Master of Applied Computer Science

CSCI 5409

Advanced Cloud Computing

Term Assignment Report (Individual)

B00924759

1.1 Problem

Many individuals seek to dispose items they no longer require or wish to replace with newer ones. They commonly leave such items on the streets for others to take or organize a garage sale. However, many genuine and interested buyers are unaware of these garage sales or may not frequent the areas where the items of interest are located. For example, someone seeking a used chair living in the West end of Halifax may be unaware of a garage sale or an office chair placed on a street in the South end.

1.2 Problems with existing solutions

The existing solutions to this problem include platforms such as Facebook Marketplace and Kijiji, which allow individuals to post advertisements for the sale of used items online. However, these solutions are not without their issues. For instance, Facebook Marketplace may lack adequate filters, making it challenging to narrow down searches based on streets/lanes or specific product categories. Additionally, Kijiji can be cumbersome to use, particularly for those who are not tech-savvy, which may deter potential buyers or sellers. Finally, both platforms have no guarantee of the reliability of the sellers, which may lead to scams or frauds.

1.3 Proposed Solution

To address this issue, a proposed solution is to develop an online platform (**Yardsale.com**), i.e., a website, that connects potential sellers and buyers of used items. The proposed website would aim to provide a simple and user-friendly platform for buyers and sellers to connect with ease. Users would be able to find items of interest quickly and efficiently by using a search bar instead of complex filters. This approach would make it easier for non-tech-savvy individuals to navigate the platform and find the items they are looking for.

2 Features

The proposed website, Yardsale.com, would offer users a range of features to facilitate the buying and selling of used items. These features include user registration, user notifications, user login, post filtering, and post creation.

1. User Registration -

User registration on Yardsale.com is a straightforward process that requires users to enter their name, a valid email address, and a password to create an account. Once registered, users can login to create and view listings for items they wish to sell or purchase.

2. User Notifications -

Upon registration, users will receive an email asking if they want to subscribe to notifications from Yardsale.com for new listings. Users can choose to unsubscribe at any time by clicking the unsubscribe link in the notification email.

3. User Login -

To access the website's features, users can log in using their registered email address and password. Once logged in, users can view existing listings and create their own posts to sell items.

4. Filter posts -

The website's search feature includes a search bar that allows users to enter keywords for the item they are looking for. The website will dynamically list items based on the search query, enabling users to filter posts effectively.

5. Create posts -

To create a listing, logged-in users can add an item description, specify the number of units for sale, offer a selling price, and provide an address for pickup. Additionally, users can upload an image of the item to assist potential buyers in making informed purchase decisions.

3.1 Implementation – Why Cloud?

Yardsale.com has been developed and deployed using the services provided by Amazon Web Services (AWS) Cloud Services. The decision to use cloud services was made to avoid the need for physical servers and hardware infrastructure, which can be costly to purchase and maintain.

One of the key advantages of using AWS Cloud Services is the flexibility it provides in managing traffic and data. Yardsale.com is likely to experience varying levels of traffic and data, and cloud services allow for scalability and elasticity to meet these fluctuations in demand. This approach ensures that the website can handle large volumes of traffic without any performance issues.

Furthermore, cloud services offer a higher level of security and reliability than traditional hosting solutions. AWS has robust security measures in place to protect against data breaches and other security threats, and they provide reliable uptime guarantees to ensure that the website is always available to users.

3.2 Services Used

Compute

1. AWS EC2

2. AWS Elastic Beanstalk (EBS)

Storage

1. AWS S3

2. AWS DynamoDB

Network

1. AWS CloudFront

General

1. AWS Secrets Manager

2. AWS Simple Notification Service (SNS)

3.2.1 EC2

EC2 is an Infrastructure as a Service (IaaS) that enables users to provision virtual machines (VMs) in the cloud environment. It lets the service consumers have complete control over the operating system, application software, define security groups and set ACLs as per requirement. Also, it allows easy integration with other services like AWS CloudFront for faster content delivery.

The frontend of the website is developed using ReactJS which is a JavaScript library that is widely used for building user interfaces for web applications.

The alternatives to EC2 and their drawbacks are

EBS – Setting up the environment to run the front-end requires a lot of configurations and adding/managing files which is easier in EC2 than EBS. Also, it is logical to put processing capabilities on EBS as it provides auto scaling.

Lambda - Since EC2 instances are dedicated to your application, they can provide better performance compared to shared resources in Lambda.

I have used the following configurations to host the frontend in AWS EC2:

i. *Amazon Machine Image*

The Amazon Linux 2023 (**ami-06e46074ae430fba6**) 64-bit Linux system with an x86 instruction set was chosen to host the frontend. Node Package Manager (npm) and AWS CLI are being installed on the instance using Linux commands through the 'user data' option in the EC2 cloud formation YAML.

ii. *Instance ID*

The website frontend has multiple dependencies that are specified in the package.json file. npm is used to install all the required packages and modules, which are then downloaded and extracted during the installation process. The installation process consumes a considerable amount of processing power and memory due to the large size of dependencies and modules.

To address this, the **t2.large** instance with 2v CPUs and 8GiB of memory was chosen over the t2.micro and t2.small instances, which proved to be too slow in running npm install and npm start. SSHing into the created t2.small and t2.mirco instances also encountered issues such as getting stuck indefinitely.

iii. Security Group

The security group was configured to allow **SSH** (port 22) and **Custom port (3000)** access from anywhere (CIDR notation 0.0.0.0/0). Access to other ports has been restricted to improve the security of the instance.

3.2.2 Elastic Beanstalk (EBS)

EBS simplifies the process of deploying and managing web applications. It supports multiple programming languages and frameworks, including Java, .NET, Node.js, PHP, Python, Ruby, and Go. Developers can specify the platform of their choice, upload the code and EBS takes care of deploying, scaling, and monitoring the application.

The backend of the code is developed using Node.js. EBS offers 3 Node.js platforms, namely Node.js 18, Node.js 16, and Node.js 14, all running on 64-bit Amazon Linux 2. By leveraging EBS, node applications can be deployed without the need to install npm and applications such as **NginX** for reverse proxying. Additionally, EBS supports automatic load balancing through the creation of new instances, ensuring efficient handling of requests. Environment variables can be easily configured for an EBS environment, thus eliminating the need to maintain a separate .env file.

Alternatives to EBS and their drawbacks -

In comparison to EC2 or Lambda, EBS provides preconfigured environments, automatic load balancing, and simplified application deployment, making it a better option for hosting the backend services of Yardsale as the backend is where all the heavy-duty processing and database operations are performed.

EC2 - The automatic scaling feature of EBS helps in efficiently handling many requests by creating new instances. In contrast, EC2 instances need to be manually configured for load balancing.

Lambda - EBS allows easy configuration of environment variables, eliminating the need for maintaining a .env file, which is not directly supported in Lambda.

I have used the following configurations to host the backend in AWS EBS :-

i. *Platform* – **Node.js**

ii. *Platform Branch* – **64bit Amazon Linux 2 v5.8.0 running Node.js 16**

iii. *Environment variables* - aws_access_key_id , aws_secret_access_key, aws_session_token, secret_name, s3_bucket_name.

iv. *Load Balancing* - Enabled

3.2.3 Simple Storage Service (S3)

Amazon Simple Storage Service (S3) is a cloud-based storage service that allows users to store and retrieve data at anytime from anywhere on the web.

Yardsale allows users to upload images of the items they wish to sell. It is important to keep the images secure from unauthorized access and ensure quick access to avoid latency while loading the website. As S3 allows storing images, configure Access Control Lists (ACLs) and can be easily integrated with Content Delivery Network (CDN) services like CloudFront it is chosen to save images uploaded by users.

Alternatives to S3 and their drawbacks -

DynamoDB – It's a NoSQL database. Images need to be stored as image buffers. Difficult to integrate with CDNs to deliver images as compared to S3.

AWS AppSync and Neptune - While both services can be used to store and retrieve data, they are not optimized for storing and serving images

AWS Athena – It allows SQL querying of data stored in S3. Less efficient compared to storing images in S3 using a unique key and fetching the key saved in a No-SQL database to render images.

I have used the following configuration to store images in S3 –

i. *Bucket Name* – yardsale-post-images

ii. *Enable ACLs* – Yes

iii. *Encryption Key* - **Amazon S3 managed keys** (SSE-S3)

3.2.4 DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service provided by AWS. It is highly available and automatically scales up or down based on the demand and capacity.

User and Post details for Yardsale are saved into DynamoDB tables. Users enter details through forms on the frontend. This data is sent to the backend as key-value pairs in a JSON format through HTTP requests. As the input is JSON, it can be easily pushed into a No-SQL document database like DynamoDB without the need of writing complex queries.

Alternatives to DynamoDB and their drawbacks are -

AWS Neptune - Not suitable for storing user and post details as they are graph databases that are designed to handle complex relationships between data points.

AWS Athena - Athena is primarily used for querying data stored in S3 using SQL.

RDS - User and posts details do not involve complex queries for retrieval. Also, SQL databases are not suitable for frequent read/write operations.

I have used the following configuration for DynamoDB -

i. *Table Name* - users, posts

ii. *Hash Key* -

For 'users' table - 'email'

For 'posts' table - 'post_id'

iii. *ReadCapacityUnits* - 5

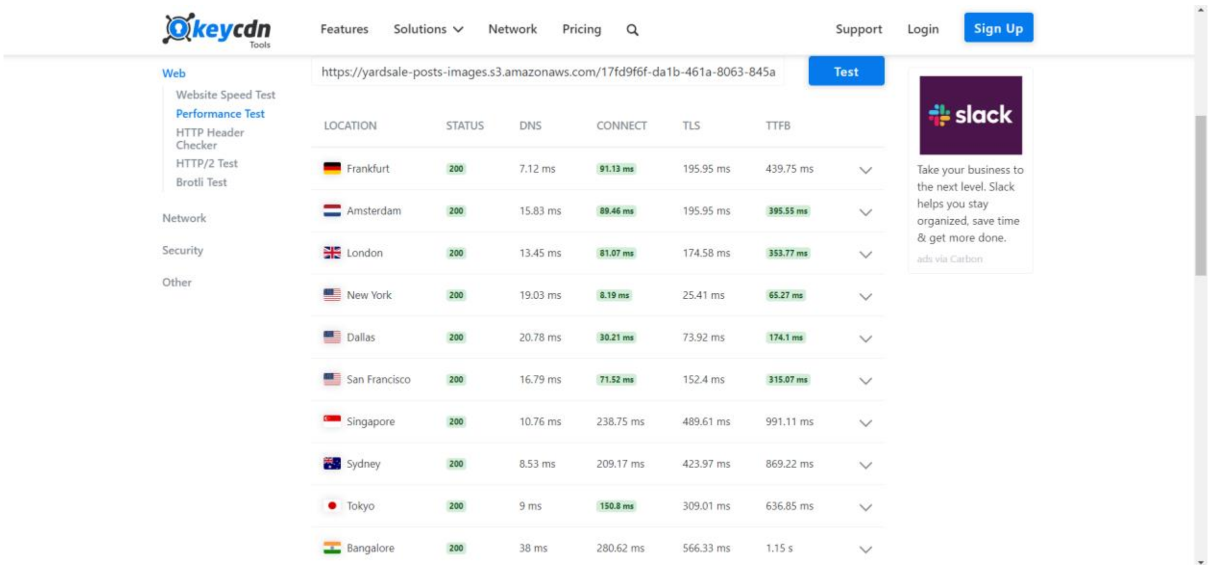
iv. *WriteCapacityUnits* - 5

3.2.5 CloudFront

Amazon CloudFront is a content delivery network (CDN) service offered by AWS that allows businesses and developers to distribute content to end-users with low latency, high transfer speeds, and high availability. It also offers security features such as SSL/TLS encryption and access control via AWS Identity and Access Management (IAM).

The main feature of Yardsale is to lists items that are sold online. It is crucial to lists all the posts with images from the S3 bucket as quick as possible on the webpage. A content delivery network like CloudFront caches content at edge locations around the world, making it easily accessible to end-users from their nearest location, resulting in faster delivery and improved user experience. CloudFront can be easily integrated with S3.

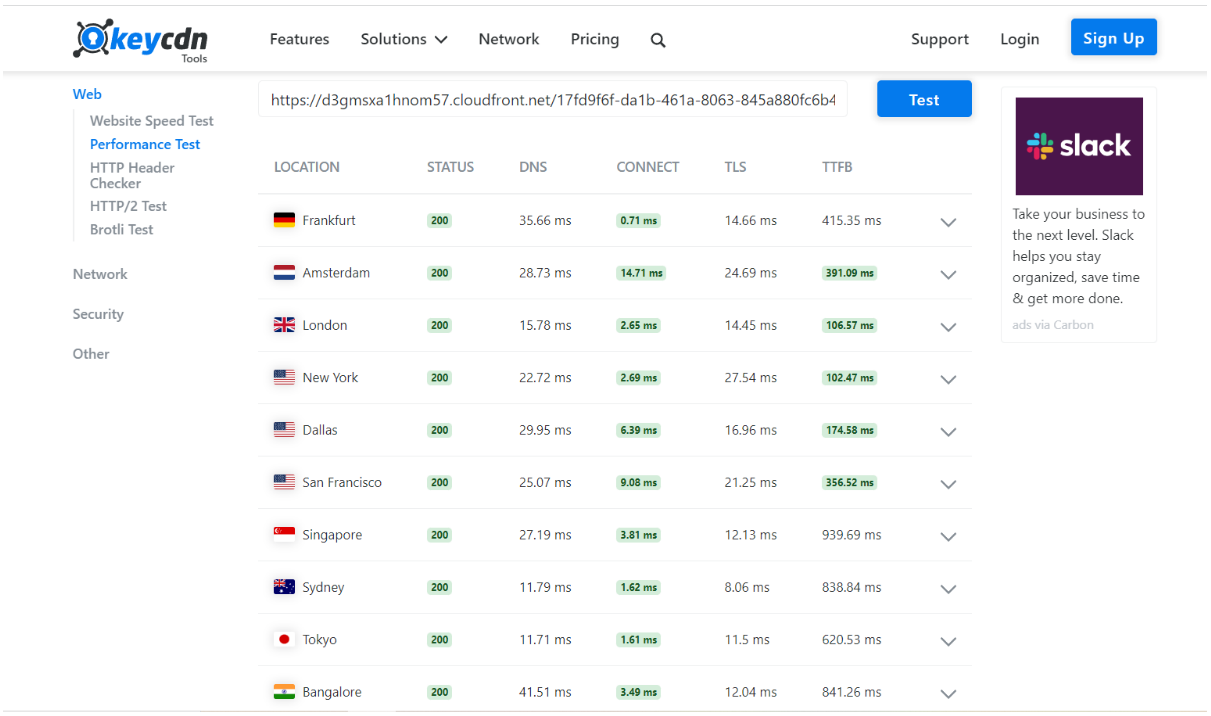
The below images show the time required to fetch an S3 resource using the S3 URL and CloudFront URL.



https://yardsale-posts-images.s3.amazonaws.com/17fd9f6f-da1b-461a-8063-845a

LOCATION	STATUS	DNS	CONNECT	TLS	TTFB
Frankfurt	200	7.12 ms	91.13 ms	195.95 ms	439.75 ms
Amsterdam	200	15.83 ms	89.46 ms	195.95 ms	395.55 ms
London	200	13.45 ms	81.07 ms	174.58 ms	353.77 ms
New York	200	19.03 ms	8.19 ms	25.41 ms	65.27 ms
Dallas	200	20.78 ms	30.21 ms	73.92 ms	174.1 ms
San Francisco	200	16.79 ms	71.52 ms	152.4 ms	315.07 ms
Singapore	200	10.76 ms	238.75 ms	489.61 ms	991.11 ms
Sydney	200	8.53 ms	209.17 ms	423.97 ms	869.22 ms
Tokyo	200	9 ms	150.8 ms	309.01 ms	636.85 ms
Bangalore	200	38 ms	280.62 ms	566.33 ms	1.15 s

Figure 1: Fetch times of object using S3 object



https://d3gmsxa1hnom57.cloudfront.net/17fd9f6f-da1b-461a-8063-845a880fc6b4

LOCATION	STATUS	DNS	CONNECT	TLS	TTFB
Frankfurt	200	35.66 ms	0.71 ms	14.66 ms	415.35 ms
Amsterdam	200	28.73 ms	14.71 ms	24.69 ms	391.09 ms
London	200	15.78 ms	2.65 ms	14.45 ms	106.57 ms
New York	200	22.72 ms	2.69 ms	27.54 ms	102.47 ms
Dallas	200	29.95 ms	6.39 ms	16.96 ms	174.58 ms
San Francisco	200	25.07 ms	9.08 ms	21.25 ms	356.52 ms
Singapore	200	27.19 ms	3.81 ms	12.13 ms	939.69 ms
Sydney	200	11.79 ms	1.62 ms	8.06 ms	838.84 ms
Tokyo	200	11.71 ms	1.61 ms	11.5 ms	620.53 ms
Bangalore	200	41.51 ms	3.49 ms	12.04 ms	841.26 ms

Figure 2: Fetch time of object using CloudFront distribution

Table 1: Tabulating observations of KeyCDN

Location	Time (S3 URL)	Time (CloudFront URL)
San Francisco (USA)	315.07 milli seconds	102.47 milli seconds
London (UK)	353.77 milli seconds	106.67 milli seconds
Sydney (AUS)	869.22 milli seconds	838.84 milli seconds
Bangalore (IND)	1.15 seconds	841.26 milli seconds

It can be concluded that CloudFront reduces latency by many folds hence is the most efficient way to fetch images from S3 bucket for the website.

The following configuration has been used for CloudFront –

i. *Price Class* - Use all edge locations (best performance)

ii. *Supported HTTP versions* - **HTTP/2, HTTP/1.1, HTTP/1.0**

iii. *Origins* - S3 bucket

3.2.6 Simple Notification Service (SNS)

SNS enables the creation and publication of messages that can be delivered to a variety of subscribers, such as email addresses, SMS text messages, and HTTP/HTTPS endpoints.

Yardsale uses AWS SNS to allow users to subscribe to email notifications for new item listings on the website. When a user subscribes, their email address is added to the subscription list. When a new item is listed, AWS SNS publishes an email notification to all subscribers on the list. User can unsubscribe any-time by clicking the 'unsubscribe' link.

Alternatives to SNS and their drawbacks

SES – SNS allows creation and management of topic to which multiple subscribes can be added. While SES can also be used to send emails to multiple recipients, it requires managing recipient lists manually, which can be time-consuming and error prone.

The following configuration has been used for SNS –

i. *Topic Name* – yardsale-notification

ii. *Protocol* – email

3.2.7 Secrets Manager

AWS Secrets Manager is a fully managed service that enables users to store and retrieve secrets such as database credentials, API keys, and other sensitive information.

Creating services like SNS, CloudFormation, EBS, EC2 and S3 require AWS credentials. These resources would be configured and billed to the account linked with the AWS credentials, thus making it crucial to keep all the credentials and resource details confidential. Yardsale would keep these credentials with AWS Secrets Manager. Additionally, Secrets Manager offers features such as automatic rotation of secrets and integration with AWS services, making it a convenient and secure option for managing credentials.

Alternative to Secrets Manager and its drawbacks

AWS Parameter Store - Parameter Store supports a limited number of secret types, including strings, string lists, and secure strings. It does not offer automatic rotations.

4. Deployment Model

The '**Public Cloud**' deployment model has been chosen to deploy Yardsale website. In a public cloud, cloud resources are owned and managed by a cloud service provider like AWS, Azure or Google. These resources are provided to the people for a cost over the internet.

There are various reasons why a public cloud has been chosen as a deployment model. Firstly, it eliminates the need to manage costly and time-consuming on-premises hardware. Second, public cloud providers offer high scalability and elasticity to ensure that the infrastructure can be scaled up or down as per demand and cloud consumers pay only for what is used. Thirdly, a public cloud provides access to a wide range of services and features that can be used to enhance the functionality and performance of the website like AWS S3, Secrets Manager, Elastic Beanstalk etc. Finally, public cloud model provides high reliability and availability, with the ability to distribute resources across multiple regions and availability zones, ensuring that the website can continue to operate even in the event of a failure in one location.

5. Delivery Model

The cloud delivery model for Yardsale website is **Software-as-a-Service (SaaS).** The website provides a service to its users where they can create an account, view posts from other users, and create their own postings. Yardsale website is accessible through the internet and users do not have to install any software or maintain any hardware to access the service. The website is hosted on AWS, and users can access the service using their web browsers. Moreover, users do not have the ability to customize the software or integrate it with other systems as Yardsale does not provide any APIs or integrations.

Overall, the lack of user control over the infrastructure and software stack and its lack of customization options makes it a clear SaaS delivery model.

Individual services used in the project have their own delivery models. Table below lists all the services used in the project and their delivery models.

Table 2: Services and their delivery model

Service	Delivery Model
EC2	Infrastructure as a Service
EBS	Infrastructure as a Service
S3	Platform as a Service
DynamoDB	Database as a Service
CloudFront	Platform as a Service
Secrets Manager	Software as a Service
SNS	Software as a Service

6. Final system architecture

The system is deployed to cloud using "**Infrastructure as Code**" through AWS CloudFormation. This involves creating a YAML file that describes the resources, their dependencies, and their configurations. CloudFormation then utilizes this YAML file to construct and deploy the web application.

Figure 3 depicts how the services interact with each other. The front-end on EC2 communicates with the backend on EBS created, load balanced EC2 instances via HTTPS requests utilizing the Axios library. These EC2 instances retrieve credentials stored as secrets from the Secrets Manager service and access the S3 bucket to store image data. Images from the S3 bucket are delivered to the front-end through AWS's CloudFront content delivery network. User and post details are transmitted from the front-end to the back-end in JSON format. These JSON documents are inserted into tables created on DynamoDB. Values from the DynamoDB tables are retrieved, processed, and returned to the front-end as HTTPs responses with appropriate status codes. The EC2 instances created by EBS retrieve the AWS SNS topic ARN to subscribe users to the SNS topic or to publish email to the subscribers.

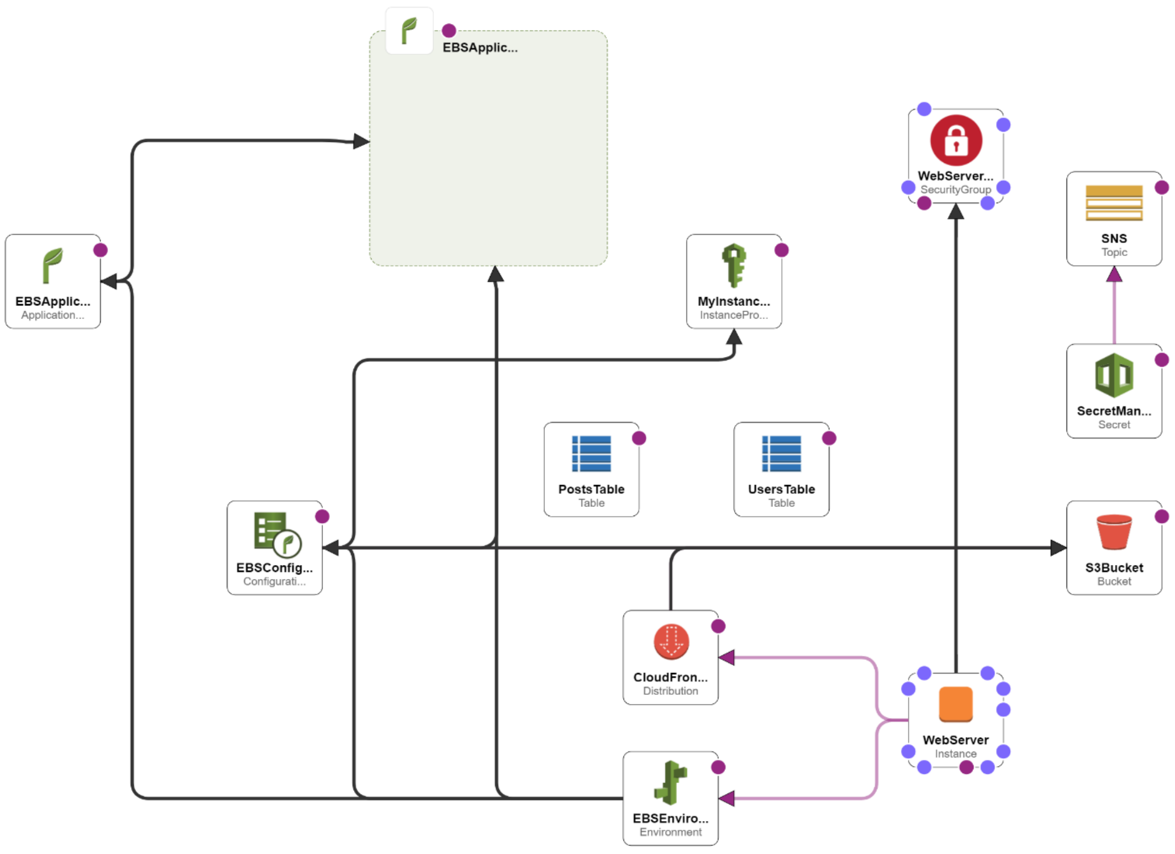


Figure 3: Interaction between the AWS services used for Yardsale

The system architecture depicted in **Figure 4** shows the final design of the web application. To handle the varying levels of traffic that the website may receive, we use the "**Service Load Balancing Architecture**". This architecture distributes incoming network traffic across multiple compute resources to improve the availability, scalability, and fault tolerance of the application.

Elastic Beanstalk is used to create a load balancer and an auto scaling group that contains similar EC2 instances. This allows the system to scale out or scale in depending on the network traffic.

By using this service load balancing architecture, the system can handle increasing levels of traffic without any downtime. As the load increases, Elastic Beanstalk automatically adds more EC2 instances to the auto scaling group to handle the extra traffic. This ensures that the website remains available and responsive to users even during peak traffic periods.

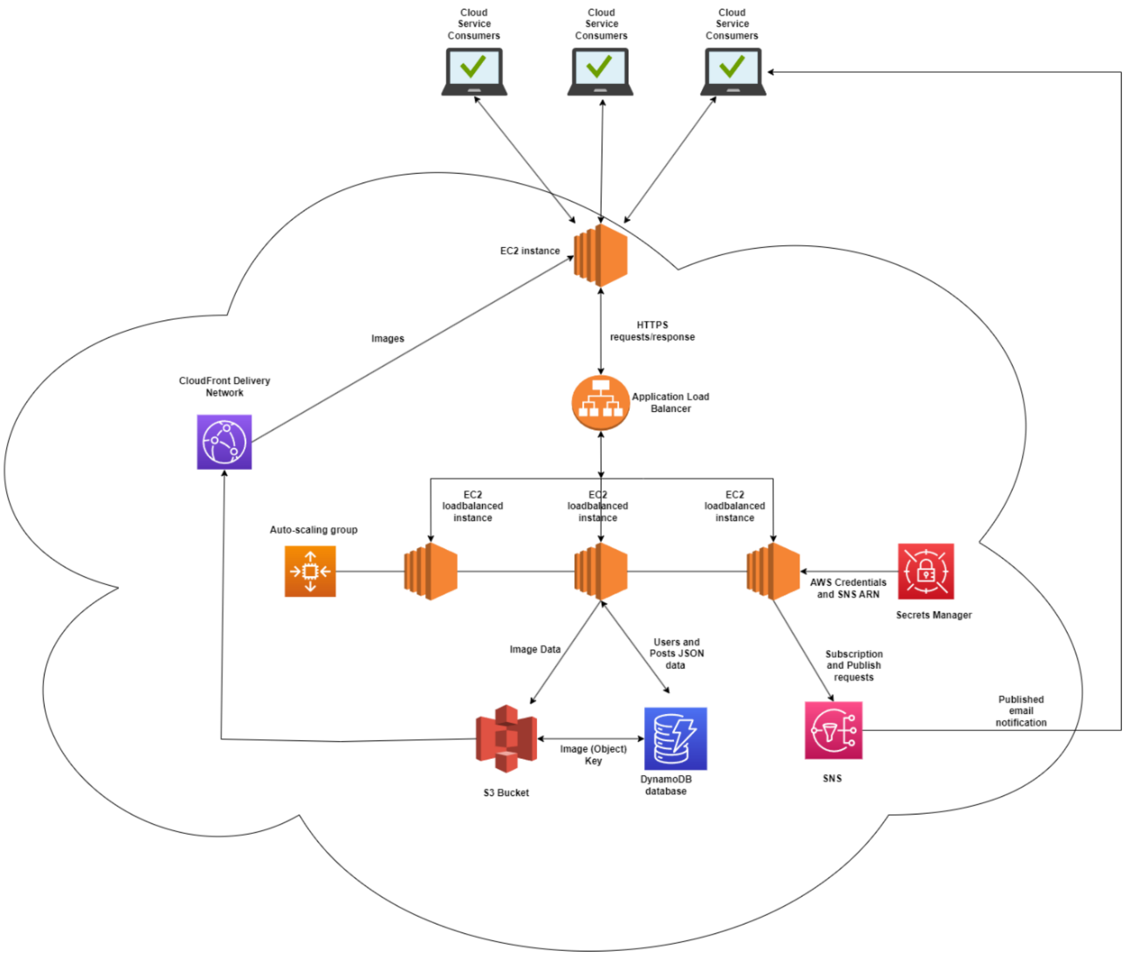

Cloud Service Consumers
Cloud Service Consumers
Cloud Service Consumers
EC2 instance
HTTPS requests/response
Images
CloudFront Delivery Network
Application Load Balancer
EC2 loadbalanced instance
EC2 loadbalanced instance
EC2 loadbalanced instance
Auto-scaling group
AWS Credentials and SNS ARN
Secrets Manager
Image Data
Users and Posts JSON data
Subscription and Publish requests
Published email notification
Image (Object) Key
S3 Bucket
DynamoDB database
SNS

Figure 4: System Architecture of Yardsale

7. Security Analysis

At the network level, a security group has been configured to allow access only on the required ports, thereby creating a virtual firewall. This restricts unauthorized access to the EC2 instance, enhancing the security of our application.

To secure the transfer of user data during transit, HTTPS (HTTP Secure) protocol has been used. This ensures that sensitive information like user email and passwords cannot be intercepted by attackers using network sniffing tools like Wireshark. Additionally, the Bcrypt library is used to hash user passwords with 10 rounds of salt before storing them in the database table. This ensures that even if an attacker gains access to the database, they cannot easily obtain user passwords.

The EBS backend is not exposed to users of the website, which further limits the exposure of our application to potential attackers. The EBS backend is also reverse proxied using NGINX to listen on port 80 and not any other custom port. This limits the scope of potential attacks as attackers may not be able to target specific ports to exploit vulnerabilities.

The application uses an API call to AWS secrets manager to access AWS credentials and the topic ARN for SNS, which has a list of subscribed email addresses. We do not save this sensitive information in a .env file that can be compromised. This ensures that AWS credentials and other sensitive information are not exposed to potential attackers.

For storage access control, an S3 bucket is used to save images. This bucket can only be accessed by the bucket owner and uses Amazon S3 managed keys (SSE-S3). This restricts access to the S3 bucket, reducing the possibility of data breaches.

Overall, our application architecture employs several security mechanisms at different layers to ensure that data is secure. While these mechanisms significantly enhance the security of our application, it is still vulnerable to some threats. For instance, if an attacker manages to gain access to the EC2 instance, they may be able to bypass the virtual firewall and access sensitive data. To address this vulnerability, we could enhance the security of our EC2 instance by implementing additional access controls such as multi-factor authentication, intrusion detection, and prevention systems, and log monitoring. Additionally, we could implement data encryption at rest to ensure that even if an attacker gains access to our storage, they cannot read or modify the data.

8. Cost Analysis

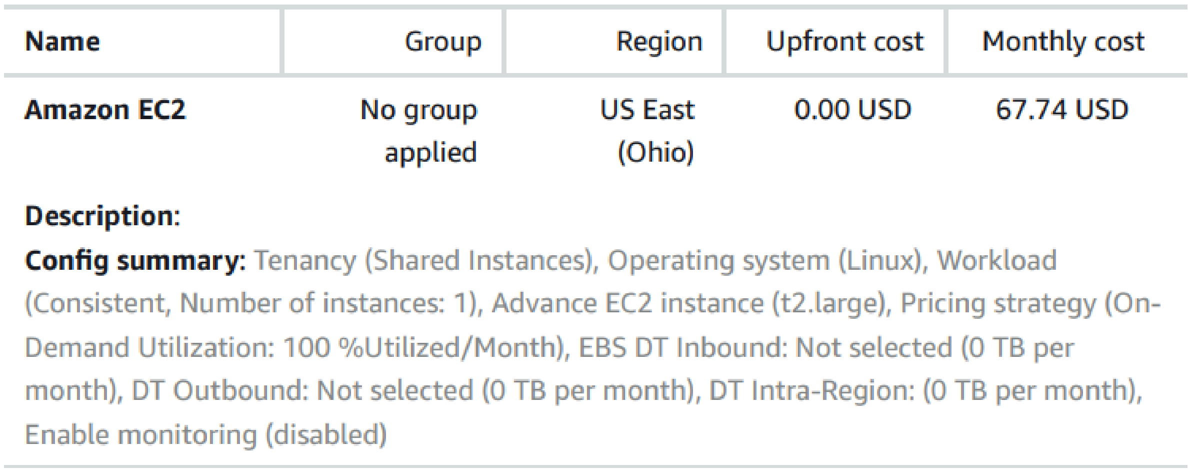
<u>On Going Costs on AWS</u>

The on-going cost of running the application is calculated using AWS Calculator. The detailed report is available at -
https://calculator.aws/#/estimate?id=129c0f496dcab72a468b48c9dab8252027c69e3f

EC2 (Frontend)

Specifications – Shared instance, Linux O.S., 1 t2.large instance, on-demand pricing option for maximum flexibility.

Monthly price – USD 67.74



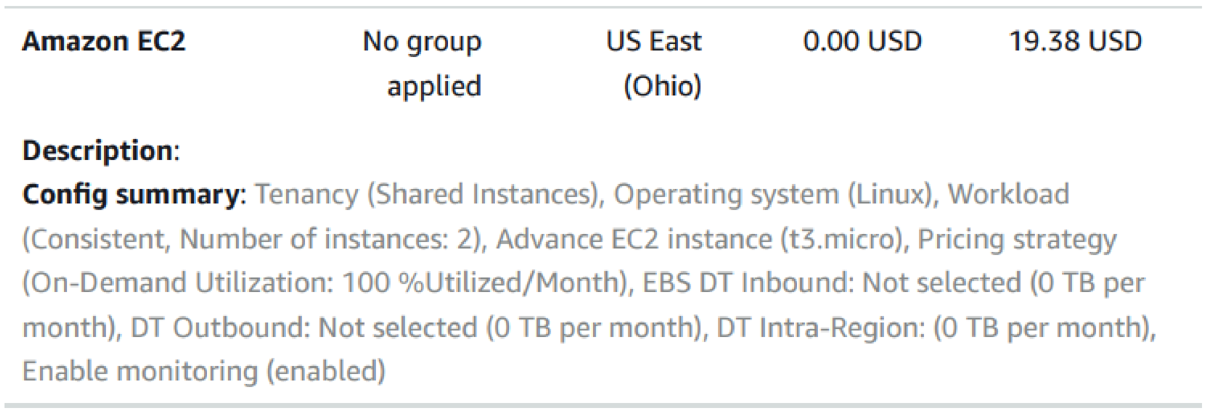
Name	Group	Region	Upfront cost	Monthly cost
Amazon EC2	No group applied	US East (Ohio)	0.00 USD	67.74 USD

Description:
Config summary: Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1), Advance EC2 instance (t2.large), Pricing strategy (On-Demand Utilization: 100 %Utilized/Month), EBS DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month), Enable monitoring (disabled)

EC2 (EBS created instance for Backend)

Specifications – Shared instance, Linux O.S., 2 instances for load balancing, on-demand pricing option for maximum flexibility**.**

Monthly Price – USD 19.38



Amazon EC2	No group applied	US East (Ohio)	0.00 USD	19.38 USD

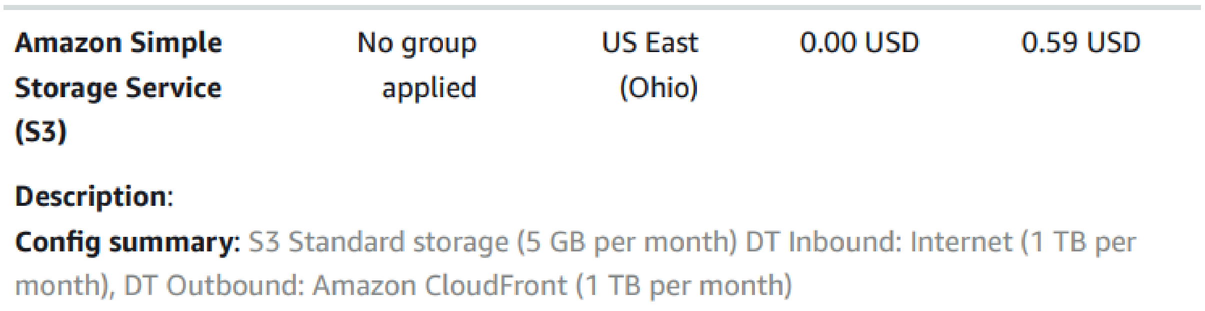
Description:
Config summary: Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 2), Advance EC2 instance (t3.micro), Pricing strategy (On-Demand Utilization: 100 %Utilized/Month), EBS DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month), Enable monitoring (enabled)

S3 (Storing Images)

Amazon S3 storage usage is calculated in binary gigabytes (GB), where 1 GB is 2^{30} bytes. This unit of measurement is also known as a gibibyte (GiB), defined by the International Electrotechnical Commission (IEC). Similarly, 1 TB is 2^{40} bytes, i.e. 1024 GBs.

Specifications – Standard storage, 1Mb average object size, 5GB of data stored per month, 1TB inbound and 1TB outbound traffic via CloudFront.

Monthly Cost – USD 0.59



Amazon Simple Storage Service (S3)	No group applied	US East (Ohio)	0.00 USD	0.59 USD

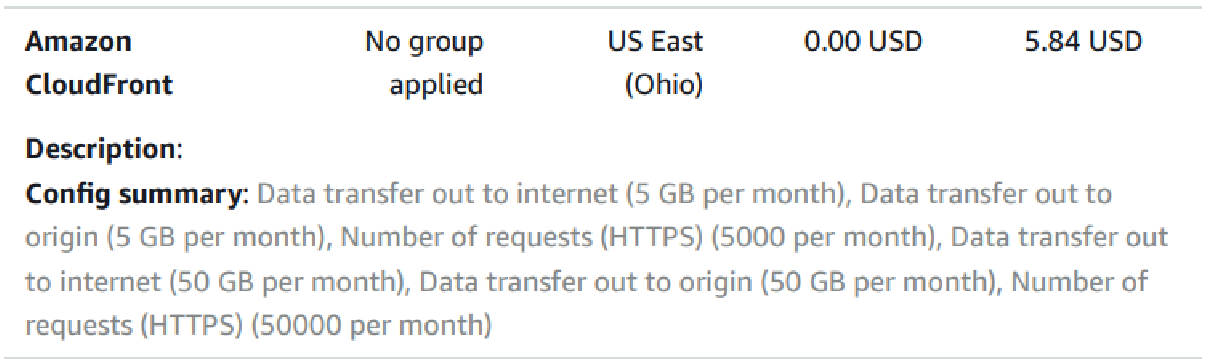
Description:
Config summary: S3 Standard storage (5 GB per month) DT Inbound: Internet (1 TB per month), DT Outbound: Amazon CloudFront (1 TB per month)

CloudFront (Retrieving Images)

Effective December 1, 2021, the first 1TB per month of data transfer out to the internet and 10 million http/https requests from all Edge locations will be free (excludes CloudFront China).

Specifications – Data transfer out to origin (50GB per month), 50000 HTTPS requests per month for Canada and Data transfer out to origin (5GB per month), 5000 HTTPS requests per month.

Monthly Price – USD 5.84



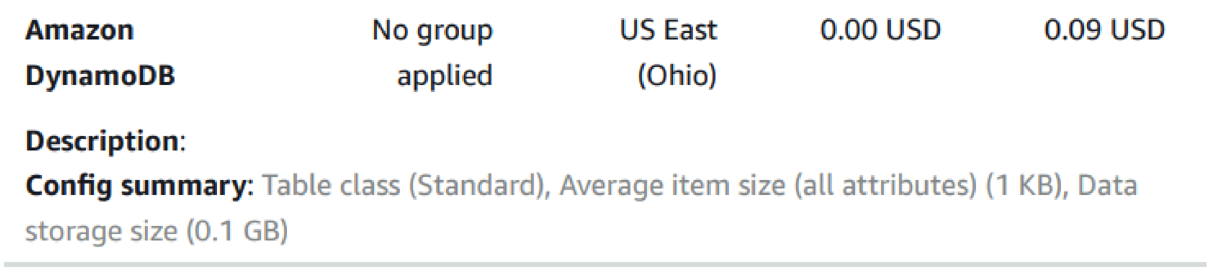
Amazon CloudFront	No group applied	US East (Ohio)	0.00 USD	5.84 USD

Description:
Config summary: Data transfer out to internet (5 GB per month), Data transfer out to origin (5 GB per month), Number of requests (HTTPS) (5000 per month), Data transfer out to internet (50 GB per month), Data transfer out to origin (50 GB per month), Number of requests (HTTPS) (50000 per month)

DynamoDB (Storing user and post details)

Specifications – Standard table class, average item size of all attributes 1KB, Data storage size (0.1 GB)

Monthly Price – USD 0.09 i.e., USD 0.18 for 2 tables.



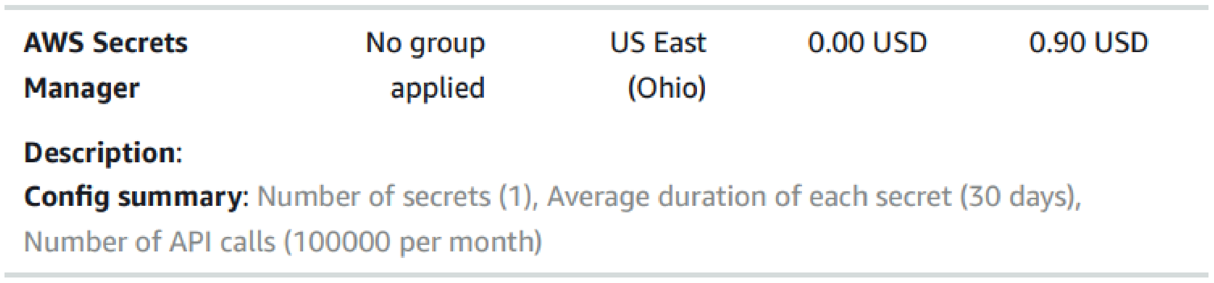
Amazon DynamoDB	No group applied	US East (Ohio)	0.00 USD	0.09 USD

Description:
Config summary: Table class (Standard), Average item size (all attributes) (1 KB), Data storage size (0.1 GB)

Secrets Manager (Storing secrets)

Specifications – 1 secret, about 30 days of secret duration, 100,000 requests per month.

Monthly Cost – USD 0.90



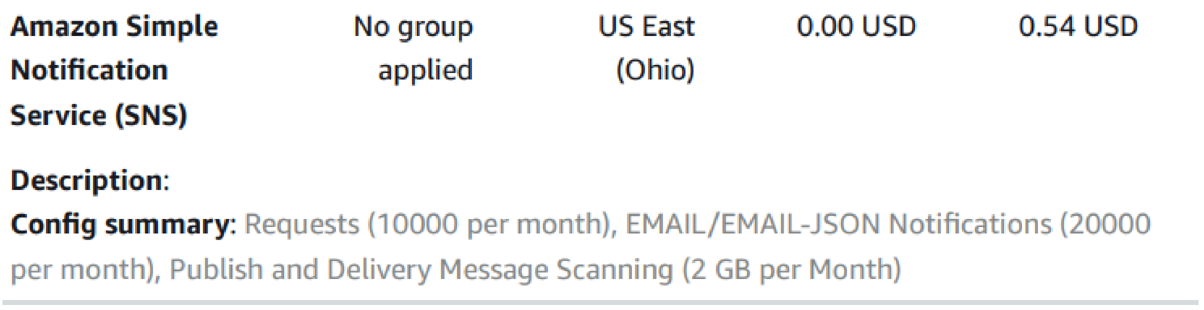
AWS Secrets Manager	No group applied	US East (Ohio)	0.00 USD	0.90 USD

Description:
Config summary: Number of secrets (1), Average duration of each secret (30 days), Number of API calls (100000 per month)

Simple Notification Service

Specifications – 10000 request per month, 20000 email notifications and 2GB per month.

Monthly Cost - USD 0.54

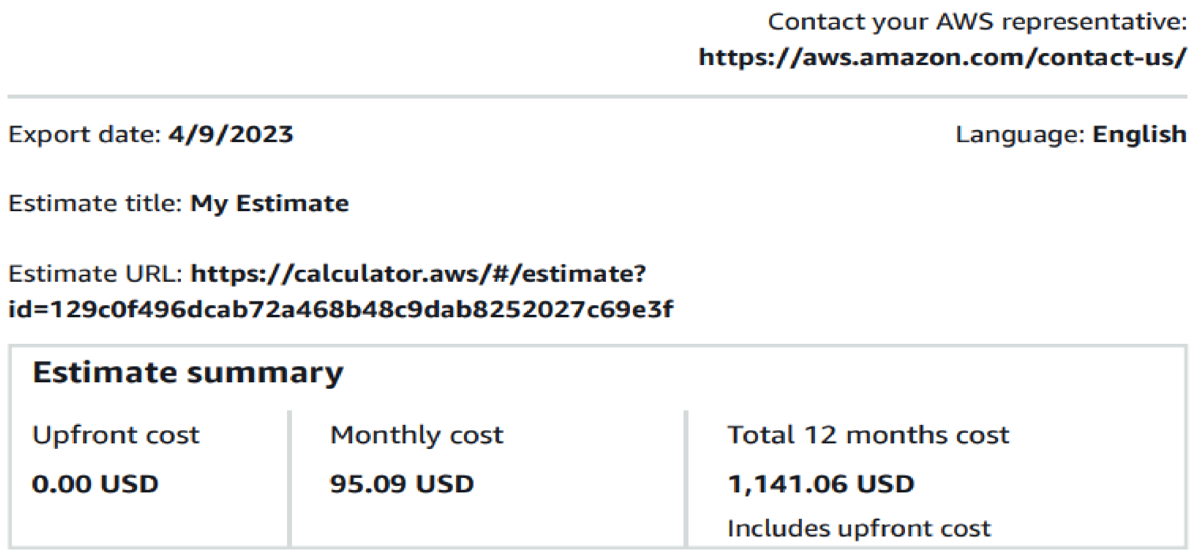


Amazon Simple Notification Service (SNS)	No group applied	US East (Ohio)	0.00 USD	0.54 USD

Description:
Config summary: Requests (10000 per month), EMAIL/EMAIL-JSON Notifications (20000 per month), Publish and Delivery Message Scanning (2 GB per Month)

Total Ongoing Cost

The total monthly cost for running the application is **USD 95.09** and yearly cost is USD **1141.06**



Contact your AWS representative:
https://aws.amazon.com/contact-us/

Export date: **4/9/2023**

Language: **English**

Estimate title: **My Estimate**

Estimate URL: **https://calculator.aws/#/estimate?id=129c0f496dcab72a468b48c9dab8252027c69e3f**

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	**95.09 USD**	**1,141.06 USD** Includes upfront cost

8.1 Cost to reproduce the architecture in a private cloud

1.Server to host the front-end

HCLs **PowerEdge T150 Tower Server** with 16GB of RAM, 480GB of SSD SATA ISE, no operating system with C20 RAID and RAID controller. Detailed specs could be found at https://i.dell.com/sites/csdocuments/Product_Docs/en/poweredge-t150-spec-sheet.pdf

Price – USD 1500-1650 which may vary as per specifications.

2.Server to host the backend

HCLs **PowerEdge T350 Single Processor Tower Server** C1 RAID, 480 GB SSD SATA Hard Drive and a 2.8Ghz processor. Detailed specs could be found at https://www.dell.com/en-ca/shop/servers-storage-and-networking/poweredge-t350/spd/poweredge-t350/pe_t350_tm_vi_vp

Price – USD 1883-2400 depending upon specifications.

3.Virtualization Software

VMware vSphere Standard provides an entry solution for basic consolidation of applications to slash hardware costs while accelerating application deployment. Details of the features could be found at https://store-us.vmware.com/vmware-vsphere-standard-5653277300.html

Price – USD 1394.00

4. Database storage to store images and data.

KIOXIA Supermicro SSG-1029P-NES32R provides SSD storage with optimized performance for MongoDB database. The performance testing results could be found at https://www.supermicro.com/solutions/Solution-Brief_Kioxia_in_SMCI.pdf .

Price – approximately USD 2000-2500

5. O.S. Licensing

If using an **open source** O.S. these charges are avoided.

6. Email services

Mailgun is a flexible, scalable, and secure email messaging API. Mailgun's foundation package starts at $35/month and allows 50000 emails in a month, inbound email routing, email address verification access and other features. Further pricing can be found at https://www.mailgun.com/pricing/ .

Price – USD 35/month

7. Maintenance technician

Maintenance technicians are paid around CAD 30 (USD 23.07), according to Glassdoor.com, an hour to perform server related maintenance and other networks.

Wages – USD 3700/month.

8. Physical Infrastructure cost

Webservers need to be kept in a secure location with strict access, temperature, and humidity controls. The average monthly cost to rent a space, secure it with password locks and place thermostats for temperature control. For a small-scale infrastructure in a city like Halifax, the cost estimate would be as follows.

Estimates include property rent, cooling fans, thermostats, password locks, basic furniture. Estimate is based on free quote service provided for, Pennant Point Data Centre, 233 Sambro Creek Road B3V 1L8 Sambro, Nova Scotia, on https://www.datacentermap.com/canada/halifax/pennant-point.html

Rental facility cost – USD 3000-4000/ month

9.Power consumption

As of September 2021, the electricity rates for Nova Scotia Power (NSPower) are approximately 15.616 cents per kilowatt-hour (kWh) for the first 500 kWh of electricity consumed per month, and 18.211 cents per kWh for any additional usage beyond that.

Calculations: -

KIOXIA CD8-V series Mixed-use SSD power consumption – 300W

Monthly power consumption = 0.3 kW * 720 hrs (30 days x 24 hrs) = 216kWh

KIOXIA PM7-V Series 4G SAS Enterprise power consumption - 400W

Monthly power consumption = 0.4 kW * 720 hrs (30 days x 24 hrs) = 288 kWh

KIOXIA Supermicro SSG-1029P-NES32R power consumption – 14W

Monthly power consumption = 0.14 kW * 720 hrs (30 days x 24 hrs) = 100 kWh

Total Monthly power consumed = 604 kWh

Power bill = (First 500 kWh x Electricity rate) + (Additional kWh x Electricity rate)
Power bill = (500 kWh x 15.616 cents/kWh) + (104 kWh x 18.211 cents/kWh)
Power bill = $77.93 + $18.94
Power bill = $96.87

10. Bandwidth
Enterprise bandwidth to host websites or email servers for ISP like **EastLink** in a city like Halifax which includes up to 5 IPs. More details about the Advanced Plus plan could be found at https://business.eastlink.ca/internet

Cost – 409.95/month

Total Cost to re-produce the architecture on a private could be

Up-front Costs = USD 6777 – 7944 (Summing 1,2,3,4,5)

Monthly recurring costs = USD 7241.82 – 8241.82 (Summing 6,7,8,9,10)

8.3 Cost Monitoring

Elastic Beanstalk creates an auto scaling group of EC2 instances to scale out in event of heavy traffic. EC2 instances are the most significant contributor to the cloud costs as they are the primary compute resources responsible for running the web application. Hence, it is crucial to closely monitor the EC2 instances. Cloud monitoring tools such as Amazon CloudWatch can be used to monitor EC2 instances' usage, performance, and costs. Alerts can be setup on CloudWatch for specific metrics, such as CPU utilization or disk space usage.

9. Future Advancements

The following features could be added in future

i. Admin Content Deletion – Admin could delete posts that seem to be fake or based on reports of cheating by customers.

ii. User subscription filters – Users can subscribe to new posts featuring only a certain product of interest and not all posts. This could be implemented by integrating AWS Simple Email Service (SES). User details document in DynamoDB would have another field which would be an array of ‘items of interest’. Whenever a new post is added, code would fetch all users that have the post’s item description in the array and then send an email.

iii. Machine Learning based content moderation – Using AWS Machine Learning, a ML based content moderation system can be integrated to check if the images are in-line with the created posts. This would mitigate fake posts.

References

[1] "AWS documentation." [Online]. Available: https://docs.aws.amazon.com/. [Accessed: 12-Apr-2023].

[2] R. Peterson, "Cloud computing architecture and components," *Guru99*, 17-Mar-2023. [Online]. Available: https://www.guru99.com/architecture-of-cloud-computing.html. [Accessed: 12-Apr-2023].

[3] *Arcitura patterns*. [Online]. Available: https://patterns.arcitura.com/cloud-computing-patterns/design_patterns/service_load_balancing. [Accessed: 12-Apr-2023].

[4] "Welcome to LWC communities!," *Welcome to LWC Communities!* [Online]. Available: https://www.awsacademy.com/. [Accessed: 12-Apr-2023].