Student Visa Application Site Using Google Cloud Platform and Next.js

Overview

> The project involves building a cloud infrastructure for a student visa application site using Next.js and deploying it on Google Cloud Platform (GCP). The system includes a microservices architecture, database integration, and an automated CI/CD pipeline to ensure efficiency and reliability.

Project Details

Local Development Environment

1. **Project Initialization**:
    - Created a new Next.js project using the command: `bash`

      ```
      npx create-next-app visa-application
      cd visa-application
      ```

2. **Running Local Server**:
    - Used the command: `bash`

      ```
      npm run dev
      ```

    - This starts the local development server for testing and development purposes.

Database Setup

1. **Creating Cloud SQL Database**:
    - Used Google [Cloud SQL] to create a [MySQL] (or PostgreSQL) database.
    - Configured database instance, including username and password.
2. **Connection Configuration**:
    - Stored database credentials in an `.env.local` file: `env`

```
DATABASE_HOST=<your-database-host>
DATABASE_NAME=<your-database-name>
DATABASE_USER=<your-database-user>
DATABASE_PASSWORD=<your-database-password>
```

3. **Database Connection Module**:
   - Created a connection module in `lib/db.js`:

```js
import mysql from 'mysql2/promise';

const pool = mysql.createPool({
  host: process.env.DATABASE_HOST,
  user: process.env.DATABASE_USER,
  password: process.env.DATABASE_PASSWORD,
  database: process.env.DATABASE_NAME,
});

export default pool;
```

Application Development

1. **Application Page**:
   - Developed a visa application form in `pages/apply.js`:

```js
import { useState } from 'react';

const Apply = () => {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    const res = await fetch('/api/submit', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ name, email }),
    });
    const result = await res.json();
    console.log(result);
  };

  return (
    <div>
      <h1>Apply for a Visa</h1>
      <form onSubmit={handleSubmit}>
        <input type="text" value={name} onChange={(e) =>
setName(e.target.value)} placeholder="Name" required />
        <input type="email" value={email} onChange={(e) =>
setEmail(e.target.value)} placeholder="Email" required />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
};

export default Apply;
```

2. **API for Submitting Applications**:
   - Developed an API endpoint in `pages/api/submit.js`:

```js
import pool from '../../lib/db';

export default async function handler(req, res) {
  if (req.method === 'POST') {
    const { name, email } = req.body;
    try {
      const [result] = await pool.query('INSERT INTO
applications (name, email) VALUES (?, ?)', [name, email]);
      res.status(200).json({ message: 'Application
submitted successfully', id: result.insertId });
    } catch (error) {
      res.status(500).json({ message: 'Error submitting
application', error: error.message });
    }
  } else {
    res.status(405).json({ message: 'Method Not Allowed'
});
  }
}
```

Cloud Infrastructure Setup

1. **GCP Project Creation**:
   - Created a new project in [Google Cloud Console].
2. **Google Kubernetes Engine (GKE)**:
   - Set up a `Kubernetes` cluster to deploy the `application`.
3. **Docker Configuration**:
   - Created a `Dockerfile` to containerize the Next.js application: `Dockerfile`

```
FROM node:14-alpine
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
CMD ["npm", "start"]
EXPOSE 3000
```

4. **Kubernetes Deployment Configuration**:
    - Created a deployment configuration in `kubernetes/deployment.yaml`: `yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: visa-application
spec:
  replicas: 3
  selector:
    matchLabels:
      app: visa-application
  template:
    metadata:
      labels:
        app: visa-application
    spec:
      containers:
      - name: visa-application
        image: gcr.io/<your-project-id>/visa-
application:latest
        ports:
        - containerPort: 3000
        env:
        - name: DATABASE_HOST
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: host
        - name: DATABASE_USER
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: user
        - name: DATABASE_PASSWORD
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: password
        - name: DATABASE_NAME
          valueFrom:
            secretKeyRef:
                name: db-credentials
                key: name
```

Build and Deployment

1. **Building and Pushing Docker Image**:
   - Built and `pushed` the Docker image to Google Container Registry: `bash`

   ```
   docker build -t gcr.io/<your-project-id>/visa-application:latest .
   docker push gcr.io/<your-project-id>/visa-application:latest
   ```

2. **Deploying in GKE**:
   - Deployed the application using `kubectl`: `bash`

   ```
   kubectl apply -f kubernetes/deployment.yaml
   ```

Performance Monitoring and Reporting

Performance Monitoring:

> - Used Google Cloud Monitoring to track performance and set up alerts for potential issues.