CSE 546 — Project 1 Report

Shubham Chawla (1227415724)

Anurag Banerjee (1225497546)

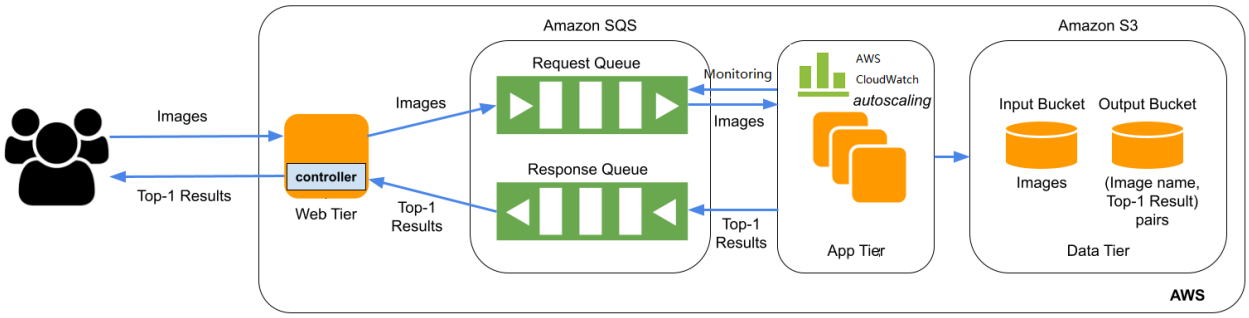
Vedant Munjaji Pople (1225453164)

1. Problem statement

This project aims to develop an IaaS cloud application, that can provide image recognition as a service, where the input consists of an Image and the user will obtain the object image(label) as the output. Various AWS cloud resources like EC2, SQS, and S3 buckets are used to perform Image recognition on images provided by the users. The application will handle multiple requests concurrently. It will automatically scale out when the requests increase and scale in when the requests decrease.

2. Design and implementation

2.1 Architecture



While Web Tier provides an API for users to send their requests containing images, the entire architecture uses various AWS services. Once the request comes on the Web Tier's controller, it uploads the file to an S3 bucket while simultaneously posting the image's key to an SQS request queue, waiting for the classification results. Scaling to match the request load, once an App Tier starts, it listens to the SQS request queue for the image's key. App Tier downloads the image from the input S3 bucket, runs the classification's script on it, and then uploads the results in an output S3 bucket while sending an acknowledgment message to the response queue. Web Tier listens to this queue and prints the output on its console.

2.1.1 Components:

1. AWS EC2: EC2 allows users to create virtual machines in the AWS cloud for working on applications. This project uses the EC2 instance for usage at the App Tier and Web Tier.
2. AWS SQS: SQS allows outgoing and incoming messages in our architecture. SQS helps scale and decouple services.
3. AWS S3: This project uses S3 for storing results from the App. S3 has the ability of scalability of storage through application programming interfaces. Buckets will store data in a key-value form for any file format.

2.1.2 Frameworks:

This project uses the JAVA Springboot framework for REST applications at the App Tier and Web Tier.

2.2 Autoscaling

Autoscaling is done at the App Tier. When the Web Tier instances get a new request, the input is put into the input SQS queue. The Web Tier can also have concurrent requests served by different threads and each thread listens to the SQS response queue for the classification results.

The project implements Step scaling, where the metrics from Cloudwatch are considered for making decisions about scaling up and scaling down App Tier instances. The metrics are fetched from the given Cloudwatch dashboard, as M1 as M2, relating to the number of messages in the request queue and App Tier instances available.

The successive triggers in scaling up include checking for the difference between the metrics to get the number of messages in the queue and the number of App Tier instances hence spawned using the difference until the maximum number of instances i.e. 20 is reached.

For scaling down the difference of metrics is used to get the number of instances running. If the requests in the queue are less than the number of App Tier instances, then the number of instances is reduced.

2.3 Member Tasks

Explain in detail what is each member’s task is in the project.

Shubham Chawla:

Shubham started with the development of the App Tier application of the project. This part entailed writing a Springboot application that listens to an SQS queue for image keys. Once received, the application downloads the image from the S3 bucket and executes an image classification script on that image. The python script's results are extracted and uploaded to another S3 bucket, and the final acknowledgment is sent to Web Tier using an S3 response queue. Here, platform-dependent paths, such as script name, script path, and download directory, are picked via Spring's run arguments for environment-focused development.

Anurag Banerjee:

Anurag was responsible for developing the Web Tier of the project, which receives requests from the workload generator, sends the message to an SQS request queue, and uploads the image to an S3 bucket. The Springboot application then waits for an acknowledgment from the App-Tier on a dedicated SQS response queue and displays the result on the console of the Web-Tier instance. AWS-related properties are passed via run arguments, ensuring environment-focused development and security.

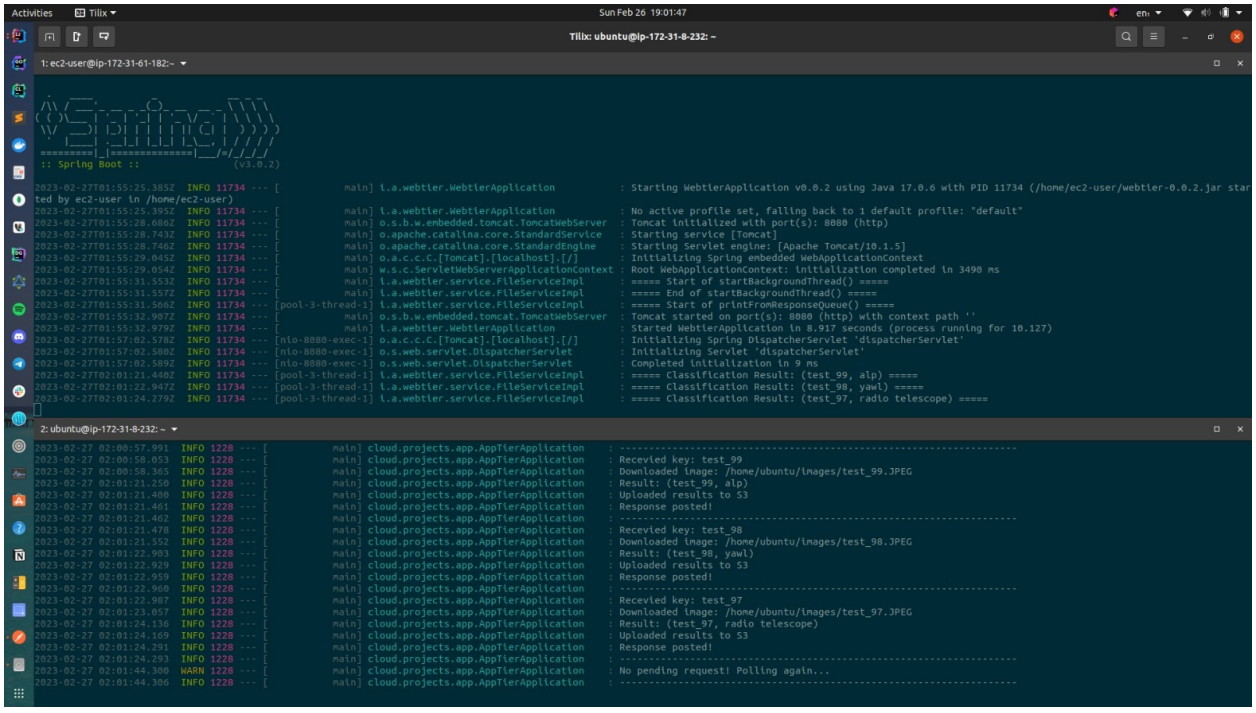
Vedant Munjaji Pople:

Vedant focused on Web and App Tier's functioning on AWS. This responsibility primarily entailed ensuring the load balancing of App Tier's instances. While working on CloudWatch and capturing the metrics of the Auto Scaling Group, App Tier's instance count was the function of the number of visible messages on the SQS request queue. Vedant maintained the project documentation, ensuring milestone resolutions.

3. Testing and evaluation

The code was tested and evaluated rigorously throughout development. The Web Tier endpoints are tested to get correct outputs for the mappings of uploading and downloading the images. Then the downloaded images are tested in the App Tier as well. A dummy python script was used for testing the classification of the images received. The image upload to the S3 bucket and download on the file directory of the App Tier is tested as well. The Scaling up and down of the EC2 instances was ensured using the workload generator provided. The CloudWatch provided metrics and graphs for evaluating the performance of the project's EC2 instances.

The output of the code tested is as follows:



4. Code

4.1 Web-Tier

Configuration:

There are 2 main configuration files in the Web Tier module. The SQS configuration file configures the SQS queues. The queues take in AWS regions, AWS access key, and AWS secret key for setting up the client. The client is created using Client builder. The builder then uses the region, and credentials provider to build the SQS client. Similarly in the S3 Configuration, the client requires the AWS region, AWS access key, and secret key. The S3 client is made using the builder, with the region, and credentials as the input parameters.

Controller:

The controller specifies the mappings for the input images of the model. The controller uses the file service to make the mappings for uploading the images. The response for uploading images uses the IO Exception to make sure the file is uploaded correctly.

Services:

The services use the AWS credentials, like region, access key, and secret key to make sqs client and s3 client implemented from the configuration modules. The uploadFile method in the services class uses the file as the key. A putObjectResponse object is made using the builder, which uses the bucket name, key and content details to build the object. Then the URL is generated from the S3 client for the same, using the builder, key, and bucket name. Then the object and URL are logged into the console. Similarly, the Queue URL is also generated using the builder and queue name. The message request is made using the builder, using the report url as the message body, and a delay of 5 seconds is introduced. The exceptions due to Amazon service exceptions and Amazon client exceptions are caught using the try-catch block and respective error messages are logged to the console. Override methods to download files, delete the file, and list all files are declared at the end.

4.2 App-Tier

Python Services:

This module consists of working with the python script to classify images in the App Tier. The path to the python script is imported. This path is given to the process builder and which builds the script, the script path, and gets the absolute path of the image. The script is run, where the builder starts the process and the result string is initialized to null. The input stream is converted to a string. If there is any error caught during this process, the exception is thrown with the error message. If there are furthermore errors in the script, classificationexception error messages are thrown.

S3 Services:

The S3 services take in AWS credentials like access key, secret key, region, input bucket name, output bucket name, and directory to download images as the parameters. These parameters are assigned their respective values from the credentials. The builder then builds the S3 client from the credentials provided. The downloadImage method creates a request to make an Object request using buckets and keys.

A new file is then created using the request built from the builder, formatted for the image format, and saved in the download directory. If there is no existing file in that, a new file is created, and the S3 request reads all bytes from it.

SQS Services:

The SQS services use the AWS access key, secret key, region, and SQS client. The setup, sqs client builds region, provider, and build. The Queue attributes build queue URL and a new concurrent hash map called cache. After completing the setup, the getNextKeyFromSQS method gets the text messages. The response receives messages from the response, and then the key is generated from the response body, inserted into the hash map, and then the key is returned. An acknowledgment is then sent to the logger in the form of a String. The SQS message is built using the builder, and the message for which the result has been displayed is removed from the hash map.

4.3 Programs Installed

1. Java is used in this project. The project uses the Springboot framework for the development of the Web Tier and the working of the App Tier. Springboot provides dependency injection, and easy API development annotations and XML configurations.

NOTE: Both App Tier and Web Tier require Java 17 to run.