Problem Statement

Part 1 A:

Create a GKE cluster with the following details:

I. Name: cc-interview-<candidate_name>
II. Location type: Zonal
III. Zone: asia-south1-a
IV. Nodepool:
    A. Number of nodes: 3
    B. Machine type: n1-standard-1

Part 1 B:

Deploy a 2-tier application in different namespaces (Frontend application in frontend-ns and backend in backend-ns). The choice of application technologies will be dependent on the candidate.

The applications must be able to communicate with each other across namespaces.

Part 1 C:

Apply HPA to the frontend application. Do a load test on the application in order to test the functionality of the HPA. We need to consider the HPA events to confirm the same.

Note: Load test can be done by using any 3rd party tool.

Part 2A:

Setup Datadog dashboards to monitor the GKE cluster i.e. node level monitoring.

Part 2B:

Setup Datadog dashboards for Kubernetes pod level monitoring.

Solution:

This Application is based on two-tier architecture. These all file let us know how to run, deploy app on kubernative environment, monitoring and Load Test.

Directory Structure:

The directory is distributed in three part.

1. Code
2. DataDog
3. K6s

1. Code: This Part contain all the file related the Docker and Kubernetes.
2. DataDog: This is monitoring tool. This directory has all the code related to that.
3. K6s: This is Load Testing tool. It has all the file related to that.

1. Code

All the file and folders.

a. Docker Directory: It has File related to Web-application and Docker file.
b. Backend: It has kubernate files related to MySQL database.
c. Frontend: It has kubernate files related to Frontend App.
d. Namespaces: It has kubernate files to create Namespaces.

Command to run the application:

Step-1: Build the Docker File and Push Image

First, move to Docker Directory

a. For Build the Code
   i. $ docker build -t <Docker-UserName> /movie .
      Eg. docker build -t ritikgupta/movie .
b. Push the Image
   i. $ docker login
   ii. $ docker push ritikgupta/movie

Step-2: Create NameSpace

This will create two namespaces for the frontend and backend. Move to Namespace directory.

a. Create Frontend namespace
   i. $ kubectl apply -f frontend-ns.yml
b. Create Backend namespace
   i. $ kubectl apply -f backend-ns.yml

Step-3: Create Backend

This will create Kubernetes backend with Mysql. Move to Backend Folder.

a. Create Persistent Volume
   Note: If you are working on GKE then no need to create PV. It will automatically create during PVC.
   i. $ kubectl apply -f mysql-pv.yml
b. Create Persistent Volume Claim
   i. kubectl apply -f mysql-pvc.yml
c. Create Secrets
   i. Use CMD: echo -n <password> | base64
      Then copy the output into mysql-secrets.yml in value.
   ii. $ kubectl apply -f mysql-secrets.yml
d. Create deployment
   i. $ kubectl apply -f mysql-deployment.yml
e. Create Service
   i. $ kubectl apply -f mysql-service.yml

Step-4: Create Frontend

This will create Kubernetes frontend with Docker Image. Move to Frontend Folder.

a. Create secrets
    i. Use CMD: echo -n <password> | base64
       Then copy the output into webapp-secrets.yml in value.
    ii. $ kubectl apply -f webapp-secrets.yml
b. Create Deployment
    i. $ kubectl apply -f webapp-deployment.yml
c. Create Service
    i. $ kubectl apply -f webapp-service.yml
d. Create Extername for Backend Mysql Service
    i. $ kubectl apply -f webapp-externalname.yml
e. Create HPA
    i. $ kubectl apply -f webapp-hpa.yml

Step-5: Create Database

a. Create a Pod for to login MYSQL
    i. $ kubectl run -it --rm --image=mysql:5.6 --namespace=backend-ns --restart=Never mysql-client -- mysql --host mysql --password=<Decoded-Password>
b. Create Database
    i. $ CREATE DATABASE flaskapi;
    ii. $ USE flaskapi;
c. Create Table
    i. $ CREATE TABLE movie (id INT PRIMARY KEY AUTO_INCREMENT, movie_name VARCHAR(255), director_name VARCHAR(255),ratings INT)

Step-6: Get the application URL

a. kubectl get svc -n frontend-ns
    i. Select flask-Service External-IP:PortNo.
       Eg.: 35.200.187.12:5000

Step-7: Load test

For installing K6
Ref: https://k6.io/docs/getting-started/installation/

We will use the K6 tool to test the spike test.

a. First, run Watch command in Terminal
    i. $ kubectl get hpa flaskapi-deployment-hpa -n frontend-ns --watch
b. In another Terminal, Move to the K6 directory
    i. We have to change the URL in spike_test.js file <External IP>:5000/movieslist
    ii. $ k6 run spike_test.js

c. Check Watch terminal to how Replicas scaling.

i. Before Load Test

```
ritikgupta912536@cloudshell:~ (cc-interview-sandbox)$ kubectl get hpa flaskapi-deployment-hpa -n frontend-ns --watch
NAME                     REFERENCE                          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   0%/30%    2         10        2          8h
```

ii. After Load Test

```
ritikgupta912536@cloudshell:~ (cc-interview-sandbox)$ kubectl get hpa flaskapi-deployment-hpa -n frontend-ns --watch
NAME                     REFERENCE                          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   0%/30%    2         10        2          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   33%/30%   2         10        2          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   33%/30%   2         10        2          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   33%/30%   2         10        3          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   73%/30%   2         10        3          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   73%/30%   2         10        3          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   73%/30%   2         10        4          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   62%/30%   2         10        4          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   62%/30%   2         10        4          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   62%/30%   2         10        7          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   98%/30%   2         10        7          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   98%/30%   2         10        7          8h
flaskapi-deployment-hpa   Deployment/flaskapi-deployment   98%/30%   2         10        10         8h
```

See the Replicas

Step-7: Monitoring with Data Dog

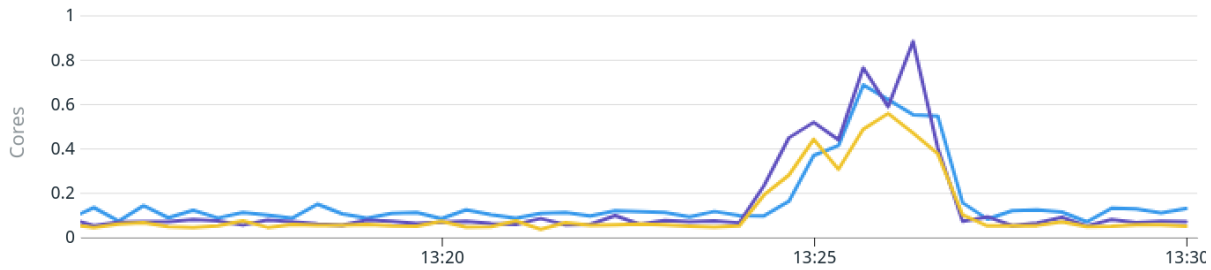
For Installing DataDog: https://github.com/HoussemDellai/datadog-prometheus-k8s

Data Dog have prebuild dashboard for monitoring for Node and Pod Level

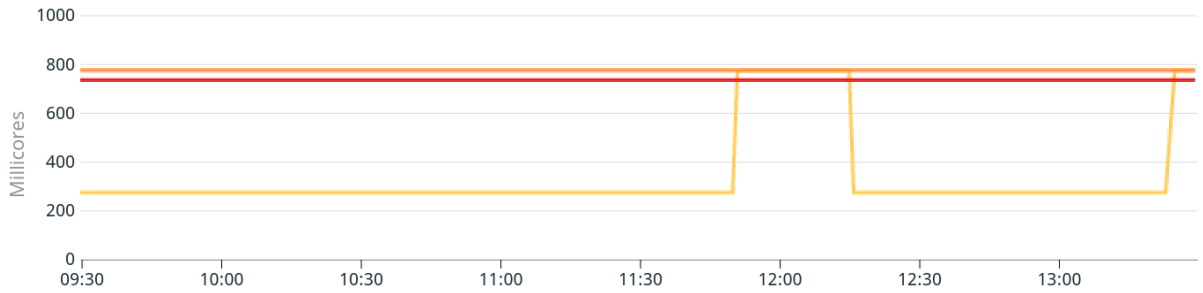
1. In DataDog go Dashboard -> Dashboard List -> Kubernates Node Overview
   a. During the Load test

CPU utilization per node

CPU usage per node



Total Kubernetes CPU requests per node



2. In DataDog go Dashboard -> Dashboard List -> Kubernates Pod Overview
    a. During the Load Test