**Open your Google cloud developer console**

Create your project iD

On the Tab of your Google cloud console

click on select ***New Project***  and

create your project ID with a unique name mine Project name is **Paul-kubent**

**For security purposes you can add IAM (identity Access management)**

On the Navigation menu Select IAM and Admin click on **IAM and** select your Project ID click on the link of your job a page will open you can select ***owner or*** ***editor role*** to give access to people you want them to access your project.

**open your cloud shell I use Google cloudshell for this project.**

**Write the following code in your cloud shell to active your cloud shell for global configuration**

gcloud auth list

gcloud config list project

**To active the cloudshell for global configuration**

gcloud config set project **Paul-kubent**

**To set up kubnet for google type the code in your cloudshell to set up the location of your kubnet.**

gcloud config set compute/zone us-central1-b

**my outbut location**

NAME: io

LOCATION: us-central1-b

MASTER\_VERSION: 1.21.5-gke.1302

MASTER\_IP: 104.155.188.177

MACHINE\_TYPE: e2-medium

NODE\_VERSION: 1.21.5-gke.1302

NUM\_NODES: 3

STATUS: RUNNING

**After you set the zone, create a clusters**

Enable the kubernetes API create service accounts

or use the code blow on your cloudshell to create kubernetes clusters

gcloud container clusters create io

**Get the sample code**

**Clone the GitHub repository from the Cloud Shell command line:**

**I use this command to clone my code in googlecloud to github**

gsutil cp -r gs://spls/gsp021/\* .

**output like this**

Copying gs://spls/gsp021/orchestrate-with-kubernetes/labs/provisioning-ubuntu-on-gce.md...

Copying gs://spls/gsp021/orchestrate-with-kubernetes/labs/rolling-out-updates.md...

/ [156 files][201.4 KiB/201.4 KiB] 6.1 KiB/s

==> NOTE: You are performing a sequence of gsutil operations that may

run significantly faster if you instead use gsutil -m cp ... Please

see the -m section under "gsutil help options" for further information

about when gsutil -m can be advantageous.

**Change into the directory needed for this lab:**

cd orchestrate-with-kubernetes/kubernetes

**output like this**

ugomesa-sa@cloudshell:~/orchestrate-with-kubernetes/kubernetes(**Paul-kubent**)$

List the files to see what you're working with:

ls

**create kubectl create command. I use to lunch a single instance of the nginx container:**

kubectl create deployment nginx --image=nginx:1.10.0

**output**

NAME READY STATUS RESTARTS AGE

nginx-56cd7f6b6-wf4jh 1/1 Running 0 20s

**use the code under to create loadbalancer**

kubectl expose deployment nginx --port 80 --type LoadBalancer

**List our services now using the kubectl get services command:**

kubectl get services

**output**

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.72.0.1 <none> 443/TCP 13m

nginx LoadBalancer 10.72.3.241 34.121.46.187 80:30172/TCP 39s

**the External IP to this command to hit the Nginx container remotely:**

curl http://34.121.46.187:80

**Creating Pods**

**Pods can be created using pod configuration files. Take a moment to explore the monolith pod configuration file.**

**Run the following:**

cat pods/monolith.yaml

**Create the monolith pod using kubectl:**

kubectl create -f pods/monolith.yaml

**Examine your pods. Use the kubectl get pods command to list all pods running in the default namespace:**

kubectl get pods

**the output**

NAME READY STATUS RESTARTS AGE

monolith 1/1 Running 0 65s

nginx-56cd7f6b6-wf4jh 1/1 Running 0 8m30s

**Once the pod is running, use kubectl describe command to get more information about the monolith pod:**

kubectl describe pods monolith

**output**

Normal Scheduled 2m34s default-scheduler Successfully assigned default/monolith to gke-io-default-pool-4cecf6e4-2qt1

Normal Pulling 2m31s kubelet Pulling image "kelseyhightower/monolith:1.0.0"

Normal Pulled 2m30s kubelet Successfully pulled image "kelseyhightower/monolith:1.0.0" in 1.792493587s

Normal Created 2m29s kubelet Created container monolith

Normal Started 2m29s kubelet Started **container monolith**

kubectl create -f services/monolith.yaml

Interacting with Pods

**In the second terminal run this in your cloudshell**

kubectl port-forward monolith 10080:80

**output:**

**Forwarding from 127.0.0.1:10080 -> 80**

**Handling connection for 10080**

**IN THE FIRST TERMINAL :**

curl <http://127.0.0.1:10080>

**output:**

{"message":"Hello"}

**To check the health check of kubect use the google firewall code**

gcloud compute firewall-rules create allow-monolith-nodeport \

--allow=tcp:31000

**OUTPUT**

NAME: allow-monolith-nodeport

NETWORK: default

DIRECTION: INGRESS

PRIORITY: 1000

ALLOW: tcp:31000

DENY:

**Deployment of kubect**

* **auth** - Generates JWT tokens for authenticated users.
* **hello** - Greet authenticated users.
* **frontend** - Routes traffic to the auth and hello services

**To get auth deployment use this code**

cat deployments/auth.yaml

this show the detils of the works you want to deploy

**this code below deployed my work**

kubectl create -f deployments/auth.**yaml**

**output**

deployment.apps/auth created:

to create and expose the hello deployment:

kubectl create -f deployments/hello.yaml

kubectl create -f services/hello.yaml

the code create frontend Deployment.

kubectl create configmap nginx-frontend-conf --from-file=nginx/frontend.conf

kubectl create -f deployments/frontend.yaml

kubectl create -f services/frontend.yaml

**Interact with the frontend by grabbing it's External IP and then curling to it:**

kubectl get services frontend ( the code help you to get the information of your fortend

**output:**

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

frontend LoadBalancer 10.72.5.29 34.67.129.4 443:30120/TCP 66s

curl -k https://34.67.129.4

Endpoints: 10.68.1.5:443

Deployments

App: Hello

Replica 3

Pod

Hello

Node2

Pod

Hello

Node1

Pod

Hello

Node3