**Task 1**

I have developed three images namely:

1. *Frontend-react*
2. *Backend-flask*
3. *Nginx*

There are separate folders for every image and Dockerfiles are also present there that I used to make image.

**Front-End:**

* In frontend I wrote a Dockerfile to make image of react app
* It will copy all the file from present directory to container
* It will install all necessary packages
* It will run on npm run command
* I have exposed port 3000 in Dockerfile

**Back-End:**

* In Backend I wrote a Dockerfile to make image of flask app
* It will copy all the file from present directory to container
* It will install all necessary packages from requirments.txt
* It will run on gunicorn command
* I have exposed port 5000 in Dockerfile

**Nginx:**

* In nginx I wrote a Dockerfile to make image of nginx
* It will copy nginx config file to container
* Now what is in nginx config file
* In nginx I have specified that when user enter URL (here we have set localhost) on port 80 it will be redirected to frontend

**Docker-compose:**

In docker compose I wrote services for each container (frontend, backend and nginx)

In nginx we said that nginx is dependent on front end and back end which means we made connection between frontend and backend with nginx

Next in frontend I said that frontend is dependent on backend

And now on docker-compose up –build command it will run our app on localhost port 80

As mentioned above I have created Docker images using the Dockerfile and then stored in the docker hub repository. Instead of creating the every time we can utilize the existing stored docker images.

I have prepared two docker-compose files.

1: Docker-compose.yml file we can build and create the containers i.e docker-compose –f Docker-compose.yml up -d -build

2. As I mentioned Bring the application with existing docker images i.e docker-compose –f docker-compose1.yml up -d

All these application i.e frontend and backend I have deployed in the google cloud server.

**Application can be accessed :** [**http://34.125.142.252:80/[nginx**](http://34.125.142.252:80/%5bnginx) **is serving the application]**

**Task 2:**

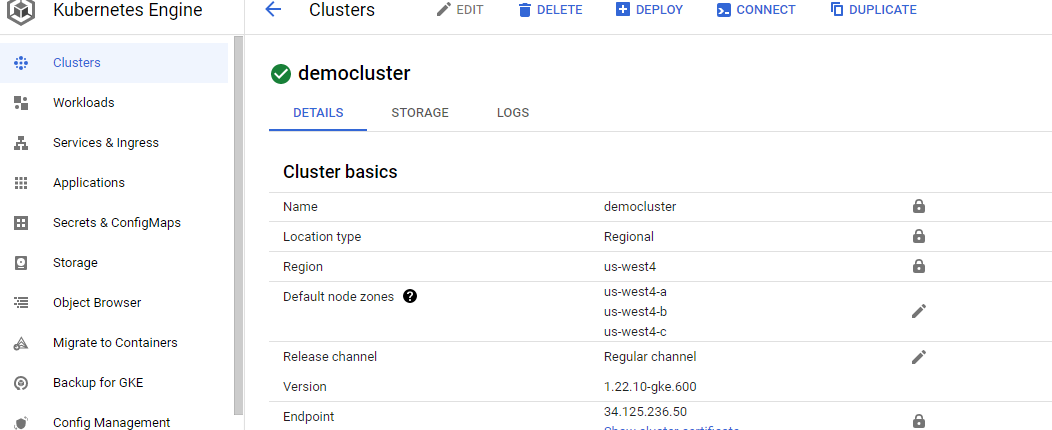
As mentioned I have dockerized the application as I am having all the images in my docker hub.

Created a kubernetes cluster in the Google Cloud Provider.

Created the following below mentioned manifest files for serving the application the google cloud

1. Created deployement file with 1 replicaset for maintaining the application desired state for each front end and back end.
2. Created the service file for communicating with the deployment i.e created front end service with type LoadBalancer exposed the application through the outside world for accessing the end user. And then created the backend service with type ClusterIP as its exposed within the cluster.

**Application can be accessed :**  [**http://34.125.191.197:3000/**](http://34.125.191.197:3000/) **[application is serving via the LoadBalancer service provided by google cloud]**

****

**Task 3**

As we we have already developed the docker images and stored in the docker hub.

Now I have created a 3 machines in the google cloud and then installed all the kuberenetes related software’s in the all the 3 machines.

Made 1 machine as master and remaining 2 nodes made as worker node.

Following are the Cluster Set up instructions

In Master Node

1. Installed docker for running the containers
2. Install the kubeadm,kubelet,kubectl
3. Initialize the Kubernetes Master Node

Sudo kubeadm init –ignore-preflight-errors=all

1. Installed the weave networking driver

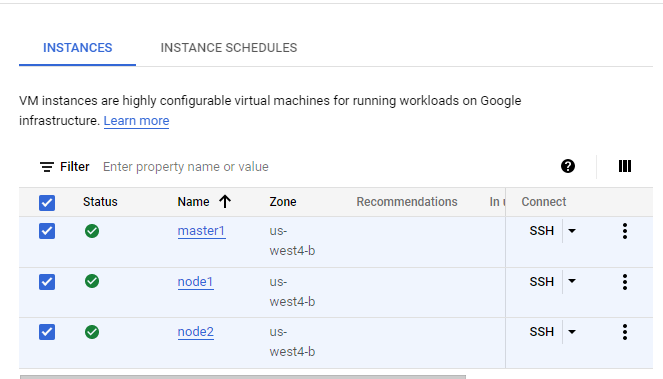
Note : run this command : kubeadm token create --print-join-command for getting the token for joining the master node .

In worker node

1. Installed the docker for running the containers
2. Installed kubeadm,kubelet,kubectl
3. As mentioned above copy the token and then execute in the worker node with kubeadm join <token id>

Now your cluster is ready to serve your dockerized application.

With this approach we can deploy the complex micro services application as we are having the 3 nodes cluster, if requires we can add worker nodes as much as it requires for maintaining application desired state.



Created the followings in the cluster

1. 2 deployments I have created for frontend with replicas 1 and then backend with replicas 1
2. 2 services created for serving the application for outside the world to the end user created client-cluster-ip-service(front end) with type NodePort and then 1 backend service created and exposed internal in the cluster with the type of ClusterIP.

Note : I am attaching the all the kubernetes manifest file and dockerfile and compose file.