GCP CLOUD

Upload a project to GCP

Prerequisites

1. **Google Cloud Account**: Make sure you have a Google Cloud account. You can create one here.
2. **Google Cloud SDK**: Install the Google Cloud SDK on your local machine. You can download it here.

Steps to Deploy a Next.js Application on Google Cloud Platform

1. **Create a New Project in Google Cloud Console**:
   - Go to the Google Cloud Console.
   - Click on the project drop-down and select "New Project".
   - Give your project a name and click "Create".
2. **Set up Google Cloud SDK**:
   - Initialize the SDK and log in to your Google Cloud account:

     ```
     gcloud init
     ```

3. **Prepare Your Next.js Application**:
   - Make sure your Next.js application is ready for deployment. If you haven't already, create a Next.js app:

     ```shell
     npx create-next-app@latest study-visa-app
     cd study-visa-app
     ```

4. **Create `app.yaml` Configuration File**:
   - In the root of your Next.js project, create a file named `app.yaml` with the following content:

```yaml
runtime: nodejs18
env: standard
handlers:
  - url: /.*
    secure: always
    script: auto
```

5. **Modify `package.json` for App Engine**:
   - Update your `package.json` to include a start script for production:

```
{
  "scripts": {
    "start": "next start",
    "build": "next build",
    "dev": "next dev",
    "lint": "next lint"
  }
}
```

6. **Deploy Your Application**:
   - Deploy your application to Google Cloud App Engine:

```
gcloud app deploy
```

7. **Access Your Deployed Application**:
   - After deployment is complete, you can access your application using the URL provided by App Engine.

Steps to Create a Private Cloud Infrastructure

1. ***Set Up Virtualization Platform:**
   - Choose a virtualization *platform* like VMware vSphere, OpenStack, or `Proxmox`.
   - Install and configure the virtualization *platform* on your physical servers.

2. ***Install Kubernetes:**
    - Set up a *Kubernetes* cluster to manage containerized applications.
    - You can install Kubernetes using tools like `kubeadm`, `Minikube` for development, or managed solutions like *Rancher*.
3. ***Set Up Docker:**
    - Install Docker on each node of your Kubernetes *cluster* to run containerized *applications*.
4. ***Deploy Next.js Application:**
    - Containerize your Next.js *application* using Docker.
    - Deploy the Docker container to your *Kubernetes* cluster.
5. ***Set Up Networking and Storage:**
    - Configure networking using `tools` like Calico or Flannel.
    - Set up persistent storage using solutions like `Ceph`, NFS, or `GlusterFS`.
6. ***Set Up Continuous Integration/Continuous Deployment (CI/CD):**
    - Use tools like Jenkins, GitLab CI, or GitHub Actions to automate the deployment process.

Detailed Steps

1. Set Up Virtualization Platform

***Using Proxmox VE:**

1. **Install `Proxmox` VE on your physical server:**
    - Download the *Proxmox* VE ISO from the Proxmox website.
    - Boot from the ISO and follow the installation instructions.
2. **Create Virtual Machines (VMs) for Kubernetes Nodes:**
    - Log in to the `Proxmox` web interface.
    - Create VMs for the Kubernetes master and *worker* nodes.

2. Install Kubernetes

1. **Install `kubeadm`, `kubelet`, and `kubectl` on each VM:**

```shell
sudo apt-get update && sudo apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
```

2. **Initialize the Kubernetes master node:**

```
sudo kubeadm init
```

3. **Set up the kubeconfig for the master node:**

```shell
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

4. **Install a network plugin (e.g., Calico):**

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

5. **Join worker nodes to the cluster:**
   - On each worker node, run the join command provided by `kubeadm init`.

3. Set Up Docker

1. **Install Docker on each node:**

```shell
sudo apt-get update
sudo apt-get install -y docker.io
sudo systemctl enable docker
sudo systemctl start docker
```

4. Deploy Next.js Application

1. **Create a Dockerfile for your Next.js application:**

```
# Dockerfile
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000
CMD ["npm", "start"]
```

2. **Build and push the Docker image to a container registry:**

```shell
docker build -t yourusername/study-visa-app .
docker push yourusername/study-visa-app
```

3. **Create Kubernetes deployment and service files:**

   ***deployment.yaml:**

YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: study-visa-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: study-visa-app
  template:
    metadata:
      labels:
        app: study-visa-app
    spec:
      containers:
      - name: study-visa-app
        image: yourusername/study-visa-app:latest
        ports:
        - containerPort: 3000
```

***service.yaml:**

YAML

```yaml
apiVersion: v1
kind: Service
metadata:
  name: study-visa-app
spec:
  selector:
    app: study-visa-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 3000
  type: LoadBalancer
```

4. **Apply the Kubernetes configuration:**

```shell
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

5. Set Up Networking and Storage

- **Networking**: Use Calico or Flannel for networking.
- **Storage**: Use Ceph, NFS, or GlusterFS for persistent storage.

6. Set Up CI/CD

- Use Jenkins, GitLab CI, or GitHub Actions to automate the build and deployment process.