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

SDUEmergency (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 needs to be able to scale its environment, adapt its disaster recovery plan, and roll out new continuous deployment capabilities to update its software at a fast pace. Google Cloud has been chosen to complete/replace their current colocation facilities.

Existing technical environment

SDUEmergency 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. Some processing of various algorithms (emotion detection, adaptation manager, etc.) and some storage are needed for location and emotional data.

SDUEmergency thinks that local processing and storage would create **performance** and **energy consumption** issues. The managers are unsure about a suitable architecture: processing on a server, on a cloud, or establishing a hybrid architecture. They would like to make their decisions based on the above-mentioned non-functional requirements.

They think they would need to use load balancers.

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

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.

Requirements

- Provide a system response time of less than one second.
- The ability to scale to 2000 users with negligible performance drop (maybe with dynamic scaling).
- Decrease infrastructure administration costs.
- Ensure compliance with regulations.
- Provide consistent logging, log retention, monitoring, and alerting capabilities.

Executive statement

An on-premises strategy may work but has required a significant investment of time and money in training our team on distinctly different systems, managing similar but separate environments, and responding to outages. Many of these 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

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. Add a file called *.env* in the root of your project and add the following environment variables:

```
.env
PORT=3000
MONGO_ATLAS_DB_NAME=value
MONGO_ATLAS_USERNAME=value
MONGO_ATLAS_PASSWORD=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 a db-name, username and password: https://www.mongodb.com/atlas/database. You'll also need to whitelist your IP address on MongoDB Atlas.

   You also need to setup 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 setup 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.