Kristian (Chris) Litsis **User_Id:** wg20498

Coursework Final Report - EMATM0051 LSDE

Part 1: "Data is not safe on the Public Cloud"

Saving photos, data, documents, videos, and reports on a computer's storage has become a little old-fashioned. Today, people are more frequently saving their files in the cloud, storage space not located in their devices but on servers hosted by third-party providers. The positives of storing data and files in the cloud are many: You won't need your computer. If your device crashes, then you'll still have access to your files. But is data safe in public cloud?

As we learned in the course, although data is saved in a public cloud it remains secure. AWS provides different tools to provide security in their clouds because security is the highest priority at Amazon Web Services. AWS shared responsibility model provides physical security of the facilities where their services operate. They are responsible for protecting the infrastructure that runs in the cloud from hardware, software, networking. On the other hand, the customer is responsible for the encryption in the upper layer.

Firstly AWS gives the user the responsibility to configure a virtual private cloud (VPC). Amazon VPC gives your section isolation on the cloud where you can launch AWS resources in a virtual network that you define where the user has complete control over the virtual networking environment. Users can control how the resources are exposed to the internet by allowing only trusted traffic through Network access control lists (ACLs) that act as a firewall at the subnet level. Rules automatically apply to all instances in the subnets, therefore, providing an additional layer of defense if the security group rules are too permissive.

Secondly, Amazon provides its customers AWS Identity and Access Management (IAM) to control access to compute, storage, database, and application services in the Cloud. IAM can be used to handle authentication, enforce and specify authorization policies so that you can choose which users can access which services. It is a tool that centrally manages access to launching, configuring, managing, and terminating resources in your AWS account. It also has the ability to specify exactly which API calls the user is authorized to make to each service.

Additionally, the customer can configure an internet gateway as part of the VPC so that the webserver can be reached by using the AWS Management Console or a Command Line Interface like the AWS CLI. When the customer uses the interface to access the instance, the connection requires the use of a secure shell ssh key that is created by the user when they created the instances in order to protect the data from others requiring access to that resource. An AWS Multi-Factor Authentication (MFA) is used to add an extra layer of protection on top of your user name and password. With MFA enabled, when a user signs in to an AWS Management Console, they will be prompted for their username and password, as well as for an authentication code from their AWS MFA device. These multiple factors together provide increased security for your AWS account settings and resources.

Next customers can create Amazon EC2 security groups which can be used to help secure instances within a VPC. A security group acts as a virtual firewall and enables you to specify which inbound or outbound network traffic to allow. Traffic that is not explicitly allowed is automatically denied. The platform allows the use of multiple security groups because they act at the instance level, not the subnet level, so each instance can be assigned to a different group providing more protection to data.

Last but not least, Amazon provides a monitoring tool called AWS CloudTrail that logs all the API requests to the user account. It enables governance, compliance, operational auditing, and risk auditing of your AWS account and is enabled on account creation by default of all AWS accounts and keeps a record of the last 90 days of account management event activity which can be gathered by the user. It can be configured to respond to a threatening activity by defining workflows that execute when events are detected. For example, a workflow can be created to add a specific policy to an Amazon S3 bucket when CloudTrail logs an API call that makes that bucket public. Given all the explanations about Amazon services, I would argue that data is relatively secure. We, as architecture engineers should focus on using the best features that would give us more security. After all, we can save all the critical files to the private subnet and block all the access outside the company. So everything is up to us and how we implement the design.

Below I represent three scenarios where data should not be stored in the public cloud.

Reduced Coverage
The second scenario can be when your company region is not covered by the cloud service so it can affect your performance, especially if the data is stored in locations far from the customer base. So the latency will not be good enough and the company will lose some customers.

Critical Data handling
Critical data are those that some would avoid storing in public could. This is because we cannot know the physical location of the storage, and local laws and jurisdictions that perform in that specific location.

Remote location
Cloud requires a high-speed Internet connection as a basic requirement to perform. So if the company is planning to have offices in a remote location or where Internet service is not so stable then moving to the cloud would not be very effective.

Bibliography:

1. AWS Documentation. (n.d.). Retrieved from https://docs.aws.amazon.com/index.html

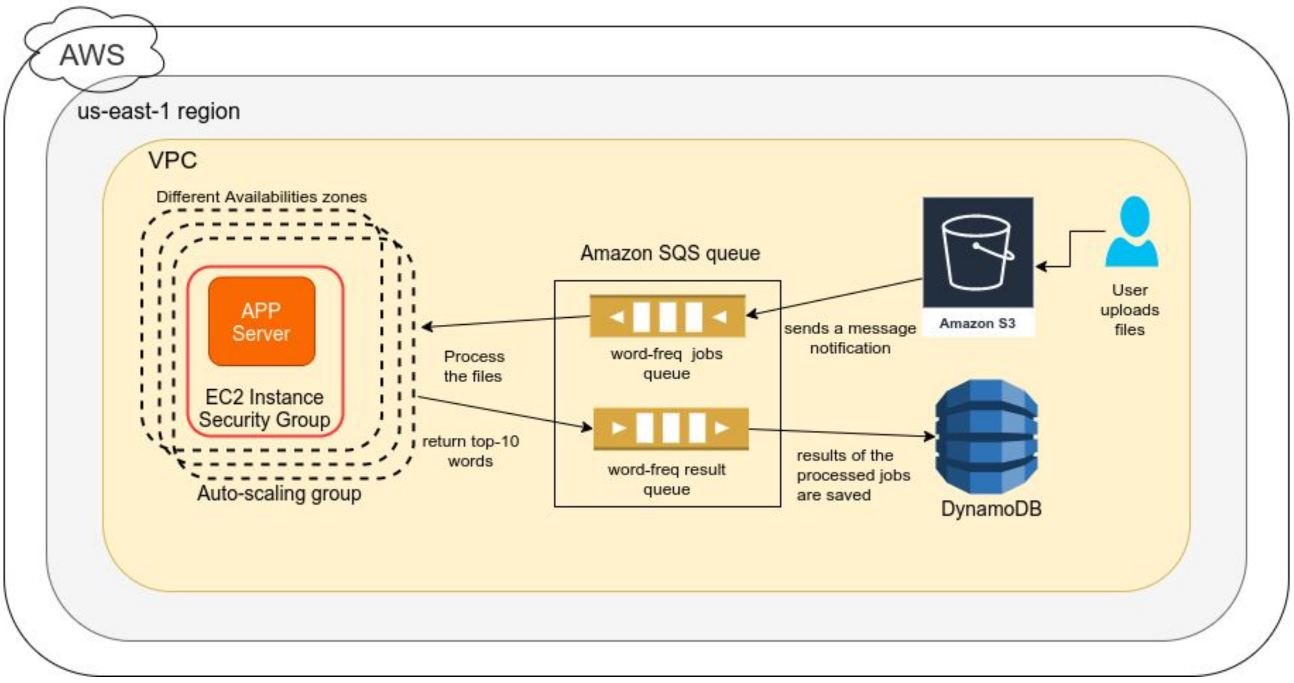
2. https://awsacademy.instructure.com/courses/1366/modules

Part 2: Scaling the WordFreq Application

• Brief summary of how the application works

The following report influences design and provides the implementation of application deployment in Amazon Cloud using the AWS Educate account. The application that we will run in the cloud will be Wordfreq. WordFreq is a complete, working application, built using the Go programming language. The application takes advantage of Amazon Elastic Compute Cloud, Amazon Simple Storage Service, Amazon Simple Queue Service, and Amazon DynamoDB to collect and report the top 10 most common words of a text file all in the AWS Cloud.

The application uses a number of different AWS services and below we will explain every service and why it is used:



First, the application uses an AWS S3 as the main storage for uploading files. Amazon S3 is an object storage service that offers high scalability, data availability, security, and performance. It is designed for 99.999999999% , known as 11 9's of durability, and stores data for millions of applications worldwide. In our case many text files are uploaded to the S3 bucket named **af-wordfreq-dec01**. The bucket has upload notifications enabled, such that when a file is uploaded, a message notification is automatically added to the wordfreq SQS queue ready for a worker to take the job and start the computing.

The application will be using Amazon SQS services which is a fully managed message queuing service that enables you to separate and scale microservices, distributed systems, and server-less applications. Using SQS, we can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available. There are two queues used for the application. One named **wordfreq-jobs** is used for holding notification messages of newly uploaded text files from the S3 bucket. These messages are known as 'jobs', or tasks to be performed by the application, and specify the location of the text file on the S3 bucket. A second queue named **wordfreq-results** is used to hold messages containing the 'top-10' results of the processed jobs.

To store the result of the processed jobs we created a NoSQL database table named **wordfreq** by using DynamoDB. DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. For key we used the text file name and the value will be the top-10 words alongside their frequency in the document.

Last, we will be using Amazon EC2, which is a web service that provides secure, resizable compute capacity in the cloud. The application runs on an Ubuntu Linux EC2 instance, where we will configure all the specific sizes required for the best performance. Given the instruction in our coursework EC2 will set up and identify the S3, SQS, and DynamoDB resources for the application.

• The design process to architect the scaling behaviours (task B)

To overcome many issues related to the performance of our application we should take advantage of all the different tools and services that the cloud provided us. The two great benefits of using the cloud are scalability and cost-effectiveness. Cloud scalability is very easy to customize to the exact need that the application. It also adds flexibility and speed because of traffic change and growth through time so it needs frequent updates of resources. Thanks to the cloud, we can monitor the demand and minimize the costs and the waste of resources. And last but not least, making the application scalable can reduce disaster recovery and failures.

There are basically two types of scalability that we used, the vertical and horizontal scaling both provided by Amazon cloud services. With vertical scaling, we can add and subtract power to an existing server, so in other words to increase the capacity and the type of the instance. On the other hand, we use horizontal scaling for adding more resources like instances to spread out the workload across the system.

We combined those two policies to increase performance and storage by creating an auto-scaling group containing a collection of Amazon EC2. These instances are treated as a logical grouping of servers for the purpose of automatic scaling and management. By using the auto-scaling group we can perform health check replacements when the instances are failing and also create some scaling policies. There are two scaling policies that belong to the step scaling category. When step adjustments are applied, they increase or decrease the current capacity of the Auto Scaling group. These adjustments vary based on the size of the alarm that will trigger. I used this policy because it is also recommended by the AWS community as a better choice. With step scaling, the policy can continue to respond to additional alarms, even while a scaling activity or health check replacement is in progress. Unlike the other policy, this policy doesn't require cool-down time to perform health checks in the instances, therefore, all alarms that are breached are evaluated by Amazon EC2 Auto Scaling as it receives the alarm messages.

The first policy that I created is the **Increase EC2** policy. When configuring step scaling or simple scaling, an alarm must be created in CloudWatch, and then define two policies, one for scaling out and the other for scaling in. This allows instances to launch and terminate in response to the CloudWatch alarms. The **Increase EC2** policy is connected with the **ADD** alarm . This alarm is a metric alarm. I decided to monitor the Amazon SQS because we use our queue to count the number of jobs that require our application to run and it would not be a good choice to go with the CPU or other metrics because target tracking scaling policy based on a custom Amazon SQS queue metric is a dynamic scaling and can adjust to the demand curve of the application more effectively. The metric that I decided to monitor is the NumberOfMessagesSent which is the number of messages added to a queue. My policy measures the sample count of the messages that go to **wordfreq-jobs** queue and use a threshold greater than 10 messages for 1 data point within 5 minutes. If the alarm is triggered then the action used is the Increase EC2 policy of the auto-scaling group which adds two additional EC2 instances for the group. I chose the interval to be within 5 minutes because I wanted the system to give it a little time to scale in order to see the demand of the messages.

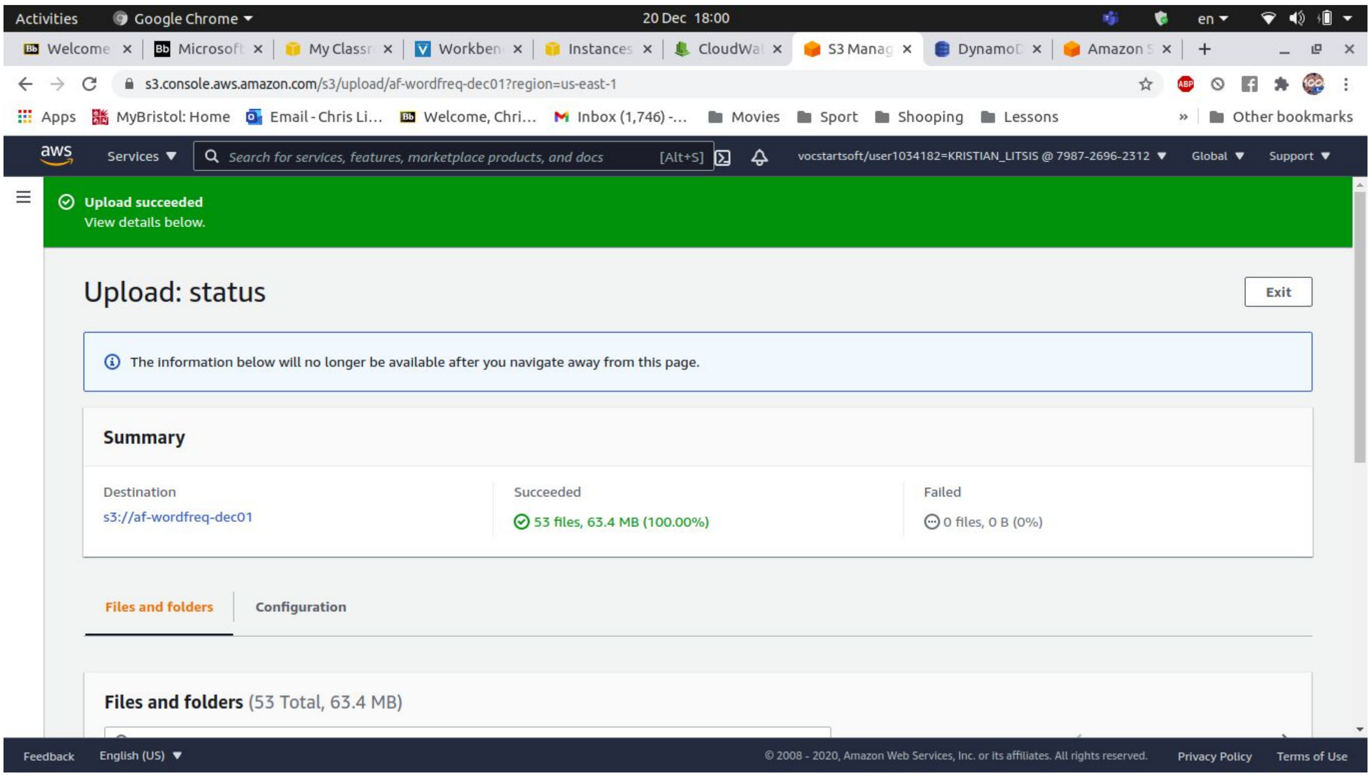
The second policy that I created is the **Remove Scaling Instances** policy. This is also a step scaling policy and it is also required to create an alarm in CloudWatch that is connected with the auto Scaling group. The alarm that is connected is the **Remove** alarm. This alarm is also a metric alarm that monitors the Amazon SQS to count the number of messages that are added to the **wordfreq-jobs** queue. The policy measures the sample count of the messages using a threshold smaller than 8 messages for 1 data point within 5 minutes. If the alarm is triggered then the action used is the Remove Scaling Instances policy of the auto-scaling group which removes two capacity EC2 units of that group. I chose the interval to be within 5 minutes because I wanted the system to give it a little time to scale in order to see the demand of the messages.

• An overview of the testing and results (task C)

After scaling our system by setting up the auto-scaling infrastructure as we described above, it was time to test in order to measure the performance. We decided to upload in our S3 bucket around 50 text documents. Files are downloaded by public databases and are in txt-format only with no longer than 5MB each.

Screenshots of the uploaded files.

To perform the load test we uploaded a total of 53 text-documents with a total memory size of 63.4MB as is shown in the screenshot below. After waiting for a little time, all the files were uploaded with success to the S3 bucket and the following Uploaded Succeeded status was shown on the screen with the green notification.



Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

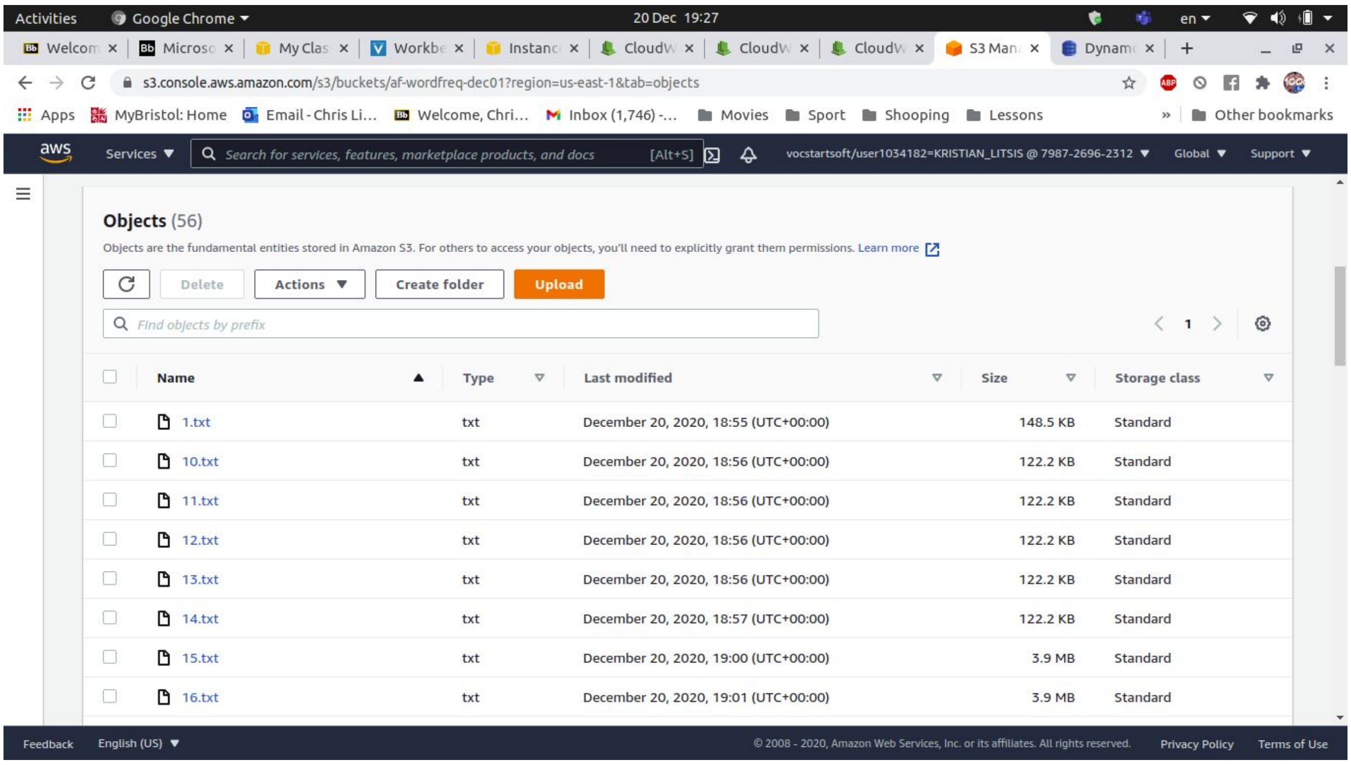
Summary

Destination	Succeeded	Failed
s3://af-wordfreq-dec01	53 files, 63.4 MB (100.00%)	0 files, 0 B (0%)

Files and folders (53 Total, 63.4 MB)

Activities
Google Chrome
20 Dec 18:00
s3.console.aws.amazon.com/s3/upload/af-wordfreq-dec01?region=us-east-1
vocstartsoft/user1034182=KRISTIAN_LITSIS @ 7987-2696-2312
Exit
Files and folders
Configuration
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

After the upload we can enter in our S3 bucket **af-wordfreq-dec01** , go to the section of objects and see all the files that are uploaded to the bucket.



s3.console.aws.amazon.com/s3/buckets/af-wordfreq-dec01?region=us-east-1&tab=objects

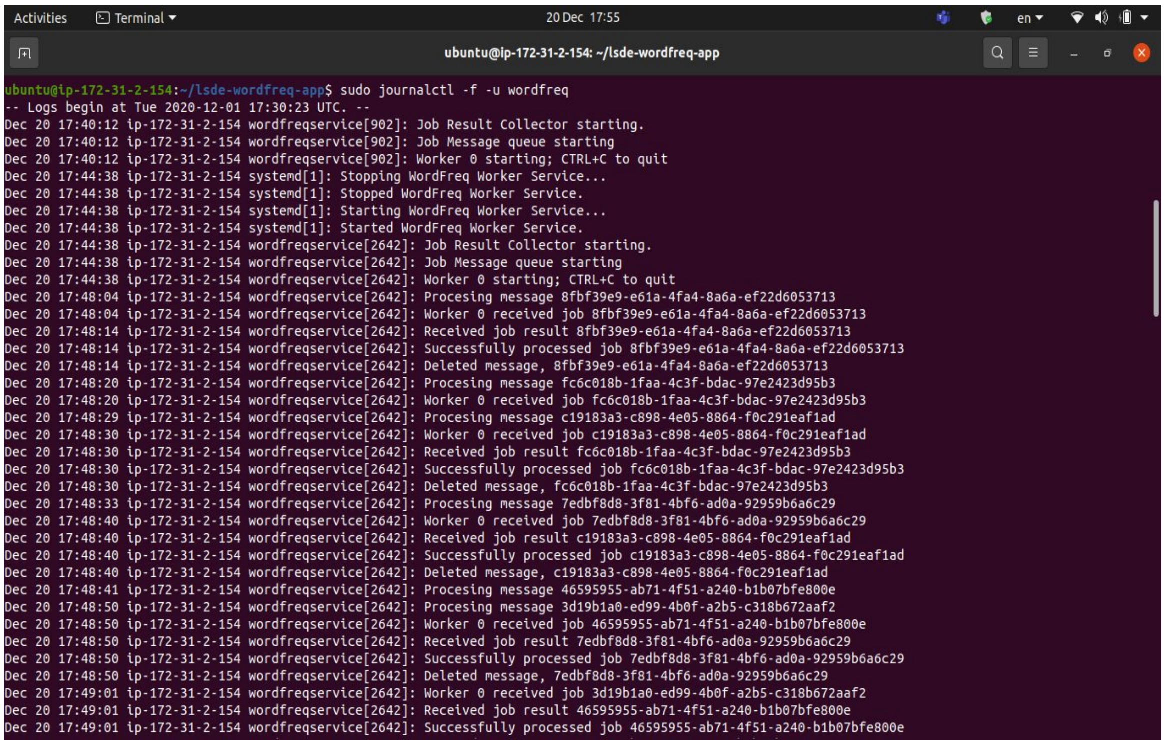
Objects (56)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

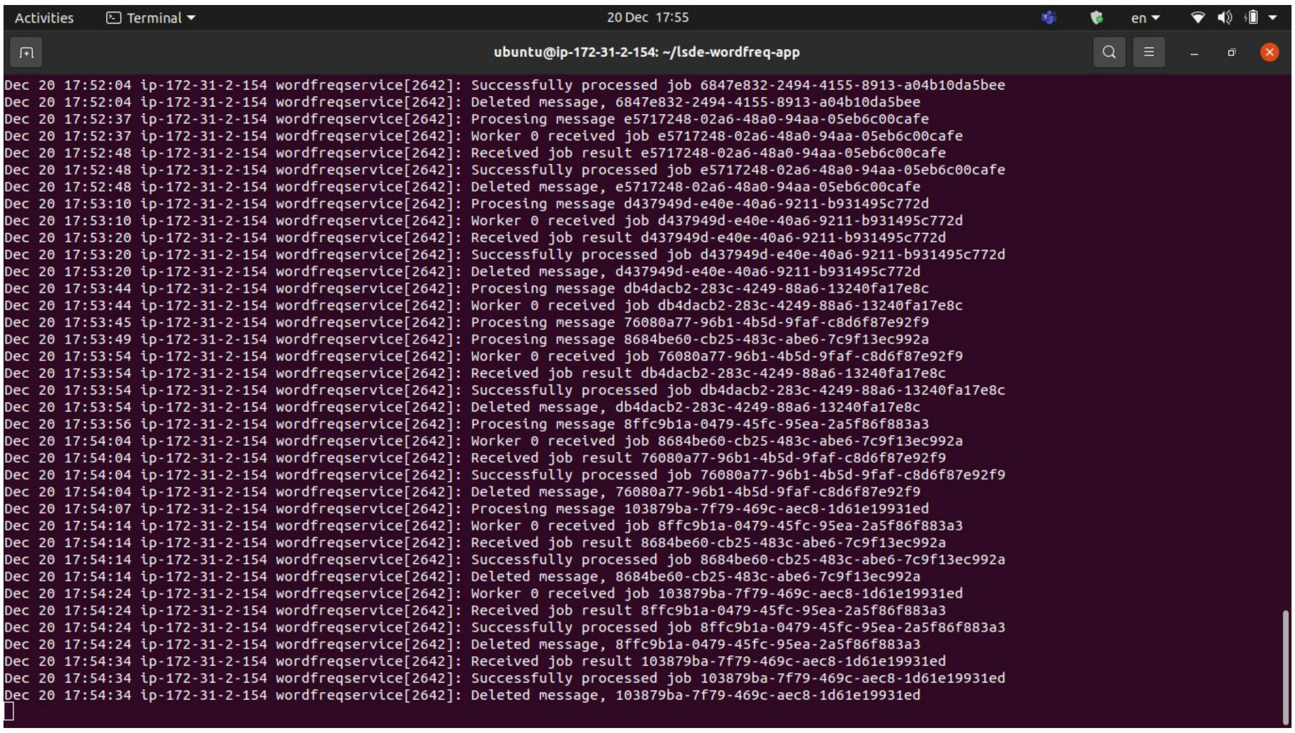
Name	Type	Last modified	Size	Storage class
1.txt	txt	December 20, 2020, 18:55 (UTC+00:00)	148.5 KB	Standard
10.txt	txt	December 20, 2020, 18:56 (UTC+00:00)	122.2 KB	Standard
11.txt	txt	December 20, 2020, 18:56 (UTC+00:00)	122.2 KB	Standard
12.txt	txt	December 20, 2020, 18:56 (UTC+00:00)	122.2 KB	Standard
13.txt	txt	December 20, 2020, 18:56 (UTC+00:00)	122.2 KB	Standard
14.txt	txt	December 20, 2020, 18:57 (UTC+00:00)	122.2 KB	Standard
15.txt	txt	December 20, 2020, 19:00 (UTC+00:00)	3.9 MB	Standard
16.txt	txt	December 20, 2020, 19:01 (UTC+00:00)	3.9 MB	Standard

SQS queue page showing message status

Below we present the working queue that receives different jobs for processing each text document. I connected in the EC2 instances by SSH connection and used the specific command sudo journalctl -f -u wordfreq which views the output logs from the running wordfreq-worker service. Each message has a unique id. There are several stages that the messages pass, first it processes the message, then a worker receives the job, after that it receives the job result, return successfully processed and then delete the message from the queue.



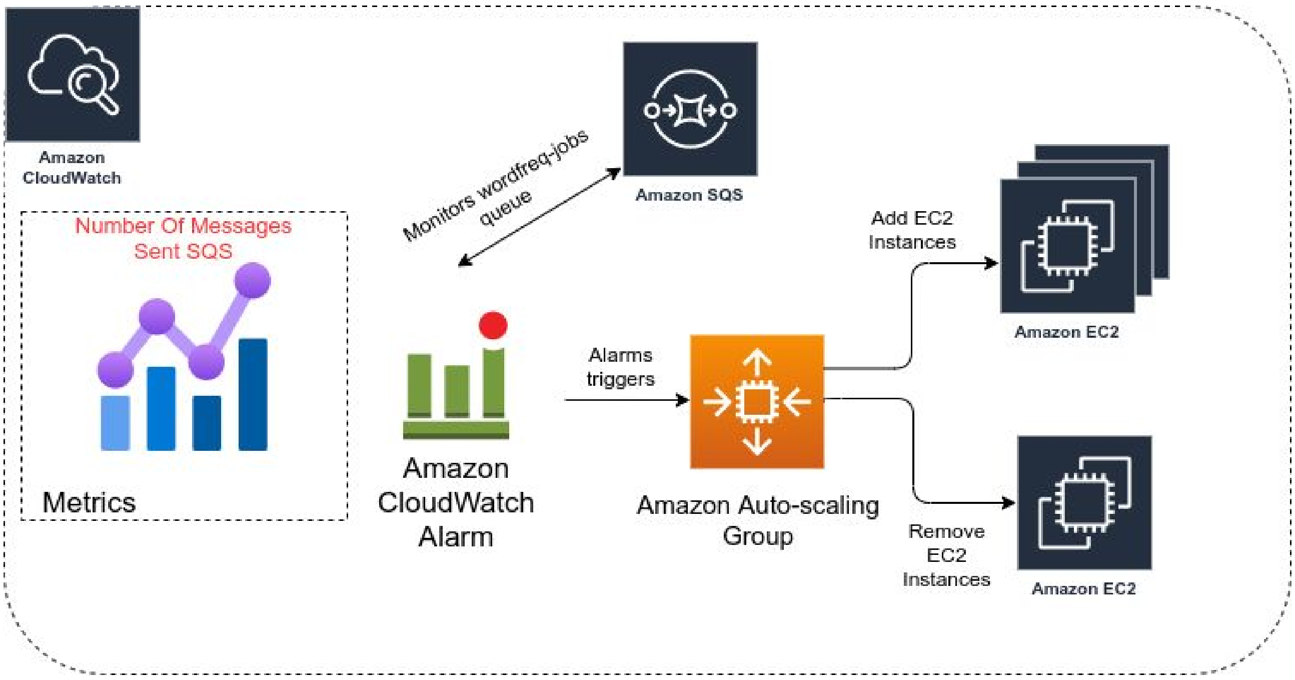
```
ubuntu@ip-172-31-2-154:~/lsde-wordfreq-app$ sudo journalctl -f -u wordfreq
-- Logs begin at Tue 2020-12-01 17:30:23 UTC. --
Dec 20 17:40:12 ip-172-31-2-154 wordfreqservice[902]: Job Result Collector starting.
Dec 20 17:40:12 ip-172-31-2-154 wordfreqservice[902]: Job Message queue starting
Dec 20 17:40:12 ip-172-31-2-154 wordfreqservice[902]: Worker 0 starting; CTRL+C to quit
Dec 20 17:44:38 ip-172-31-2-154 systemd[1]: Stopping WordFreq Worker Service...
Dec 20 17:44:38 ip-172-31-2-154 systemd[1]: Stopped WordFreq Worker Service.
Dec 20 17:44:38 ip-172-31-2-154 systemd[1]: Starting WordFreq Worker Service...
Dec 20 17:44:38 ip-172-31-2-154 systemd[1]: Started WordFreq Worker Service.
Dec 20 17:44:38 ip-172-31-2-154 wordfreqservice[2642]: Job Result Collector starting.
Dec 20 17:44:38 ip-172-31-2-154 wordfreqservice[2642]: Job Message queue starting
Dec 20 17:44:38 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 starting; CTRL+C to quit
Dec 20 17:48:04 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 8fbf39e9-e61a-4fa4-8a6a-ef22d6053713
Dec 20 17:48:04 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 8fbf39e9-e61a-4fa4-8a6a-ef22d6053713
Dec 20 17:48:14 ip-172-31-2-154 wordfreqservice[2642]: Received job result 8fbf39e9-e61a-4fa4-8a6a-ef22d6053713
Dec 20 17:48:14 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 8fbf39e9-e61a-4fa4-8a6a-ef22d6053713
Dec 20 17:48:14 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, 8fbf39e9-e61a-4fa4-8a6a-ef22d6053713
Dec 20 17:48:20 ip-172-31-2-154 wordfreqservice[2642]: Procesing message fc6c018b-1faa-4c3f-bdac-97e2423d95b3
Dec 20 17:48:20 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job fc6c018b-1faa-4c3f-bdac-97e2423d95b3
Dec 20 17:48:29 ip-172-31-2-154 wordfreqservice[2642]: Procesing message c19183a3-c898-4e05-8864-f0c291eaf1ad
Dec 20 17:48:30 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job c19183a3-c898-4e05-8864-f0c291eaf1ad
Dec 20 17:48:30 ip-172-31-2-154 wordfreqservice[2642]: Received job result fc6c018b-1faa-4c3f-bdac-97e2423d95b3
Dec 20 17:48:30 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job fc6c018b-1faa-4c3f-bdac-97e2423d95b3
Dec 20 17:48:30 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, fc6c018b-1faa-4c3f-bdac-97e2423d95b3
Dec 20 17:48:33 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 7edbf8d8-3f81-4bf6-ad0a-92959b6a6c29
Dec 20 17:48:40 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 7edbf8d8-3f81-4bf6-ad0a-92959b6a6c29
Dec 20 17:48:40 ip-172-31-2-154 wordfreqservice[2642]: Received job result c19183a3-c898-4e05-8864-f0c291eaf1ad
Dec 20 17:48:40 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job c19183a3-c898-4e05-8864-f0c291eaf1ad
Dec 20 17:48:40 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, c19183a3-c898-4e05-8864-f0c291eaf1ad
Dec 20 17:48:41 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 46595955-ab71-4f51-a240-b1b07bfe800e
Dec 20 17:48:50 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 3d19b1a0-ed99-4b0f-a2b5-c318b672aaf2
Dec 20 17:48:50 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 46595955-ab71-4f51-a240-b1b07bfe800e
Dec 20 17:48:50 ip-172-31-2-154 wordfreqservice[2642]: Received job result 7edbf8d8-3f81-4bf6-ad0a-92959b6a6c29
Dec 20 17:48:50 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 7edbf8d8-3f81-4bf6-ad0a-92959b6a6c29
Dec 20 17:48:50 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, 7edbf8d8-3f81-4bf6-ad0a-92959b6a6c29
Dec 20 17:49:01 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 3d19b1a0-ed99-4b0f-a2b5-c318b672aaf2
Dec 20 17:49:01 ip-172-31-2-154 wordfreqservice[2642]: Received job result 46595955-ab71-4f51-a240-b1b07bfe800e
Dec 20 17:49:01 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 46595955-ab71-4f51-a240-b1b07bfe800e
```



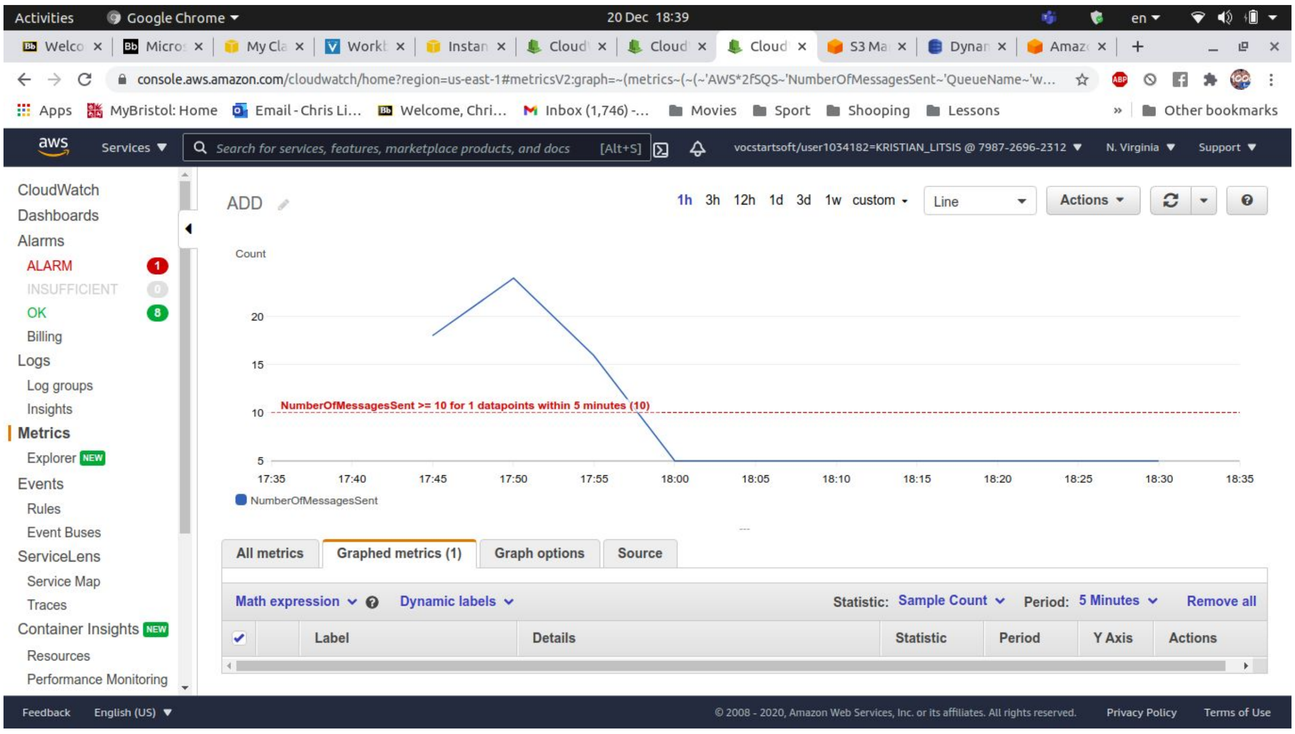
```
Dec 20 17:52:04 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 6847e832-2494-4155-8913-a04b10da5bee
Dec 20 17:52:04 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, 6847e832-2494-4155-8913-a04b10da5bee
Dec 20 17:52:37 ip-172-31-2-154 wordfreqservice[2642]: Procesing message e5717248-02a6-48a0-94aa-05eb6c00cafe
Dec 20 17:52:37 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job e5717248-02a6-48a0-94aa-05eb6c00cafe
Dec 20 17:52:48 ip-172-31-2-154 wordfreqservice[2642]: Received job result e5717248-02a6-48a0-94aa-05eb6c00cafe
Dec 20 17:52:48 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job e5717248-02a6-48a0-94aa-05eb6c00cafe
Dec 20 17:52:48 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, e5717248-02a6-48a0-94aa-05eb6c00cafe
Dec 20 17:53:10 ip-172-31-2-154 wordfreqservice[2642]: Procesing message d437949d-e40e-40a6-9211-b931495c772d
Dec 20 17:53:10 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job d437949d-e40e-40a6-9211-b931495c772d
Dec 20 17:53:20 ip-172-31-2-154 wordfreqservice[2642]: Received job result d437949d-e40e-40a6-9211-b931495c772d
Dec 20 17:53:20 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job d437949d-e40e-40a6-9211-b931495c772d
Dec 20 17:53:20 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, d437949d-e40e-40a6-9211-b931495c772d
Dec 20 17:53:44 ip-172-31-2-154 wordfreqservice[2642]: Procesing message db4dacb2-283c-4249-88a6-13240fa17e8c
Dec 20 17:53:44 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job db4dacb2-283c-4249-88a6-13240fa17e8c
Dec 20 17:53:45 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 76080a77-96b1-4b5d-9faf-c8d6f87e92f9
Dec 20 17:53:49 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 8684be60-cb25-483c-abe6-7c9f13ec992a
Dec 20 17:53:54 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 76080a77-96b1-4b5d-9faf-c8d6f87e92f9
Dec 20 17:53:54 ip-172-31-2-154 wordfreqservice[2642]: Received job result db4dacb2-283c-4249-88a6-13240fa17e8c
Dec 20 17:53:54 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job db4dacb2-283c-4249-88a6-13240fa17e8c
Dec 20 17:53:54 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, db4dacb2-283c-4249-88a6-13240fa17e8c
Dec 20 17:53:56 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 8ffc9b1a-0479-45fc-95ea-2a5f86f883a3
Dec 20 17:54:04 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 8684be60-cb25-483c-abe6-7c9f13ec992a
Dec 20 17:54:04 ip-172-31-2-154 wordfreqservice[2642]: Received job result 76080a77-96b1-4b5d-9faf-c8d6f87e92f9
Dec 20 17:54:04 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 76080a77-96b1-4b5d-9faf-c8d6f87e92f9
Dec 20 17:54:04 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, 76080a77-96b1-4b5d-9faf-c8d6f87e92f9
Dec 20 17:54:07 ip-172-31-2-154 wordfreqservice[2642]: Procesing message 103879ba-7f79-469c-aec8-1d61e19931ed
Dec 20 17:54:14 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 8ffc9b1a-0479-45fc-95ea-2a5f86f883a3
Dec 20 17:54:14 ip-172-31-2-154 wordfreqservice[2642]: Received job result 8684be60-cb25-483c-abe6-7c9f13ec992a
Dec 20 17:54:14 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 8684be60-cb25-483c-abe6-7c9f13ec992a
Dec 20 17:54:14 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, 8684be60-cb25-483c-abe6-7c9f13ec992a
Dec 20 17:54:24 ip-172-31-2-154 wordfreqservice[2642]: Worker 0 received job 103879ba-7f79-469c-aec8-1d61e19931ed
Dec 20 17:54:24 ip-172-31-2-154 wordfreqservice[2642]: Received job result 8ffc9b1a-0479-45fc-95ea-2a5f86f883a3
Dec 20 17:54:24 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 8ffc9b1a-0479-45fc-95ea-2a5f86f883a3
Dec 20 17:54:24 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, 8ffc9b1a-0479-45fc-95ea-2a5f86f883a3
Dec 20 17:54:34 ip-172-31-2-154 wordfreqservice[2642]: Received job result 103879ba-7f79-469c-aec8-1d61e19931ed
Dec 20 17:54:34 ip-172-31-2-154 wordfreqservice[2642]: Successfully processed job 103879ba-7f79-469c-aec8-1d61e19931ed
Dec 20 17:54:34 ip-172-31-2-154 wordfreqservice[2642]: Deleted message, 103879ba-7f79-469c-aec8-1d61e19931ed
```

EC2 instance page showing launched / terminated instances during this process

Below we will show the design and screenshots of how the system scales after receiving more jobs for processing, and how they scale down when the instances are not necessarily needed in a 1 and 5 minute time window.



After the construction of the scaling architecture we used CloudWatch services for the creation of the alarms that will be triggered when a specific metrics threshold would be passed. Below we see the ADD alarm that monitors the work freq-jobs SQS and checks if the number of messages sent in the queue is more than 10. In case this happens, it will run Add instance policy in the auto-scaling group that will add two additional EC2 instances for better performance. This graph shows the messages in the time period of 5 minutes that I used.

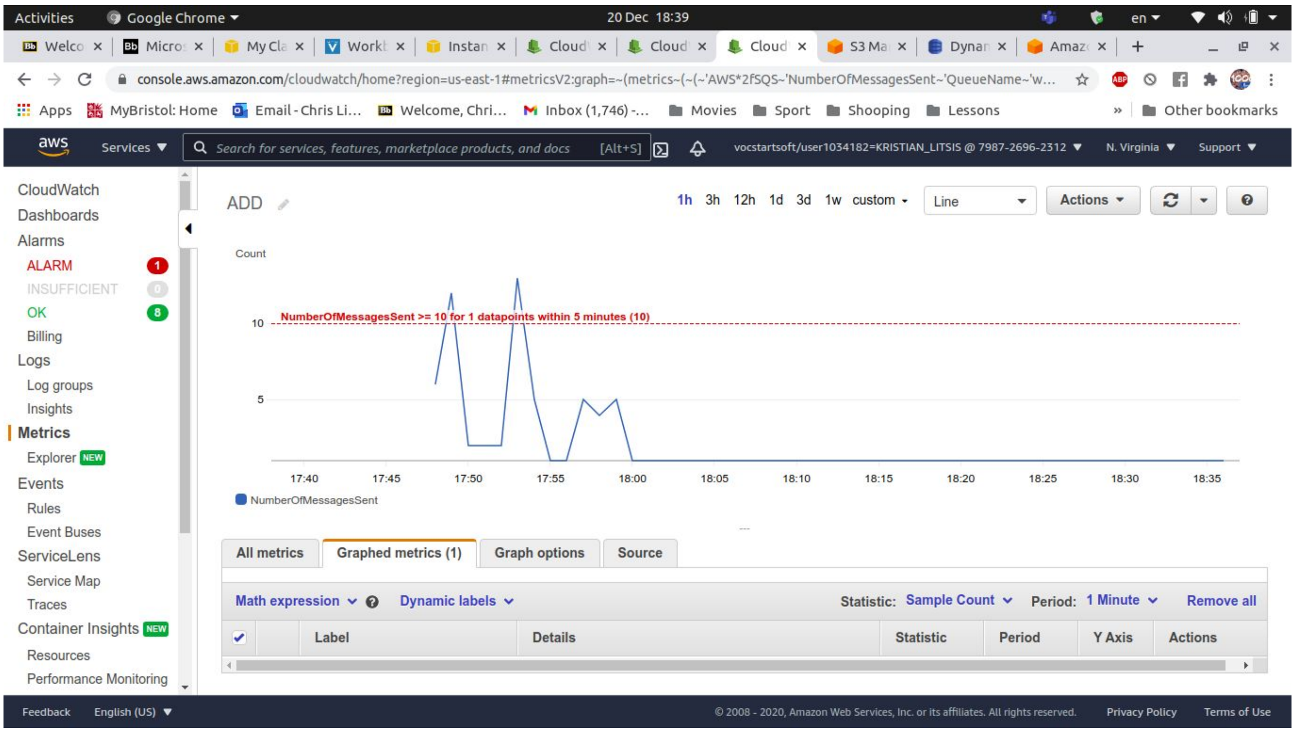


NumberOfMessagesSent >= 10 for 1 datapoints within 5 minutes (10)

NumberOfMessagesSent

Statistic: Sample Count   Period: 5 Minutes

We can see the same alarm but in the time period of 1 minute. As we see we have some high and lows so I think it would not be better for the policy to depend on 1 minute because it would add and remove instances frequently and there would be a delay introduced so this is the reason why I went with 5 minute time window, which is also the default in the amazon documentation.



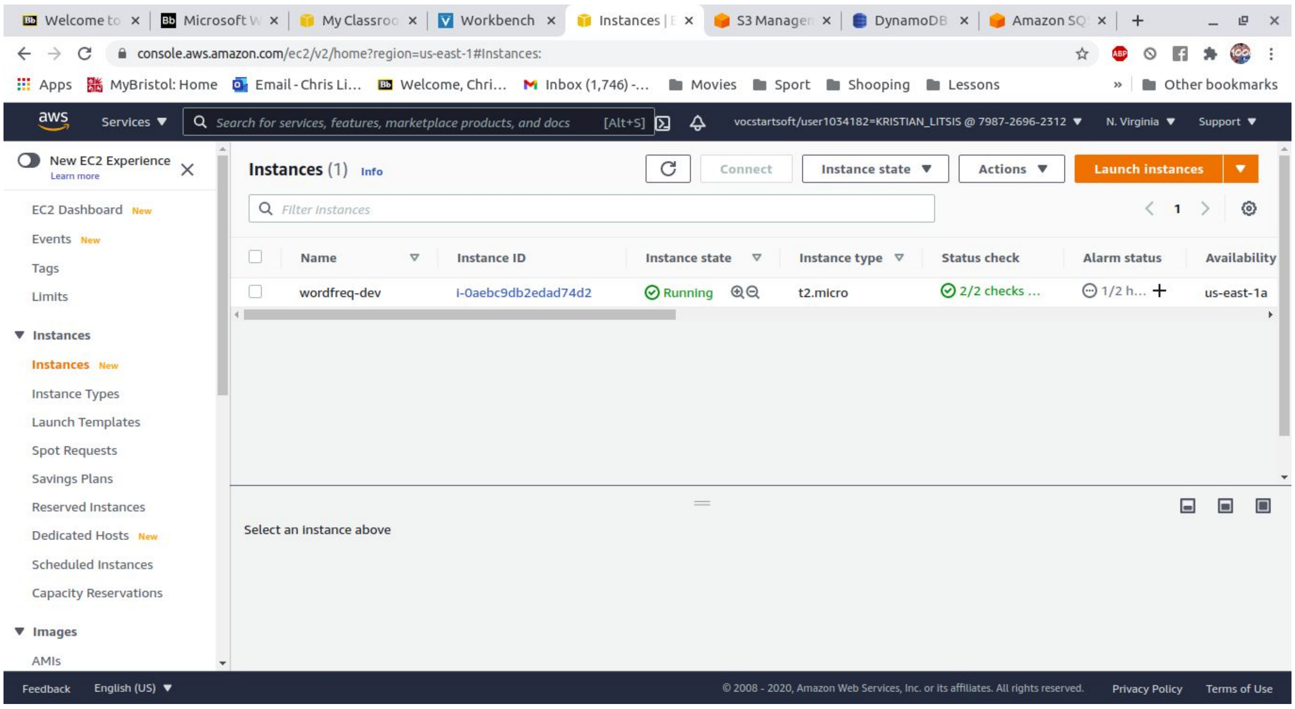
NumberOfMessagesSent >= 10 for 1 datapoints within 5 minutes (10)

NumberOfMessagesSent

Statistic: Sample Count   Period: 1 Minute

Kristian (Chris) Litsis **User_Id:** wg20498

We have one default EC2 instance that always runs the application. I decided for this instance to be detached from the auto-scaling group because I could let this instance be inside in the group and make the desired and minimal size 1 but it would run always even when I would sign out from the account and for this reasons I didn’t want to spend the money of AWS Educate account. Another reason for doing this is because the auto-scaling group doesn't let you add more than 4 instances inside the group and by making a EC2 detached outside the group I can run 5 instances which gives me more computational power.

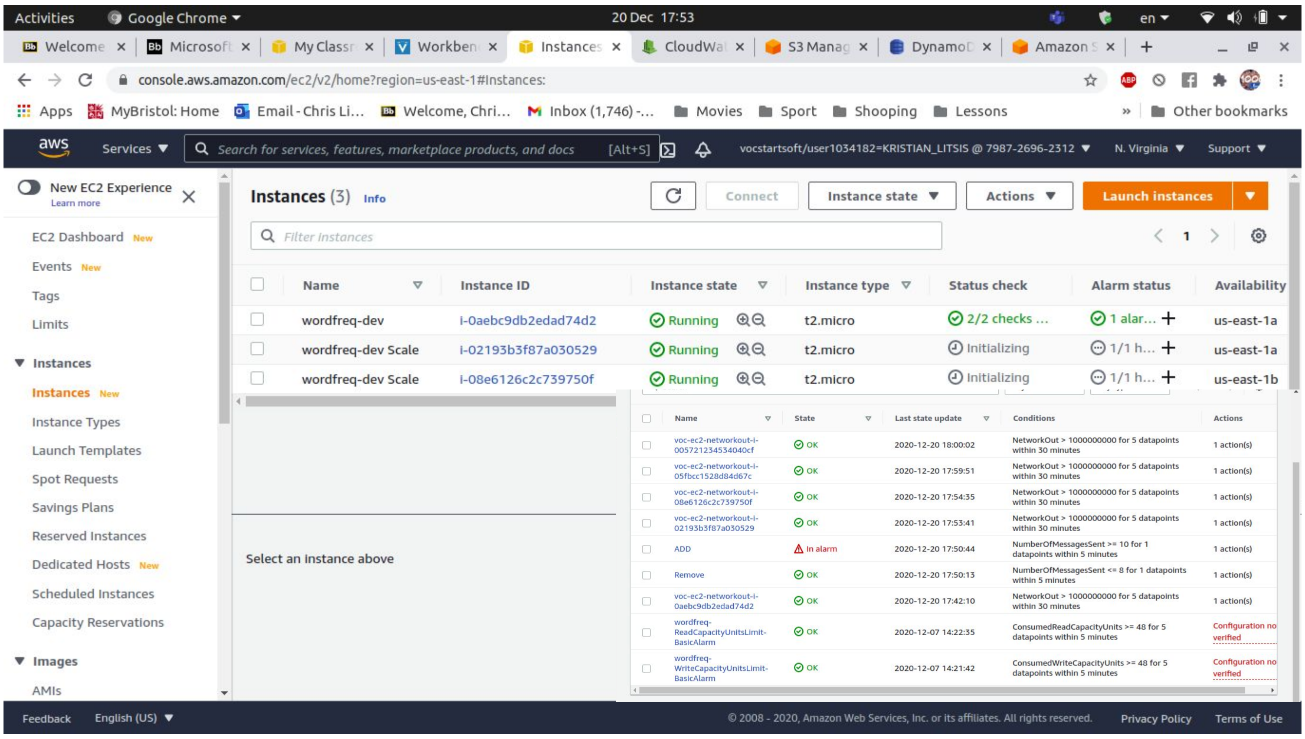


Instances (1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
wordfreq-dev	i-0aebc9db2edad74d2	Running	t2.micro	2/2 checks ...	1/2 h...	us-east-1a

Select an instance above

We perform the test for the auto-scaling group by uploading more files to the S3 bucket we see that the alarm ADD triggers and 2 additional EC2 instances named wordfreq-dev Scale lunches and after passing the checks can run the application.



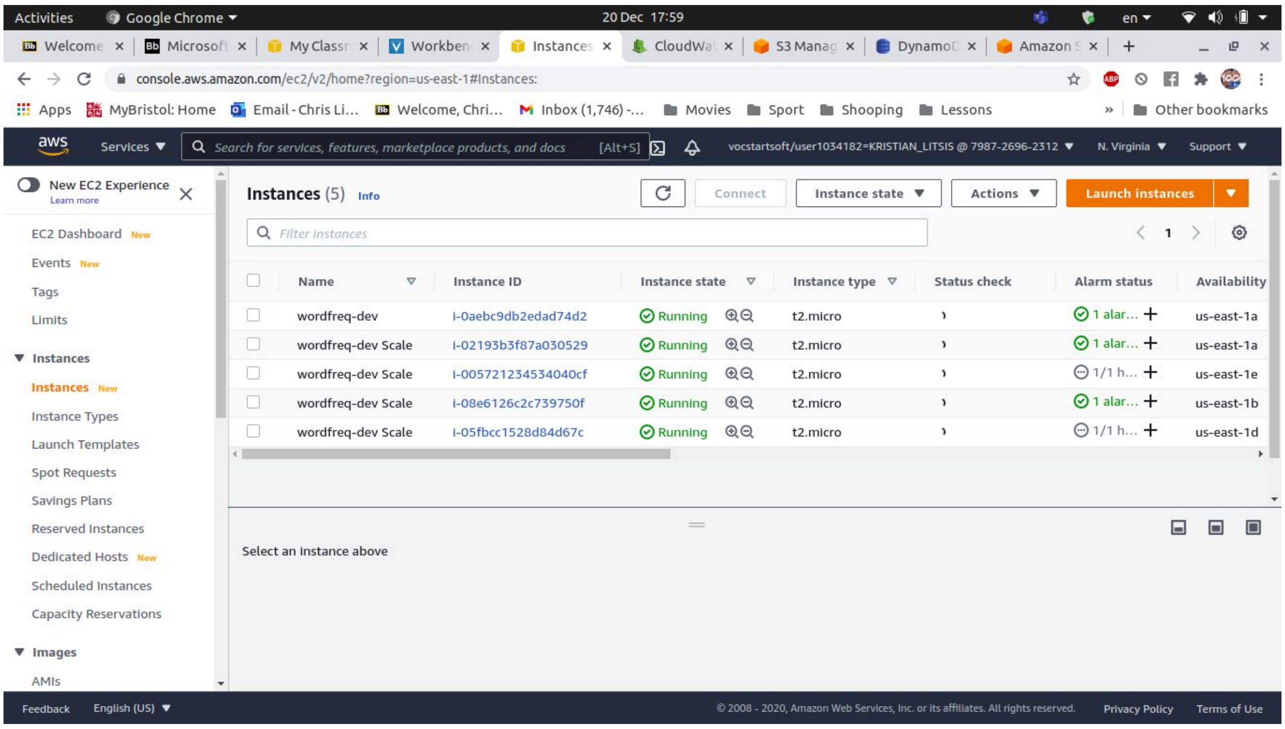
Instances (3)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
wordfreq-dev	i-0aebc9db2edad74d2	Running	t2.micro	2/2 checks ...	1 alar...	us-east-1a
wordfreq-dev Scale	i-02193b3f87a030529	Running	t2.micro	Initializing	1/1 h...	us-east-1a
wordfreq-dev Scale	i-08e6126c2c739750f	Running	t2.micro	Initializing	1/1 h...	us-east-1b

Select an instance above

Name	State	Last state update	Conditions	Actions
voc-ec2-networkout-i-005721234534040cf	OK	2020-12-20 18:00:02	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
voc-ec2-networkout-i-05fbcc1528d84d67c	OK	2020-12-20 17:59:51	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
voc-ec2-networkout-i-08e6126c2c739750f	OK	2020-12-20 17:54:35	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
voc-ec2-networkout-i-02193b3f87a030529	OK	2020-12-20 17:53:41	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
ADD	In alarm	2020-12-20 17:50:44	NumberOfMessagesSent >= 10 for 1 datapoints within 5 minutes	1 action(s)
Remove	OK	2020-12-20 17:50:13	NumberOfMessagesSent <= 8 for 1 datapoints within 5 minutes	1 action(s)
voc-ec2-networkout-i-0aebc9db2edad74d2	OK	2020-12-20 17:42:10	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
wordfreq-ReadCapacityUnitsLimit-BasicAlarm	OK	2020-12-07 14:22:35	ConsumedReadCapacityUnits >= 48 for 5 datapoints within 5 minutes	Configuration no verified
wordfreq-WriteCapacityUnitsLimit-BasicAlarm	OK	2020-12-07 14:21:42	ConsumedWriteCapacityUnits >= 48 for 5 datapoints within 5 minutes	Configuration no verified

After passing a little time and having more files uploaded we can see that we launch the maximum number of EC2 instances permitted by the AWS Educate account in different availability zones and we can process every file faster.

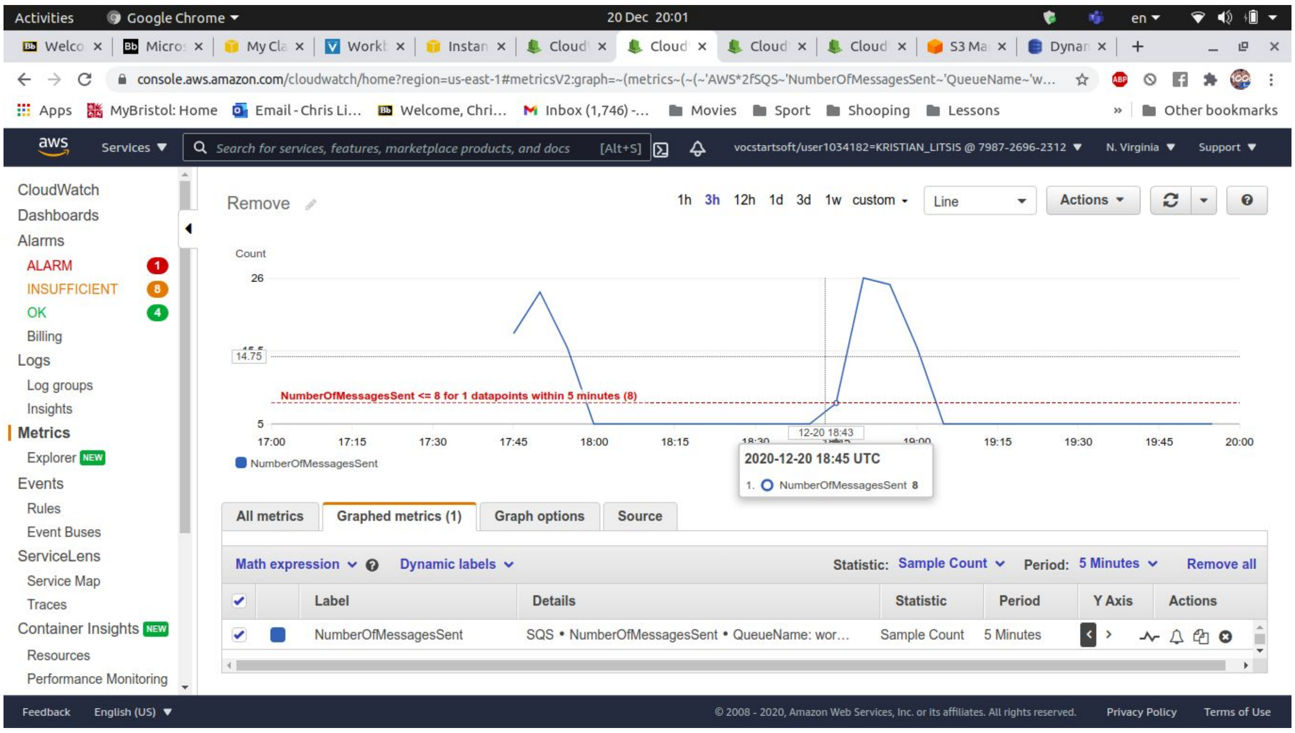


Instances (5) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
☐	wordfreq-dev	i-0aebc9db2edad74d2	Running	t2.micro		1 alar...	us-east-1a
☐	wordfreq-dev Scale	i-02193b3f87a030529	Running	t2.micro		1 alar...	us-east-1a
☐	wordfreq-dev Scale	i-005721234534040cf	Running	t2.micro		1/1 h...	us-east-1e
☐	wordfreq-dev Scale	i-08e6126c2c739750f	Running	t2.micro		1 alar...	us-east-1b
☐	wordfreq-dev Scale	i-05fbcc1528d84d67c	Running	t2.micro		1/1 h...	us-east-1d

Select an instance above

Below we will see how the Remove alarm triggers. The system monitors the number of messages sent to be lower than the threshold of 8 messages and applies the Remove Instance step policy of the auto-scaling group which will remove two EC2 instances. For the same reason that I explained above in the ADD alarm I decided to go with the time period of 5 minutes to remove the additional cost in time for adding and removing instances frequently.



Remove

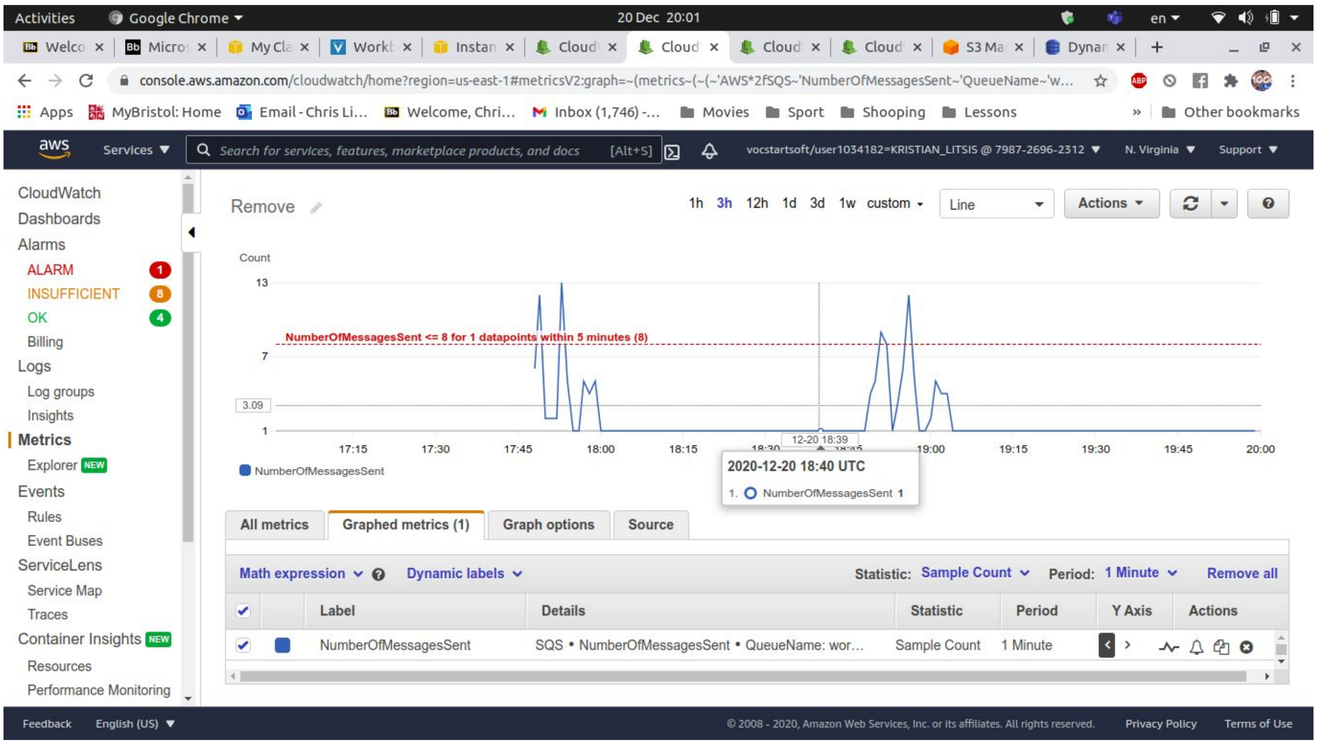
NumberOfMessagesSent <= 8 for 1 datapoints within 5 minutes (8)

2020-12-20 18:45 UTC

1. NumberOfMessagesSent 8

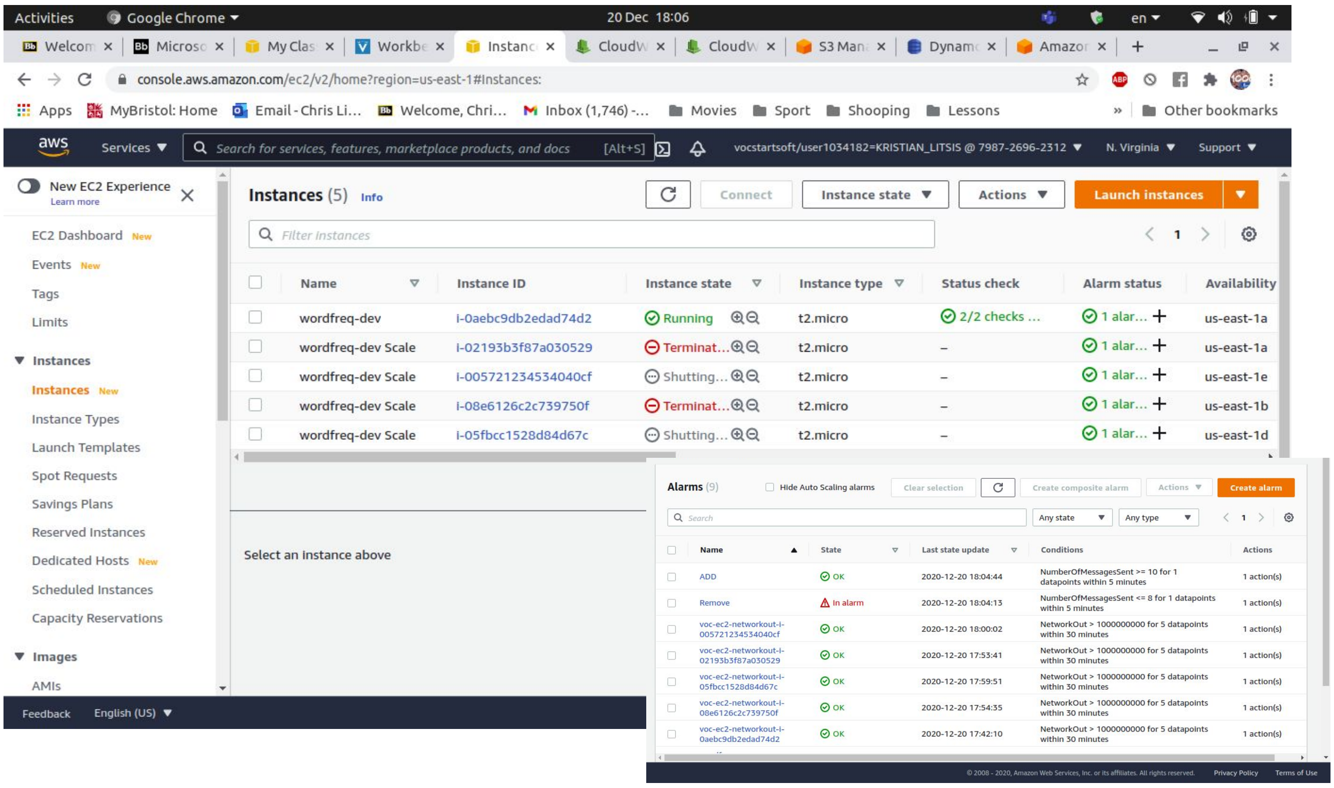
Statistic: Sample Count Period: 5 Minutes

	Label	Details	Statistic	Period	Y Axis	Actions
✓	NumberOfMessagesSent	SQS • NumberOfMessagesSent • QueueName: wor...	Sample Count	5 Minutes		



Label	Details	Statistic	Period	Y Axis	Actions
NumberOfMessagesSent	SQS • NumberOfMessagesSent • QueueName: wor…	Sample Count	1 Minute		

And we can see how the alarm performs when the number of messages are below the threshold. The Remove alarm will trigger and EC2 instances will be removed from the group.

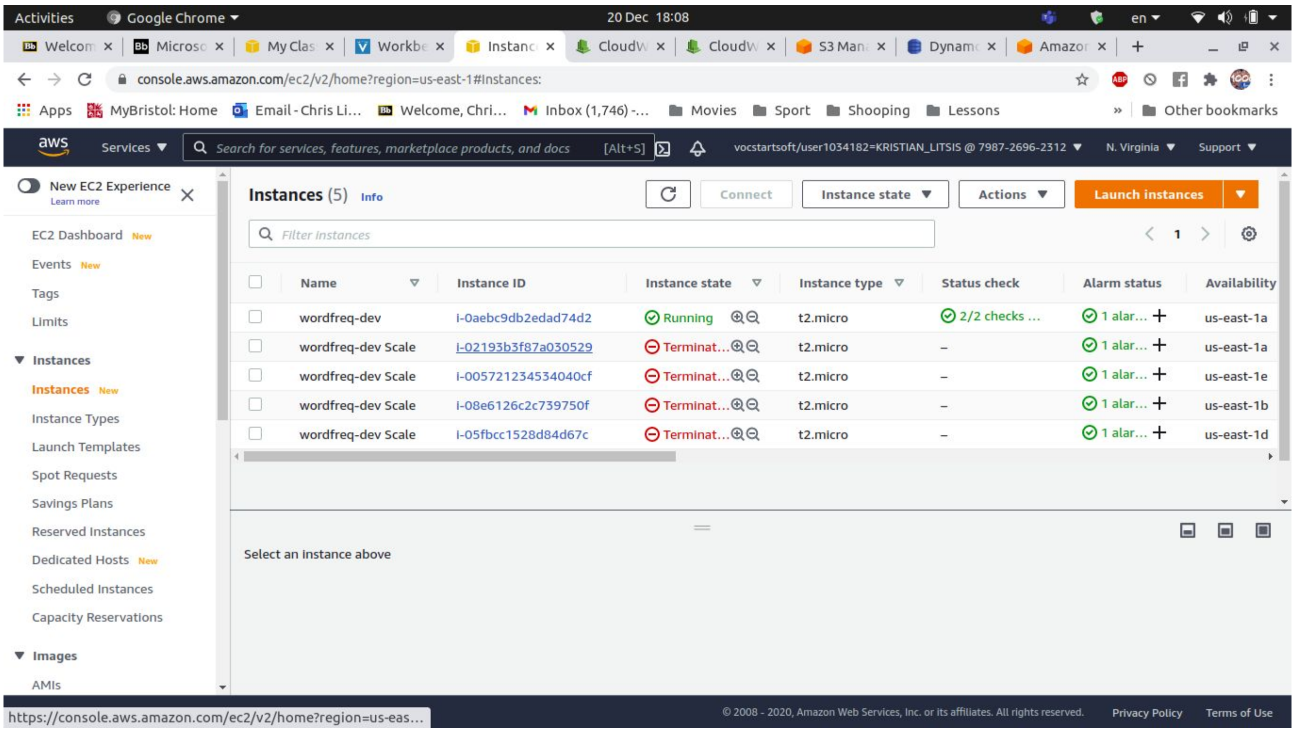


Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
wordfreq-dev	i-0aebc9db2edad74d2	Running	t2.micro	2/2 checks …	1 alar…	us-east-1a
wordfreq-dev Scale	i-02193b3f87a030529	Terminat…	t2.micro	–	1 alar…	us-east-1a
wordfreq-dev Scale	i-005721234534040cf	Shutting…	t2.micro	–	1 alar…	us-east-1e
wordfreq-dev Scale	i-08e6126c2c739750f	Terminat…	t2.micro	–	1 alar…	us-east-1b
wordfreq-dev Scale	i-05fbcc1528d84d67c	Shutting…	t2.micro	–	1 alar…	us-east-1d

Alarms (9)

Name	State	Last state update	Conditions	Actions
ADD	OK	2020-12-20 18:04:44	NumberOfMessagesSent >= 10 for 1 datapoints within 5 minutes	1 action(s)
Remove	In alarm	2020-12-20 18:04:13	NumberOfMessagesSent <= 8 for 1 datapoints within 5 minutes	1 action(s)
voc-ec2-networkout-i-005721234534040cf	OK	2020-12-20 18:00:02	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
voc-ec2-networkout-i-02193b3f87a030529	OK	2020-12-20 17:53:41	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
voc-ec2-networkout-i-05fbcc1528d84d67c	OK	2020-12-20 17:59:51	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
voc-ec2-networkout-i-08e6126c2c739750f	OK	2020-12-20 17:54:35	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)
voc-ec2-networkout-i-0aebc9db2edad74d2	OK	2020-12-20 17:42:10	NetworkOut > 1000000000 for 5 datapoints within 30 minutes	1 action(s)

When all the processing is close to finish, the Reduce alarm will remove all the instances from the auto-scaling group and only the default EC2 will run as it is required as a minimum to have always an instance running that will be waiting for jobs.



Instances (5)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
wordfreq-dev	i-0aebc9db2edad74d2	Running	t2.micro	2/2 checks ...	1 alar...	us-east-1a
wordfreq-dev Scale	i-02193b3f87a030529	Terminat...	t2.micro	–	1 alar...	us-east-1a
wordfreq-dev Scale	i-005721234534040cf	Terminat...	t2.micro	–	1 alar...	us-east-1e
wordfreq-dev Scale	i-08e6126c2c739750f	Terminat...	t2.micro	–	1 alar...	us-east-1b
wordfreq-dev Scale	i-05fbcc1528d84d67c	Terminat...	t2.micro	–	1 alar...	us-east-1d

Select an instance above

I wanted to mention that because of the low internet connection, when I tried to run with bigger CPU processing power for the new auto-scaling group, the result wouldn’t change much and this because it needed much time to upload the files in the S3 bucket(bottleneck) so I couldn’t measure how much improvement I had in the system. However, using more CPU should make the processing much faster. Overall, in approximately 10 minute, all 55 files were processed.

• The optimisation steps (task D)

1) Increase resilience and availability of the application against component failure.

New auto-scaling group with bigger EC2 instance type

I Created a new auto-scaling group with bigger EC2 instance types for better performance. I also created multiple subnets in different availability zones to increase availability. This can be seen also when the alarm ADD triggers and adds additional EC2 instances that are located in different availability zones(5 total). Auto-scaling groups use EC2 instances status check to determine the health and, if unhealthy, will terminate the instance and launch a new instance.

Bucket Versioning S3

By using this feature we keep multiple variants of an object in the same S3 bucket. We can easily recover from unintended user actions or application failures. When enabled, if Amazon S3 receives multiple write requests for the same object simultaneously, it stores all of the objects.

On-Demand Backup and Restore DynamoDB

I created a DynamoDB on-demand backup named **TableBackUp** with the capability to create full backups of the table for long-term retention and archival for regulatory compliance needs. With backup, we can restore the table data anytime with a single click on the AWS Management Console or with a single API call. It uses a new and unique distributed technology that lets you complete backups in seconds regardless of table size and doesn't affect performance.

Replication rules S3

Replication rules enable automatic, asynchronous copying of objects across Amazon S3 buckets. Objects may be replicated to a single destination bucket or multiple destination buckets. Destination buckets named **s3://my1backupbucket** I used is in the same account and Region as the source bucket.

2) Increase security of the application for data protection and prevent unauthorised access.

Created security group no inbound rules from outside.

A security group acts as a virtual firewall and enables you to specify which inbound or outbound network traffic to allow. Traffic that is not explicitly allowed in rules is automatically denied. The platform allows the use of multiple security groups because they act at the instance level, not the subnet level, so each instance can be assigned to a different group providing more protection to data. I wanted also to move the instances in the private subnets for more security but it didn't permit me to create a Nat gateway for the isolation of the subnet.

IAM group and user

I Created an IAM EC2-Admin group with a user1 as the user in that group. AWS IAM is a web service that helps to secure and control access to AWS resources. We use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources. The permissions were AdministratorAccess, but I couldn't connect with my subnets because it didn't permit me.

Default encryption S3

Amazon S3 default encryption provides a way to set the default encryption behaviour for an S3 bucket. I used this to secure my **af-wordfreq-dec01** bucket. It encrypts all objects when they are stored in the bucket. The objects are encrypted using Amazon S3-managed keys (SSE-S3). which is a server-side encryption. We could use customer master keys (CMKs) stored in AWS Key Management Service (AWS KMS) but it required additional cost so we wanted the free option.

Block public access (bucket settings) S3

We Block all public access so no one else outside could access the data. For any request made through an access point, Amazon S3 evaluates and blocks all public access.

3) The application is as cost-effective as possible.

- Because I used an auto-scaling group with the cheapest EC2 Instance we ensured that it would be very cost-effective. I also wanted to use more EC2 instances with smaller sizes than little with bigger types. And the threshold that I used for 5 minutes in the alarms ensures to not let instances to run for much time but only when it is required. For the other tools, I used all the services that were free for encryption and replication.

- Use of Amazon S3 Intelligent Tiering in the lifecycle rules which is used to automatically save costs for data with unknown/changing access patterns. It stores objects in four access tiers. The first two are low latency access tier for frequent and infrequent access while the two others are for asynchronous and rare access.

Further Improvements:

- Encryption Type in DynamoDB but with additional cost.
- IAM with different roles to work for better security.
- Private subnet with NAT Gateway also for better security.

• Alternative data processing architectures

1. AWS EMR(Hadoop)

AWS's Elastic MapReduce (EMR) service runs Hadoop's MapReduce in the AWS cloud and can be used for our application. It is a distributed file system that maps data located on the cluster which gives faster data processing. It is more cost-effective and easy to use, no need to monitor alarms. More elastic and can be run on containers and configure EC2 firewall settings automatically controlling network access.

2. Apache Spark

It uses in-memory techniques to achieve significant speed-ups via implicit parallelism and execution-plan optimization, and is highly fault-tolerant: this makes it much faster than MapReduce. More complex, multi-stage, or iterative data processing by using RDD stored in memory and can be cached. They can recover from failure without costly replication.