Final Project

DevOps Tirana

Introduction

You are hired as a DevOps Engineer to deploy a web application in the cloud. The cloud provider that the company uses is AWS. The client is requesting a reliable infrastructure that can scale dynamically, based on the incoming traffic.

What does the application consist of

The client application is a simple online bookstore. It includes the following components:

- A frontend web application that is written in React.js.
- An API that contains the business logic, built in Node.js.
- A Postgres database for persisting the data.

What is already prepared

The code for the whole application, including the database part, is already built for you by the developers. It is pushed to GitHub and it is organized in two directories:

- The **api** directory: Where the backend logic of the application is written.
- The **web_client** directory: Where the frontend logic is written.

Your responsibilities

As a DevOps Engineer, your duties include the following:

- Build the applications and prepare them for execution.
- Create the infrastructure where the applications will be hosted.
- Automate the processes (i.e. the build phase for the application and the infrastructure creation).
- Apply the best practices during your work.

How to build the application code

Both the frontend and the backend of the application are written in JavaScript, so the build commands that you have to use are the same. The developers have already included the build scripts in the corresponding package.json files of the client and api.

The application infrastructure

The client requirements are the following - they want to host the application in a reliable and scalable infrastructure, while minimizing the costs as much as possible.
To achieve this, the following infrastructure is proposed by the system architects.

The client application

Since the client application is written as an SPA (Single Page Application) in React.js, we can build the code and host it as a static website in an S3 bucket. The costs are minimal and we have all the reliability and availability that we require.

The API

The API needs to be scalable. It is suggested to use a container service within AWS. There is no preference in terms of what service we use. We could go with EKS or ECS.

The database

We are using a Postgres database. We want the database to be reliable, available all the time, easily set up, backed up with minimal effort, while also keeping the costs low. To fulfill all the requirements, it is suggested to use a managed service instead of a custom solution. We will go with the Postgres database engine in RDS.

Other tools

Application security and monitoring

All the layers of the applications must be secured (e.g. use security groups, IAM roles and policies, secure the database, etc…)
For the API, you are also advised to use some monitoring mechanisms to inspect the basic metrics (e.g. CPU utilization, network in/out, memory, etc…). Logging is also important, so make sure to configure a logging tool.

Infrastructure as code

The infrastructure will be built using IAC (Infrastructure as Code) tools. It is required that you use Terraform.

Automation tools

The pipelines to build the code and deploy both the code and the infrastructure will be built in GitHub, using GitHub Actions. For the infrastructure part, Terraform Cloud will be used to host the infrastructure state and variables/secrets.