In this lab, you create a Google Kubernetes Engine cluster containing several containers, each containing a web server. You place a load balancer in front of the cluster and view its contents.
In this lab, you learn how to perform the following tasks:

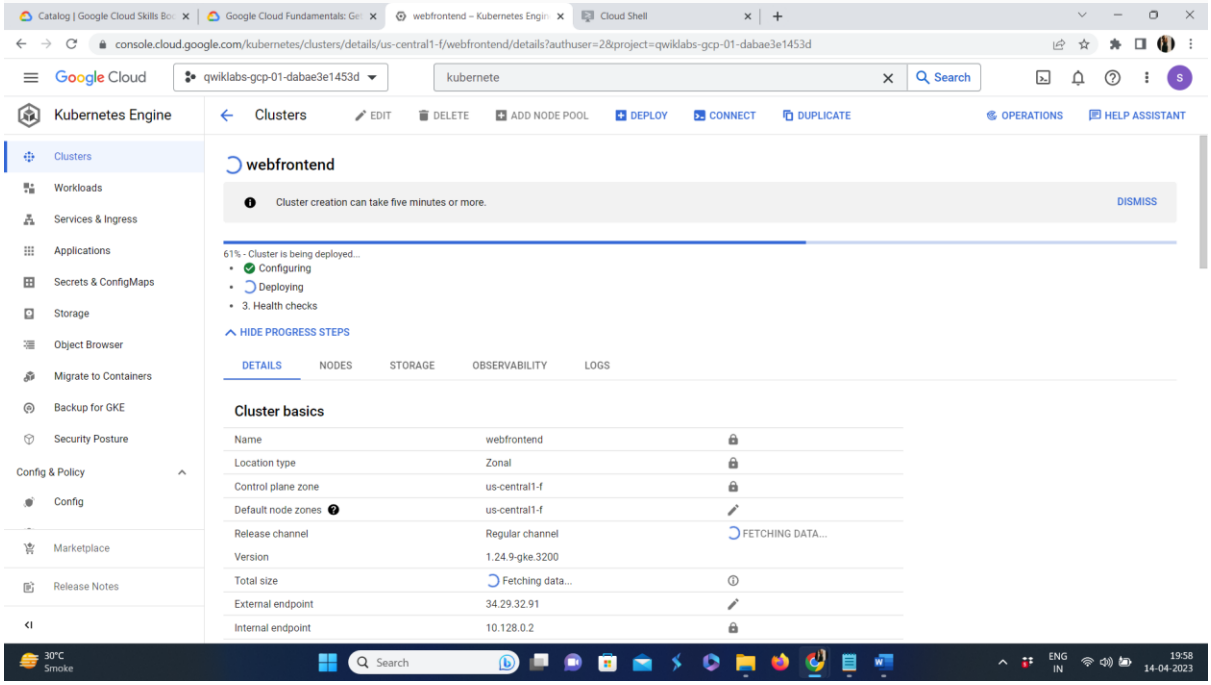
- Provision a Kubernetes cluster using Kubernetes Engine.
- Deploy and manage Docker containers using kubectl.

Task 2. Confirm that needed APIs are enabled

- Kubernetes Engine API
- Container Registry API

1. In Google Cloud console, on the top right toolbar, click the Activate Cloud Shell button.
2. export MY_ZONE="Zone"
3. Start a Kubernetes cluster managed by Kubernetes Engine. Name the cluster webfrontend and configure it to run 2 nodes: gcloud container clusters create webfrontend --zone $MY_ZONE --num-nodes 2

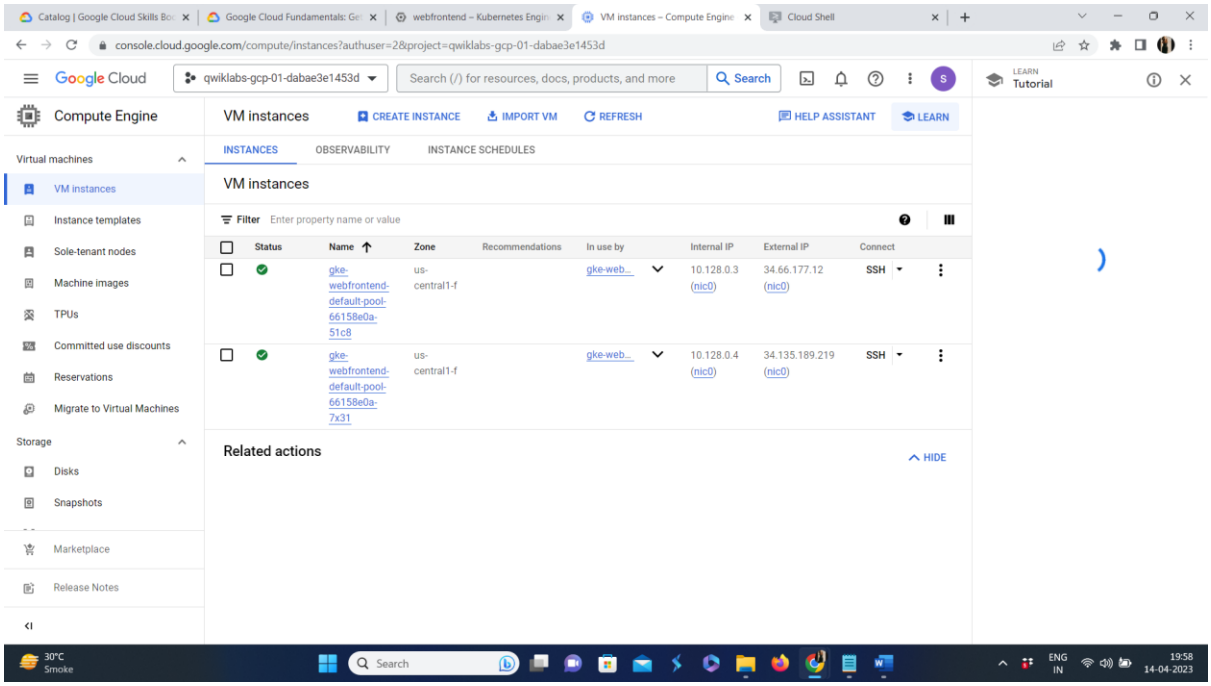
```
Your Cloud Platform project in this session is set to qwiklabs-gcp-01-dabae3e1453d.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ export zone=us-central1-f
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ gcloud container clusters create webfrontend --num-nodes 2 --zone $zone
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters,
please pass the `--no-enable-ip-alias` flag
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a default location policy
 is applied. For Spot and PVM it defaults to ANY, and for all other VM kinds a BALANCED policy is used. To change the default values use the `--location-pol
icy` flag.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).

Creating cluster webfrontend in us-central1-f... Cluster is being configured...working.
```



4. After the cluster is created, check your installed version of Kubernetes using the kubectl version command: kubectl version

```
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl version
WARNING: This version information is deprecated and will be replaced with the output from kubectl version --short.  Use --output=yaml|json to get the full v
ersion.
Client Version: version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.3", GitCommit:"9e644106593f3f4aa98f8a84b23db5fa378900bd", GitTreeState:"clean", BuildD
ate:"2023-03-15T13:40:17Z", GoVersion:"go1.19.7", Compiler:"gc", Platform:"linux/amd64"}
Kustomize Version: v4.5.7
Server Version: version.Info{Major:"1", Minor:"24", GitVersion:"v1.24.9-gke.3200", GitCommit:"92ea556d4e7418d0e7b5db1ee576a73f8fc47e91", GitTreeState:"clean
", BuildDate:"2023-01-20T09:29:29Z", GoVersion:"go1.18.9b7", Compiler:"gc", Platform:"linux/amd64"}
WARNING: version difference between client (1.26) and server (1.24) exceeds the supported minor version skew of +/-1
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$
```

5. View your running nodes in the GCP Console. On the Navigation menu (Navigation menu icon), click Compute Engine > VM Instances.



6. Your Kubernetes cluster is now ready for use.

Task 4. Run and deploy a container

1. From your Cloud Shell prompt, launch a single instance of the nginx container. (Nginx is a popular web server.) : kubectl create deploy nginx --image=nginx:1.17.10 - In Kubernetes, all containers run in pods. This use of the kubectl create command caused Kubernetes to create a deployment consisting of a single pod containing the nginx container. A Kubernetes deployment keeps a given number of pods up and running even in the event of failures among the nodes on which they run. In this command, you launched the default number of pods, which is 1.

```
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl create deploy nginx --image=nginx:1.17.10
deployment.apps/nginx created
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$
```

2. View the pod running the nginx container: kubectl get pods

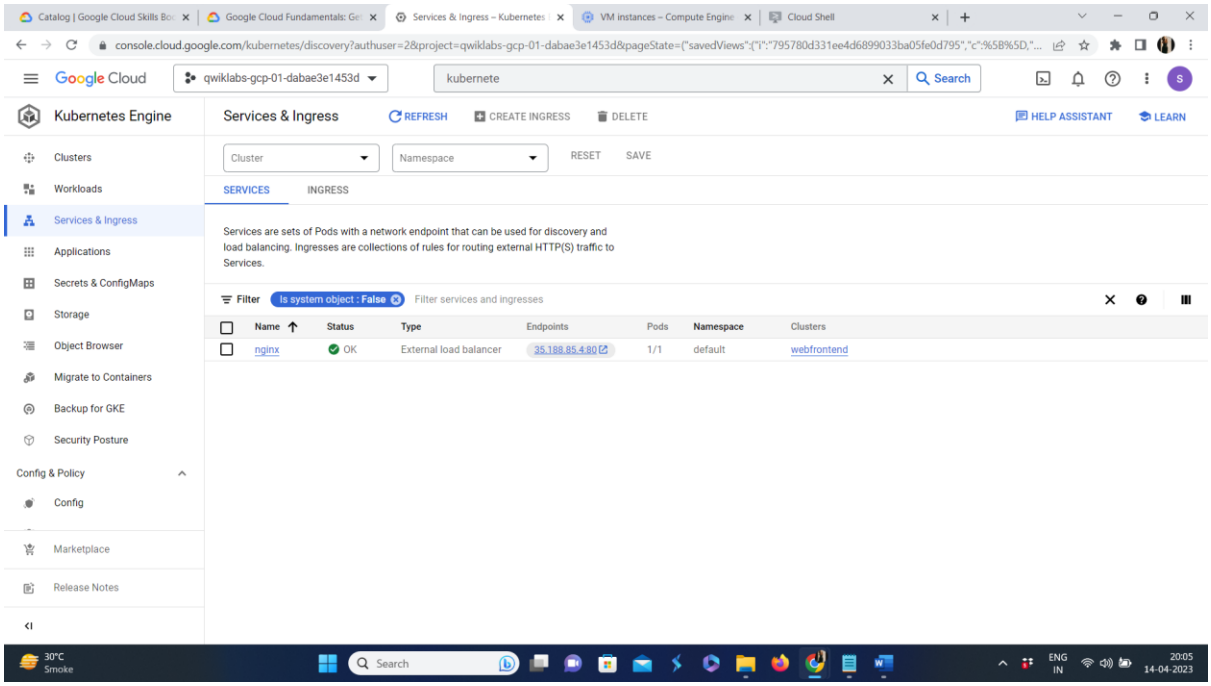
```
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-5fc59799db-47b4w   1/1     Running   0          20s
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$
```

3. Expose the nginx container to the Internet: kubectl expose deployment nginx --port 80 --type LoadBalancer - Kubernetes created a service and an external load balancer with a public IP address attached to it. The IP address remains the same for the life of the service. Any network traffic to that public IP address is routed to pods behind the service: in this case, the nginx pod.

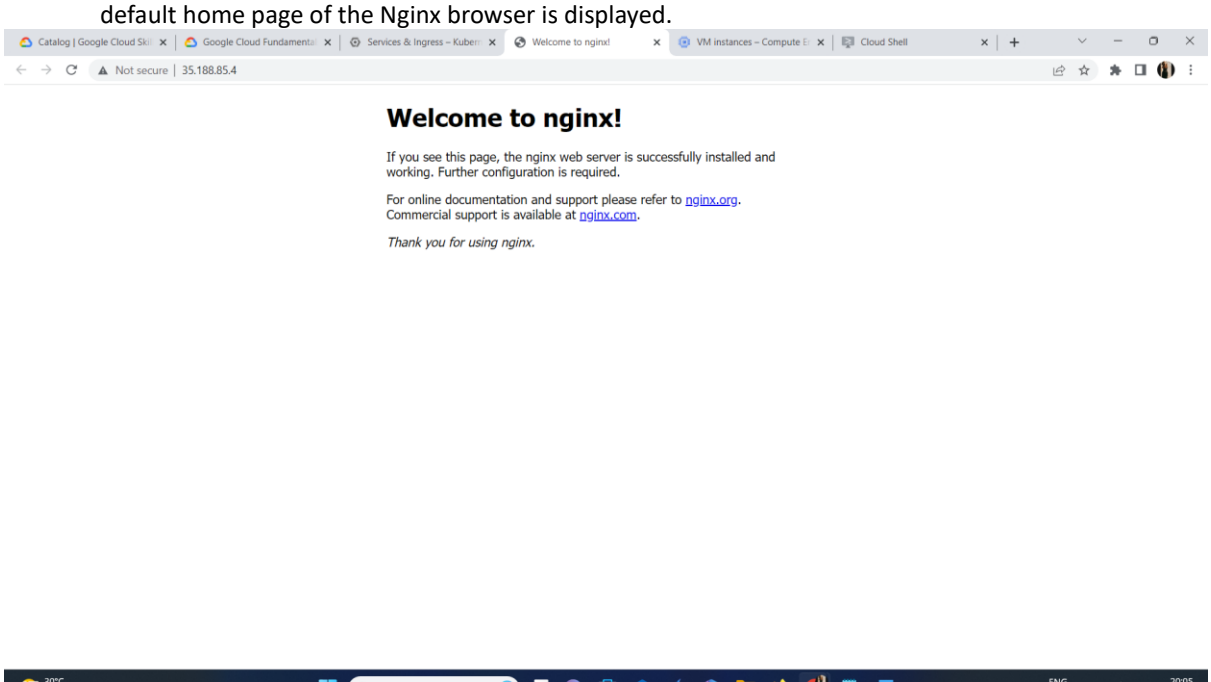
```
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl expose deployment nginx --port 80 --type LoadBalancer
service/nginx exposed
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           48s
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$
```

4. View the new service: kubectl get services - You can use the displayed external IP address to test and contact the nginx container remotely.

```
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl get services
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP      10.16.0.1      <none>        443/TCP        5m10s
nginx        LoadBalancer   10.16.10.137   <pending>     80:32163/TCP   31s
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$
```



5. Open a new web browser tab and paste your cluster's external IP address into the address bar. The default home page of the Nginx browser is displayed.



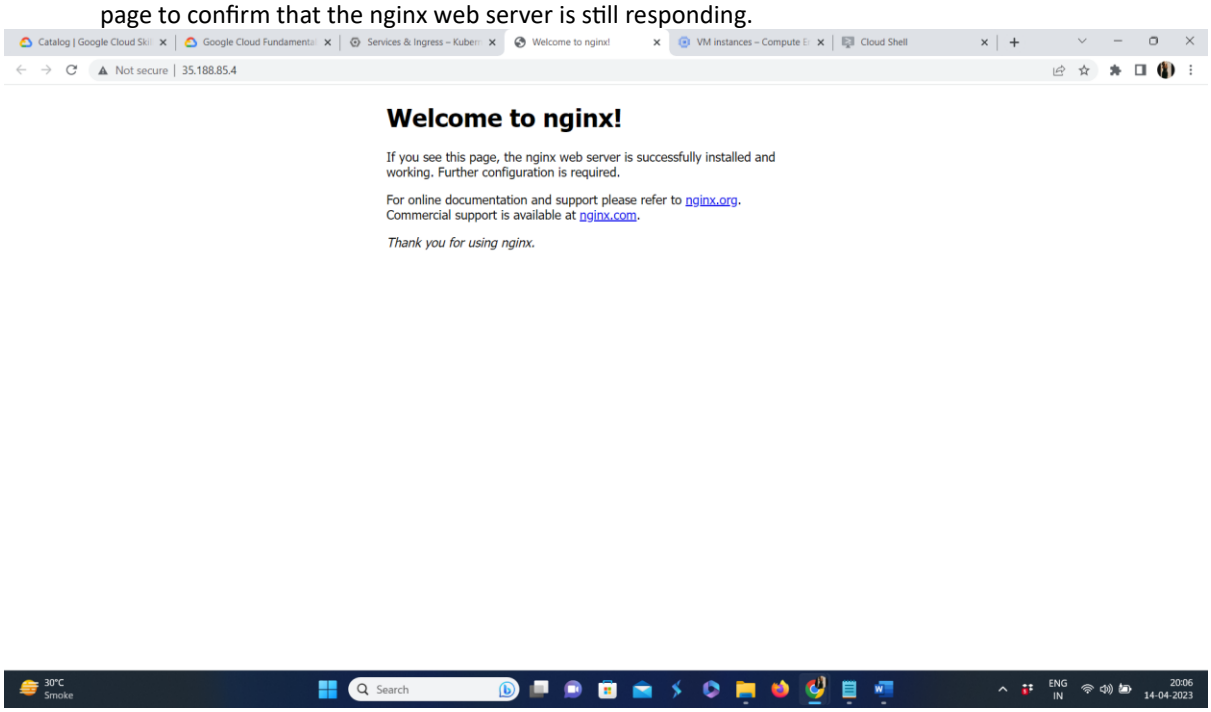
6. Scale up the number of pods running on your service: kubectl scale deployment nginx --replicas 3 - Scaling up a deployment is useful when you want to increase available resources for an application that is becoming more popular.

```
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl scale deployment nginx --replicas 3
deployment.apps/nginx scaled
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl get pods
NAME                     READY   STATUS              RESTARTS   AGE
nginx-5fc59799db-47b4w   1/1     Running             0          2m21s
nginx-5fc59799db-59ss8   0/1     ContainerCreating   0          5s
nginx-5fc59799db-dbfcl   1/1     Running             0          5s
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$
```

7. Confirm that Kubernetes has updated the number of pods: kubectl get pods
8. Confirm that your external IP address has not changed: kubectl get services

```
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$ kubectl get services
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP      10.16.0.1      <none>         443/TCP        6m39s
nginx        LoadBalancer   10.16.10.137   35.188.85.4    80:32163/TCP   2m
student_00_84da249777ed@cloudshell:~ (qwiklabs-gcp-01-dabae3e1453d)$
```

9. Return to the web browser tab in which you viewed your cluster's external IP address. Refresh the page to confirm that the nginx web server is still responding.



In this lab, you configured a Kubernetes cluster in Kubernetes Engine. You populated the cluster with several pods containing an application, exposed the application, and scaled the application.