Cloud Computing and Edge-Cloud Continuum Course

GCP Project description

The students need to prove their understanding of cloud technologies and architectures. They are supposed to understand the design, development, and management of dynamic solutions that improve the quality of services. The approaches such as muti-cloud and hybrid clouds should be considered as well.

Use-case: Emergency Routing

Company Overview

OdenseEmergency (imaginary) is a provider of emergency handling software used by governments/organizations for their end users: first responders, firefighters, and citizens. They provide their software as a service to organizations globally.

Solution Concept

Due to rapid global warming and fires that occur around the world, this business has been growing exponentially year over year. The company provides a mobile app and needs to be able to assure good performance in real-time while considering the users' device limits. In addition, scaling its environment, adapting its disaster recovery plan, and rolling out new continuous deployment capabilities to update its software quickly are desirable if the resources allow them. Google Cloud has been chosen to complete/replace their current colocation facilities.

Existing technical environment

OdenseEmergency developed an application that routes people to safe areas in disaster. The application captures location and human emotions data and processes the data to change the application interface. The change is in line with helping people to make better decisions and evacuate safely and quickly. The application captures six types of basic emotions (joy, surprise, fear, anger, disgust, sadness) plus neutral, uses various adaptation techniques (such as rule-based, reinforcement learning-based, etc.), and performs four types of UI adaptation (change in background color, font size, street vs. satellite view, pop-up information). It is worth mentioning that the target emotions (that the app wants to push the users to) are joy, surprise, and fear, and they get a reward in RL. The other emotions get punishment, and neutral is set to zero.

OdenseEmergency thinks that local processing and storage would create **performance** and **energy consumption** issues (***these are the two main non-functional requirements, you are free to consider more such as scalability***), but they are not sure about it. The managers are unsure about a suitable architecture: processing on a phone, a server, a cloud, establishing a hybrid architecture, etc. They would like to make their decisions based on the above-mentioned non-functional requirements.

They need to think about how moving virtual machines (VMs) instances between zones or regions and letting them connect would impact latency.

They think they might need to use load balancers.

Furthermore, they would like to assess network service tiers (premium vs. standard) and see how critical they would be before implementation.

They also need to decide about containerization and data storage methodologies.

They would like to ensure compliance with privacy regulations, as human data is captured.

Executive statement

An on-premises strategy may work but has required a significant resource investment. Many of the outages have resulted from misconfigured systems and inadequate capacity to handle spikes in traffic. We want to use Google Cloud to leverage a performant and energy-efficient platform that can span multiple environments seamlessly and provide a consistent and stable user experience that positions us for future growth. However, a lot of pre-assessments are required.

The abilities to assess:

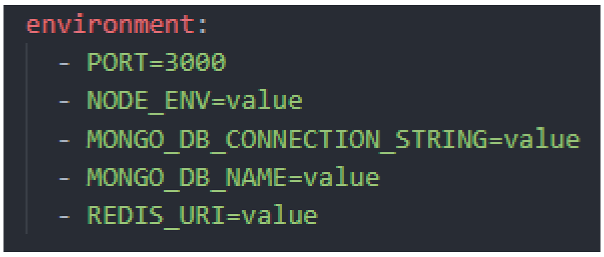
- Design and plan cloud solution architectures
- Manage and provision the cloud solution infrastructure
- Design for performance and energy efficiency
- Analyze and optimize technical and business processes
- Manage implementations of cloud architectures
- Verify solution and operations feasibility/quality

Getting started instructions:
You will need an API key for Google Maps, credentials for MongoDB, a Redis Server, GCP credits, versions of both the app and the backend.

The frontend and backend are provided as ZIP-files.

Setting up and running the backend:

1. Download and unzip the provided ZIP-file.
2. Download and install NodeJS: https://nodejs.org/en/download/
3. Open the project in your IDE of choice
4. Install external dependencies with the command *"npm install"*
5. Edit the environment variables in the outermost docker-compose file



```
environment:
  - PORT=3000
  - NODE_ENV=value
  - MONGO_DB_CONNECTION_STRING=value
  - MONGO_DB_NAME=value
  - REDIS_URI=value
```

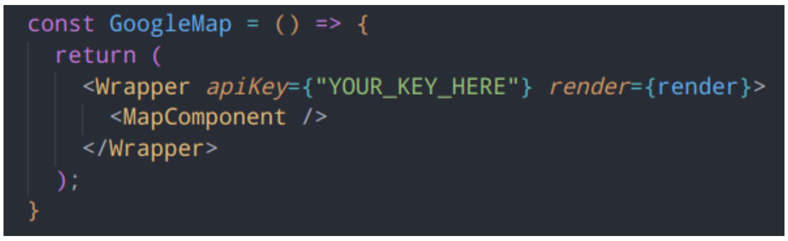
   This configuration will start the server on port 3000. You will need to setup a MongoDB database on MongoDB Atlas and add the database name and the connection string: https://www.mongodb.com/atlas/database. You'll also need to whitelist your IP address on MongoDB Atlas.

   You also need to set up a Redis server and add the relevant URI. You can get up to $300 of credits on Aiven if you don't want to run it locally: https://aiven.io/

6. When everything is set up you can run the server with the command *"npm run dev".*

Setting up and running the frontend

1. Download and unzip the provided ZIP-file.
2. Open the project in your IDE of choice
3. Install external dependencies with the command *"npm install"*
4. Add a Google Maps API key (https://developers.google.com/maps/get-started) in the file *src/components/GoogleMap.tsx*



```
const GoogleMap = () => {
  return (
    <Wrapper apiKey={"YOUR_KEY_HERE"} render={render}>
      <MapComponent />
    </Wrapper>
  );
}
```

   The API key needs access to the following APIs:

   – Maps JavaScript API
   – Geocoding API

- Directions API

5. Run the project with the command *“npm start”*

Please note that the application is designed for mobile screens, so you might prefer to use your browser’s developer tools to view the app in a smaller resolution.

Important note:
The frontend always tries to reach the backend on localhost. If you deploy the services, the URL in *Frontend/src/common/Constants.ts* must be updated