DAL-COLLAB

By

JEMS PATEL

Contents

Project Introduction:

DalCollab is a web application designed specifically for students at Dalhousie University to upload and share their projects. The platform facilitates collaboration and knowledge sharing among students by allowing them to view projects posted by their peers. The primary users of this application are Dalhousie University students who are engaged in various academic and extracurricular projects. The performance target for this application includes fast response times for uploading and retrieving projects, secure storage and transfer of data, and high availability to ensure students can access the platform at any time.

- **Primary Users: T**he primary users are students at Dalhousie University, engaged in various academic and extracurricular projects.
- **Performance Targets:**
    - **Fast Response Times:** Ensuring quick uploads and retrieval of projects to provide a seamless user experience.
    - **Secure Data Storage and Transfer:** Protecting user data and project information through secure storage solutions and encrypted data transfer.
    - **High Availability:** Ensuring the platform is always accessible to accommodate the varied schedules of students.

AWS Services:

Compute:

- **EC2 (Elastic Compute Cloud)**: The frontends and backends of the application were constructed using EC2 instances [8] because they are highly customizable, scalable and can be easily integrated into other AWS services. We choose EC2 instances [8] to host our frontend (Next.js) and backend (Spring Boot) applications; this is made possible by the fact that they offer the flexibility required to configure the environment precisely to cater for specific application requirements.
    - **Alternative: AWS Elastic BeanStalk**
      While it automates deployment, scaling, and monitoring of applications thereby making their management easier [13], AWS Elastic Beanstalk has less control over the environment than EC2 [8]. The flexibility of EC2 allows us to configure the environment precisely to meet the application's requirements. It offers scalability, enabling us to adjust the computing power based on traffic and load, ensuring consistent performance. Consequently, we prefer finer control and integration capabilities provided by EC2.

- **Lambda**: The Lambda [10] function was used for handling peculiar tasks like sending notifications etc. Lambda functions can perform certain duties such as sending a notification whenever there is a new project or account creation. This serverless approach is cost-effective and scalable since it adjusts with workload accordingly such that charges only run when the function is executed.
    - **Alternative: EC2**
      Running these tasks on EC2 [8] involves managing server instances or containers, which increases operational complexity and cost. While these alternatives offer more control, Lambda's serverless model provides simplicity and cost-effectiveness, making it the preferred choice for handling discrete, event-driven tasks.

Storage:

- **RDS (Relational Database Service)**: RDS [9] is used to host the MySQL database that keeps project data, user accounts and other related information. As a result of this, operational overhead is reduced since it deals with the management of databases such as backups, updates and scaling while we concentrate on application development.
    - **Alternative: DynamoDB**
      Amazon DynamoDB [16] is a fully managed NoSQL database service that provides fast and predictable performance. However, it may not be ideal for applications requiring complex queries and transactional operations, which are better supported by MySQL in RDS. RDS offers a balance of performance, cost, and functionality, making it the preferred choice for this project.

Network:

- **API Gateway**: Amazon API Gateway [11] is selected to create, publish, maintain, monitor, and secure the API endpoints. API Gateway [11] is used to create and manage the API endpoints for the application. It serves as the entry point for all client requests routing them to appropriate backend services such as EC2 instances and Lambda functions. The gateway ensures that they are secure scalable APIs which can be maintained easily.
    - **Alternative: Lambda Function URLs**
      While AWS Lambda Function URLs can be used to create HTTP endpoints for Lambda functions, they lack the full feature set of APIS Gateway, such as throttling, API lifecycle management, and detailed monitoring. API Gateway ensures that the API is secure, scalable, and easy to maintain, making it the better choice for managing API endpoints.

General:

- **SNS (Simple Notification Service)**: SNS [12] is the service that can be used to send emails on projects upload or when a student creates an account. This helps to keep the users informed of the important actions such as these promptly thereby improving user engagement and experience.
    - **Alternative: Amazon SES**
      On the other hand, Amazon SES [15] could have been chosen for email sending, however, SNS was preferred because of its easy integration with simple notification services. SNS [12] ensures that users are promptly informed of important actions, enhancing user engagement and experience.

- **Secret Manager**: Secrets Manager [19] is a great platform for storing credentials needed by RDS databases. It securely manages sensitive information, such as database passwords, ensuring that the credentials are not hardcoded in the application code and are protected from unauthorized access.
    - **Alternative: AWS Parameter Store**
      However, while AWS Parameter Store [17] can store configuration data and secrets as well, some of the additional features provided by Secrets Manager include automatic rotation of credentials and tighter integration with other AWS services. Thus, making it more appropriate to use Secrets Manager for secure management of sensitive information.

Deployment Model:

Therefore, I decided to use AWS's public cloud for DalCollab because it has a well-developed service market and has an elastic platform. When the frontend and backend are installed in EC2 instances [8], it is likely to make optimizations depending on working conditions. Amazon RDS [9] functions as a MySQL database server, providing features such as automated database backup, user-pooling and automated database deletion or resizing, when necessary, which reduces management and administrative time used. These serve as backends that take care of tasks like notifications hence being cost effective and scalable. Additionally, API Gateway supports the API operations through safe and flexible access to business backends. For one, AWS has multiple data centres worldwide that improve application accessibility and performance while AWS has a transparent pricing structure that facilitates making applications cost-effective. Generally, integration with all AWS features as well as the variety of services available will meet the requirements of DalCollab for efficient implementation and later administration of this software package.

Delivery Model:

EC2 [8]: Infrastructure as a service

AWS Lambda [10]: Function as a service

RDS [9]: Platform as a service

API Gateway [11]: Platform as a service

SNS [12]: Software as a service

Secret Manager [19]: Software as a service

For DalCollab, we utilized a range of AWS service models to optimize deployment and management of the application.

- **EC2 (Elastic Compute Cloud):** Used as Infrastructure as a Service (IaaS), EC2 has adequate computing capabilities to host a frontend app (Next.js) and a backend app (Spring Boot). This choice allows one to freely set up the environment and modulate the amount of traffic which is processed to maintain constant quality. [8]
- **AWS Lambda:** Acting as Function as a Service (FaaS), Lambda [10] is used to perform such functions as sending notifications. This serverless model supports cost-optimisation and is elastic, only charging the execution time thus cutting down on operational costs.

- **RDS (Relational Database Service):** When offered as Platform as a Service (PaaS), RDS [9] takes the responsibility for MySQL database, its backups, updates and scaling. The stated way significantly minimizes the operational pressure that accompanies database management while drastically enabling the enhancement of development and performance.
- **API Gateway:** Also, a Platform as a Service (PaaS), the API Gateway [11], enables the formation of API endpoints, its administration and protection. It provides a clean and efficient interface between the client requests and the backend services when dealing with the authentication, routing of the requests and scaling.
- **SNS (Simple Notification Service):** Employed as Software as a Service (SaaS), SNS [12] is used for sending email notifications to users. It simplifies the process of managing notifications and ensures users are informed of important updates without requiring infrastructure management.
- **Secrets Manager:** Deployed as Software as a Service (SaaS), Secrets Manager [19] securely manages sensitive information such as database credentials. It protects against unauthorized access and reduces the complexity of handling secrets in application code.

Thus, the choice of these service models as IaaS, FaaS, PaaS, and SaaS allows DalCollab to use AWS's opportunities optimally and minimize time and resource consumption, as well as proposing a secure, efficient, and cost-effective solution to our users.

Architecture:

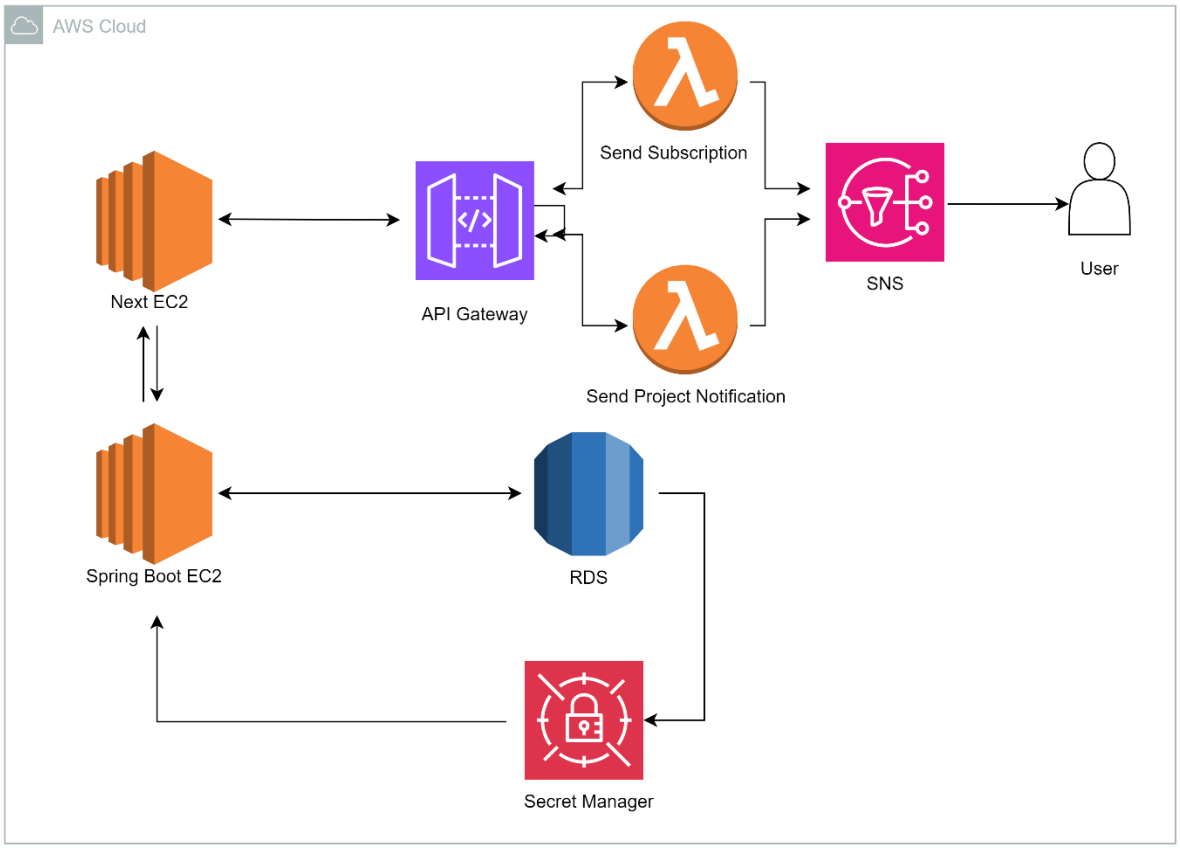

Figure. Architecture Diagram for the SERVICE HUB. [7]

Architecture Overview:

- EC2: Dalcollab's frontend (react.js) and backend (springboot) is hosted on the ec2 with the help of the docker. [8]
- RDS: RDS is used for storing all the data in the MySQL database. [9]
- API Gateway: Api gateway routes the API request to the lambda. [11]
- Lambda: Handle the logic of sending the email to the users. [10]
- Secret Manager: It created for storing the credentials for the RDS. [19]
- SNS: Its main purpose is to send the notifications to the users.[12]

Flow of the cloud architecture:

- User can access the web application from the EC2 [8] in which our both frontend and backend is hosted.
- After that, user will see the dashboard of the dalcollab in which user can do register and login.

- When user do register, at that time the lambda function for sending the subscription request is sent to user regarding subscribe the channel.
- Once user do register and login, user will see the dashboard of the dalcollab in which user can see all the projects that are listed by other students.
- User can also add the project of themselves by giving all the details for that along with the facility to edit and delete the project.
- User can filter the services as per the technology type of the project.
- In this whenever the new project is added by any user, it will trigger another lambda function and send the notifications to all other users regarding that by giving all the descrition.

Data Storage:

Most of the structure data for DalCollab is kept in the Amazon RDS which has MySQL database database for user details projects and its associations metadata. Moreover, Lambda functions might use temporary data in memory at run-time and any storage requirements of any kind are met through RDS. For the ones that require notification and sensitive data for example 9redentials, then Amazon SNS [12] and Secrets Manager are used respectively for notification and secure management. Such a setup helps to minimize loss off data as it organizes data storage and management within the application space effectively and with high security.

Programming Languages:

- **Frontend (Client): Next.js**
  For DalCollab, Java was used for backend development with Spring Boot, chosen for its robustness and scalability in building enterprise-level applications.
- **Backend (Server): Spring Boot**
  JavaScript (with TypeScript) was used for the frontend with Next.js, providing an interactive and dynamic user interface.
- **Lambda Functions: Python**
  Python was utilized for Lambda functions, chosen for its simplicity and efficiency in handling serverless tasks such as notifications and other backend processes. This combination ensures a well-rounded, scalable, and efficient application.

How is the system deployed to the cloud?

In order to deploy the system to the cloud for the first time, it can be achieved by using AWS cloud formation which automates the provisioning of the resources by running he cloud formation file and create all the resources which is defined in the cloud formation.

Other than that, in order to deploy any changes that are made in the frontend and backend and want to get that updated version then just need to build the image of the

backend and frontend with the newly updated code. After that just need to reconnect the EC2 ssh and update the newly image so it will automatically make changes.

For updating the lambda function, just make sure that to deploy the new lambda function after making changes.

Data Security:

- **JWT Authentication:**
  Dalcollab Uses a JWT authentication [18] to secure user interactions with the application. It's basically using a token-based authentication as Upon successful login, users receive a JWT that contains encoded user information and claims. This token is used for subsequent API requests, ensuring that each request is authenticated without the need for re-entering credentials. It is stateless as server do not worry about storing it and it will manage It in the front-end side.
- **Secret Manager:**
  This application stores the credentials of the database in the secret manager [19] for the security purpose. This will ensure the security of the user's data. It securely manages sensitive information, such as database passwords, ensuring that the credentials are not hardcoded in the application code and are protected from unauthorized access.

Vulnerability:

Despite having the strong security, it has some vulnerabilities as in this application it is storing the jwt token in the local storage so there are some chances for exposing the token to the user and will affect by the unauthorised user and attack.

In this right now, this application is not encrypting the password to store in the database, so it has increased the risk to get attacked by unauthorised user.

This application is not fully serverless as it has only two lambda function and all other API are running on the spring boot. So now its tightly coupled with each other and its better to do all the whole application server less on the lambda function.

Cost Estimation:

Private Cloud Cost Estimation:

- **Compute Resources**
  - **Server**: Purchase mid-range servers which is equivalent to the t2.m Gb micro ec2 instance or servers with the same specifications. Servers with the 4 vCPUs and 2 GB RAM would be sufficient.
  - **Estimated Cost**: Estimate $858 per server [1]
- **Serverless Functions**
  - We can use the tool like OpenFaas and Apache open whisk which is open-source platform for building the serverless functions.
  - **Estimated Cost:** Approx $200 [2]
- **Database Storage**
  - **Server:** Servers with the good Memory, CPU and storage in order to handle the RDS instance capacity. For this, we need approx 1 TB SSD or HDD.
  - **Estimation Cost:** Approx $90 [3]
- **API Management**
  - **Server and Storage:** Need mid- range servers which is equivalent to the 4 vCPU and 8 GB RAM.
  - **Estimated Cost:** $849 per year [4]
- **Notifications**
  - To replace the SNS, we can use the tools like Apache Kafka and Postfix/Dovecot.
  - **Estimated Cost:** It could cost around $1000-$1400 annually depending on the traffic. [5]
- **Secret Manager**
  - To use the tool like the secret manger on primes, we can use the tool HarshiCorp Vault.
  - **Server:** Need additional server for handling the HarshiCorp Valult.
  - **Estimated Cost:** Approx $300-$400 [6]

Total On-Premises Costs:

- **Compute Resources: $858**
- **Serverless Functions: $200**
- **Database Storage: $90**
- **API Management: $849**
- **Notifications: $1000-$1400**
- **Secret Manager: $300-$400**

Total Estimated On-Primises Cost: $6000-$6500 (Initial Setup)

Costly Cloud Mechanism:

- In the dalcollab, EC2 Instance is used for hosting the frontend and backend in which cost can increase as per the increasing the traffic for requesting the web applications functionality.
- RDS is used here for storing the MYSQL database for the project data, user accounts related information. In this it will increase as per the continuous uptime, storage requirements and backup storage.
- In the API Gateway, Charges depend on the number of the API requests and data transfer. So, the cost will increase if the request is increased.
- In the lambda function, cost is based on the number of invocations and execution time.
- In the SNS that I have used for sending the notifications costs are depending on the number of subscriptions which is increased as per the user creation.
- In the secret manager, the cost is very minimal as now it has only stored a RDS credentials.
- By comparing all the services, **EC2** [8] has the highest potential for escalating the costs as it depends on the higher usage, addition instances or updated instance types.

Future Improvements:

In the dalcollab, many enhancements can be made to make it very good and capable website. First, by adding the functionality of participating in each other's project, it will not only increase the relation between the students but also will get the information of the different projects which will increase the knowledge of them.

In the future, it will be great if it has a chat feature in which students can talk with each other and discuss some topics which enhance the knowledge and information. Having this on the website, they can share the problems while developing the projects to each other and they can get the help from another students.

It would be very beneficial if the whole application is serverless so it will be able to handle the loads that are coming on the website. By doing this this application will become the scalable, secure and fully serverless.

References:

[1] ServerMall, Server HPE DL360 Gen10, [Online]. Available: https://servermall.com/catalog/servers/server-hpe-dl360-gen10/. [Accessed August 06 2024]

[2] OpenFaaS, GKE Multi-Stage, [Online]. Available: https://www.openfaas.com/blog/gke-multi-stage/. [Accessed August 06 2024]

[3] Newegg, 1TB SSD, [Online]. Available: https://www.newegg.ca/p/pl?d=1tb+ssd. [Accessed August 07 2024]

[4] CAST AI, Traefik vs NGINX, [Online]. Available: https://cast.ai/blog/traefik-vs-nginx/#:~:text=NGINX%20pricing&text=A%20commercial%20solution%20like%20NGINX,also%20comes%20with%20professional%20assistance.. [Accessed 2024]

[5] DZone, What Apache Kafka Costs, [Online]. Available: https://dzone.com/articles/what-apache-kafka-costs-pricing-out-open-source-di. [Accessed August 06 2024]

[6] HashiCorp, Vault Pricing, [Online]. Available: https://www.hashicorp.com/products/vault/pricing. [Accessed August 05 2024]

[7] Draw.io, [Online]. Available: https://www.drawio.com/. [Accessed August 07 2024]

[8] Amazon, Amazon EC2 Concepts, [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html. [Accessed August 01 2024]

[9] Amazon, Amazon RDS, [Online]. Available: https://aws.amazon.com/rds/. [Accessed August 02 2024]

[10] Wikipedia, AWS Lambda, [Online]. Available: https://en.wikipedia.org/wiki/AWS_Lambda. [Accessed August 03 2024]

[11] Amazon, API Gateway, [Online]. Available: https://aws.amazon.com/api-gateway/. [Accessed August 03 2024]

[12] Amazon, Amazon SNS, [Online]. Available: https://docs.aws.amazon.com/sns/. [Accessed August 02 2024]

[13] Amazon, Elastic Beanstalk, [Online]. Available: https://docs.aws.amazon.com/elastic-beanstalk/. [Accessed August 05 2024]

[14] Amazon, Lambda URL Configuration, [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/urls-configuration.html. [Accessed August 06 2024]

[15] Amazon, SES, [Online]. Available: https://docs.aws.amazon.com/ses/. [Accessed August 06 2024]

[16] Amazon, DynamoDB, [Online]. Available: https://docs.aws.amazon.com/dynamodb/. [Accessed August 06 2024]

[17] Amazon, Systems Manager Parameter Store, [Online]. Available: https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html. [Accessed August 06 2024]

[18] Medium, Implement JWT Authentication in Spring Boot, [Online]. Available: https://medium.com/@tericcabrel/implement-jwt-authentication-in-a-spring-boot-3-application-5839e4fd8fac. [Accessed August 06 2024]

[19] Medium, AWS Secret Manager, [Online]. Available: https://abiabi0707.medium.com/aws-secret-manager-f9963a6b4be0. [Accessed August 02 2024]