-03-2021

Fig. 6

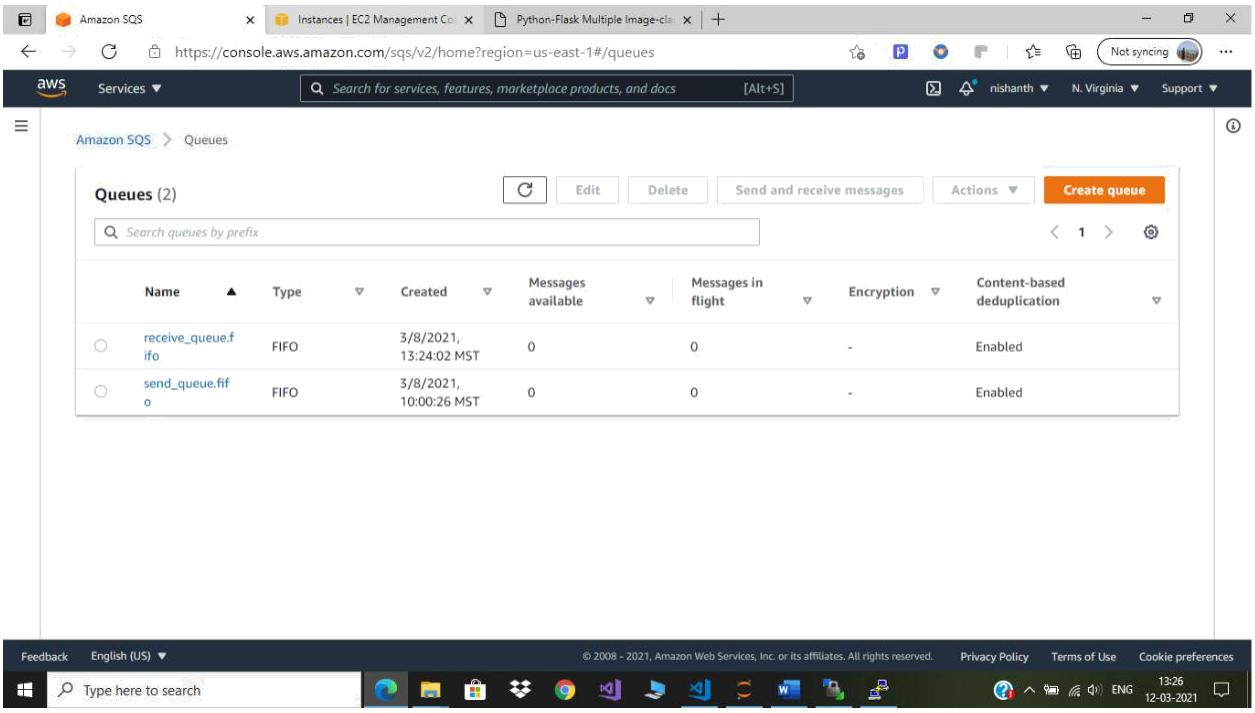
Amazon SQS
Instances | EC2 Management Co
Python-Flask Multiple Image-cla
https://console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Not syncing
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
nishanth
N. Virginia
Support
Amazon SQS > Queues
Queues (2)
Edit
Delete
Send and receive messages
Actions
Create queue
Search queues by prefix
1
Name
Type
Created
Messages available
Messages in flight
Encryption
Content-based deduplication
receive_queue.fifo
FIFO
3/8/2021, 13:24:02 MST
0
0
-
Enabled
send_queue.fifo
FIFO
3/8/2021, 10:00:26 MST
0
0
-
Enabled
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
13:26
12-03-2021

Fig. 7

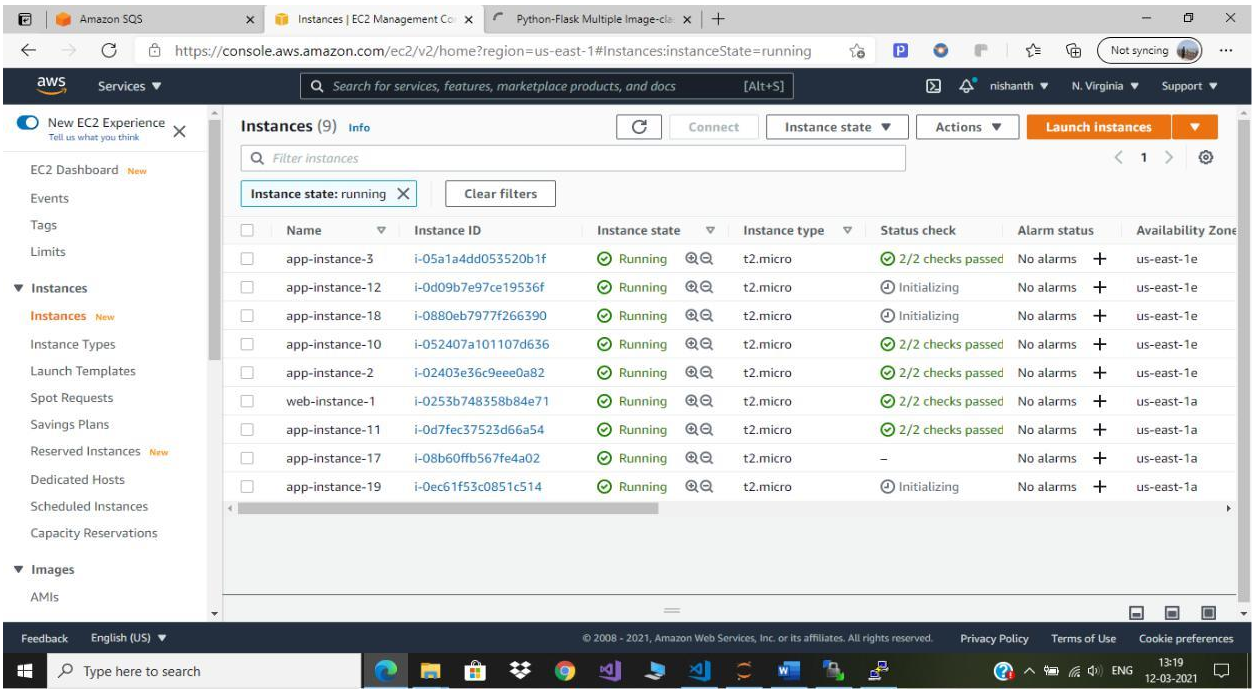
Amazon SQS
Instances | EC2 Management Co
Python-Flask Multiple Image-cla
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:instanceState=running
Not syncing
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
nishanth
N. Virginia
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Capacity Reservations
Images
AMIs
Instances (9) Info
Connect
Instance state
Actions
Launch instances
Filter instances
1
Instance state: running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
app-instance-3 i-05a1a4dd053520b1f Running t2.micro 2/2 checks passed No alarms us-east-1e
app-instance-12 i-0d09b7e97ce19536f Running t2.micro Initializing No alarms us-east-1e
app-instance-18 i-0880eb7977f266390 Running t2.micro Initializing No alarms us-east-1e
app-instance-10 i-052407a101107d636 Running t2.micro 2/2 checks passed No alarms us-east-1e
app-instance-2 i-02403e36c9eee0a82 Running t2.micro 2/2 checks passed No alarms us-east-1e
web-instance-1 i-0253b748358b84e71 Running t2.micro 2/2 checks passed No alarms us-east-1a
app-instance-11 i-0d7fec37523d66a54 Running t2.micro 2/2 checks passed No alarms us-east-1a
app-instance-17 i-08b60ffb567fe4a02 Running t2.micro - No alarms us-east-1a
app-instance-19 i-0ec61f53c0851c514 Running t2.micro Initializing No alarms us-east-1a
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
13:19
12-03-2021

Fig. 8

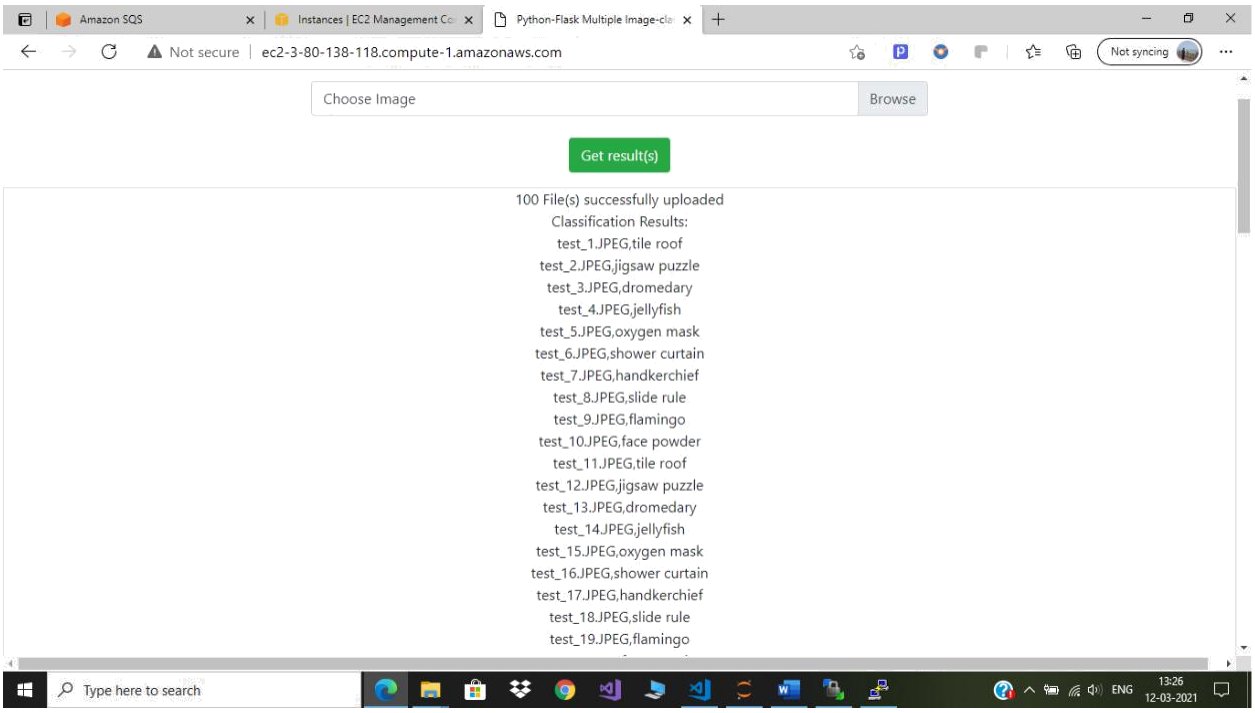
Amazon SQS
Instances | EC2 Management Co
Python-Flask Multiple Image-cla
Not secure
ec2-3-80-138-118.compute-1.amazonaws.com
Not syncing
Choose Image
Browse
Get result(s)
100 File(s) successfully uploaded
Classification Results:
test_1.JPEG,tile roof
test_2.JPEG,jigsaw puzzle
test_3.JPEG,dromedary
test_4.JPEG,jellyfish
test_5.JPEG,oxygen mask
test_6.JPEG,shower curtain
test_7.JPEG,handkerchief
test_8.JPEG,slide rule
test_9.JPEG,flamingo
test_10.JPEG,face powder
test_11.JPEG,tile roof
test_12.JPEG,jigsaw puzzle
test_13.JPEG,dromedary
test_14.JPEG,jellyfish
test_15.JPEG,oxygen mask
test_16.JPEG,shower curtain
test_17.JPEG,handkerchief
test_18.JPEG,slide rule
test_19.JPEG,flamingo
Type here to search
ENG
13:26
12-03-2021

Fig. 9

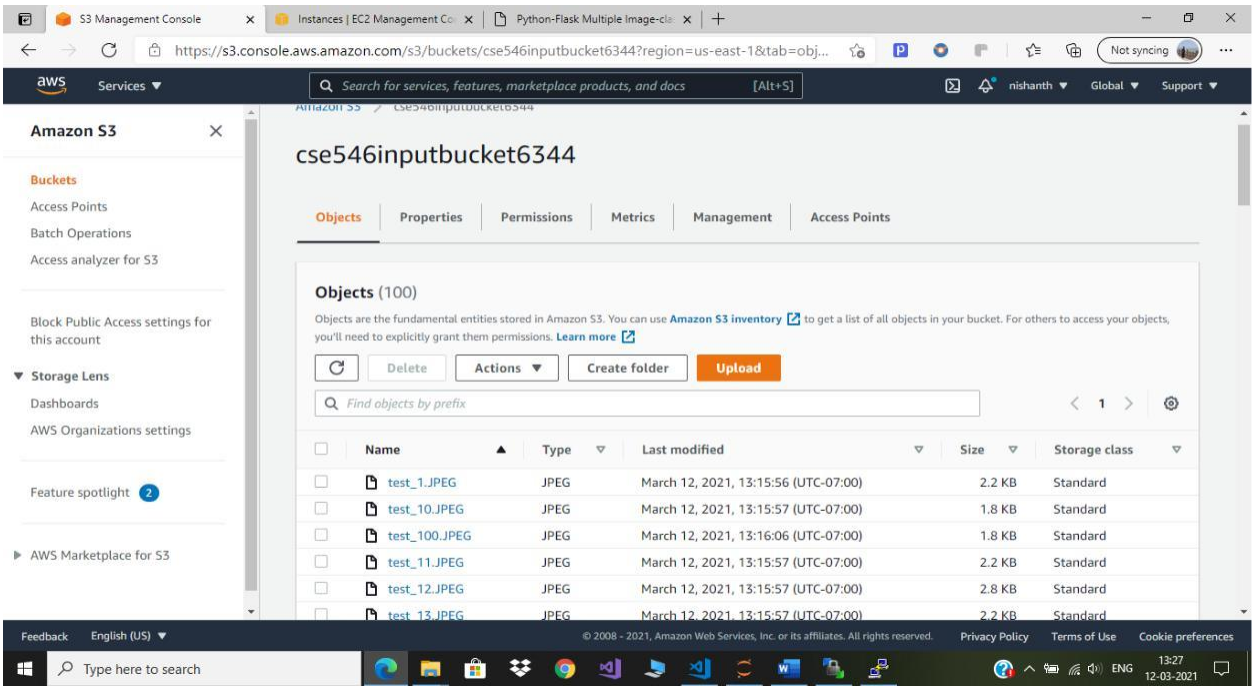
S3 Management Console
Instances | EC2 Management Co
Python-Flask Multiple Image-cla
https://s3.console.aws.amazon.com/s3/buckets/cse546inputbucket6344?region=us-east-1&tab=obj...
Not syncing
aws
Services
Search for services, features, marketplace products, and docs
[Alt+S]
nishanth
Global
Support
Amazon S3
Buckets
Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
cse546inputbucket6344
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (100)
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
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class
test_1.JPEG JPEG March 12, 2021, 13:15:56 (UTC-07:00) 2.2 KB Standard
test_10.JPEG JPEG March 12, 2021, 13:15:57 (UTC-07:00) 1.8 KB Standard
test_100.JPEG JPEG March 12, 2021, 13:16:06 (UTC-07:00) 1.8 KB Standard
test_11.JPEG JPEG March 12, 2021, 13:15:57 (UTC-07:00) 2.2 KB Standard
test_12.JPEG JPEG March 12, 2021, 13:15:57 (UTC-07:00) 2.8 KB Standard
test_13.JPEG JPEG March 12, 2021, 13:15:57 (UTC-07:00) 2.2 KB Standard
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
13:27
12-03-2021

Fig. 10

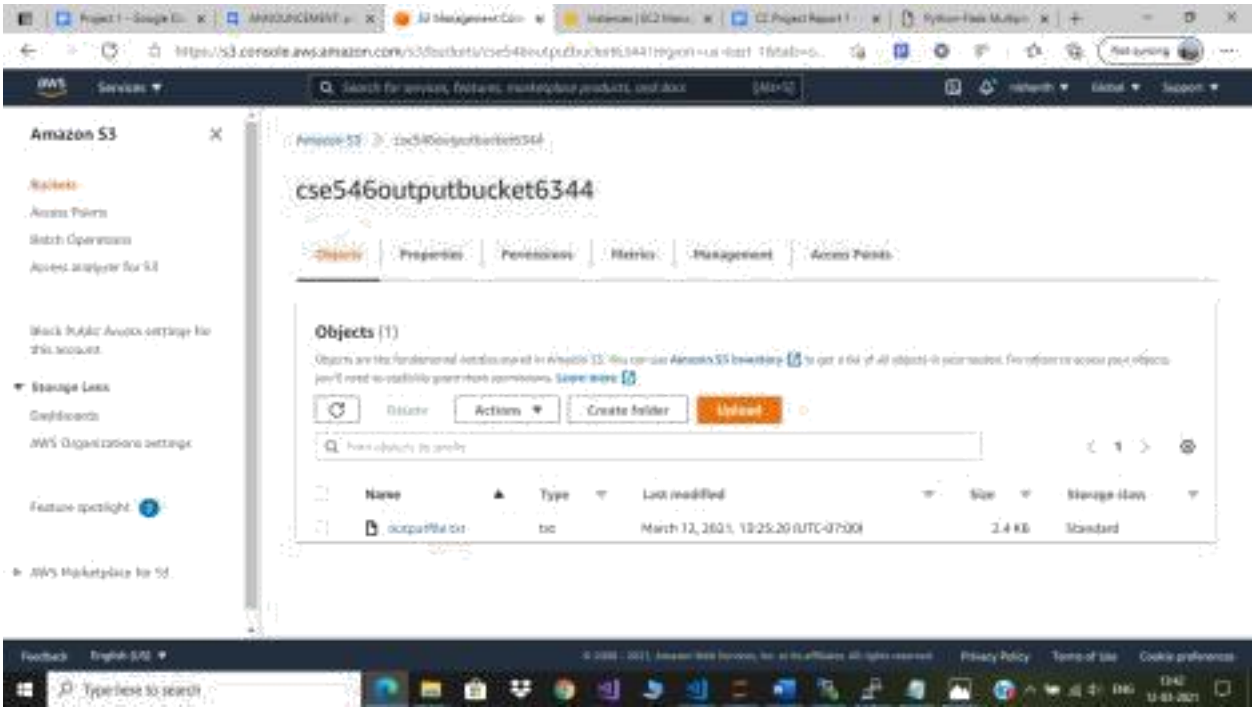
Amazon S3
cse546outputbucket6344
Objects (1)
Actions
Create folder
Upload
Name
Type
Last modified
Size
Storage class
outputfile.txt
txt
March 12, 2021, 13:25:20 (UTC-07:00)
2.4 KB
Standard

Fig. 11

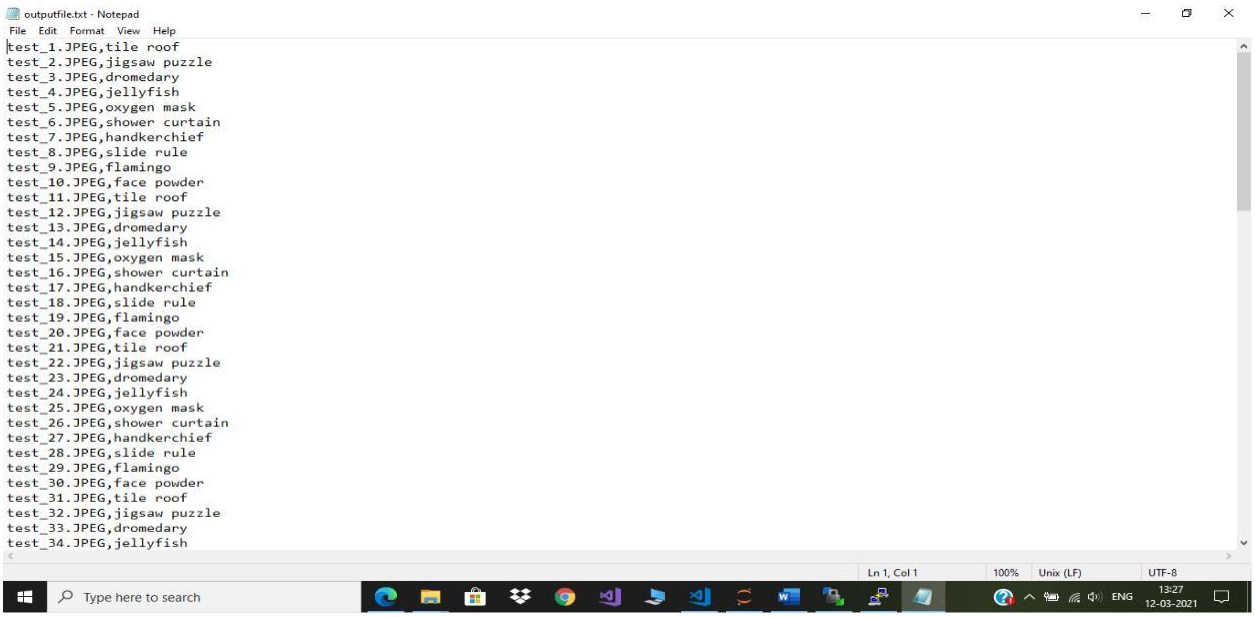
outputfile.txt - Notepad
File Edit Format View Help
test_1.JPEG,tile roof
test_2.JPEG,jigsaw puzzle
test_3.JPEG,dromedary
test_4.JPEG,jellyfish
test_5.JPEG,oxygen mask
test_6.JPEG,shower curtain
test_7.JPEG,handkerchief
test_8.JPEG,slide rule
test_9.JPEG,flamingo
test_10.JPEG,face powder
test_11.JPEG,tile roof
test_12.JPEG,jigsaw puzzle
test_13.JPEG,dromedary
test_14.JPEG,jellyfish
test_15.JPEG,oxygen mask
test_16.JPEG,shower curtain
test_17.JPEG,handkerchief
test_18.JPEG,slide rule
test_19.JPEG,flamingo
test_20.JPEG,face powder
test_21.JPEG,tile roof
test_22.JPEG,jigsaw puzzle
test_23.JPEG,dromedary
test_24.JPEG,jellyfish
test_25.JPEG,oxygen mask
test_26.JPEG,shower curtain
test_27.JPEG,handkerchief
test_28.JPEG,slide rule
test_29.JPEG,flamingo
test_30.JPEG,face powder
test_31.JPEG,tile roof
test_32.JPEG,jigsaw puzzle
test_33.JPEG,dromedary
test_34.JPEG,jellyfish
Ln 1, Col 1
100%
Unix (LF)
UTF-8

Fig. 12

The application's output, i.e., the image classification results, was evaluated with respect to the provided actual image tags and the classification results were totally accurate.

4. Code

The project uses three python programs that implement the different abstractions of the boto3 services, that have been used in the Flask app, which would run in the 'app.py' file. All the Three services are independent of each other and from the main module. They are loosely coupled such that they could be reused in other projects too with minimal or no changes. The three python modules using AWS SDKs in this project are;

1. SQSservice: 'SQSservice' contains methods that are required to perform basic operation on SQS Service like creating, deleting, sending, and receiving messages using 'sqs' resource of boto3 SDKs in Python.

2. S3Service: The boto3 SDKs also provide services to handle S3 buckets. 'S3Service' contains the methods that represent the Simple Storage Service (S3) at higher level client visibility. The methods help in uploading and downloading files to and from the S3 buckets.

3. EC2Service: 'EC2Service' contains methods that are required to perform basic operations on EC2, such as starting or terminating (stopping) an instance.

4.1 Web-Tier Module

Web-tier Module consists of a Flask code 'app.py' which acts as driving functionality of the web-tier. In this module, we would be mapping the URI to the backend API through Flask. The Frontend HTML and Javascript are also mapped using the rendering template of the flask. The main function takes input from the user through the Html form which is built using the bootstrap library. The files are then sent to the S3 and its Object URL is generated. The generated File Urls are pushed into the SQS queue (Request queue) to be later read by app-tier instances. After pushing the input image url to the SQS, the Web-Tier polls for messages from the output queue (Response queue), which are then stored in the S3 bucket and displayed to the user using Jinja template.

4.2 App-Tier Module

For each input image, an app-tier Instance processes it to get the classification result using the 'image_classification.py' python code. It is the heart of the application as it takes care of delivering the output from the deep learning model to the user. The App-Tier Module First polls the message from the SQS, classifies the image using a deep learning model, and sends the output back to the web-tier through SQS. It also scales down the application, by shutting down the instance it runs on, after it sees no message in the Request queue.

4.3 Installation of the application

1. Web-tier

We need to type in the following commands in the Ubuntu shell, before we can run the Flask app that would run on an Apache server. Before that, we should create a folder called 'flaskproject' in /home/ubuntu directory and add all the application files in this folder. The commands to be executed later are given below;

1. sudo apt-get update
2. sudo apt-get python3-pip
3. pip 3 install virtualenv
4. sudo apt-get install apache2 libapache2-mod-wsgi-py3
5. sudo ln -sT ~/flaskproject /var/www/html/flaskproject
6. sudo a2enmod wsgi

Then, we need to edit a configuration file whose path is /etc/apache2/sites-enabled/000-default.conf, wherein we will add the following lines after the line containing "var/www/html";

```
WSGIDaemonProcess flaskproject threads=5
WSGIScriptAlias / /var/www/html/flaskproject/app.wsgi

<Directory flaskproject>
        WSGIProcessGroup flaskproject
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
</Directory>
```

Now, the environment for the Flask application is setup and we need to execute the below given three commands to run the application on EC2 instance's localhost, so that we can type in the public DNS of the EC2 instance in our local browser to access the web-tier, i.e the frontend of our application.

2. App-tier

We need to have a folder named 'classifier' in /home/ubuntu of our app-instance EC2 and add the files 'image_classification.py' and 'imagenet-labels.json' in that folder. After this, we add the 'pystart.sh' shell script in /home/ubuntu and edit the crontab, in order to run the classification code automatically when the instance starts. The crontab can be edited by typing the commands;

1. crontab -e
2. @reboot /home/ubuntu/pystart.sh

With these configurations installed in the EC2 instance, we can build an AMI from this instance, so that we can create instances from this AMI, for autoscaling. All the app-instances created while auto-scaling the application will be using this AMI ID, so that all our configurations are up and running.