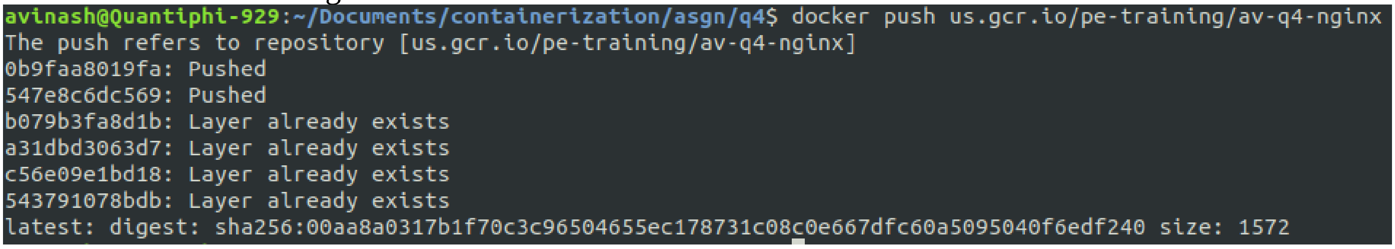
Kubernetes (EKS or GKE)

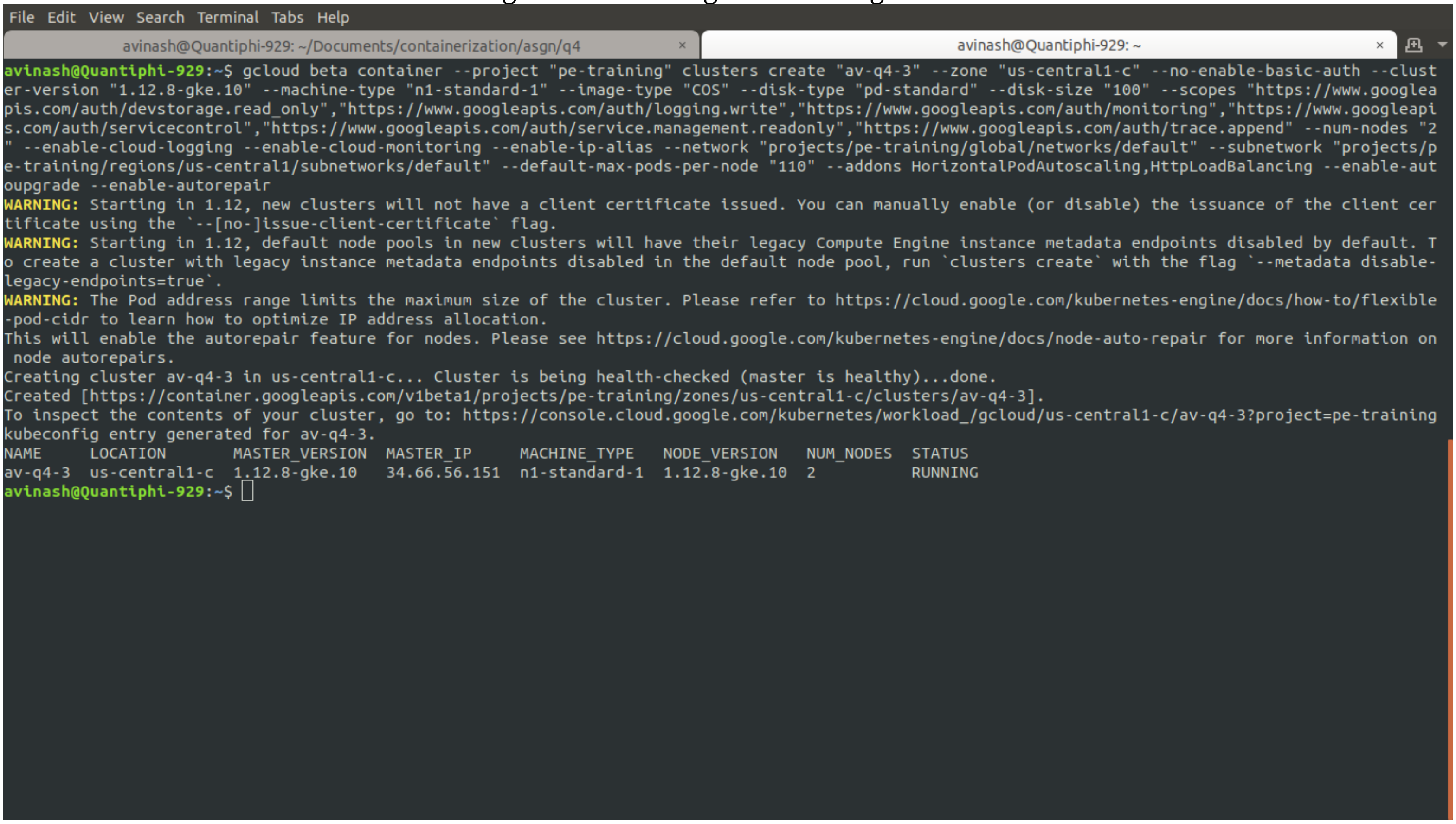
1: Write Script to creating Microservice using Nginx webserver to show default webpage. Create Kubernetes configuration files. Expose this microservice on ClusterIP, NodePort and as a LoadBalancer. Create a custom webpage to show which pod the page is loading from (it should automatically change with every refresh).

1. Build an image for your app.
2. Push that image on GCR.



```
avinash@Quantiphi-929:~/Documents/containerization/asgn/q4$ docker push us.gcr.io/pe-training/av-q4-nginx
The push refers to repository [us.gcr.io/pe-training/av-q4-nginx]
0b9faa8019fa: Pushed
547e8c6dc569: Pushed
b079b3fa8d1b: Layer already exists
a31dbd3063d7: Layer already exists
c56e09e1bd18: Layer already exists
543791078bdb: Layer already exists
latest: digest: sha256:00aa8a0317b1f70c3c96504655ec178731c08c0e667dfc60a5095040f6edf240 size: 1572
```

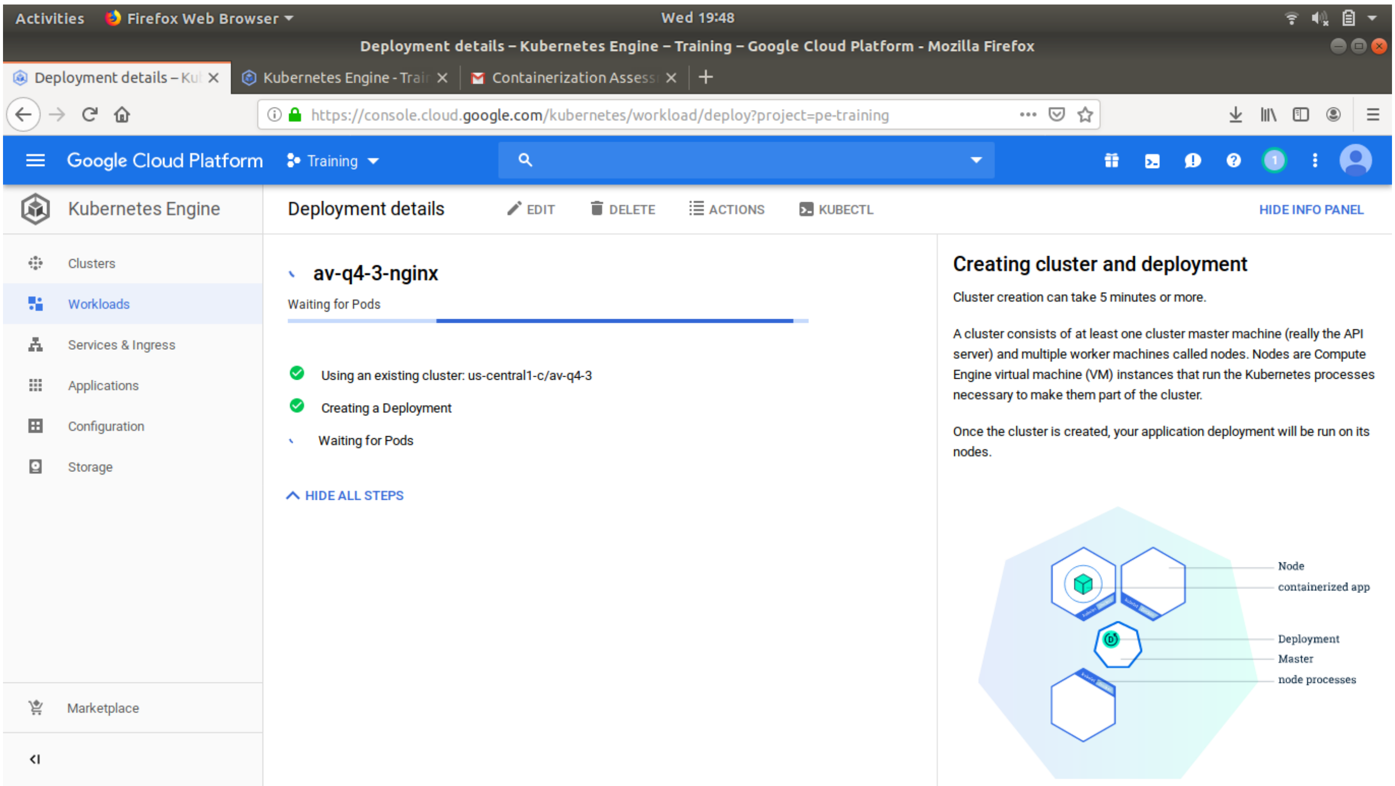
3. Create a Kubernetes Engine Cluster using the following command:



File Edit View Search Terminal Tabs Help
avinash@Quantiphi-929: ~/Documents/containerization/asgn/q4
avinash@Quantiphi-929: ~

```
avinash@Quantiphi-929:~$ gcloud beta container --project "pe-training" clusters create "av-q4-3" --zone "us-central1-c" --no-enable-basic-auth --clust
er-version "1.12.8-gke.10" --machine-type "n1-standard-1" --image-type "COS" --disk-type "pd-standard" --disk-size "100" --scopes "https://www.googlea
pis.com/auth/devstorage.read_only","https://www.googleapis.com/auth/logging.write","https://www.googleapis.com/auth/monitoring","https://www.googleapi
s.com/auth/servicecontrol","https://www.googleapis.com/auth/service.management.readonly","https://www.googleapis.com/auth/trace.append" --num-nodes "2
" --enable-cloud-logging --enable-cloud-monitoring --enable-ip-alias --network "projects/pe-training/global/networks/default" --subnetwork "projects/p
e-training/regions/us-central1/subnetworks/default" --default-max-pods-per-node "110" --addons HorizontalPodAutoscaling,HttpLoadBalancing --enable-aut
oupgrade --enable-autorepair
WARNING: Starting in 1.12, new clusters will not have a client certificate issued. You can manually enable (or disable) the issuance of the client cer
tificate using the `--[no-]issue-client-certificate` flag.
WARNING: Starting in 1.12, default node pools in new clusters will have their legacy Compute Engine instance metadata endpoints disabled by default. T
o create a cluster with legacy instance metadata endpoints disabled in the default node pool, run `clusters create` with the flag `--metadata disable-
legacy-endpoints=true`.
WARNING: The Pod address range limits the maximum size of the cluster. Please refer to https://cloud.google.com/kubernetes-engine/docs/how-to/flexible
-pod-cidr to learn how to optimize IP address allocation.
This will enable the autorepair feature for nodes. Please see https://cloud.google.com/kubernetes-engine/docs/node-auto-repair for more information on
 node autorepairs.
Creating cluster av-q4-3 in us-central1-c... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1beta1/projects/pe-training/zones/us-central1-c/clusters/av-q4-3].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-central1-c/av-q4-3?project=pe-training
kubeconfig entry generated for av-q4-3.
NAME     LOCATION       MASTER_VERSION  MASTER_IP     MACHINE_TYPE   NODE_VERSION   NUM_NODES  STATUS
av-q4-3  us-central1-c  1.12.8-gke.10   34.66.56.151  n1-standard-1  1.12.8-gke.10  2          RUNNING
avinash@Quantiphi-929:~$
```

4. Go to Kubernetes Engine > Workloads and create a deployment, select your image and click on continue.
5. Give the application a suitable name, select your cluster, fill-in other relevant details and click on DEPLOY.



av-q4-3-nginx

Waiting for Pods

Using an existing cluster: us-central1-c/av-q4-3

Creating a Deployment

Waiting for Pods

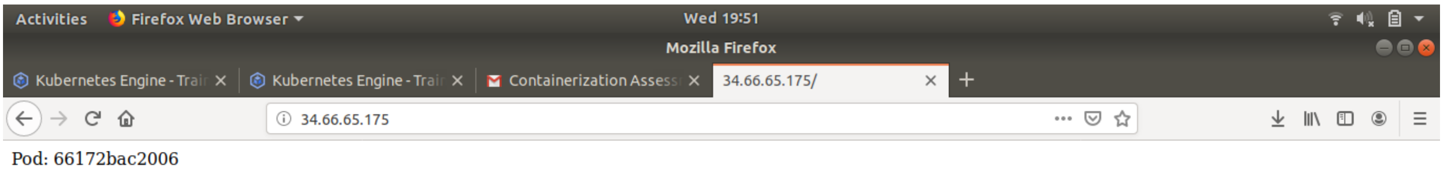
Creating cluster and deployment

Cluster creation can take 5 minutes or more.

A cluster consists of at least one cluster master machine (really the API server) and multiple worker machines called nodes. Nodes are Compute Engine virtual machine (VM) instances that run the Kubernetes processes necessary to make them part of the cluster.

Once the cluster is created, your application deployment will be run on its nodes.

6. In the deployent details page, click on actions and then click on expose. Set the port as 80 and select the service type as Cluster IP and click on EXPOSE.
7. Repeat step 6 twice, setting the service type as Node port and Load balancer.
8. Copy your LoadBalancer’s endpoint and paste it in your browser’s address bar.
9. You’ll see your pod’s container id. Refresh the page several times to see how the LB transfers the request to another pod.



Pod: 66172bac2006