Coursework – EMATM0051 Large Scale Data Engineering [FinTech]

Part 1 – The Public Cloud is ideal for data processing

Public cloud is the next technology that will be adopted by everyone in time. The technology era is slowly ending, while the cloud era is slowly rising prominence, not only due to the cost of building hardware infrastructure but also the perceivable delay that is apparent when using physical data warehousing. The covid-19 pandemic has made it more apparent about the requirements of easy access to data, that is more secure than transferring files and the need for collaboration over the internet.

Many enterprises are slowly converting towards a cloud infrastructure to not only reduce the current cost of running the business but also to reach out to a farther market and customers that are stuck in their homes. Public cloud has also led to a boom in remote work jobs as it keeps the leisure of life safe, while keeping the efficiency same, or sometimes even better.

The public cloud being the next thing leads to the question of data processing which has also been in the rise in the recent years, the requirement of large datasets that must be stored and processed for hours for a chance of a proper prediction or pattern outcome among data scientists has been apparent. The resources required to run proper algorithms is nothing to scoff at, the importance of CUDA in complex machine learning algorithms has become essential, which for research or learning is a costly endeavor. The public cloud and the services it can provide through various cloud platforms ease up the issue of incurring costs in advance, as marketed by various cloud platforms, the services provided are pay per use, and will become cheaper the more people use it as the cost for the data warehouse has already been incurred and repaid. It also saves up the idea of constant upgrading of different peripherals to match the current technology.

The AWS services that make it worthwhile and apparent that public cloud is ideal for data processing include AWS compute optimize, AWS autoscaling, AWS S3, AWS X-Ray, AWS Glue. While AWS X-Ray and AWS Glue are the new services that have been provided for application management, monitoring and optimization, while Glue is a dream come true for doing data analytics on the cloud. The long existing services like compute optimize, auto scaling and S3 make cloud a stable choice, the auto scaling and compute optimize makes sure you do not run out of resources while your code is running and algorithm, which has happened to me more times than I'd like to admit. While S3 makes sure you don't run out of storage while using API to keep getting the latest data for proper analysis of a trend, a stock, or any real-time data that exists globally.

There are a few instances where the public cloud does feel effy, including the issues with data-usage and cookie-usage by different companies, the issue with data privacy, which lead to some situations where data processing can do more harm than good. One of the primary reason is lack of competition if you get trapped under one cloud platform with no way to shift to another without incurring massive changes in personnel and relearning the entire framework, data processing of data that can easily lead to privacy leaks and social networking data processing which has caused enough problems world-wide, in politics and fake news.

The public cloud is ideal for data processing if utilized with proper care and set guidelines for the user and the provider.

Part 2 Scaling the WordFreq Application

Tasks-

> Install the application

> Design and Implement Autoscaling

> Perform Load Testing

> Optimize the Wordfreq architecture

> Final Repot

Report -

The application calculates the frequency of words in a text file. It considers all the unique words in the text file separated by _space_ and if any word is repeated that is added to the count, the final calculation is done at the end of the file and then it is sorted before given the printout. The file is read the queue service one at a time and stored into DynamoDB. The DynamoDB table does not sort the values, it stores all values as they are sorted when the output is to be displayed to the user.

The application considers only plain text files and does not take complex characters in the filename.

The autoscaling configuration I used to implement were t1.micro and t3 micro instances, over the existing free tier t2.micro that can be changed depending on the day and work, to speed through the process of changing between these instances, they were stored as launch instance types and could convert the auto scaling groups as per the requirements.

Dynamic scaling policies (1) Info

Target Tracking Policy

Policy type:
Target tracking scaling

Enabled or disabled?
Enabled

Execute policy when:
As required to maintain Average CPU utilization at 50

Take the action:
Add or remove capacity units as required

Instances need:
300 seconds to warm up before including in metric

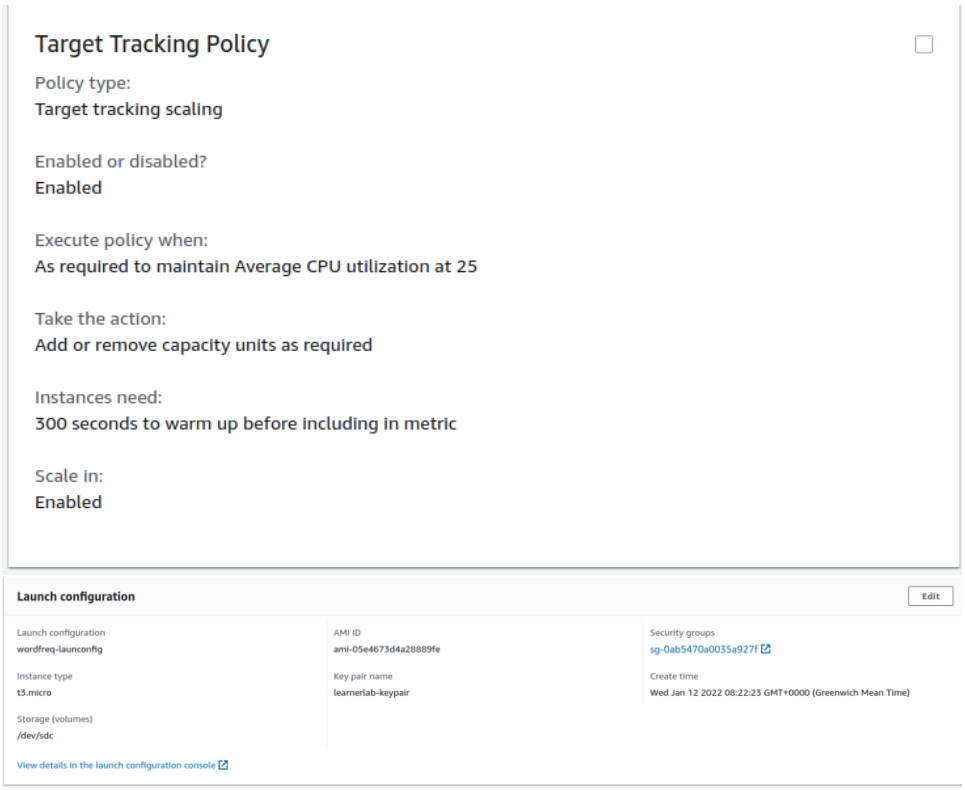
Scale in:
Enabled

Successful	Launching a new EC2 instance: i-0cdd7040b072e59ea	At 2022-01-12T08:06:11Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2022 January 12, 08:06:13 AM +00:00	2022 January 12, 08:06:31 AM +00:00
Successful	Terminating EC2 instance: i-0d72cdef0bde9058e	At 2022-01-12T08:06:01Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2022 January 12, 08:06:01 AM +00:00	2022 January 12, 08:11:21 AM +00:00
Successful	Launching a new EC2 instance: i-0d72cdef0bde9058e	At 2022-01-12T08:04:10Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2022 January 12, 08:04:13 AM +00:00	2022 January 12, 08:04:29 AM +00:00
Successful	Terminating EC2 instance: i-00e10859514b28016	At 2022-01-12T08:04:00Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2022 January 12, 08:04:01 AM +00:00	2022 January 12, 08:09:18 AM +00:00
Successful	Launching a new EC2 instance: i-00e10859514b28016	At 2022-01-12T07:52:10Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2022 January 12, 07:52:13 AM +00:00	2022 January 12, 07:52:30 AM +00:00
Successful	Terminating EC2 instance: i-00a0607225d9675f1	At 2022-01-12T07:52:00Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2022 January 12, 07:52:01 AM +00:00	2022 January 12, 07:57:27 AM +00:00
Successful	Terminating EC2 instance: i-0f28b6994c63656c7	At 2022-01-12T07:22:42Z a monitor alarm TargetTracking-wordfreq-autos-group-AlarmLow-717c9f86-d1c0-47c3-b1e6-fe580cb0eb85 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 1. At 2022-01-12T07:22:50Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2022-01-12T07:22:50Z instance i-0f28b6994c63656c7 was selected for termination.	2022 January 12, 07:22:50 AM +00:00	2022 January 12, 07:28:43 AM +00:00
Successful	Launching a new EC2 instance: i-00a0607225d9675f1	At 2022-01-12T07:07:04Z a monitor alarm TargetTracking-wordfreq-autos-group-AlarmHigh-3fff858c-8bbf-4231-845a-e29cf65ae821 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2022-01-12T07:07:09Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2022 January 12, 07:07:13 AM +00:00	2022 January 12, 07:12:44 AM +00:00
Successful	Launching a new EC2 instance: i-0f28b6994c63656c7	At 2022-01-12T07:02:05Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2022 January 12, 07:02:08 AM +00:00	2022 January 12, 07:02:41 AM +00:00

AWS Notifications <no-reply@sns.amazonaws.com>
Wed 1/12/2022 8:06 AM
To: Akshat Pande

Service: AWS Auto Scaling
Time: 2022-01-12T08:06:30.790Z
RequestId: a455f935-4ae1-f56e-4fad-a4b1a29e09b5
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 035036656553
AutoScalingGroupName: wordfreq-autos-group
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:035036656553:autoScalingGroup:990af58b-a68d-40a3-a62e-7867990fda43:autoScalingGroupName/wordfreq-autos-group
ActivityId: a455f935-4ae1-f56e-4fad-a4b1a29e09b5
Description: Launching a new EC2 instance: i-0cdd7040b072e59ea
Cause: At 2022-01-12T08:06:11Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.
StartTime: 2022-01-12T08:06:13.885Z
EndTime: 2022-01-12T08:06:30.790Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-0cdd7040b072e59ea
Details: {"Subnet ID":"subnet-03aeead090ca288fc","Availability Zone":"us-east-1a"}
Origin: EC2
Destination: AutoScalingGroup

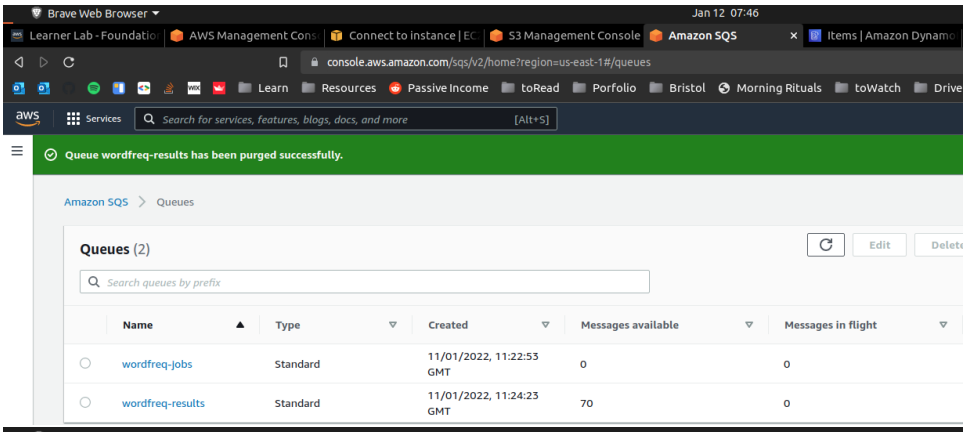
It was also dynamically scaled based on the health of the instances and CPU utilization, after doing multiple test loads and checking networks, the CPU utilization was around 40% for file uploads of around 150mb with all files having separate unique data. Hence, the dynamic settings were brought down to 30% to avoid halting due to instance timeout times (errors occurred in run_worker.sh journal) and added load balancer in case of multiple parallel uploads to the Wordfreq Application.

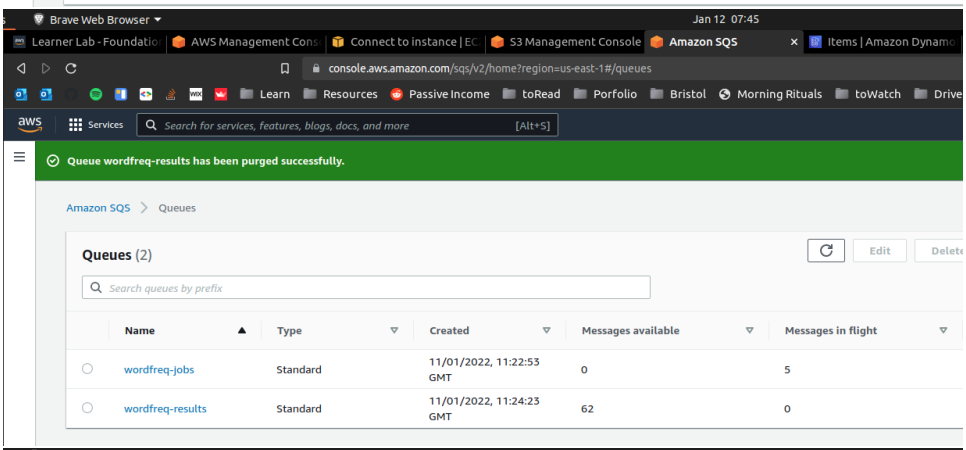


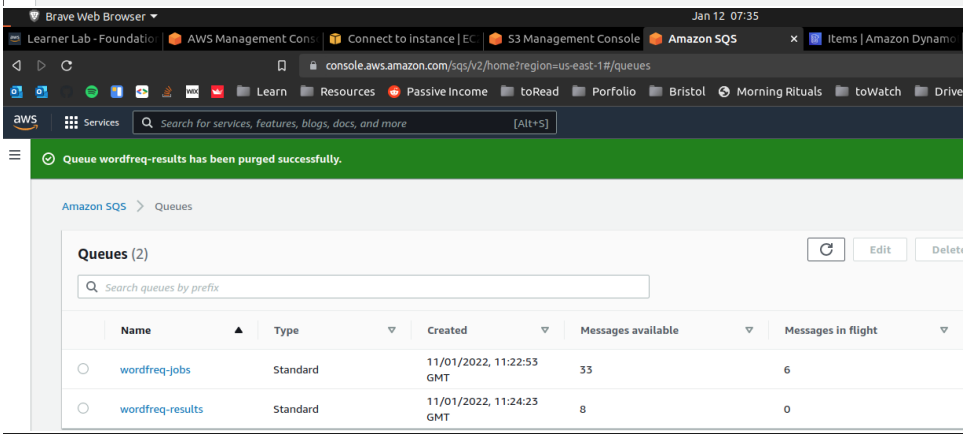
Building and Testing

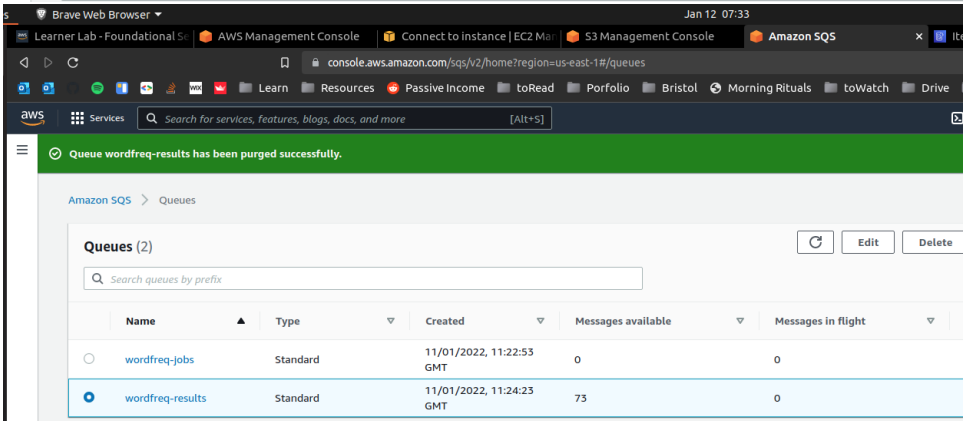
After following the guidelines to setup, the app, the first few steps were to setup the auto-scaling configuration and the auto scaling groups that would be applicable to the wordfreq application.

I primarily tested the wordfreq application with the default settings and the backed-up AMI image with the same instance for testing out the application. The delay of 10 seconds aside, there were inconsistencies during the testing when going through the journal and the SQS, messages in the wordfreq-job were being restored back due to instance timeouts and exceptions in filenames took the entire process into a slow procedure throughout all files with failed messages.

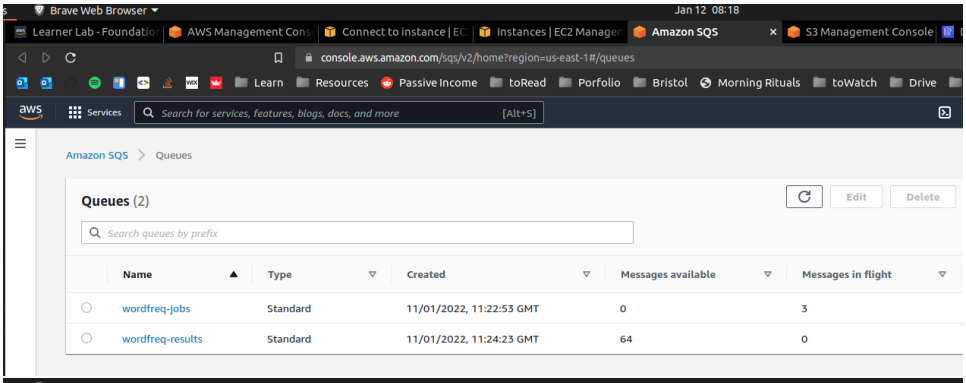

Brave Web Browser
Jan 12 07:46
Amazon SQS
console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Search for services, features, blogs, docs, and more [Alt+S]
Queue wordfreq-results has been purged successfully.
Amazon SQS > Queues
Queues (2)
Edit
Search queues by prefix
Name Type Created Messages available Messages in flight
wordfreq-jobs Standard 11/01/2022, 11:22:53 GMT 0 0
wordfreq-results Standard 11/01/2022, 11:24:23 GMT 70 0

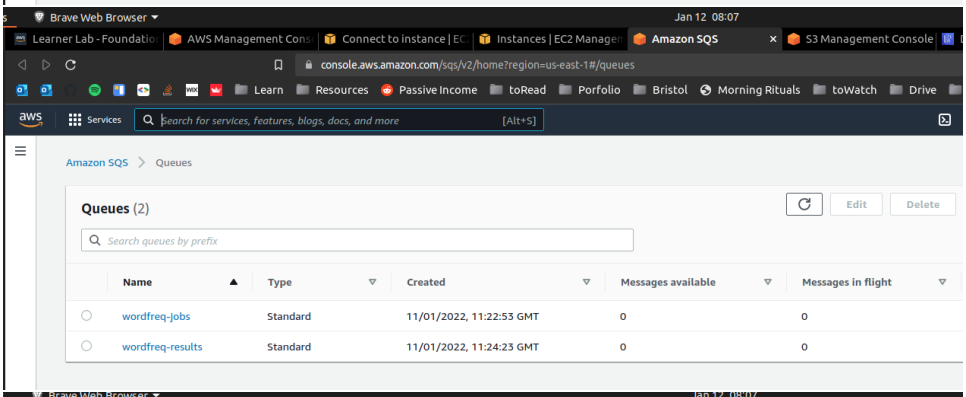

Brave Web Browser
Jan 12 07:45
Amazon SQS
console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Search for services, features, blogs, docs, and more [Alt+S]
Queue wordfreq-results has been purged successfully.
Amazon SQS > Queues
Queues (2)
Edit
Search queues by prefix
Name Type Created Messages available Messages in flight
wordfreq-jobs Standard 11/01/2022, 11:22:53 GMT 0 5
wordfreq-results Standard 11/01/2022, 11:24:23 GMT 62 0

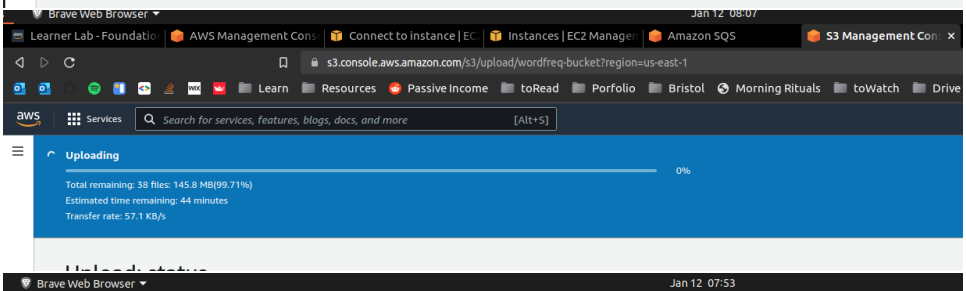

Brave Web Browser
Jan 12 07:35
Amazon SQS
console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Search for services, features, blogs, docs, and more [Alt+S]
Queue wordfreq-results has been purged successfully.
Amazon SQS > Queues
Queues (2)
Edit
Search queues by prefix
Name Type Created Messages available Messages in flight
wordfreq-jobs Standard 11/01/2022, 11:22:53 GMT 33 6
wordfreq-results Standard 11/01/2022, 11:24:23 GMT 8 0

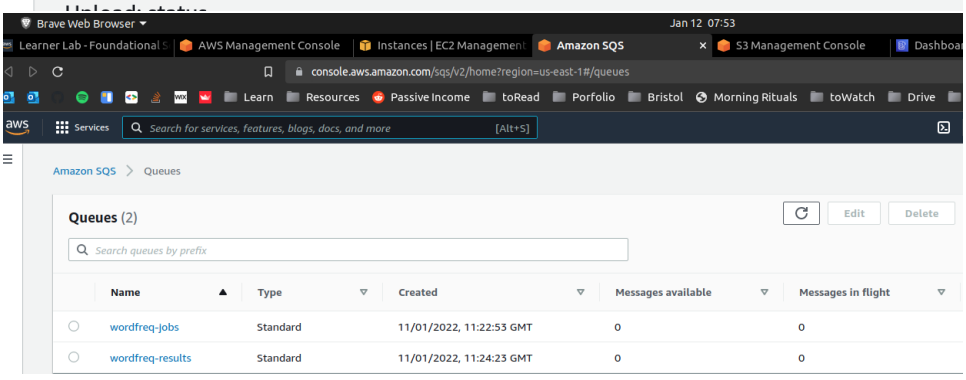

Brave Web Browser
Jan 12 07:33
Amazon SQS
console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues
Search for services, features, blogs, docs, and more [Alt+S]
Queue wordfreq-results has been purged successfully.
Amazon SQS > Queues
Queues (2)
Edit
Delete
Search queues by prefix
Name Type Created Messages available Messages in flight
wordfreq-jobs Standard 11/01/2022, 11:22:53 GMT 0 0
wordfreq-results Standard 11/01/2022, 11:24:23 GMT 73 0

After the primary load testing with two different instance types, t1,t3 (except the primary instance used) , which had a time of around 15-20 mins depending on uploading time and processing time as shown below. I also setup a notification email service for changes in the autoscaling formation and termination to closely look at the costs that could be incurred.

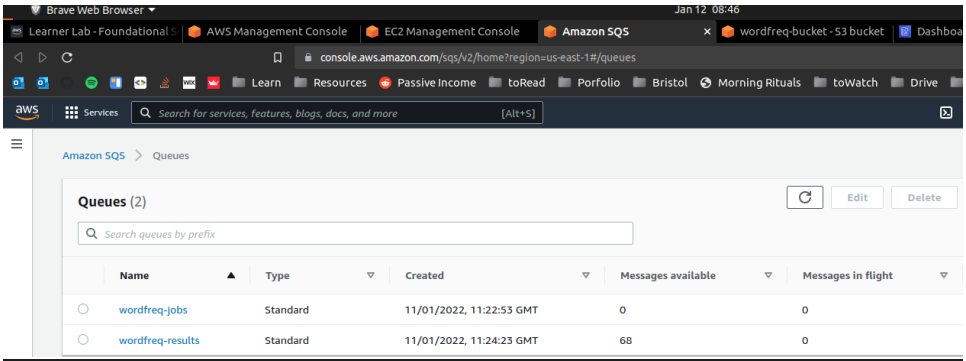


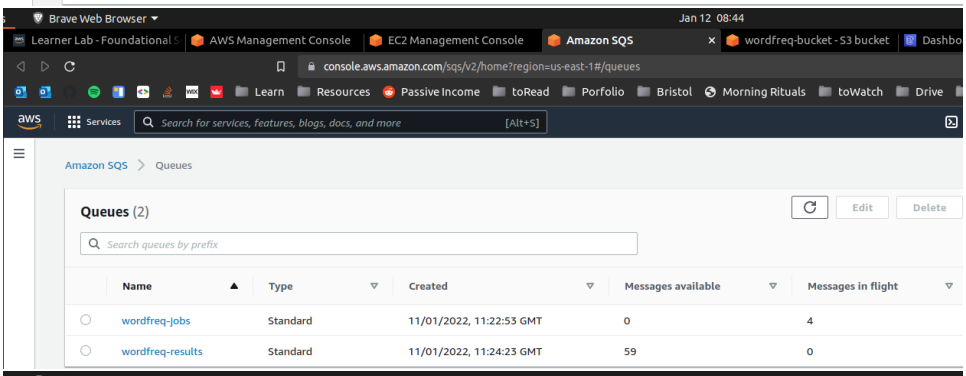


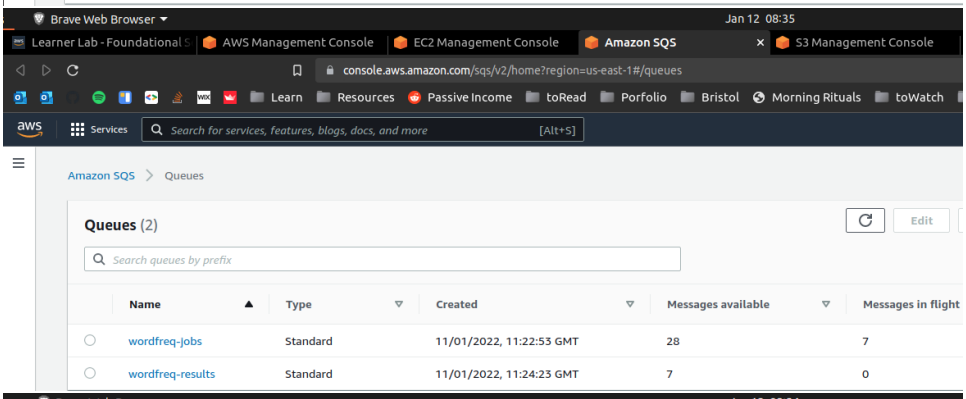


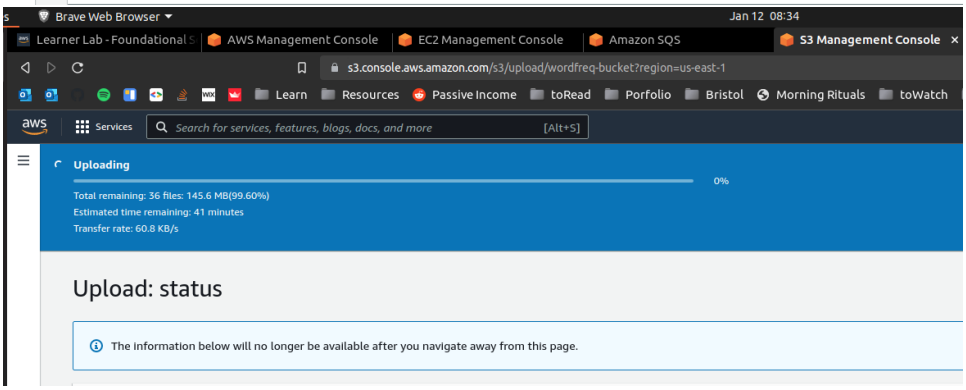


The autoscaling group was changed to a t3.micro with a different dynamic scaling setting, the cpu utilization requirements for new instances was reduced to 30% to decrease the time required to process files between instances and not trigger the instance timeout function in the sqs, leading to a rollback of the message request.









After the settings were changed, the time was reduced to half of the previous test. Most of it was due to the improvement in the instance type and reduction in the CPU utilization requirements that lead to higher allocation of instances and less rollback occurrences.

Company B:

Requirements: extremely cost-effective, efficient (scalable and resilient), occasional use, basic security and long-term data backups required.

Design –

Primary Instance – t2.micro, processes in a reasonable amount of time with low cost and enough memory to go through files in reasonable time.

Security group – Simple ssh rule to be accessed by the users for any changes and uploads, and https rule if the application is to be distributed to other employees. Proper creation of user groups with AWS IAM services to give proper rules and service access to avoid any mishaps.

AWS Cost Optimization – optimizes the cost of all services after analyzing the usage and suggests proper AWS services that can be used to maintain the performance but lower the costs.

DynamoDB- Simple NoSQL database, same as setup to store the results of the application

S3- Primary gateway to transfer uploaded files to job for applying the wordfreq application and can be auto scaled using lower values to create additional S3 in case of huge upload.

Glacier – archive storage provided by AWS can be used for long term backup or

AWS Backup – used for backing up anything on the cloud, can store the processed files that can be transferred to S3 bucket for processing again.

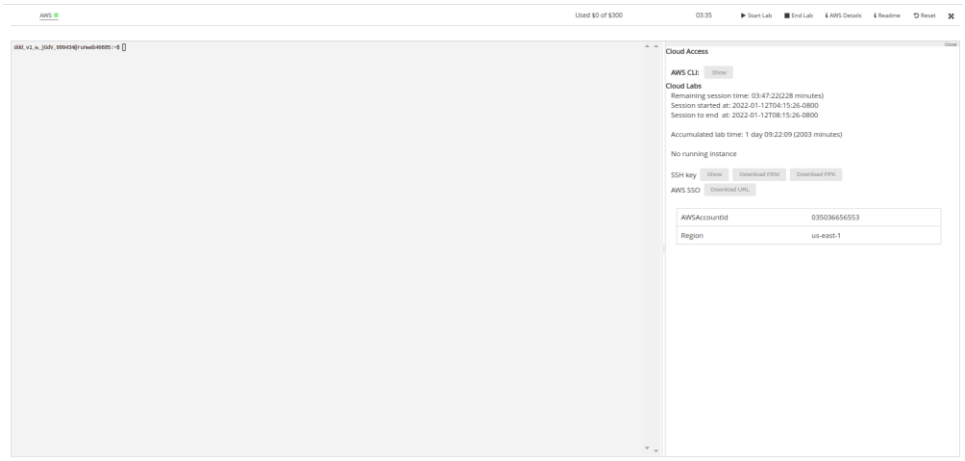
AWS compute optimizer – to optimize the compute services so we don't employ more resources than necessary given the situation.

AWS auto scaling for instances – helps in scaling in unseen circumstances but also provided proper resilience to the instances and perform health checks regularly to avoid any downtime of the application.

AWS Database migration service or AWS Snow – applicable to store the results in a physical or local field that can be kept on a physical storage which won't incur any cost at all.

Issues during the testing and working on wordfreq applications.

- No update to costs in the aws learner lab, even though it showed cost in the instances that would be incurred per hour, the update was not applied when starting or ending labs and as such I have not employed any heavy duty instances, to avoid any problems of running out of funds without realizing.



- Instance_timeout due to network speeds and working but was managed by using a faster instance and better autoscaling optimization