Cloud-Based E-Commerce Web Application

HUNG, Ching Hei (20858764)	LAU, Ho Yin (20852629)	YU, Yu Tao (20852588)

Introduction

Microservices is an architecture that structures an application as a collection of two or more services that are independently deployable and loosely coupled. It can help improve fault isolation and cost efficiency. This project aims to build a microservice web application based on cloud environment using AWS cloud platform. Since designing the web application is not the most important part, this report will mainly focus on the implementation of web pages through microservices and the usage of cloud related components. In this project, we take role as a startup e-commerce company, we will explore AWS and figure out what is the best solution for us to provide our web services to users according to our requirements.

Methodology

First Approach

We utilized Amazon Elastic Container Service (IaaS) with Fargate to deploy our application. ECS is a fully managed container orchestration service that helps efficiently deploy, manage, and scale containerized applications [1]. While Fargate is a serverless computing engine that allows us to focus on building applications without managing servers [2].

However, our team does not have a very good sense of managing infrastructure. Using ECS is beneficial as it allows managing complex microservices, but it also requires significant expertise and ongoing maintenance. For just hosting a database server, it costs our team about 3 USD per day, and considering our budget, assuming we only have 50 USD for hosting one month (AWS Lab quota), it is very costly, and our team should reconsider our plan.

Final Decision

Considering cost efficiency and the scale of our project, we decided to use EC2, API Gateway, Lambda and RDS (mainly FaaS) provided by AWS.

Auto-scaling

Considering our app is a small-scale project and the usage of the app is unknown as a startup service provider, using those services is a better approach. Those services provide automatic

scaling based on demand. Therefore, we do not need to worry about handling the sudden increases in traffic which is under-provisioning and causing our services to crash.

Cost Efficiency

Unlike traditional servers or ECS, we do not need to be charged when there's no activity. This is very beneficial for our startup budget. Using those services, we do not need to worry about over-provisioning as we only pay for the compute time used. And after deployment, it shows satisfactory result that it only costs about 0.3 USD per day to host everything which is 10 times less expensive than using ECS just to host the database server alone.

Minimal Management

As a startup, it is urgent to publish a usable and stable app. In the starting phase, using those services allows us to focus on implementing app features instead of managing architecture. As our business grows, we can still transition to ECS so that we can apply more complex architecture.

Implementation

System Architecture

Microservice architecture is an architectural and organizational approach to software development where software is composed of small independent services that communicate over APIs [3]. Each service in the web application is independently deployed.

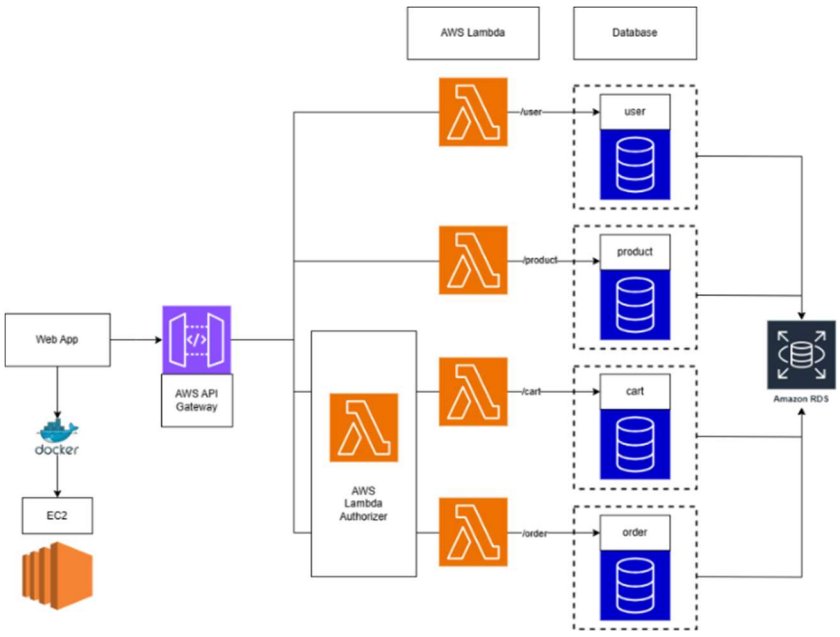

Figure 1: The microservice architecture for the project

Cloud-related Components

During the implementation, several cloud services are utilized to deploy the project, the implementation architecture is shown in Figure 1, detailed explanations are below:

Amazon API Gateway

We used Amazon API Gateway to manage our APIs. It allows us to integrate user authentication and authorization into APIs easily. One of the impressive features is that it allows us to deploy API in different stages such as development, user acceptance test and production. This enhances our development efficiency as we do not need to create different environments for different stages by ourselves.

Amazon Lambda

We used Amazon Lambda to implement microservices, allowing us to run our code without the need to manage servers. This approach offers several benefits: it automatically scales to handle varying traffic, reduces costs by charging only for the compute time used, and simplifies deployment since we can easily update functions. Additionally, Lambda integrates well with other AWS services, making it a powerful choice for building serverless applications.

Amazon Relational Database Service (RDS)

We utilized RDS to deploy our database. It offers several key benefits: it automates routine tasks like backups and patching, allowing us to focus on development. RDS is easily scalable and optimizes performance through various instance types.

Amazon Elastic Compute Cloud (EC2) & Docker Container

We utilized Docker to encapsulate the frontend service of our application. The Docker container has high portability as it packages all the required dependencies, which ensures the application can run consistently across different environments [4]. Then, EC2 is used to create a cost-effective instance to run the docker container.

Application Overview

Front-end Service

The front-end service, deployed using EC2 and docker, consists of several functions, which include user registration, login, product search bar, product pages, shopping cart and sending order. Through the API, the front end can transmit requests to the back-end server and receive corresponding responses in JSON format. By using React, we built a web page as a user interface for users to interact with the functions mentioned above. During the implementation,

CSS file is leveraged for the style customization of the web page.

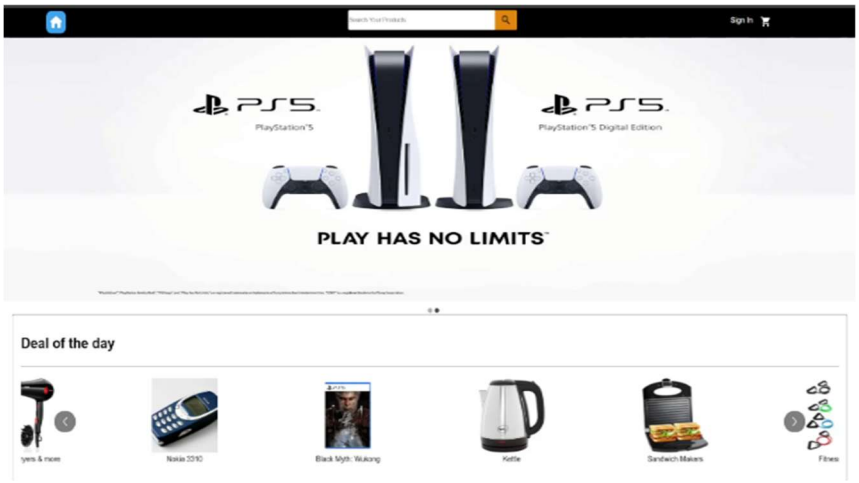

Figure 2: Main page of the web page

Back-end Server

To provide the front-end services, we deployed a back-end server and database. AWS API gateway, Lambda and RDS are used to deploy and build the server and database.

User service

It provides 2 request calls: login, and user registration using email, username and password. The server will send a token to the front-end service every time the user login and the token will be used for accessing services such as shopping carts and ordering.

Product service

This is responsible for fetching product details, such as image URL, and description from the product database. The result will be sent to the front-end service in JSON format for further processing on the web page, such as product pages and search bar.

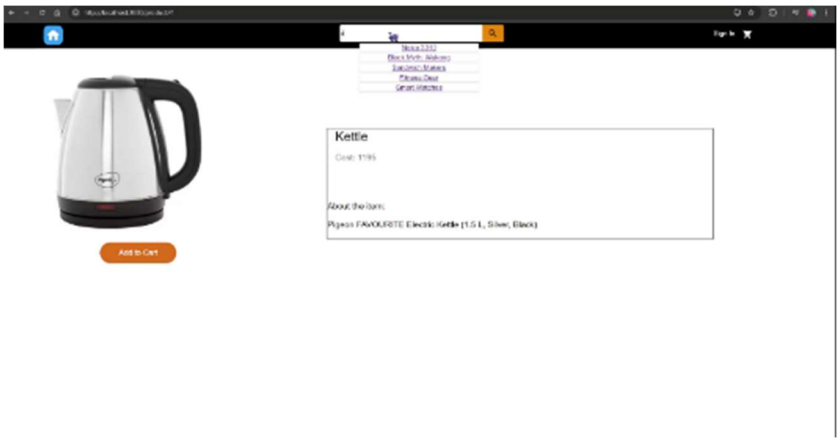

Figure 3: Search Bar and Product detail

Cart Service

A shopping cart feature has been implemented. There are three major API functions in cart service: “get cart items”, “add product to cart” and “clean cart”, which allows users to add and view products in their carts by sending requests to the database.

Order service

An order service has been implemented. There are two major API functions: “create order” and “add order items”, which sends the order information from a user to the database..

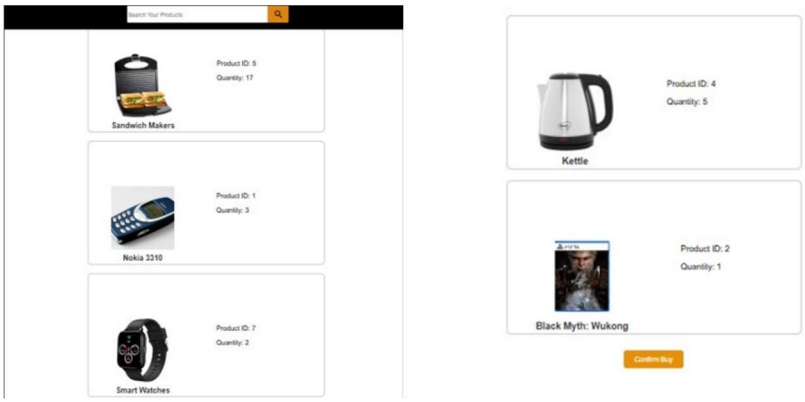

Figure 4: Order detail page

Authorizer

Through AWS API Gateway, we can implement an authorizer for registered users to access the cart service and order service. We implemented a lambda authorizer which can verify token provided by users to verify if they are registered and to prevent identity theft.

Conclusion

In conclusion, cloud components mainly focus on deploying the database and back-end server to implement stable and efficient web applications. By leveraging AWS EC2, API Gateway, Lambda and RDS, we have reduced the cost, ensured the portability of the application and made it easy to deploy in different environments.

Workload

HUNG, Ching Hei	Frontend service and website building
YU, Yu Tao	Frontend and Backend for cart and order
LAU, Ho Yin	Backend Development and Deployment

References

[1] AWS, “Amazon ECS - Run containerized applications in production,” *Amazon Web Services, Inc.* https://aws.amazon.com/ecs/

[2] AWS Fargate - Run containers without having to manage servers or clusters,” *Amazon Web Services, Inc.* https://aws.amazon.com/fargate/

[3] AWS, “What are Microservices?,” *Amazon Web Services, Inc.*, 2019. https://aws.amazon.com/microservices/

[4] AWS ,“What is Docker? | AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/docker/?nc1=h_ls