Deployment Guide: Frontend on Firebase, Backend on Google Kubernetes Engine (GKE), and Serverless Functions on Google Cloud Functions

1. Deploying a Frontend in Firebase Hosting

1. **Set Up Firebase CLI**
    - Install Firebase CLI:

npm install -g firebase-tools

2. **Log In to Firebase**
    - Log in to your Firebase account:

firebase login

    - Choose the Firebase services you want to configure, select Hosting, and follow the prompts to select your Firebase project.
3. **Build Your Node.js Application**
    - Ensure your application is built and ready for production. Run:

npm run build

    - This generates a production-ready version of your app in the directory specified during initialization build .
4. **Deploy to Firebase**
    - Deploy your application to Firebase Hosting:

firebase deploy

    - This command uploads your build files to Firebase Hosting and makes your application live.

2. Deploying a Backend in Google Kubernetes Engine (GKE)

Prerequisites:

- Google Cloud SDK

- Docker
- kubectl
- Google Cloud Project
- GKE Cluster
- YAML Files (cluster-issuer.yaml, deployment.yaml, ingress.yaml)

Steps:

5. **Configure gcloud and Push Docker Image to Google Container Registry**

a. **Authenticate Docker with Google Cloud:**

gcloud auth configure-docker us-central1-docker.pkg.dev

b. **Tag Your Docker Image:**

docker tag <local-image-name> us-central1-docker.pkg.dev/<project-id>/<repository>/<image-name>:<tag>

Example:

docker tag watermark us-central1-docker.pkg.dev/watermark1234/watermark/watermark:latest

c. **Push the Image to GCR:**

docker push us-central1-docker.pkg.dev/watermark1234/watermark/watermark:latest

6. **Deploy YAML Files to GKE**

a. **Authenticate with GKE Cluster:**

gcloud container clusters get-credentials <your-cluster-name> --zone <zone>

Example:

gcloud container clusters get-credentials my-cluster --zone us-central1-c

b. **Deploy cluster-issuer.yaml:**

kubectl apply -f cluster-issuer.yaml

c. **Deploy deployment.yaml:**

- Ensure your deployment YAML references the correct Docker image:

image: "us-central1-docker.pkg.dev/watermark1234/watermark/watermark:latest"

- Deploy:

kubectl apply -f deployment.yaml

d. **Deploy ingress.yaml:**

- Ensure the host in ingress.yaml points to your domain.
- Deploy:

kubectl apply -f ingress.yaml

7. **Verify the Deployment**
   - Check Deployments:

kubectl get deployments

- Check Services:

kubectl get services

- Check Ingress:

kubectl get ingress

- The external IP address for the ingress should be provisioned, allowing access to your service.

8. **Configure DNS**
    - Configure DNS to point the domain (e.g., alphamatrix.linkpc.net) to the IP address of the ingress controller. Set an A record pointing to the external IP.
9. **Verify HTTPS (Optional)**
    - If using Let's Encrypt, check the status of your Ingress and ClusterIssuer resources:

```
kubectl describe clusterissuer
kubectl describe ingress watermark-ingress
```

- Your application should be accessible at https://alphamatrix.linkpc.net.

3. Deploying a Backend Using Google Cloud Functions

10. **Deploy the Cloud Function**
    - Search for Cloud Run in the Google Cloud Console.
    - Select the watermarkservice2 service.
    - Click "Edit & Deploy New Revision."
    - Select the new container image URL.
    - Deploy the new revision.
11. **View Logs**
    - Access logs in Cloud Run functions and see videos related to watermarking in the specified bucket.