CSCI 5409 Advanced Topics in Cloud Computing

Project Feasibility Study Report: Tourism app

Instructor: Dr. Saurabh Dey

Date of Submission: 31 January 2020

Submitted By:

Arunachalam Hari (B00832143)

Gamidi Vamsi (B00834696)

Kase Raviteja (B00823644)

Sharma Anuj (B00825885)

DALHOUSIE UNIVERSITY

Faculty of Computer Science

Dalhousie University

Halifax, Nova Scotia

Contents

Project Requirements:

This project involves the development of a secure web application and a mobile application for Canada tourism. The functionalities are the same for both the web application and the mobile application. The key requirements for the project include the home page which contains a search page. Users can search for the national parks, beaches, hill stations, amusement parks, ski resorts, holiday destinations all over Canada. They do not need to login to query the destinations. After selecting a holiday destination, the user will be taken to the booking page. To book tickets, the user will have to create an account and login. A ticket will be generated after the completion of the payment.

The following are the main modules in the project:

- Search
    - Users can search for various places all over Canada. By using the GPS of the mobile in which the application is used, locations will be sorted based on the location.
- Signup
    - For the booking of tickets, the user can create an account using this module.
- Login
    - After creating an account, the user can log in to book the tickets.
- Booking
    - Users can book the tickets after logging into the account.
- Payment
    - The payment module handles the payments through credit/debit cards.
- Ticket Generation
    - A ticket will be generated after the payment is done.

Programming Environment:

For this application, we are going to follow the MVC structure as it will help us build a common backend for mobile and web applications. This is going to consist of:

- MySQL database:
    - We are going with this, as the data for the site is relational and we will be better served by a relational database like MySQL than a NoSQL database like MongoDB. With the kind of data that we are going to handle, MySQL will provide better performance and complete workflow control.
- Python with Flask framework:
    - We need a lightweight framework for building API and Flask is very lightweight and can be deployed in a container. We will use a flask for fetching the search data for the mobile application and web application in JSON format. We chose flask as it provides simplicity, flexibility, scalability, modularity and great control over development.
- Angular:
    - we are going with one of the most popular Frontend frameworks for the web application as it suits our needs. Angular is a feature-rich framework that provides numerous features such as form validations, data binding. Angular also provides many ready-made functionalities/features which we can make use of.

- Android Studio:
    - We are building the android application using the IDE Android Studio because it provides faster deployment for fresh builds, faster development and testing and developer-friendly user interface.
- Java and XML:
    - For the development of android applications in the android studio, we will use XML for user interface and java for controllers and models.
- Visual Studio Code:
    - We are going to use visual studio code as IDE for the development of web application

Cloud Architecture

This application needs to be deployed on the cloud and also utilize the various benefits that the cloud has to offer.

The architecture and services we are planning to use for this deployment are:

- AWS EC2

  The AWS instance where the entire application will be hosted.

- AWS ECS

Amazon EC2 Container Service (Amazon ECS) is a highly scalable, high-performance container management service that supports Docker containers and allows you to run applications easily on a managed cluster of EC2 instances. The ECS service scheduler places tasks onto container instances in the cluster, monitors their performance and health, and restarts failed tasks as needed.[1] Since we are going to be using a docker container, ECS seems like the way to go.

- The ECS has four parts to it and we will be using different AWS services to do them.
    - Elastic container repository (ECR): This is what that will hold our application images.
    - Clusters: The place where the containers are hosted.
    - Tasks: This defines how containers should be created.
    - Services: The containers that will be running and will be auto scaled.
- `Load balancing:
    - We are going to use the network load balancer as it seems to be what AWS is recommending based on the fact that it allows for TCP traffic to pass through without decrypting and encrypting again.[2] This will involve the creation of an Amazon Virtual private cloud (VPC)
- Encryption: We are going to use the free SSL certificate that AWS provides along with simple encryption techniques to encrypt the data as well.

The architecture of the system will either be like figure 1 or figure 2

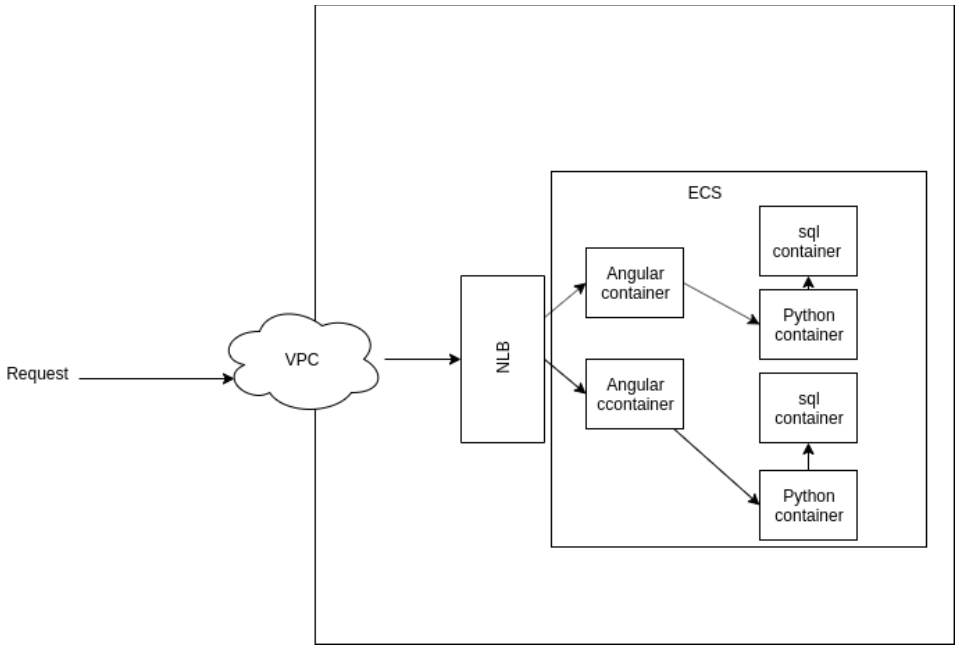

Figure 1: All three servers of the system inside a docker

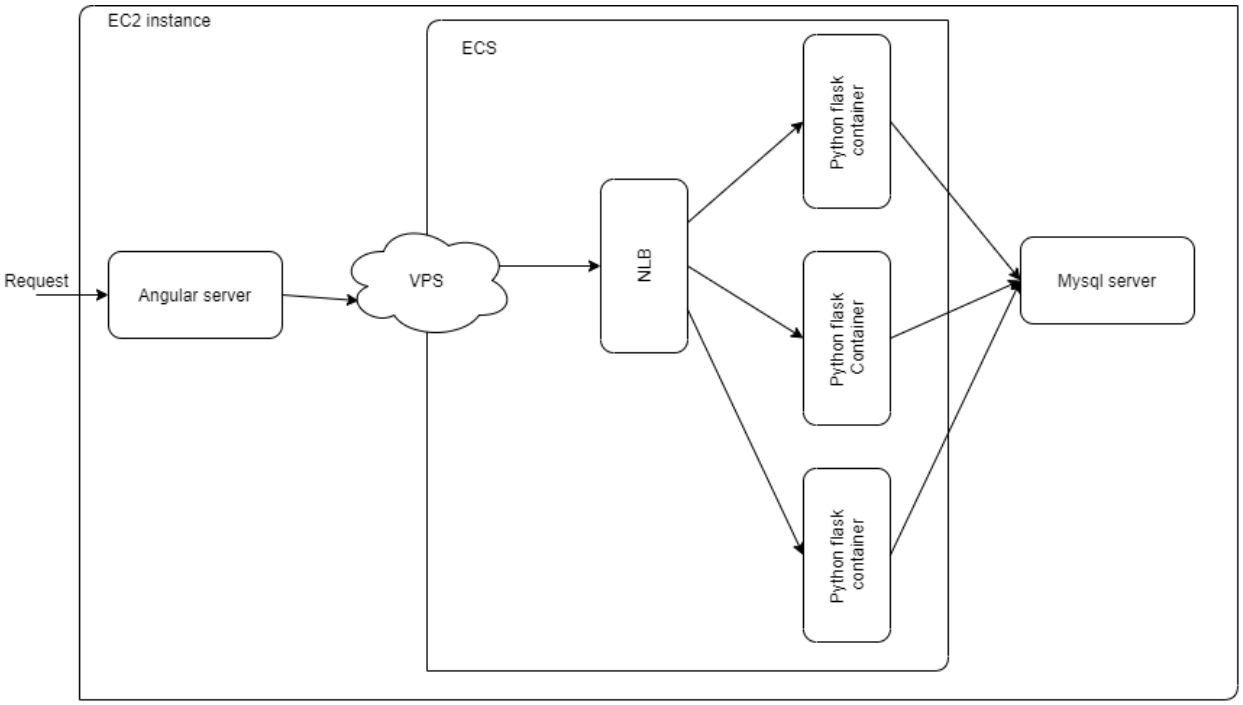

Figure 2: Only the python server inside a docker container

Cost Analysis:

The ECS system comes under the free tier and will not be charged as long as we use the EC2 launch type model [3]. The 200$ credit should be enough to use the EC2 service and complete the project barring any mishaps.

Timelines:

31 Jan- 10th Feb: Developing use cases for the project, setting up the environment (GitLab, Jenkins, AWS), Creating ERDs and implement in MySQL.

10th Feb-20th Feb: Initial Development of User Interface for below functionalities for the website and android application.

- Signup
- Login
- Search
- Booking
- Ticket generation

20th Feb- 5th March: Developing API using python and Flask for below functionalities (APIs will be consumed by both website and android application)

- Search
- Booking
- Ticket generation

5th March-15th March: Consume the APIs built-in User Interface, configuring the AWS to place the modules in dockers.

15th March-20th March: Testing

Note:

Continuous Integration and Continuous Deployment is achieved at each stage.

Project report synchronization is done at each timeline.

Weakness and strengths:

Strengths:

- The team seems to have good coordination, and this will assist us in the completion of the project.
- We are familiar with web and mobile development

Weakness:

- None of us have any experience with the cloud environment, so we can't judge the requirements and needs for the cloud design.

Challenges and Solutions:

These are the possible challenges (and solutions) that have been identified by the team to occur while working on the project:

1.) **Challenge:** Limited credit for using AWS Services.

We have a limited amount of credit ($50 per student) for using cloud services (AWS). Thus, it will be a challenge to complete the project with limited credit.

Solution: One approach to mitigate this issue will be the proper utilization of resources. Each team member must take care of stopping the EC2 instances after using it to avoid credit wastage. Since this is a group project, members can share their resources so that everyone can contribute in case a member exhausts his/her credit for using AWS services.

2.) **Challenge:** Lack of knowledge about Cloud Service.

Cloud computing is a completely new concept for the group members. None of the group members have prior experience in this field [4]. Hence, the team will face challenges while working on this project. They might face difficulties while implementing some of the cloud computing concepts.

Solution: Using online resources can provide some assistance to the team members while working on the project. Other than that, AWS documentation can provide useful information about the services provided by this cloud service provider.

Team Contributions:

- **Arunachalam** Hari: Cloud Architecture, strengths and weaknesses.
- **Gamidi** Vamsi: Project requirements, programming languages and environment.
- **Kase** Raviteja: Project requirements and timeline
- **Sharma** Anuj: Cost analysis, challenges and solutions

References:

1.*AWS Documentation,*https://aws.amazon.com/blogs/compute/using-amazon-efs-to-persist-data-from-amazon-ecs-containers/ [Accessed Jan 28, 2019]

2.*AWS Documentation*,https://aws.amazon.com/blogs/compute/maintaining-transport-layer-security-all-the-way-to-your-container-using-the-network-load-balancer-with-amazon-ecs/ [Accessed Jan 28, 2019]

3. *AWS Documentation*, https://aws.amazon.com/ecs/pricing/ [Accessed Jan 28, 2019]

4.I. Sabo, "5 AWS Limitations every CEO needs to be aware of", Cloud Academy, 2020. [Online]. Available: https://cloudacademy.com/blog/5-aws-limitations-to-be-aware-of/. [Accessed: 30- Jan- 2020].