DEPLOYING A CONTAINERIZED WEB APPLICATION

INTRODUCTION

The following document's steps taken to package a web application in a Docker container image and run the container image on a Google Kubernetes Engine (GKE) cluster. This web application was deployed as a load-balanced set of replicas that provide advantage of scaling to meet user's needs. The web application is based on node js and consists of a single page detailing just basic information.

PREREQUISITES

-Have an account on Google Cloud Platform(free-tier)
-Create a project and make sure you have billing enabled for the Google cloud Project

Made use of the Cloud Shell which comes preinstalled with gcloud, docker and kubectl command line tools.

Build the Container Image:

1. First we have to package the source code as a Docker image. The Dockerfile contains instructions on how the image is built. Download the source code and Dockerfile by running the following commands
   ```
   git clone https://github.com/neeyoma/circleci-orbs
   cd circleci-orbs
   ```

2. Set the PROJECT_ID environment variable to your Google Cloud project ID. The variable is used to associate the container image with the project's Container Registry
   ```
   export PROJECT_ID=project1-284815
   ```

3. Build and tag the Docker images
   ```
   docker build -t gcr.io/${PROJECT_ID}/circleci-orbs:v1 .
   ```

4. Verify the build was successful
   ```
   docker images
   ```

Push the Docker image to the Container Registry:

1. Configure Docker to authenticate to Container Registry
   ```
   gcloud auth configure-docker
   ```
2. Push the Docker image to Container Registry
   ```
   docker push gcr.io/${PROJECT_ID}/circleci-orbs:v1
   ```

Create GKE cluster:

1. Set Project ID and Compute Engine Zone options for gcloud tool

```
gcloud config set project $PROJECT ID
```

```
gcloud config set project $PROJECT_ID us-east1-b
```

2. Create a cluster called test-cluster
   ```
   gcloud container clusters create test-cluster
   ```
3. Check to see the cluster's instances
   ```
   gcloud compute instances list
   ```

Deploy the sample application to GKE:

1. Create a Kubernetes deployment for the Docker image

```
kubectl create deployment circleci-orbs--image=gcr.io/${PROJECT ID}/circleci-orbs:v1
```

2. Set Deployment replicas to 3(baseline)
   ```
   kubectl scale deployment circleci-orbs --replicas=3
   ```

3. Create a HorizontalPodAutoscaler resource for Deployment
   ```
   kubectl autoscale deployment circleci-orbs --cpu-percent=80 --min=1
   --max=5
   ```

4. To see the pods created
   ```
   kubectl get pods
   ```

Expose the application to the internet:

1. Use kubectl expose to generate a kubernetes service for the deployment
   ```
   kubectl expose deployment circleci-orbs --name=circleci-orbs-service
   --type=LoadBalancer --port 80 --target-port 3000
   ```
2. Get services for circleci-orbs-service
   ```
   kubectl get service
   ```
3. Copy the EXTERNAL_IP address and paste on a browser tab
   ```
   NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP
   PORT(S)        AGE
   circleci-orbs-service   LoadBalancer   10.63.244.141   34.73.158.181
   80:30858/TCP   46s
   ```