DALHOUSIE UNIVERSITY

Term Assignment

CSCI 5409

Adv. Topics in Cloud Computing

Summer 2024

Professor :

Lu Yang

Prepared By:

Banner ID : B00950942

Name : Luv Prakashkumar Patel

CSID : lppatel (lv455130)

Dalhousie University

Halifax, NS

Table of Contents :

OneStore

Project Overview:

'*OneStore'*, a cloud based storage service designed to offer over the cloud storage service. The primary purpose that it focuses to all users to store, manage and interact with files in the cloud. This project take into account various cloud technologies to provide a seamless and efficient solution to one's storage needs. This application is built leveraging the power of AWS (Amazon Web Services) services, that provide cloud resources and uses serverless architecture, which modernizes the infrastructure and provide high availability.

Top feature of the application:

- **File Upload :** Users can upload files through UI, which will get stored in the cloud and provided freedom to access it from any internet connected device.
- **View Files :** Users can see list of all uploaded files once logged in. Each file diaplays the filename and two buttons "Download" and "Delete".
- **Download File :** Users can download files back to their device using the download feature.
- **Delete File :** Users can delete files that they do not need anymore. This way, unwanted files can be removed in order to better organize the important files.

List of Services:

For OneStore, several AWS provided services are used to ensure robust, secure cloud storage solutions. Each service is selected after a thorough and extensive research, ensuring optimal performance while minimizing cost factor.

Below are the services that are used in development of this project:-

Storage

1). Amazon S3 (Simple Storage Service):

Purpose : For storing the user's uploaded files

S3 Over EBS & EFS :

S3 is more capable of handling large volume of data, especially unstructured data. On the other hand, service like EBS is better suited for tasks that require low-latency access and consistent performance, such as Operating System (OS) files, Entire Databases, or data like transactional Data.

Moreover, S3 is object storage, as well is not limited to EC2. Also, files in the S3 can be accessed programmatically , making it an ideal choice for this use case, unlike other services like EFS or EBS.

Network

Amazon VPC:

I chose to use AWS VPC as it offers a robust and scalable capacity for network infrastructure. Also, I am using EC2 instance, so this help secure the traffic to the instance by setting permission for accessibility.

API Gateway :

I chose API Gateway because it is capable to handle all the tasks involved including handling traffc for thousands of API calls concurrently, access control as well as authorization and version management of the APIs. Also, allows easy deployment of the created API. As far as security concerns, API gateway offers inbuilt protection against DDoS attacks , thus protecting the API from malicious traffic. Moreover, its auto scaling capability allows for scaling to handle increased traffic handling.

Compute

EC2 over Beanstalk :

I chose EC2 over bean stalk because EC2 provides full control over the compute resources, including the underlying OS, configuration, etc. On the other hand beanstalk provide all the above mentioned features, but it does not allow us to manage underlying infrastructure, so it becomes challenging to modify the deployment as per one's requirement. Moreover, with EC2 we can

customize the environment to match our system performance and other network and instance requirement.

Moreover, EC2 provides various purchase options , helping in cost savings by selecting plns based in ones needs. While beanstalk also support auto scaling, EC2 auto scale groups provide more fine control over the policies and instances, which the beanstalk lacks to provide.

Lambda over ECS :

While ECS provide more flexibility for execution environment, as compared to lambda, the use case for my application does not require lambdas to perform continuous running tasks, making lambda functions a more viable option. Also, cost aspect for lambda is per request, just saving cost by paying for just the time the lambda executes, Moreover, ECS is more suitable for programns with longer runtime, whereas my application require short run time for lambda to help make content available faster.

Moreover, I could have also deployed everything in ECS, as opposed to using EC2 for frontend and lambda for backend python code, but again there is restriction for using atleast two services from the list of compute services.

General

AWS Cognito:

It provides user authentication, authorization and user management for applications. It enables users to signup or signin using just their email and password. This service is used because to access personal files, user must signin in to an account to see their uploaded files. It provides secure signin using cognito user pool. This user pools are a secure user directory that can scale to millions of users. It also allows lambda trigger invocation , which

helped me send sign up mail to users who signup to the application for creating an account.

Moreover, cognito ensures that data is encrypted during transit and at rest, thus meeting security concerns. It provides 50000 monthly active users free of cost, saving cost for small companies like startups.

Amazon SNS:

To send user notification upon signup as well as when they upload a photo, SNS is a simple notification service, that works on publisher subscriber model. A user subscribes to the topic upon account creation and then receives email notification upon subsequent tasks.

SNS over SES :

Althoug I could have used SES(Simple Email Service), due to its ease of sending email notification, but as per the listed services, it was not present in the list and also failed to satisfy at least 2 services to be used under General

SNS over SQS :

I chose SNS over SQS because SQS works on queuing service. It allows messages to be sent and received between distributed application. This cases is not applicable for my application and moreover, in event of sending messages to every user, in case of any update or emergency, SQS is not capable in handling this situation where as SNS can handle this.

Moreover, when a user uploads a photo to cloud storage, they must be notified instantly, which SNS does pretty quickly. On the other hand, SQS does not inherently provide real-time message delivery due to the nature of queuing and message retrieval by consumers. Thus, SNS is suitable choice for my application, as user needs to be instantly notified of the event.

Deployment Model

Public Cloud :

- I chose the Public Cloud (Services by AWS). In this deployment model, every service is provide by AWS(Amazon Web Services). AWS handle all the physical infrastructure and all the related services, so there is no infrastructure related maintenance needs to be taken care of by the consumer. Moreover, AWS due to its diversified locations of Data Centers, they manage to achieve high availability.

- This model provide pay-as-you-use model for pricing, which helps us save cost during the low traffic period, thus reducing the burden to manage the physical infrastructure, even in tough times.

- It offers vast variety of services, which helps us focus more on the core aspects of the development, without worrying about the other things.

- Provides multi-region support, allowing us to deploy whole or parts of application in various region of maximum availability this reducing the data latency as well as increasing the availability and performance of the application.

- Reliability is very high, thus providing the confidence to the developers and reducing the stress of unavailability that might occur due to damaged systems.

- Application is highly optimised by leveraging the power of cloud computing resources, helps make robust application.

Overall, public cloud deployment model is perfect match for my application, as it provides high scalability , high availability as well as reliability, while also paying equal importance to data and application security by using various AWS services.

Delivery Model

Storage as a Service (Staas):

Why StaaS?

⇨ **Scalability :** StaaS arrangements can effortlessly scale up or down based on client needs, making it basic to oblige development or changes in request.

- ⇨ **Upgrades & Updates:** StaaS suppliers handle all overhauls and updates, guaranteeing that clients continuously have recent highlights and security improvements without manual mediation.
- ⇨ **Backup Solution :** StaaS provide backup and damage recovery as part of their service, thus ensuring data integrity and high availability

Component Delivery Model:

S3 : Storage as a Service

Lambda : Function as a Service

EC2 : Infrastructure as a Service

API Gateway : Platform as a Service

SNS : Messaging as a Service

Cognito : Platform as a Service

Final Architecture

AWS Cognito:

- ⇨ Performs user authentication which allows users to signin and signup to our application to access the features of the '*onestore*' application to upload photos to the cloud.

AWS Lambda:

- OneStoreSendNotification
  - Sends the subscription confirmation email to the user who just registered using the cognito. This lambda function is triggered by the congito user pool.

- OneStoreUploadFile
  - Lambda function to perform file upload to the s3 bucket that is passed by user through the frontend.

- OneStoreConfirmUser
    - Lambda function to confirm the user authentication during the singup process. This takes place when the user successfully enters correct verification code sent to their registered email.

- OneStoreListFiles:
    - Function to retrieve all the files that the logged in user has uploaded.

- OneStoreDeleteFile:
    - Function to delete a particular file that the user has requested to delete from the frontend. It deletes the data from the S3 bucket.

- OneStoreUploadFileNotification:
    - Function to publish the successful file upload email, when a user upload a photo using the UI into the storage (S3).

Amazon SNS (Simple Notification Service):

- Sends instant real-time email notifications to the users of the application, such as during signup as well as during successful upload action.

API Gateway:

- Serves as an entry point to access various backend lambda functions, that perform specific task as per the request sent. It receives HTTP request from the frontend and passes onto the backend services.

EC2:

- It is an instance (a virtual machine hosted on cloud), where the application's frontend is hosted, that allows other users to access the web application through the worldwide web. It is configured with internet gateway, security groups, route tables and VPC, for secured access.

AWS S3 (Simple Storage Service):

- A object storage service, used to store the user uploaded data files. Stores the data in encrypted format while data transit as well as data at rest.

Architecture Diagram

⇨ Below is the architecture diagram that is created using online free tool, which helps make diagram easy and more accurate to visualize [2]. The tool is “draw.io”. It makes make documentation ready diagram professionally and accurately, helping others understand the high level scenario of the whole project in one timeframe.

Below is the architecture diagram for “OneStore” application:-

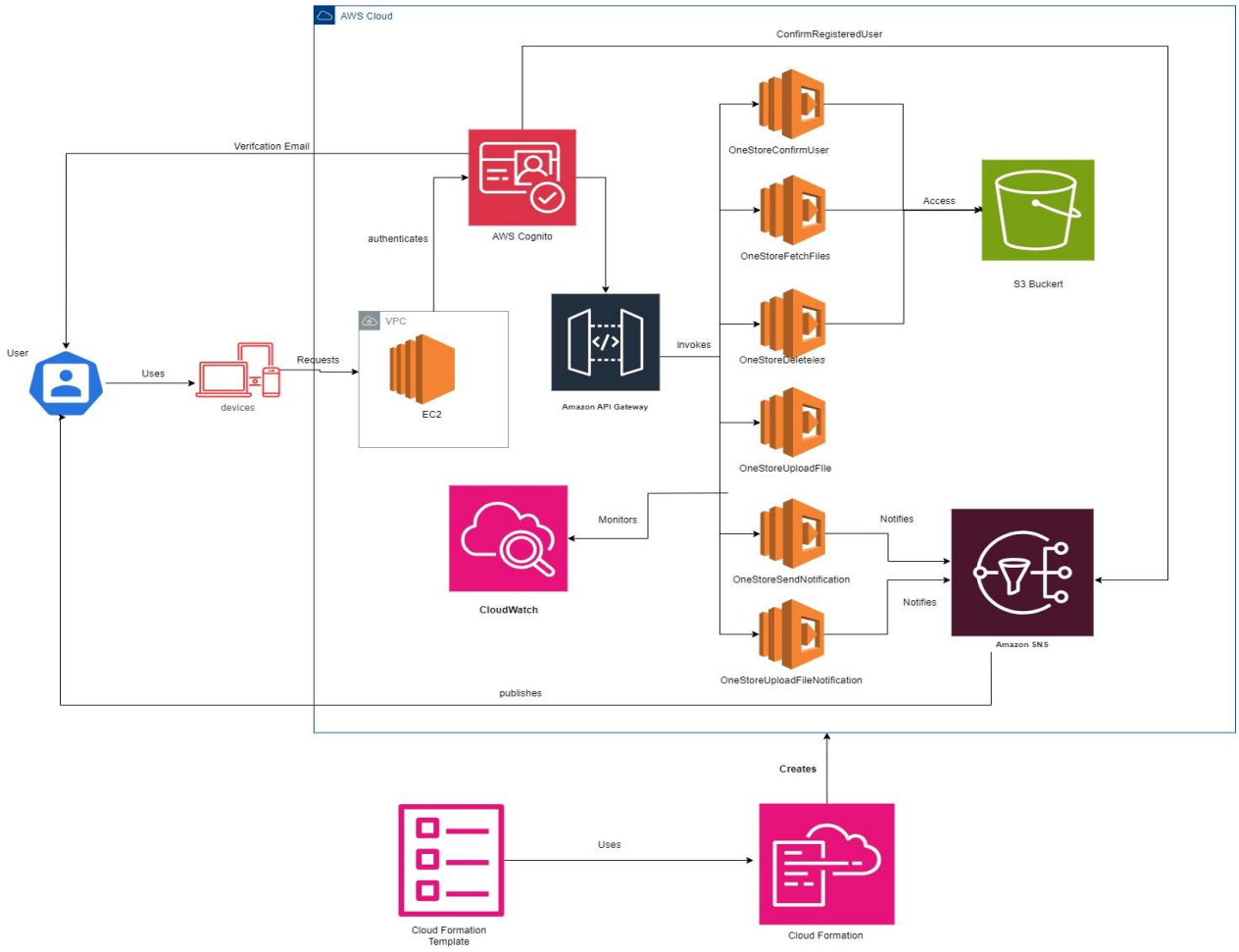

Figure 1: Application Architecture Diagram [2]

Application Stack:

Python (Python 3.12) :

- **Purpose :** For developing the aws lambda functions, while focusing on serverless development.
- **Application :** All the above mentioned lambda functions are built in this programming lamguage. It is fast and faster to develop, saving time and effort.

React JS :

- Used for development of the frontend application which is hosted in the Amazon EC2 instance.

System Deployment

How is your system deployed ?

CloudFormation

Helps automate the deployment and setup of entire infrastructure of the application. Makes the application deployment platform independent. Just by running the cloud template file using cloud formation, the whole infrastructure can be developed without manual intervention.

Frontend : Deployed in EC2 Instance, which the users can access using the public IP of the instance

Backend : All of the backend infrastructure is deployed using cloud formation, ranging from Cognito, lambda, S3, SNS, EC2 & API gateway.

System Deployment

Frontend, developed using react JS, these is containerized and the created image is uploaded in the Docker Hub. Then, during the cloudformation, when the EC2 instance is provisioned, this docker image is then pulled in the created ec2 instance and then the image is automatically run inside the ec2 instance along with required environment variables. Once the image is up and running, users can access the application UI thorugh the Public Ip of the ec2 instance.

Backend code is written in python and is present in the S3 bucket in compressed zip file format, which is then fetched during the cloud formation , and the respective lambdas are created.

Where is data stored ?

- ⇨ The data is stored in the S3 bucket. So when a user uploads any image, the image is base64 encoded and then passed to the lambda function for security purpose, where the data is then decoded and then stored as object in S3 bucket. I could have created multiple buckets for every user, but the reason is explained below.

Vulnerabilities:

- ⇨ As mentioned above, a single s3 bucket with a folder for each user is being created where the user specific images are stored. But the issue is that there is a risk of other user able to access other users data. This could have been prevented by creating unique bucket for each user and storing that user specific data into that bucket. This way, the files are isolated from each user, but as using learner lab, I do not have access to create IAM role that I can attach with each bucket with custom policy, but if in future I am able to create new IAM role, I would have created separate buckets for each user, for more security and privacy.

Security Aspect

API Gateway:

- ⇨ Using API gateway, ensures robust architecture for the OneStore application, by enabling secured and seamless access to backend logic in lambda functions. Morever, using api resources, each function has dedicated resource configured which is covered in dedicated API , used for interacting with the lambda functions.

S3 Bucket :

- ⇨ S3 is the chosen as it is the only service that allow us to block public access, thus making sure no access is present, making it suitable for my application, to store sensitive files.
- ⇨ S3 encrypts all the objects stored inside all bucket, thus aligning more with the security aspect.

- ⇨ The bucket policy setup restricts the public access from all external source, thus making sure no unauthorised user can access it without permission.
- ⇨ I have created just one S3 bucket to store user file, because as using learner lab and having just LabRole as IAM role, I am not able to provide custom access to the S3 buckets that I wanted to created for each bucket dedicated to each user, thus I am forced to use single S3 bucket to store all the user data, Still for sake of management, I am categorizing data into user folders, created for each user after they signup.

Lambda :

- ⇨ Lambda are not publicly accessible directly, thus preventing the risk of unauthorized person directly accessing it. The call can only be made using API Gateway created for each lambda resource, that perform specific tasks.

EC2 :

=> This is the place, where the frontend is deployed. The security group is defined in such a way that it only allows PORT 22 and PORT 3000 TCP access , where port 22 is for access through SSH for internal purposes and port 3000 is where the frontend will be accessible. Rest, no other port is exposed, thus aligning with security measure.

Cognito:

- ⇨ It uses TLS (Transport) to encrypt data in transit in between application and cognito service. Thus, aligning with security measure and preventing the man in the middle attacks.
- ⇨ User credentials in the user pool are held securely in user pools just eliminating the risk of data exposure.

How do all of the cloud mechanisms fit together to deliver your application?

- ⇨ So to describe how all the services, lets go through all the services , lets understand the flow of my project. So the end user will use my application using webs browser through their devices. The frontend is deployed in the EC2 instance where the docker image is pulled from the docker hub and that image is run on the ec2 instance. TO use the application, the user need to register themselves, which takes place using cognito. Once user creates their account using cognito, they are sent a verification code mail which takes place using cognito itself. After This , a lambda is triggered that will send confirm subscription mail to user which takes place using SNS. Upon subscribing, the user will be subscribed to the topic and will be able to receive future

notifications. Next, after login, the user can click on upload file, that will trigger the "/uploadFIle" resource of the API Gateway, that will call the lambda function which perform the photo upload task. Next, upon successful storage of photo in S3 bucket, that is used to store these objects, a lambda function is called that will publish a new email to user indicating successful upload. Next if the user wants to see the files that he has uploaded, then using "Myfiles" tab, they can see the files. This retrieval of the data takes place through lambda, that fetches data from the s3 bucket. Next If user wants to delete file, then they can click on the delete button and this will trigger a lambda function that will delete object from the s3 bucket. Moreover, the ec2 instance is within a custom VPC, which has security groups, subnet, etc. and the inbound rules allows traffic form port 22 and 3000 only. PORT 22 because it allows to connect to this instance using SSH client, and 3000 because ethe docker image frontend runs on the port 3000.

What would organization have to purchase to reproduce the same architecture in a private cloud ?

- Reproducing application's architecture in a private cloud with similar availability would involve purchasing a combination of hardware, software, and support services. Below is a breakdown of what might be need, along with a rough cost estimate.

- Initially, need to invest in a range of servers for compute, storage, and database functionalities. This includes compute servers, two storage servers, totaling approximately $30,000 [3]. Network infrastructure is also crucial, including enterprise firewalls, routers, and switches, which would cost around $30,000 [4]. For backup and redundancy, including UPS systems and cooling units, an additional $25,000 would be necessary [5].

- On the software side, need for virtualization solutions such as VMware vSphere for managing your servers, costing about $16,000 [6]. along with management software like VMware vCenter for $10,000 [7]. To replicate the object storage capabilities of AWS S3, I could use OpenStack Swift or MinIO, costing roughly $10,000. Identity management solutions similar to AWS Cognito and encryption tools would add approximately $8,000. Messaging solutions, like RabbitMQ or Apache Kafka, would cost around $4,000 [8] .

Cost Metrics

Resource used: Amazon Pricing Calculator [9]

Below is the expected cost that I have calculated for three categories : up-front cost, cheap and expensive. Estimates are for the application developed.

1). Up-front Cost:

- ⇨ Based on my calculations of the upfront cost, the initial cost for setting up the environment is negligible, this is because all the resources calculated are chosen precisely such that there is equal balance of performance and budget friendly.

2). Cheap Cost:

=> Below is the cost calculated based on the minimal resources used thus, focusing on the cost factor .

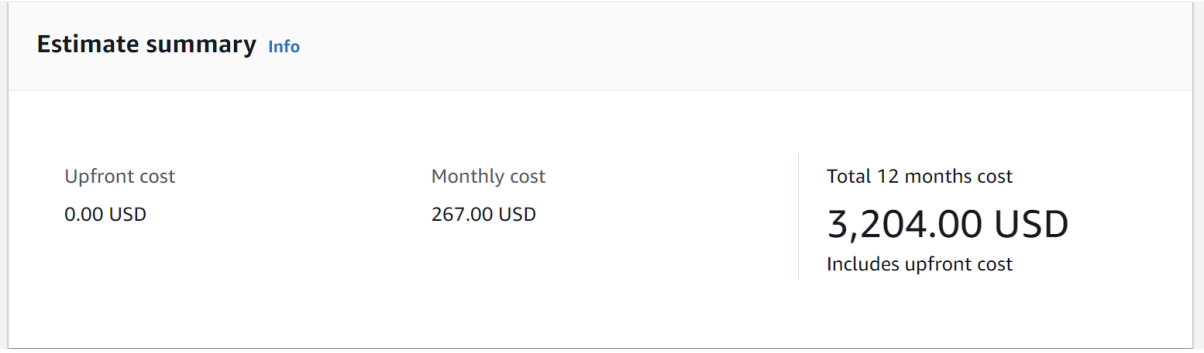

Figure 2: Cheap cost estimation

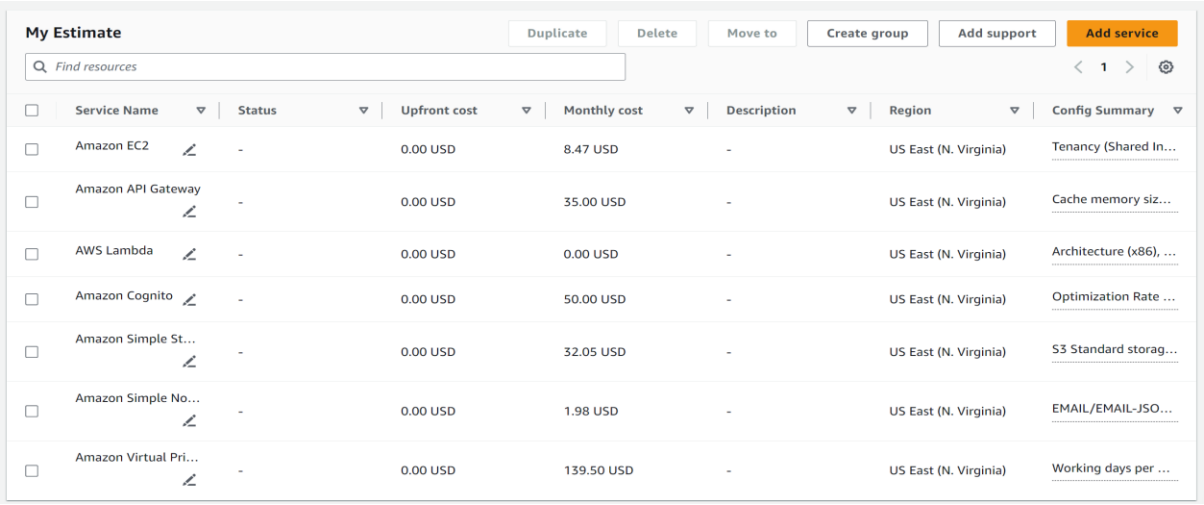

Figure 3: Detailed cost metrics

Moderate Cost :

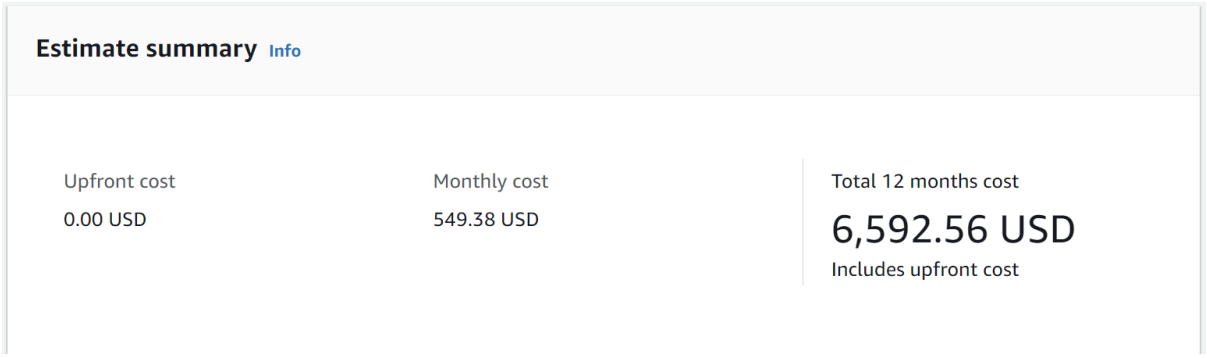

Figure 4: Moderate Cost, required for smooth functionality

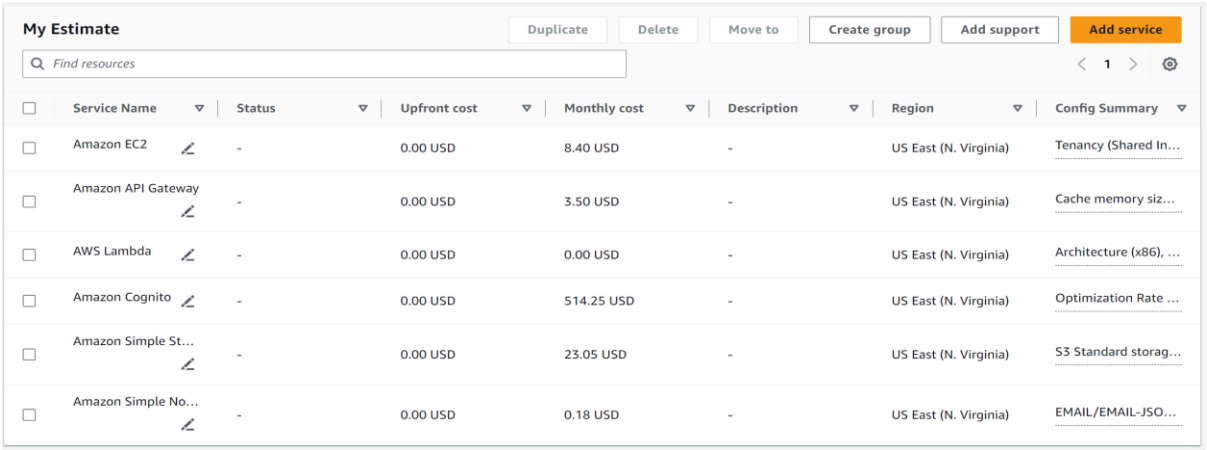

Figure 5: Detailed breakdown of cost

Expensive Cost:

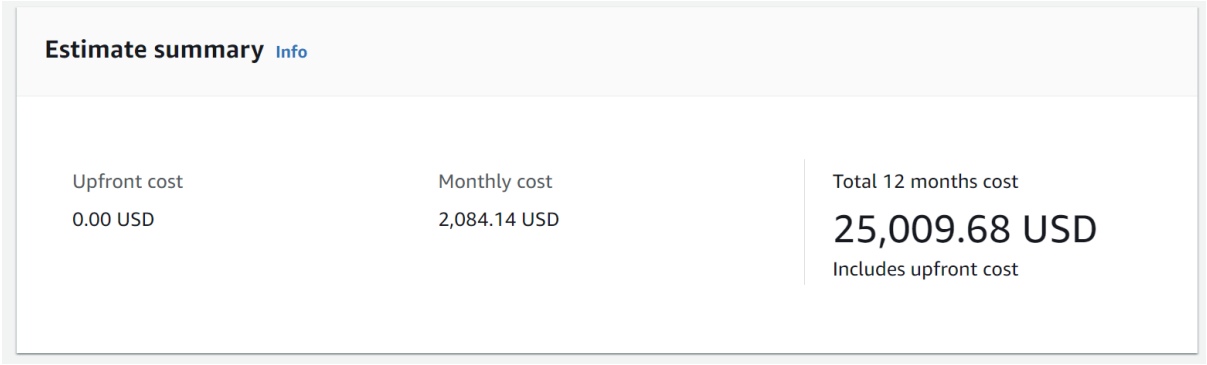

Figure 6: Expensive cost overview

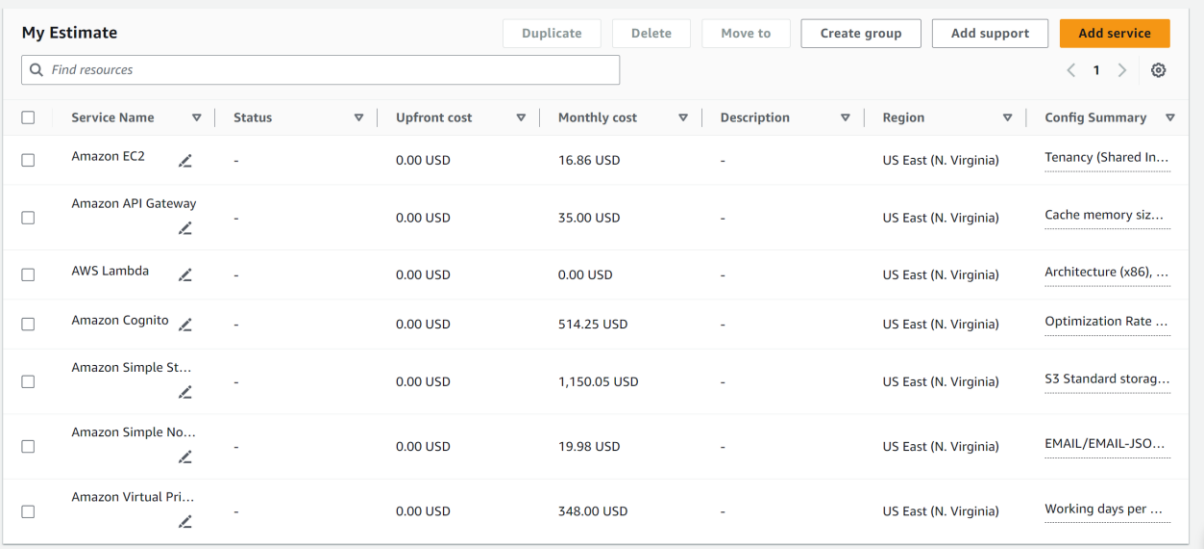

Figure 7: Expensive cost metrics detailed report

How would your application evolve if you were to continue development?

Future Development

⇨ As to continuously evolve the application, many new features can be added to make it more engaging and worthy. The features that I have thought of are as follows;

i). File sharing : Allow users to share files with other user. This can be achieved using S3 bucket with custom IAM roles and by using AWS CloudFront for content delivery.

ii). Monitoring : Enable monitoring of the services using Amazon CloudWatch for monitoring the services.

iii). Search Facility : Enable users to search thorough the list of files that they have uploaded to our application. This can be achieved using services like Amazon ElasticSearch Service, which makes full text searching and scanning possible with high performance.

References:

[1]. "Top 5 advantages of software as a ser", *IBM,* Online*,* Available: https://www.ibm.com/think/insights/software-as-a-service-advantages [Accessed: Aug 7, 2024]

[2]. "Flowchart Maker & Online Diagram Software", *Diagrams.net,* 2024,Online, Available: https://app.diagrams.net/ [Accessed : Aug 7, 2024]

[3]. "Dell PowerEdge R740 Server", *Dell Inc,*[Online], Available : https://www.dell.com/en-us/search/dell%20poweredge%20r740%20server [Accessed : Aug 2, 2024]

[4]. "Cisco ASA 5500-X with FirePOWER Services", *Cisco Systems*, [Online], Available : https://www.cisco.com/c/en/us/products/security/asa-firepower-services/index.html [Accessed : Aug 2, 2024]

[5]. "Managed Backup & Disaster Recovery Services", *Veeam Software*, [Online], Available : https://www.veeam.com/solutions/managed-services/managed-backup-dr-services.html?ad=menu-products-workloads [Accessed: Aug 5, 2024]

[6]. "VMware vSphere Standard Edition",*Insight.com.,* [Online], Available : https://www.insight.com/en_US/shop/product/BD711AAE/HEWLETT+PACKARD+ENTERPRISE/BD711AAE/VMware-vSphere-Standard-Edition---license---3-Years-24x7-Support---1-processor/ [Accessed : Aug 4, 2024]

[7]. "VMware vCenters", *Insight.com*, [Online], Available : https://www.insight.com/en_US/search.html?country=US&q=vcentere&instockOnly=false&start=0&salesOrg=2400&sessionId=AC0B6302149B9C0D76D651A5195CAC13.appprd4-2&lang=en_US&rows=50&userSegment=CES&tabType=products [Accessed: Aug 3, 2024]

[8]. "Plans & Pricing", *CloudAMQP.com*, [Online], Available : https://www.cloudamqp.com/plans.html [Accessed: Aug 7, 2024]

[9]. "AWS Pricing Calculator", *Amazon Web Services (AWS),* [Online], Available: https://calculator.aws/#/ , [Accessed: Aug 1, 2024]