1)create a folder in cloud shell my_first_app

And upload file requirements.txt,Dockerfile,Kubernetes_first_app.py on cloud shell.

2)Verify docker is present on cloud shell

```
docker --version
```

3)build image using docker command

```
docker build -t kubernetes_first_app .
```

4)verify docker images has formed

```
docker images
```

5)tag docker image to upload it to gcr registry

```
docker tag kubernetes_first_app gcr.io/<project_id>/kubernetes_first_app
```

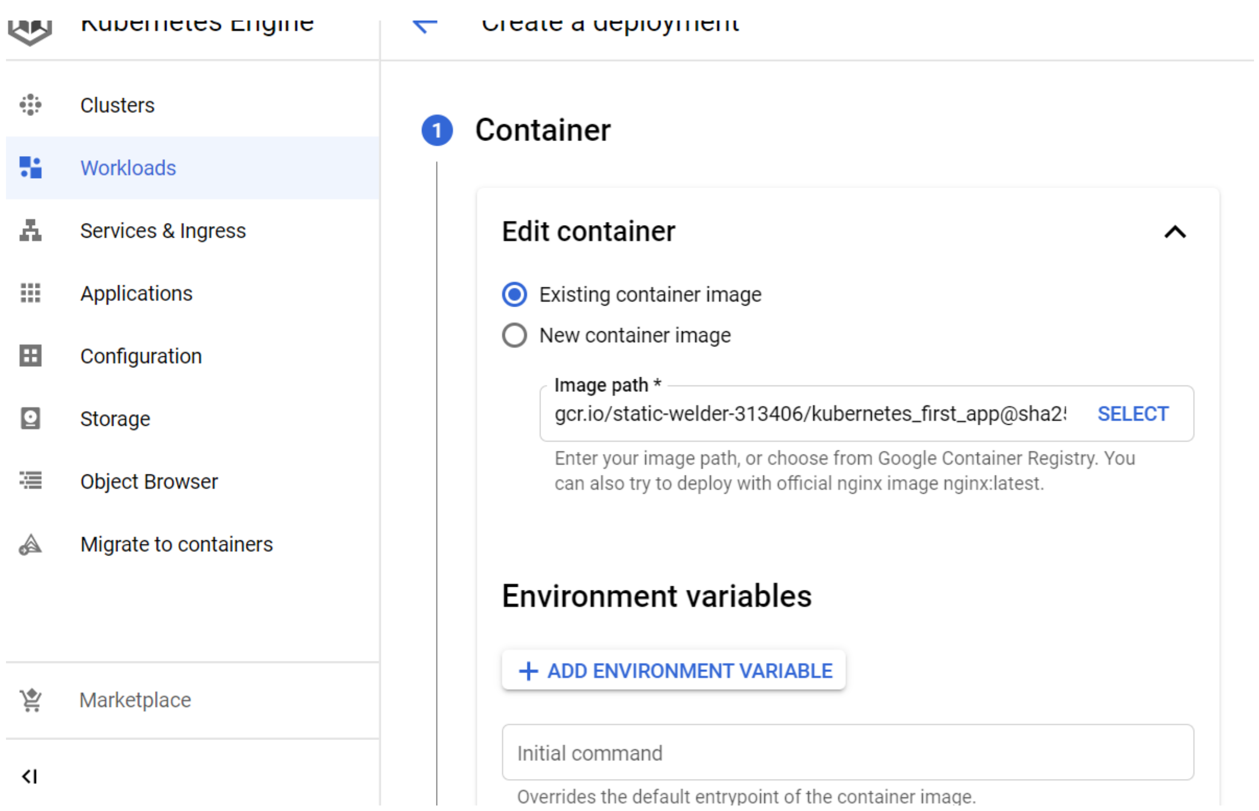
Here <project_id> shall be the project id of the qwiki lab

6)push docker image to gcr registry

```
docker push gcr.io/<project_id>/kubernetes_first_app
```

7)verify docker image has been uploaded in the gcr registry

8)Deploy your application on the Kubernetes cluster. click on deploy and select our own container image from gcr registry as shown below.



Existing container image [✓]
New container image

Image path *
gcr.io/static-welder-313406/kubernetes_first_app@sha2!

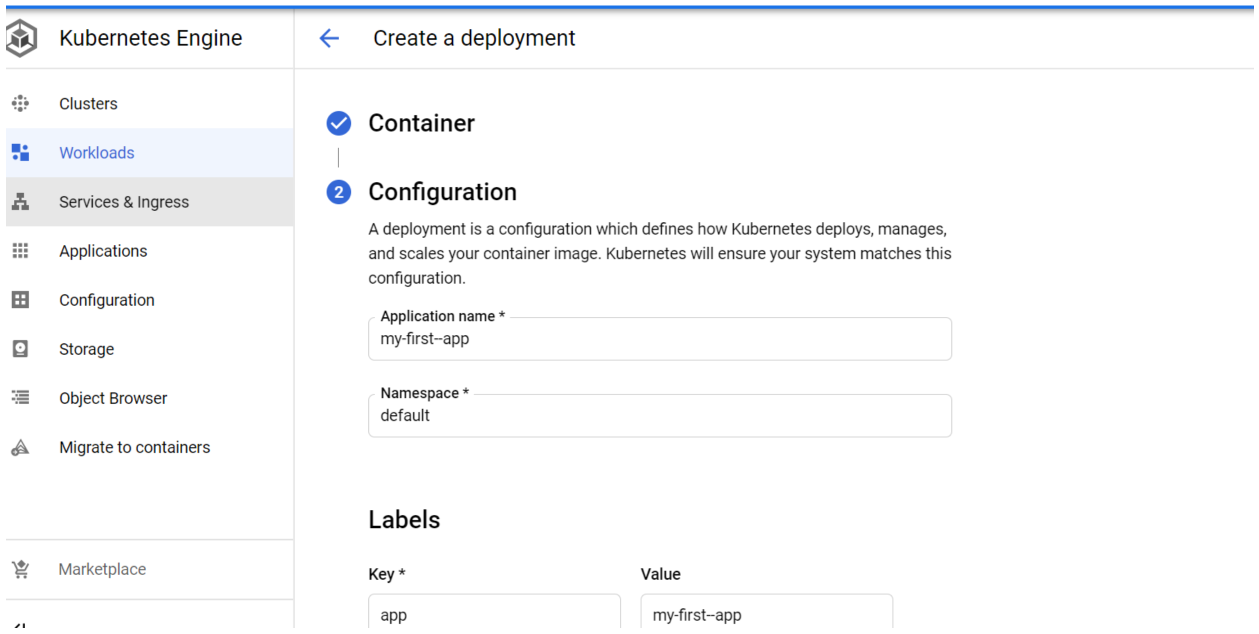
Enter your image path, or choose from Google Container Registry. You can also try to deploy with official nginx image nginx:latest.

Environment variables

Initial command

Overrides the default entrypoint of the container image.

```
Change the app name to my-first-app
```



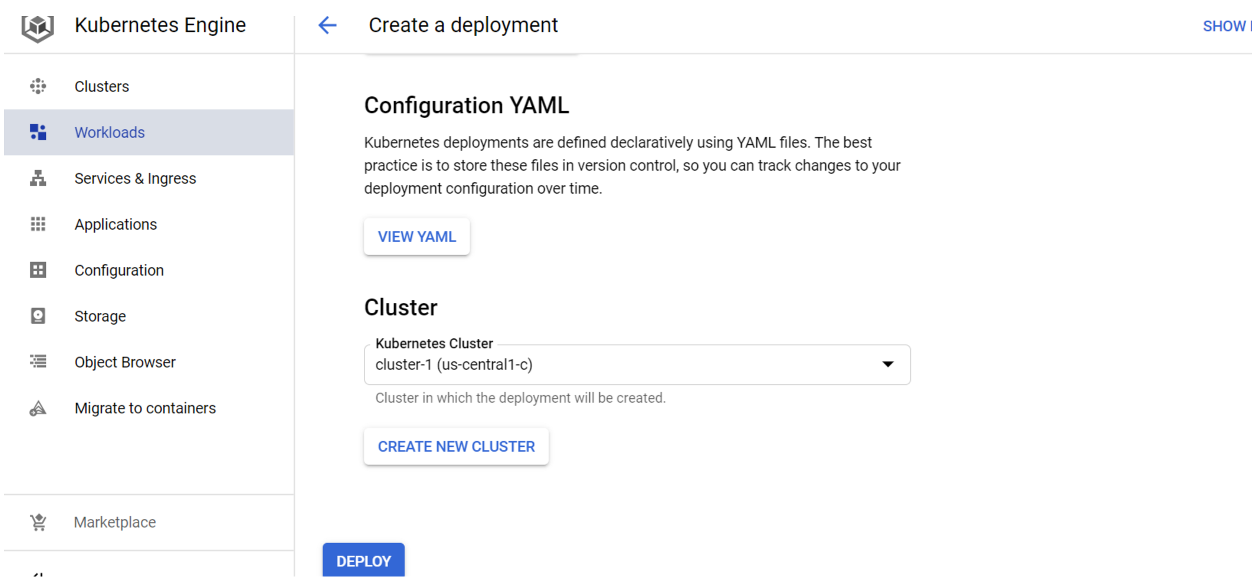
A deployment is a configuration which defines how Kubernetes deploys, manages, and scales your container image. Kubernetes will ensure your system matches this configuration.

Application name *
my-first--app

Namespace *
default

Labels

Key *	Value
app	my-first--app



Configuration YAML

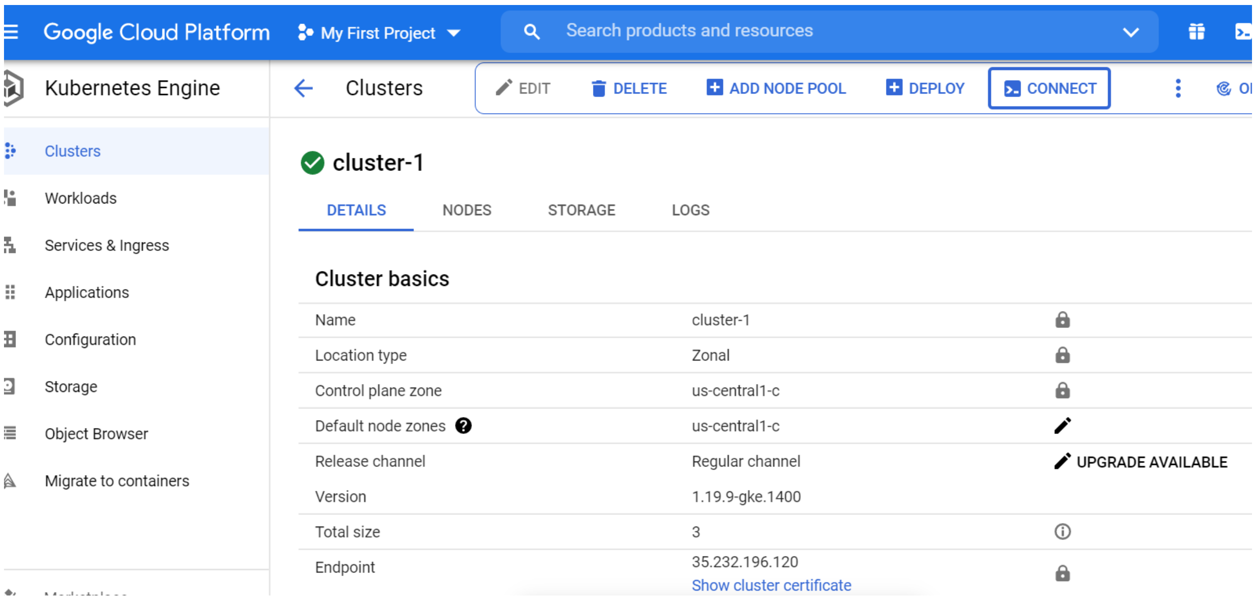
Kubernetes deployments are defined declaratively using YAML files. The best practice is to store these files in version control, so you can track changes to your deployment configuration over time.

Cluster

Kubernetes Cluster: cluster-1 (us-central1-c)

Cluster in which the deployment will be created.

9)Click on connect inside the cluster and run the command on cloud shell.



cluster-1

Cluster basics

Name	cluster-1
Location type	Zonal
Control plane zone	us-central1-c
Default node zones	us-central1-c
Release channel	Regular channel
Version	1.19.9-gke.1400
Total size	3
Endpoint	35.232.196.120

```
Output shall be like
Fetching cluster endpoint and auth data.
kubeconfig entry generated for <cluster-name>.
```

10)To view number of pods running in cluster ,hit command
kubectl get pods .To view three pods running on our cluster.

11)To view number of nodes we have in GKE cluster
kubectl get nodes

12)Go inside in one of the running pod by below pods

```
kubectl exec -it my-first--app-99456666f-sndjd -- /bin/sh
```

where my-first--app-99456666f-sndjd is the pod name
13)To see our application process running inside pod,do
```
Ps
```

14)to verify the port 3000,do
```
netstat -an |grep 3000
```

15)To test our app is running inside pod and is giving response on port 3000,we need to run curl on localhost:3000/welcome.
```
apk update
apk add curl
curl localhost:3000/welcome
exit
```

16)run the below command and observe the restart field it tells you how many time this pod has been restarted.

```
kubectl get pods
```

17)login to one of the pod again .use below command

```
kubectl exec -it my-first--app-99456666f-sndjd -- /bin/sh
```

where my-first--app-99456666f-sndjd is the pod name

kill the main application python process as below
```
ps -eaf |grep python

kill -9 <pid>
```

observe the pod ,it shall get restarted again .this can be figure out by looking at the restart count of the pod.

Kill each pod atleast once to see how the restart count changes.