Task 1 : Architecture of “MyTravel” Application :

MyTravel” website is designed as a platform which is accessible globally to share blogs related to travelling. The architecture is optimised for high availability, low latency, auto disaster recovery and milestone based notifications.

Global Accessibility of AWS

Service : Amazon Route 53 - This will ensure that the website is available to public globally with low latency and high availability

- DNS Resolution is available in Route 53 which makes it reliable and efficient so that worldwide users can access website quickly. It also supports health checks and failover routing.
- Balance traffic across multiple AWS regions which optimises latency and redundancy

Content Delivery and Latency Optimisation

Service: Amazon Cloud Front - Reduce latency and deliver content like static images and files to users through a Content Delivery Network (CDN)

- Caches the static content at the very edge locations, which delivers content faster to users

Integrates seamlessly with S3 for static file hosting and also provide enhanced security with the use of AWS Web Application Firewall (WAF). It can also leverage CloudFront functions to modify headers dynamically for enhance cache and analytics

Routing Request

Service: AWS API Gateway - It acts as a central point for a hub of API operations.

- Directs user requests regarding blog interaction, visitor data, or content uploads.
- Invokes direct AWS Lambda for backend logic processing.
- Enforces request throttling, monitoring, and secured handling of requests

Authentication

Service: AWS Cognito - It provides authentication tokens

- The sign-in, sign-up, and authentication tokens for users.
- Management of the federated login options such as Google, Facebook, or any such.
- Roles are specified for access: End Users: They access directly blogs and view content only. Admin Users: Can upload, manage or delete any type of content.

Hosting the Static Website and Content Storage

Service: Amazon S3 - It is used to host static content of website and store the travel photos and files

- It is very durable ensuring that data remains safe and available even when users are increasing. Versioning is present which helps to maintain diff versions of content uploaded in case accidental deletions or overwrites. It optimises storage costs through Intelligent Tiering by automatically moving infrequently accessed files to lower cost tier

Visitor Tracking and Milestone Monitoring

Service: Amazon DynamoDB - Tracking of no. of visitors on website and manage the logic of milestone achievement for sending congratulatory emails

- Provides single digit millisecond response time which ensure real time visitor tracking without affecting performance It handles spikes in traffic because of its auto scaling properties

Dynamic Actions and Backend Automation

Service: AWS Lambda - Handling of dynamic actions including sending congratulatory email, updating visitor milestones and initiating backup workflows

- It offers server less environment which reduce cost and we pay for only services we execute

Seamless integration with S3 for triggered based events and SES for sending mails. It also designs logic for data enrichment like retrieving visitor demographics before triggering further actions

Backup and Disaster Recovery

Service: AWS Backup and S3 Cross-Region Replication - Auto backup of data and replication of it across regions for recovery during disasters

- Automates backup scheduling and creation of period backups os S3 content and DynamoDB tables. Cross Region makes sure your data is available in another region ensuring resilience against regional outrages

Governance of Access

Service: AWS IAM Security - Enforces least privilege policies

Lambda can access S3 and DynamoDB, plus to SES. API Gateway enables permission to invoke Lambda. Admin Users can create role-wise access to manage content in S3 and API Gateway. The entire integrated system is intended to work with Cognito so that roles and permissions can easily manage dynamically.

Email Notifications for Milestone Achievements

Service: Amazon SES (Simple Email Service) - Send congratulatory email when website exceeds 1,000,000 visits

- It is cost effective, reliable and also integrates with lambda for event-driven email notifications

Supports multiple formats. It add custom visitor analytics from Dynamo DB

Monitoring and Alerts

Service: Amazon CloudWatch - Monitor application health, track metrics and set alarm for milestone notification

- Monitor application health in real time, metrics for visitor count and overall performance

Alarms can trigger lambda functions for automated workflows. Dashboard setups are done here which shows website performance in visualisations and also visitor analytics

Securing the Website

Service: AWS WAF (Web Application Firewall) - Protect site form threats like cross-site scripting, DDoS attacks

- WAF integrates with CloudFront, inspect and filter HTTP/S traffic before it reaches the origin

Customised rule engine to block specific patterns. Uses rate based rules to block IP with abnormal request rates enhancing protection

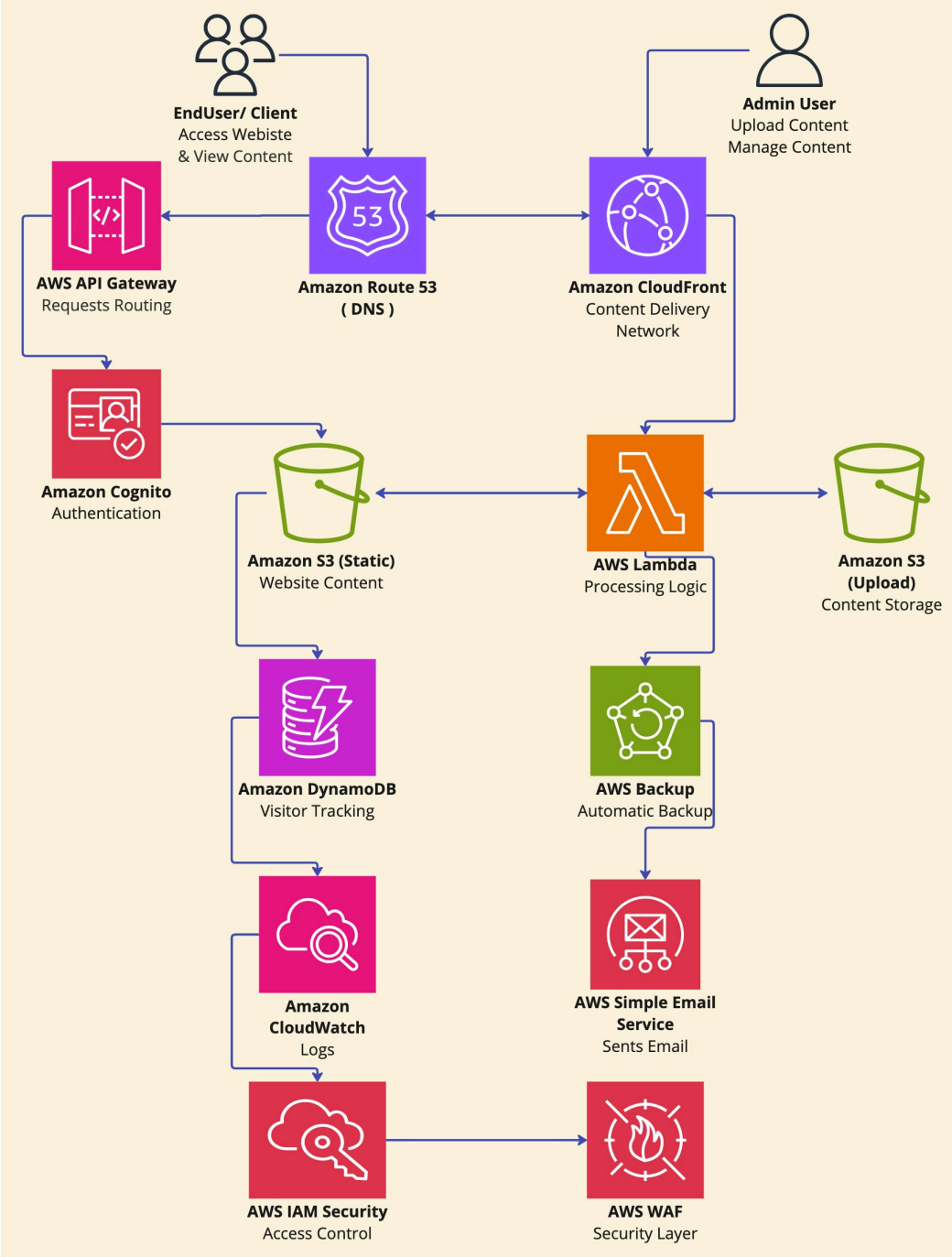


Fig 1: Architecture Diagram of "My Travel" Website Application

Task A: Installation And Configuration of The WordFreq Application

It is server less and distributed system that processes text files to find the 10 most frequently occurring words. It relies on AWS services to automate its workflow. It offers a computer capacity which can be resized and modified as per the requirements. Following are the steps of how application works:

1. **EC2 Instance :** We launched a t2.micro instance with the ubuntu server and configured the following

    - AWS Identity & Access Management (IAM) for permission allowance
    - Security groups which allows inbound and outbound traffic
    - Allocation of 30GB storage for application files and logs content

2. **S3 Bucket :** It gives users the ability to store, safeguard, and acquire information from buckets. We created the following 2 buckets

    - Uploading Bucket - User will upload text files in this bucket
    - Processing Bucket - Files present here are prepared for analysis

3. **SQS Queues :** This allows user send, store, and receive messages between software components at any scale, without loss of messages or reliance on the availability of other services. We have created 2 SQS Queues

    - Jobs Queue - As soon as files are uploaded, it receives a notification
    - Results Queue - Notification of completion of all the tasks are stored here
    - We will also update the queue policies to allow AWS entities to write queues
    - Then, we will note down the queue URLs for further tasks

4. **DynamoDB Table :** It is a key-value NoSQL database service with which you can store and access data in Amazon Web Services (AWS).

    - To store the output obtained from a processed text file, we created a table named wordfreq is created.

5. **Event Notification Configuration :**

    - We have set up the Uploading Bucket to trigger event notification to the Jobs Queue each time file is uploaded.

6. **Preparing the Development Environment :** We have installed the dependencies including :-

    - AWS CLI
    - Go Language

7. **Configuration of Worker :**

- We will edit the configuration to specify SQS queue URL's
- Then, test the work manually and configure as background service for seamless operations

8. **Setup Testing :**

- We will upload the file to uploading bucket
- Further, monitor worker logs to make sure file processing is successful and results are getting generated

9. **Bulk Processing :**

- At the end, we will upload multiple files to uploading bucket for batch processing

Backup And Optimisation

- We will backup the EC2 instance which is recommend by AMI creation
- We will also test purging mechanisms for queues and autoscaling in order to ensure optimised scaling

End to End Workflow :-

Step 1 :- File upload - Uploading text files into the S3 bucket (Uploading bucket)

Step 2 :- Event Driven Workflow - When file is uploaded, an event is triggered to notify SQS queue which is the job queue

Step 3:- Processing - Worker instance which runs on EC2, it continuously pools jobs queue for tasks. Once task is received, worker process file by counting word freq

Step 4 :- Results Storage - Processed data is stored in Dynamo DB table. Notification are sent using SQS Queue about completed task

Step 5:- Logs - Logs and output details are available for monitoring via the worker instance.

Task B : Designing and Implement Auto-scaling

To implement an auto-scaling feature for the WordFreq application, you have to configure a few AWS services, including EC2 Auto Scaling, CloudWatch, and SQS, to scale the number of EC2 worker instances according to the number of jobs in the SQS queue. The triggering of scaling will happen through CloudWatch alarms that monitor SQS queue length (i.e., number of visible messages in the queue).
The steps below show the process of implementing the auto-scaling feature for the WordFreq application and how to set up the policy.

Auto-Scaling Group for EC2 Worker Instances

1. Creating a Launch Template

- Go to the EC2 console and create a new launch template for the worker instances. It will specify the Amazon Machine Image, instance type, and security group for the EC2 worker instances.
- For AMI, we will pick the one that corresponds with the instance type for the environment where WordFreq application is running.
- We will select the instance type then depending on our requirement. At first, we are selecting t2.micro and for further implementation and comparisons, we will select other ones also including t2.nano, t2.small, t2.medium etc.

2. Creating an Auto Scaling Group

- After creating launching template, we will create auto scaling group. The configuration for the auto scaling group is:

    - Minimum Size: 1 (It makes sure that 1 worker instance is always running)
    - Desired Capacity: It started with instances ranging from 2 or 3
    - Maximum Size: We will set an upper limit for the system to avoid launching too many instances (6 instances).
    - We will choose one or more availability zones where ywe want to distribute the EC2 instances.

3. CloudWatch Metrics & Alarms

- For auto-scaling based on the number of jobs pending in the queue, we will use the SQS metric ApproximateNumberOfMessagesVisible. It indicates the number of messages that could be processed by the worker instances and are held in the SQS queue.

Configuration:

- We will create 2 cloud watch alarms for scaling
    1. Scale Out Alarm (Queue length exceeds a threshold)
    2. Scale In Alarm (Queue length falls below a threshold)

- For scale out alarm configuration, we will choose SQS as our metric and select ApproximateNumberOfMessagesVisible metric for our SQS queue. Then, we will set the

threshold to 40 which will trigger the alarm when queue length will exceed 40 messages. We will set the policy to Scale Out Policy which we will discuss in the next section. At last, we will set the period 1 minute to check metric every 60 seconds.

- For scale in also we will do the same just, we will set threshold to 40 which will trigger alarm when the queue will fall below 40 messages. Then, we will set the policy to Scale In Policy. We will keep the period time 1 minute only.

4. Auto-Scaling Policies

We will configure auto-scaling policies for both scale-out and scale-in actions according to CloudWatch alarms. The auto-scaling policies, in this case, denote what actions are taken when alarm gets triggered.

Scaling-out Policy:

This is the policy that we will initiate when the queue length becomes more than the upper threshold. It adds one EC2 worker instance to the Auto Scaling Group.

We will go to the Auto Scaling Groups in the EC2 console. ChooseAuto Scaling Group. Create a policy.

Policy Type: Simple Scaling Policy
Name: Give the policy any name (for example, Scale-Out-Policy).
Modify Capacity: Add 1 instance.
Cooldown timer: Set the cooldown period.
CloudWatch Alarm: Select the Scale-Out Alarm created above.

Scaling-In Policy:

This is the trigger for the Scale-In policy when the queue length falls below the lower threshold (say, 20 messages) and removes one EC2 worker instance in the Auto Scaling Group.

Policy Type: Choose Simple scaling.
Policy Name: Name the policy (e.g., Scale-In-Policy).
Adjust the Capacity: Set to Remove 1 instance.
Cooldown Period: Set a cooldown period
CloudWatch Alarm: Select the Scale-In Alarm created above.

5. CloudWatch Dashboard and Monitoring

- We will monitor SQS queue metrics (ApproximateNumberOfMessagesVisible) and EC2 instance metrics (CPU utilisation, instance health, etc.).
- CloudWatch Logs: We will Enable logging for the Lambda function and EC2 instances to track the state of the jobs being processed.This will also help in debugging issues, if any, regarding job processing or scaling.

6. Testing the Auto-Scaling

- Uploading Multiple Files to the S3 Upload Bucket (triggering jobs in the SQS queue).
- Checking the auto scaling group of the EC2 worker instances based on defined threshold- if the queue length is monitored as well.
- The newest launched EC2 instance should terminate when the queue length falls below the threshold.

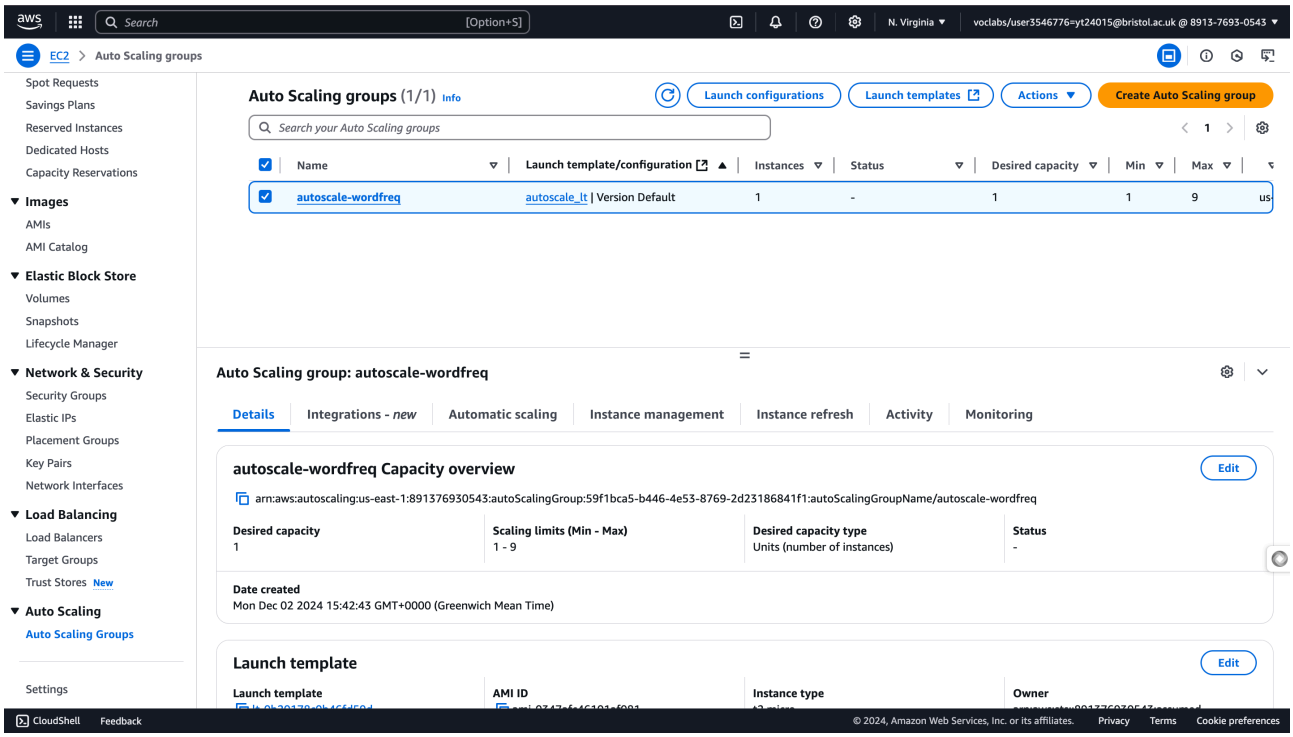


Fig 2: Auto Scaling Group Creation

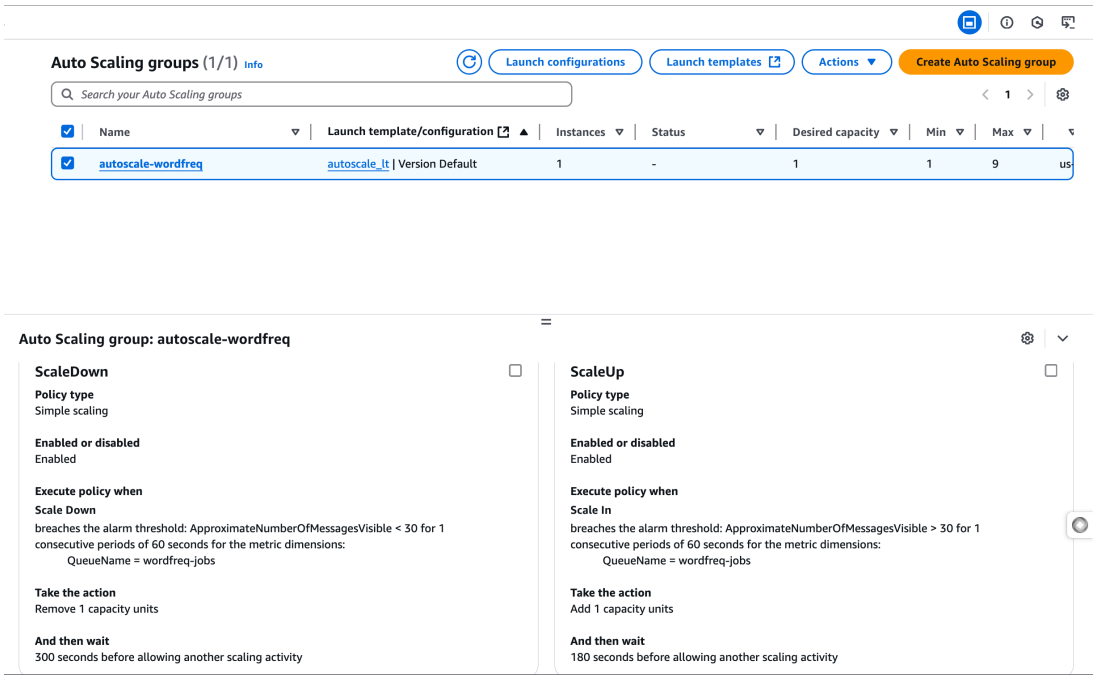


Fig 3: Auto Scaling Group Dynamic Policy Creation

Task C : Performing Load Testing

When configuring the load testing for auto-scaling WordFreq app, we will do the following to simulate the processing of 120 text files. This will test the auto-scaling infrastructure and that the scaling works as intended as well as allow us to fine-tune setup.

- First we will ensure that 120 files are uploaded to S3 Uploading Bucket i.e ary-wordfreq-dec24-uploading

- Then, we will purge the S3 Processing Bucket to ensure that no files are left in it. This will help in efficient load testing without any interference/problems

- After that, we will copy files from uploading to processing bucket. For that, we will first connect to the EC2 instance of the auto scaling group with the help of the SSH key. As mentioned the ReadMe File,

  ssh -i your-key.pem ec2-user@your-instance-public-ip

  We will run the following to connect with EC2 Instance

- Further, for copying all the files to processing, we will run the following command as per the ReadMe file

  aws s3 cp s3://ary-wordfreq-dec24-uploading s3://ary-wordfreq-dec24-processing --exclude "*" --include "*.txt" --recursive

This command will make sure all the .txt files are copied to the processing bucket, the WordFreq application will start to process them and the messages will be queued in SQS queue

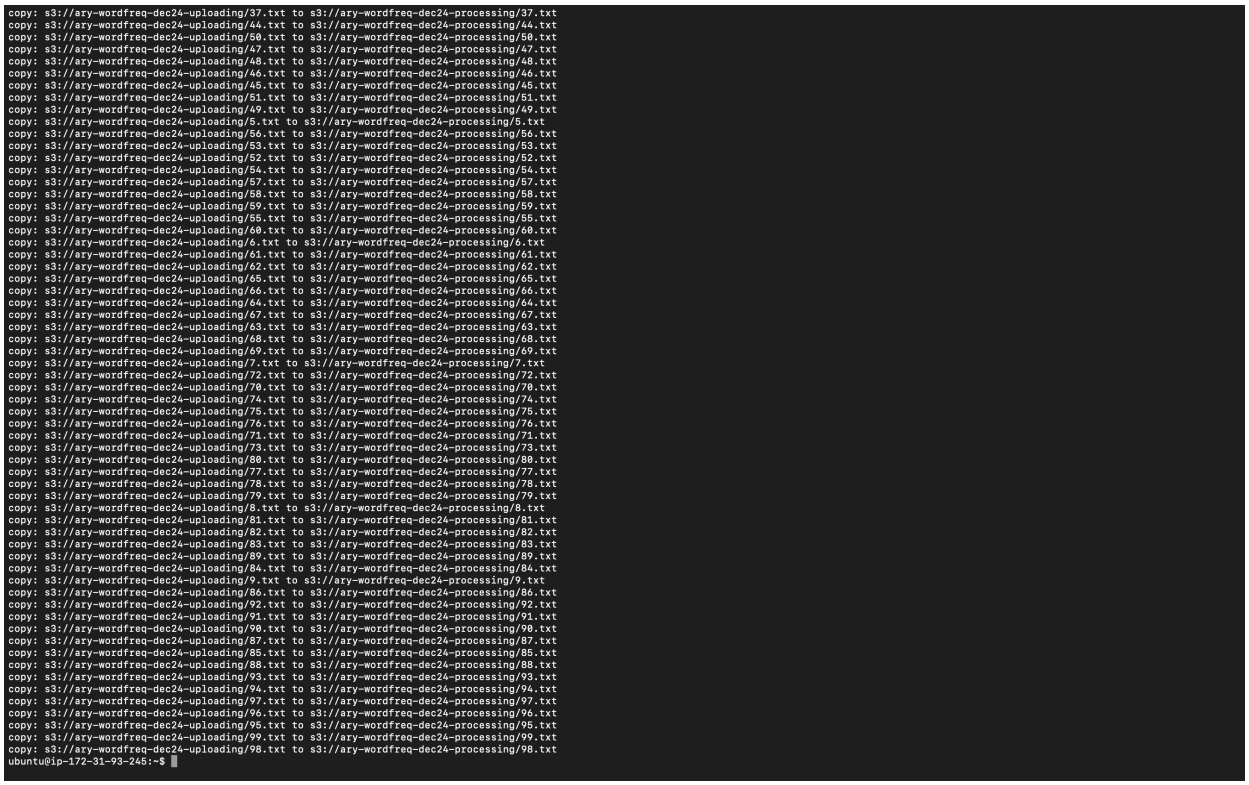


Fig 4: Processing of files by WordFreq in SSH

Measuring Scaling and Application Behaviour:

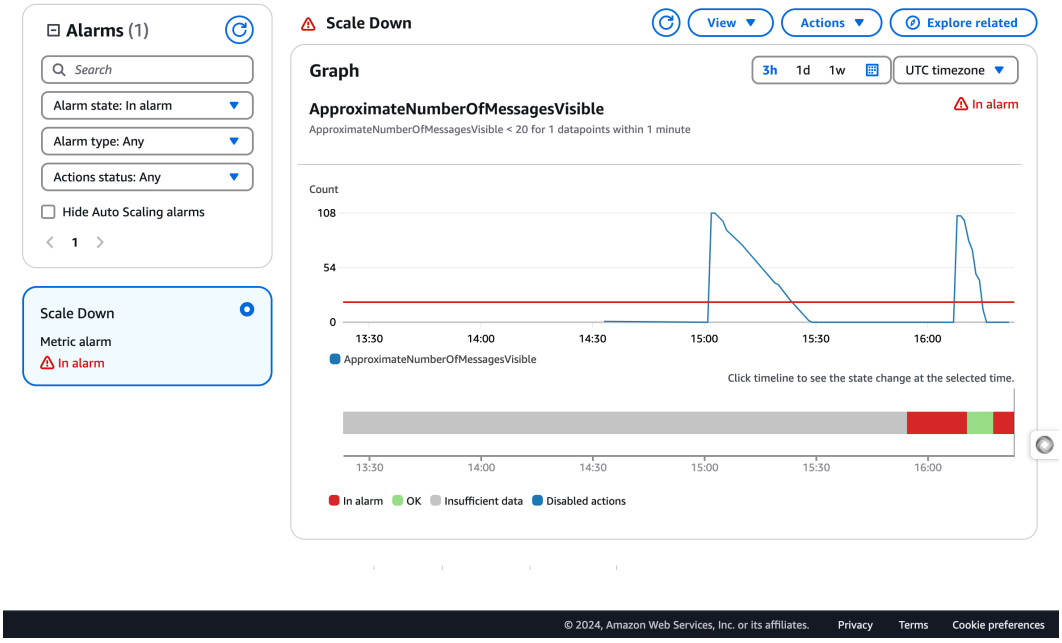


Fig 5: Scaling In and Scaling Out Behaviour

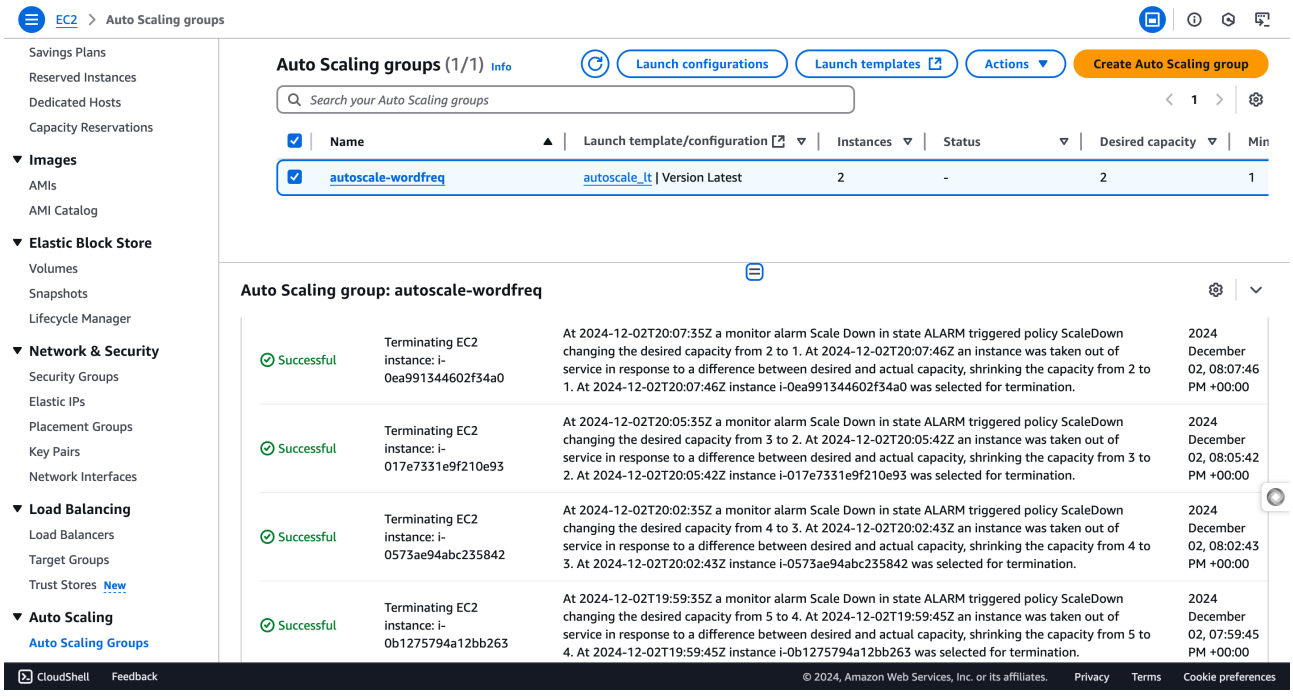


Fig 6: Status of Instances (After launching & Termination)

- Scale out (Add instances): The number of instances we have added to Auto Scaling group should ideally be visible in the Auto Scaling Group console. When the job queue (SQS) fills with tasks, it should launch new instances to take care of the load.
- Scale in (Remove instances): It marks when the job queue gets reduced (after some files processed), where it'll allow Auto Scaling to scale down existing instances. Here look for terminated instances.

- Auto Scaling Console: Go to the Auto Scaling Group in the AWS Console and view the instance count and status of each instance. We will see how instances are being launched and terminated dynamically depending on the workload.

Assess the Behaviour of SQS Queue and EC2 Instance:

Status of SQS Queue:

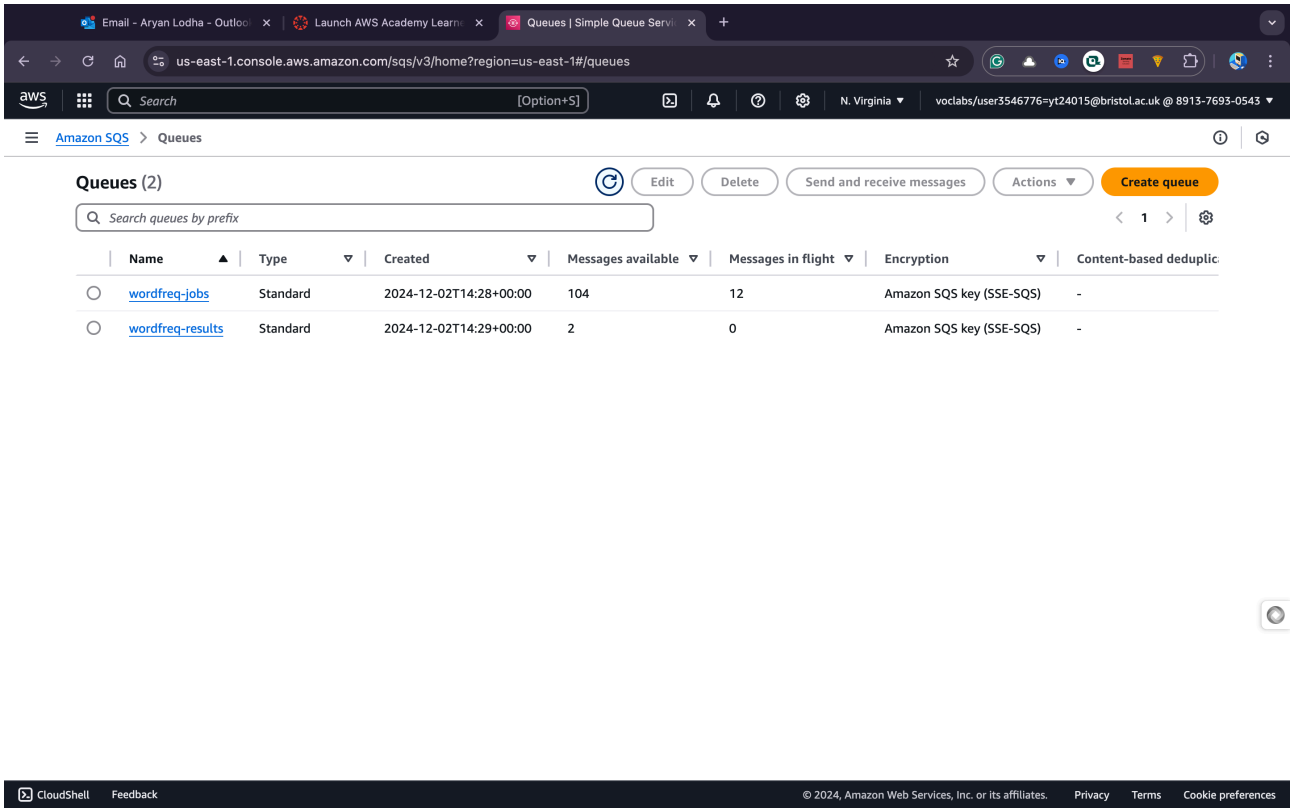


Fig7: Message Processing in Action

We will go to the SQS Console on AWS to track the number of messages in the queue (WordFreq Jobs queue). Initially, many of these will be all queued. Once the job processing by the EC2 instances begins, there would-be fewer messages in this queue.

EC2 Instances:

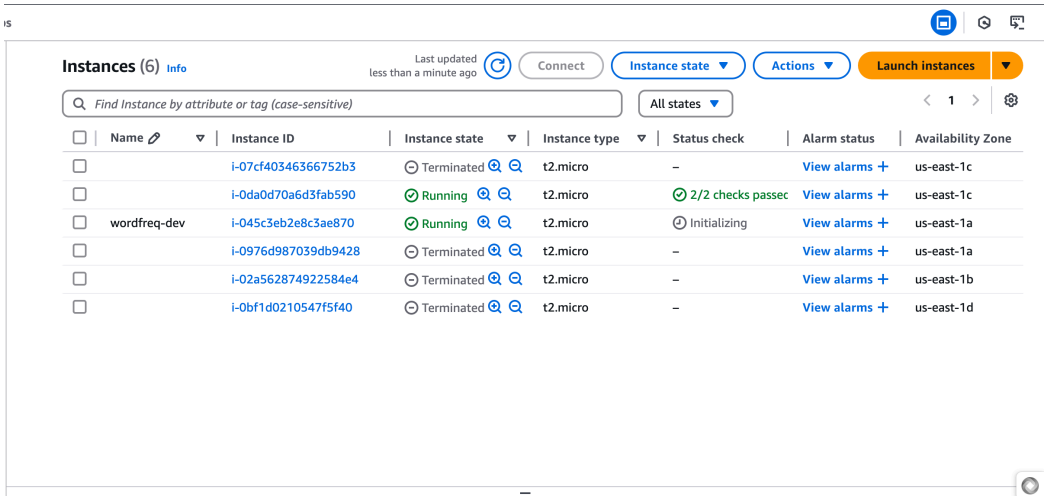


Fig 8 : Total no. of Instances Running

In the EC2 dashboard review the entire list of running and terminated instances. We will then check for numbers of active EC2 instances and then the way they are provisioned with the speed for new ones and terminated ones.

S3 Buckets:

Captured the files in processing buckets after the load test.

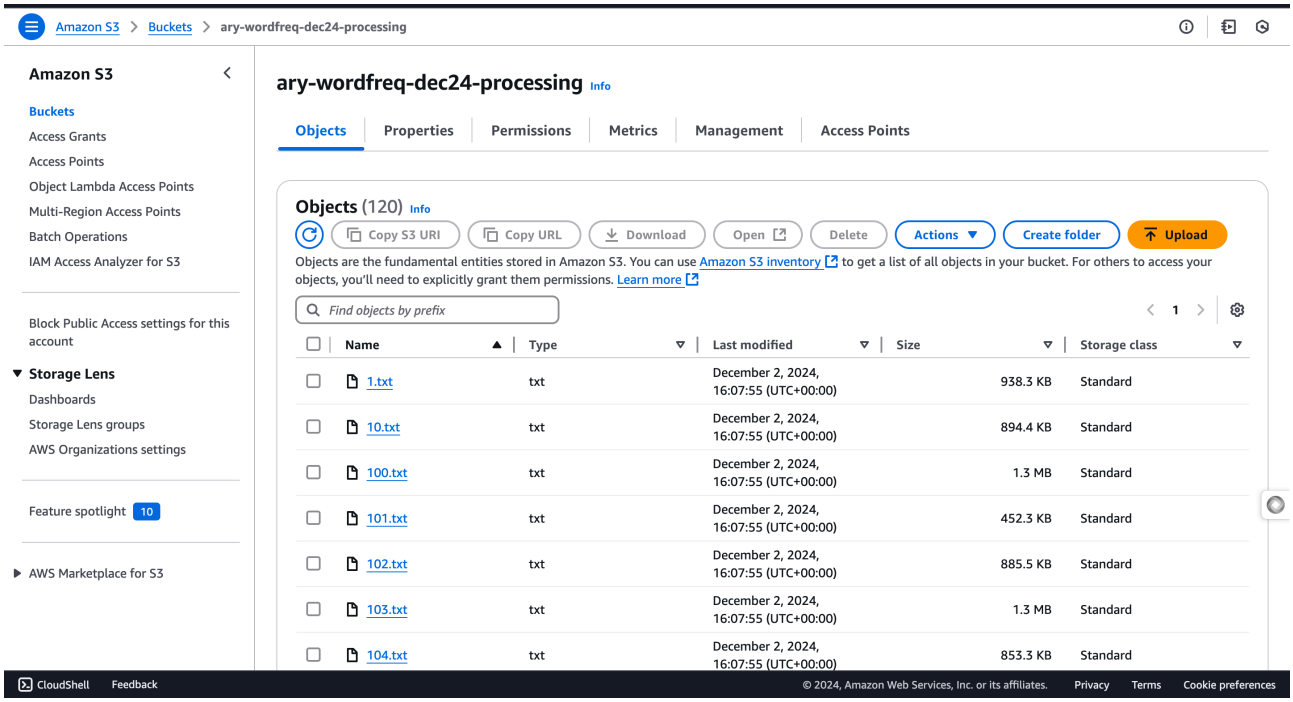


Fig 9: Processing Bucket where 120 files are copied

SQS Queue:

SQS WordFreq Jobs queue, showing the status of messages and how quickly the queue size reduces or got transferred

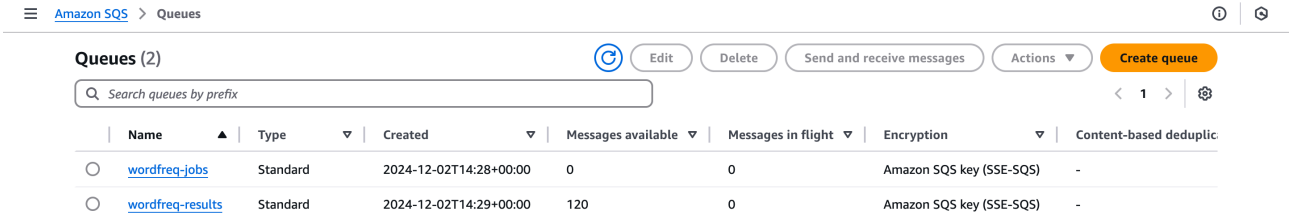


Fig 10: All the messages are moved to wordfreq-results

We have also used different EC2 instances with different configurations in order to find which performs the best

Parameters to Consider:

Instance Type: T2 Micro, T2 Small, T2 Medium

- T2 Micro: 1 vCPU, 1 GiB RAM
- T2 Small: 1 vCPU, 2 GiB RAM
- T2 Medium: 2 vCPUs, 4 GiB RAM

Cooldown Period: This is the duration that scaling operations must wait for so that one should not be resized quickly because it is likely to cause instability. Very short cool-down periods might call for frequent scales while overly long cool-down time may delay scaling when it is very essential.

Threshold: The metric value that triggers a scaling event. A higher threshold would, however, cause scaling events to happen later (more load must be present), while a lower threshold would trigger scaling events earlier (scales out sooner).

Total Time Taken: It shows the total time taken to process the files

1. T2 Micro

- **Cooldown Period**: 60 seconds
- **Threshold**: 20
- **Processing Time**: 12:53:59
- **Analysis**:
  - Because of a long cooldown (60 seconds), the system may scale more slowly in reaction to sudden traffic spikes.
  - The threshold of 20 means that the process will trigger scaling when the actual queue size reaches 20, which might be termed a mid-level trigger.
  - Long Processing Time: 12:53:59 indicates that this configuration is not as effective for dealing with the workload.

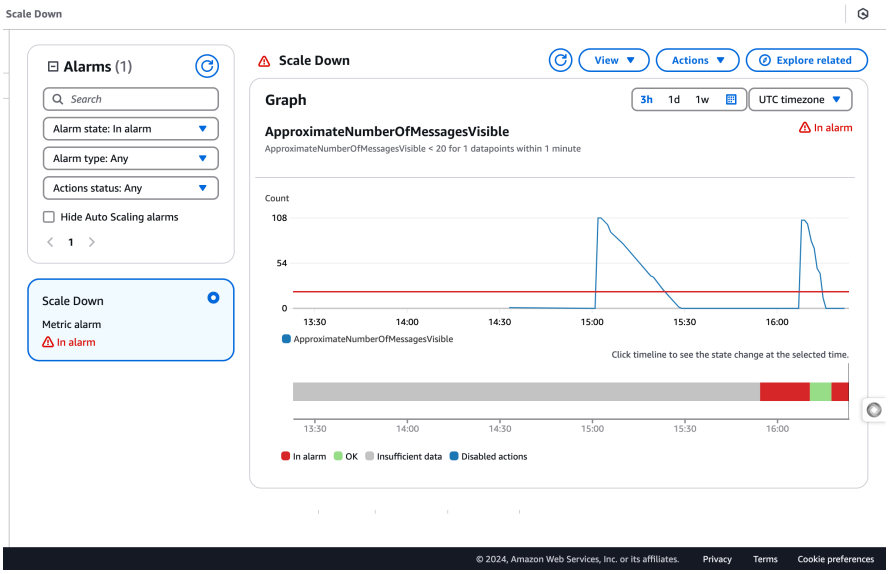


Fig 11: T2 Micro Scaling Graph

2. T2 Small

- **Cooldown Period**: 40 seconds for scale-up, 50 seconds for scale-down
- **Threshold**: 13
- **Processing Time**: 11:57:4
- **Analysis**:
  - Both the scale up and scale down cooldowns are quite short at 40 and 50 seconds respectively, thus allowing rapid scaling actions.
  - At a value of 13 the threshold is quite low, which means that there will be more frequent scaling activities in response to an increase in the workload.
  - Processing time better than T2 Micro, still not the fastest.

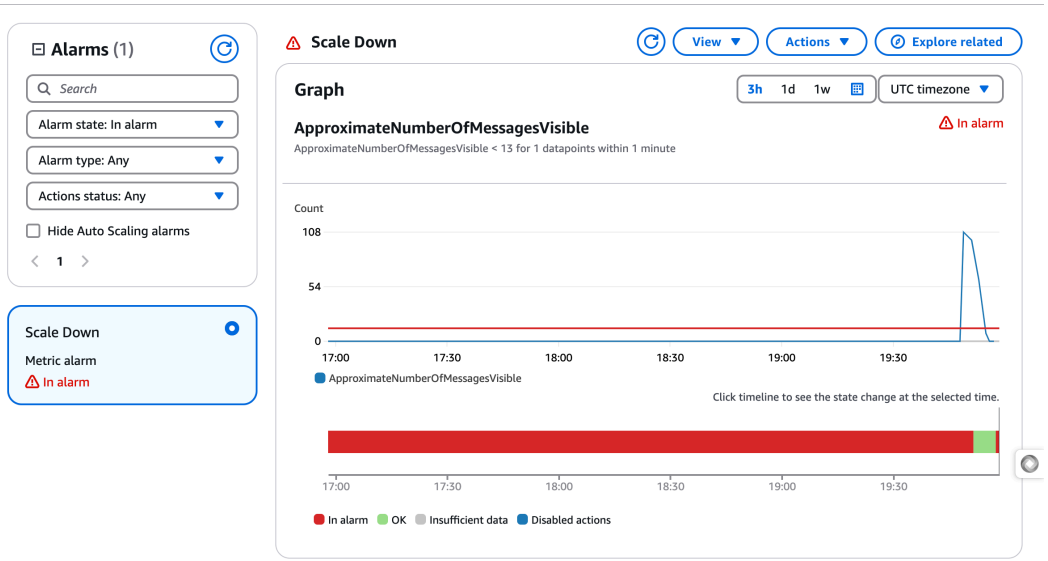


Fig 12: T2 Small Scaling Graph

T2 Medium

- **Cooldown Period**: 50 seconds
- **Threshold**: 25
- **Processing Time**: 08:07:40
- **Analysis**:
    - The cooldown is not too favourable for scaling decisions with a reasonable 50-second adjustment period.
    - The threshold of 25 is higher for scaling out in that it really waits for the maximum queue to build before it scales, which is likely to limit unnecessary scale operations.
    - Processing Time is the best among the configurations; that means this instance type is the most efficient in processing jobs.

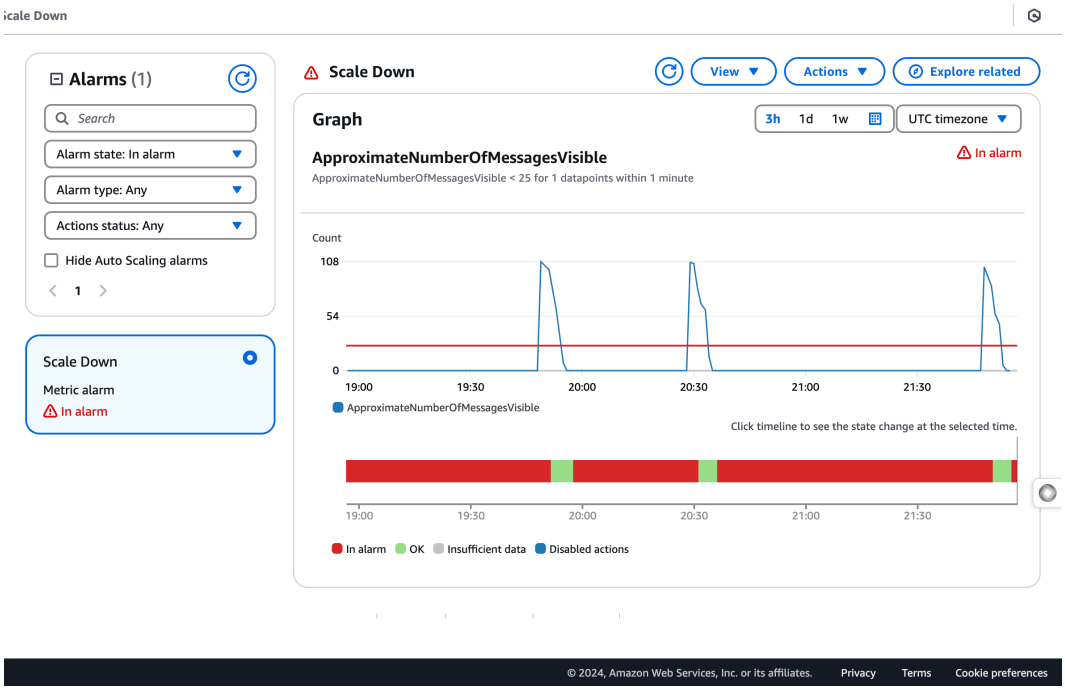


Fig 12: T2 Medium Scaling Graph

4. T2 Small

- **Cooldown Period**: 40 seconds
- **Threshold**: 30
- **Processing Time**: 12:31:04
- **Analysis**:
    - The cooldown duration seems to be ideal to allow responsiveness scaling at 40 seconds.

- Threshold of 30 is quite high, so scaling up hardly occurs, which probably saves unnecessary scaling actions.
- Processing Time is a little over that of T2 Medium, so this configuration is probably not as efficient.

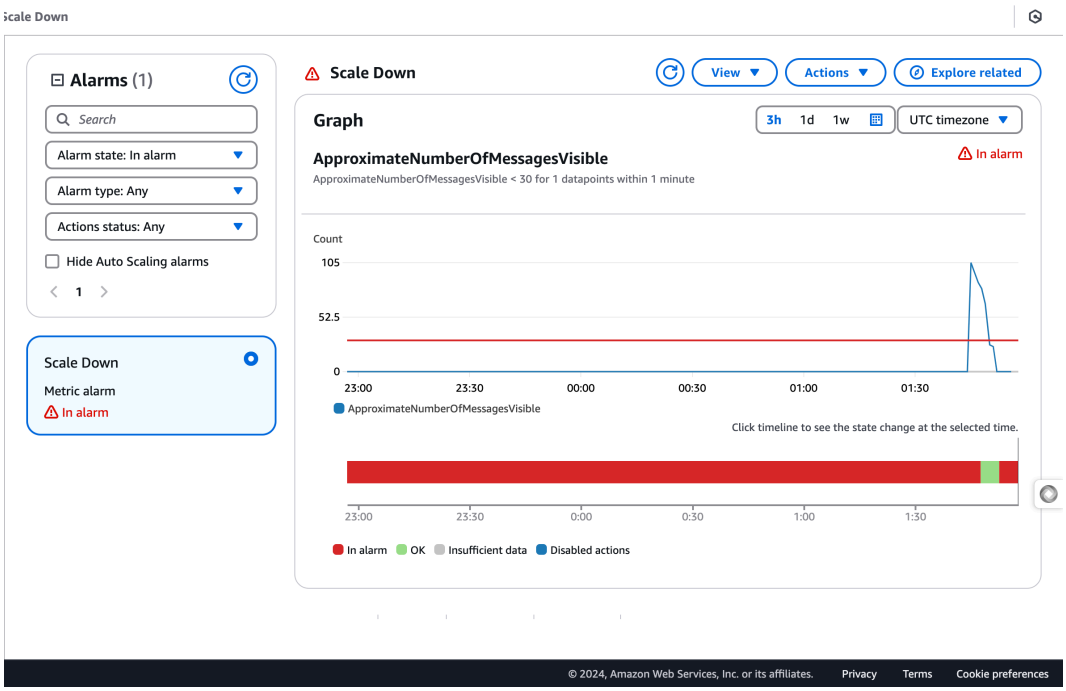


Fig 13: T2 Small Scaling Graph (Extra)

The best configuration is T2 Medium which as a cool-down period of 50 seconds, threshold 25 and the fast Processing time of around 8 minutes 7 seconds which processes jobs every efficiently

Normally, with a 50-second cooldown and a threshold equal to 25, T2 Medium scales up when it's needed and steers clear of scaling too frequently. As a result, the scaling processes will happen only when the demand for resources is genuine.

T2 Small and Micro, giving it a superiority boost, especially with multiple tasks happening at the same time. Provided with 2 virtual CPUs and 4 GiB of RAM, the T2 Medium instance performs concurrent file processing more efficiently than the T2 Micro or T2 Small, leading to completion of jobs faster, which equals an overall system performance improvement.

However, being T2 Small performs really well regarding low threshold, but the most perfect balance of top performance, efficient scale, and optimal resource usage for your WordFreq application is T2 Medium's.

Task D : Optimisation of WordFrq Architecture

1. Increasing resilience and Availability Against Component Failure

Our main aim is to ensure that if a component of an architecture fails like EC2, the application could still work with minimal downtime. This will ensure high availability

Amazon EC2 with Auto Scaling:-

- EC2: Wordfreq runs on this server. It is a virtual server. Provides customisable compute instances for running the WordFreq application.
- Auto Scaling Group : Adjust number of instances on basis of demand dynamically. Handle traffic spikes or reduced loads
- Elastic Load Balancer : Distribution of incoming traffic across all EC2 instances that are healthy

Amazon SQS :-

- Durable and availability, No message loss guarantee during failure
- Decoupling application components by queuing the tasks for asynchronous processing

Amazon CloudWatch:-

Monitor performance and health of components. Also calculate real time metrics, logs, activities and alarms

AWS Lambda:

Lambda, as per its design, is highly available and fault-tolerant. Each invocation of a function is isolated from the other, so any failure in the function invocation does not affect the remaining calls.

2. Long Term Backups of Valuable Data

When considering long term backups, the critical data should be backed up regularly and stored in a secure place which should also be cost effective

Amazon S3:-

Durable and scalable object storage device and it provide support for lifecycle policies for long term data management

S3 Glacier

- It is used for archiving older and infrequent accessed data. It configures vault locks for regular compliance

3. Cost-Effective and Efficient Application for Occasional Use

Since the application does not require immediate processing and is used occasionally, balancing of cost with performance is important which is only possible by using resources when necessary

Amazon EC2 Spot Instance:-

- Runs instances which are cost efficient for workloads that are able to handle interruptions

Provide same performance justl like on demand instance but at lower costs

- This is ideal for batch processing workloads like WordFreq, which can tolerate occasional interruptions.

AWS SQS with AWS Lambda:-

- SQS helps in queuing tasks and Lambda helps in processing those tasks in a serverless manner
- Reduces the costs by eliminating the need for always on computing resources

Amazon S3 Lifecycle Policies:-

It moves infrequently accessed data to lower cost storage classes such as S3 Glacier.

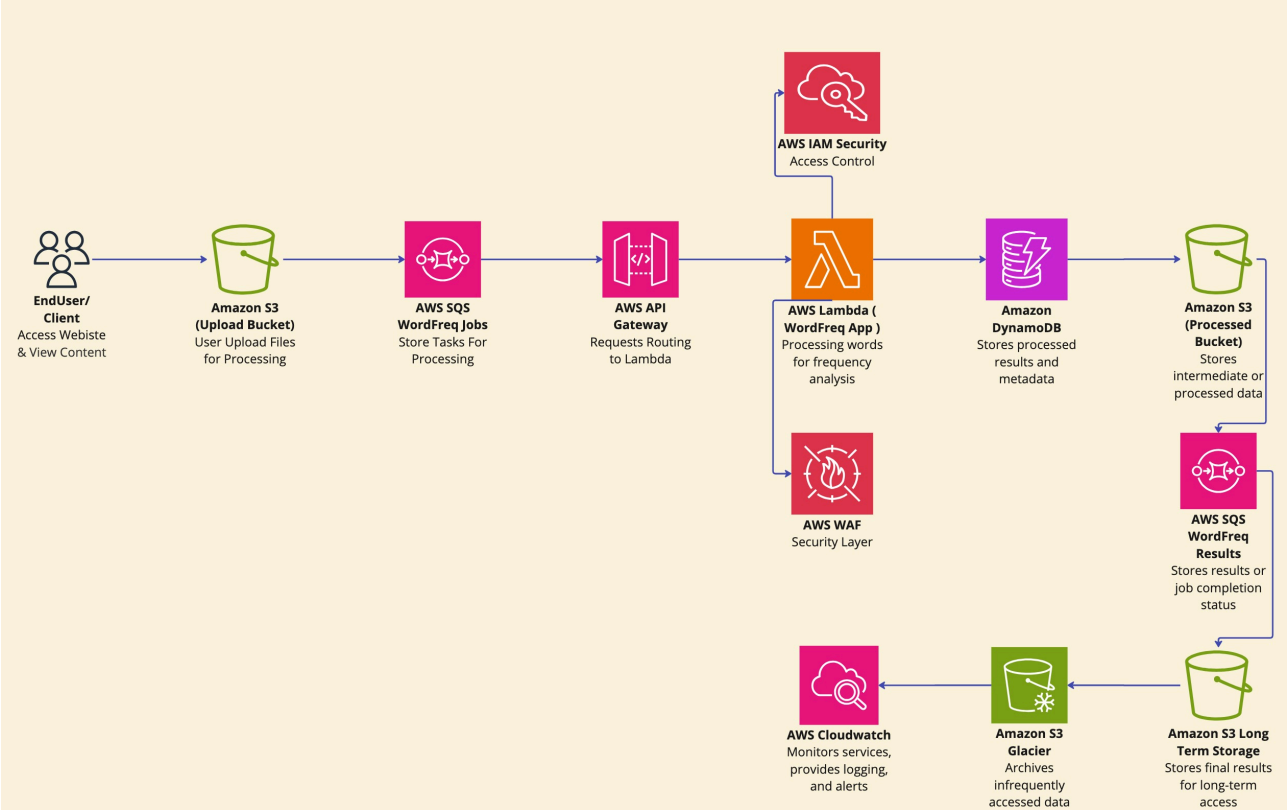
4. Prevent Unauthorised Access

AWS Identity and Access Management (IAM):-

Controls over the access to resources in centralised. It also gives role based permissions along with least privilege policies

Amazon CloudFront with Web Application Firewall (WAF):-

Low latency content distribution globally. It rrovide protection against common websites attacks and DDos attacks



Architecutre Diagram of Optimised Word Freq Application

Flow of Architecture :

- User Upload → S3 Upload Bucket: When a user uploads files into the S3 Upload Bucket, event notifications are sent to SQS WordFreq Jobs.
- SQS WordFreq Jobs → API Gateway: Messages from the SQS WordFreq Jobs are routed to API Gateway and are forwarded to Lambda for further processing.
- API Gateway → Lambda (WordFreq App): The word frequency processing results are stored in the DynamoDB and S3 Processed Bucket by Lambda.
- Lambda → DynamoDB: The processed information is stored in a database with results and metadata for quick retrieval.
- Lambda → S3 Processed Bucket: The intermediate or processed results are stored here in the S3 Processed Bucket.
- Lambda → SQS WordFreq Results: The result processing by Lambda is sent to SQS WordFreq Results for further processing.
- S3 Processed Bucket → S3 Long-Term Storage: The data that have been processed are now archived into the S3 Long-Term Storage transfer for the long term.
- S3 Long-Term Storage → S3 Glacier: For cost-effective archival, data that are accessed infrequently are moved to S3 Glacier.
- Monitoring & Logging → AWS CloudWatch: CloudWatch monitors all activities, collects logs, sends alerts for failure cases in operational health for smooth operations across services.

Task E : Further Improvements : Using Hadoop and Spark for WordFreq

All the tasks in WordFreq requires a scalable and robust solution which has the ability to handle large datasets and compute word frequencies. We will integrate Apache Hadoop and Spark with the AWS services which is a very good option for high performance data processing. The services of AWS like the Amazon Elastic Compute Cloud (EMR) helps and streamlines the deployment process of both Hadoop and Spark.

Apache Hadoop

- It is a very mature framework which is designed for processing massive datasets using MapReduce programming model.
- It is mainly used for large scale batch processing. It's distributed system of files provides fault tolerance and high throughput which makes it suitable for batch processing
- It integrates to AWS through Amazon EMR. EMR helps in simplifying the deployment and scaling of Hadoop clusters. In addition, services including S3 and IAM which helps increasing storage, accessibility and security.

Advantages :

1. **Distributed Storage and Processing :-**
   - Hadoop Distributed File System (HDFS) stores the Hadoop data and distribute it across multiple nodes which helps in processing of WordFreq more efficiently

2. **Fault Tolerance :-**
   - By copying data blocks across nodes, a guarantee is provided by Hadoop about data availability even if there is an event of hardware failure. This reliability makes it very robust for long running batch jobs

3. **Scalability :-**
   - The architecture of Hadoop allows for seamless scaling by adding more nodes to the cluster. This scalability ensures that Hadoop can handle increasing data volumes without any degradation

4. **Cost-Effectiveness :-**
   - Hadoop minimises infrastructure costs while maintaining good robust performance for computationally intensive tasks by using commodity hardware.

5. **Ecosystem Integration**:**-**
   - Hadoop is able to integrate well with other tools like Hive for SQL- like queries and also Pig for better data transformations. This flexible integration quality of Hadoop simplifies complex data workflows

In case of WordFreq, MapReduce helps in breaking down task into multiple stages which are mapping words to counts and then reducing the counts for aggregation. HDFS ensures a very high speed access to the distributed text data while the fault tolerance characteristic of Hadoop safeguards against job interruptions.

Apache Spark

Spark is built on Hadoop's strengths but it comes with crucial enhancements which makes it a more efficient and versatile alternative. Spark operates on memory-based data processing and it efficiently deals with both batch and real-time data processing. When it is coupled with AWS, Spark on EMR turns extraordinarily effective for doing iterative and high-performance tasks like WordFreq.

Advantages of Spark:

1. In-Memory Processing:-
   - The processing of intermediate data is performed in memory rather than disk (same as Hadoop). This accelerates iterative operations, making it profitable for WordFreq tasks in which the major requirement is of multiple transformations.

2. Rich APIs & Multi-Language Support:-

- Spark provides client friendly APIs in multiple languages like Python, R, Java etc. which in turn simplifies development process and testing. For example, the tool PySpark helps in data analysis and its transformation

3. Support for Real-Time and Batch Processing:-

- Spark can handle real time streaming with the help of Spark Streaming and other batch tasks which offers enhanced flexibility for different workloads.

4. Resilient Distributed Datasets (RDDs):-

- RDD in spark allows transformations to be recomputed automatically in case of any event failures and also provides fault tolerance which ensures reliability during data processing.

5. Ecosystem and Integration:-

- Sparks makes an integration with HDFS, S3 and with many other storage options. It supports a number of advance libraries useful for analytics like GraphX, Lib etc. which enhances its ability to work beyond tasks which just involve Extract, Transform and Load (ETL) tasks
- The AWS Glue is a ready-made tool for performing ETL operations, preparing and cataloging data for Spark processing.
- Spark Streaming is also capable of integrating with Amazon Kinesis to actually process real-time data streams. This is not necessary for WordFreq, yet it makes sense in the case of live text streams.

Due to its in-memory execution, Spark can be able to process the text data from Gutenberg corpus much faster than Hadoop. For instance, one could efficiently map words to frequencies and aggregate the results using DataFrames or RDDs. In addition, these transformation-and-aggregation capabilities allow one to receive real-time insights into word usage trends.

Username And Credentials for AWS Academy Account

Username : yt24015@bristol.ac.uk
Password : Aryan@120901