

Kubernetes has emerged as a powerful tool to manage and scale cloud-native applications.

A service type determines how the service is exposed to the network.

There are mainly three types of services that Kubernetes supports:

- ClusterIP - The default service that allows multiple pods in the cluster to communicate with one another.
- NodePort - It simply routes traffic from a random host port to a random container port.
- LoadBalancer - It runs on each pod and connects to the outside world, either through networks such as the Internet or within your datacenter.

*Ingress - is not considered an official Kubernetes service , However, we can configure an Ingress service by writing rules that specify which inbound connections should be routed to which services.*

Here, we are going to see the hands-on about the service type - Load Balancer.

- ☐ Load Balancer services connect our applications to the outside world, and they are used in production environments where high availability and scalability are critical. It will keep connections open to pods that are up, and close connections to those that are down.
- ☐ Load Balancer services are ideal for applications that must handle high traffic volumes, such as web applications or APIs.

☐ We can access our application using the load balancer's single IP address.

☐ When the Service type is set to LoadBalancer, Kubernetes provides functionality equivalent to type equals ClusterIP to pods within the cluster and extends it by programming the (external to Kubernetes) load balancer with entries for the Kubernetes pods.

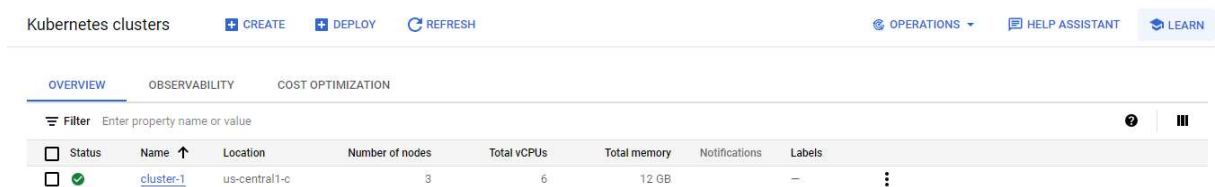
Let's get started.....

[ Create a Google Cloud Account to do this activity ]

## 1. Create Google Kubernetes cluster engine

Search for Kubernetes Engine in Google Cloud console and create a cluster.  
Select Standard Cluster and create with default settings.

From below image we can confirm with green tick mark that the cluster is activated.



The screenshot shows the 'Kubernetes clusters' page in the Google Cloud console. At the top, there are buttons for 'CREATE', 'DEPLOY', and 'REFRESH'. On the right, there are links for 'OPERATIONS', 'HELP ASSISTANT', and 'LEARN'. Below these, there are tabs for 'OVERVIEW', 'OBSERVABILITY', and 'COST OPTIMIZATION'. A filter bar is present with the text 'Filter Enter property name or value'. The main table lists the clusters with columns: Status, Name, Location, Number of nodes, Total vCPUs, Total memory, Notifications, and Labels. One cluster is listed with a green checkmark in the Status column, indicating it is active.

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	cluster-1	us-central1-c	3	6	12 GB	—	⋮

## 2 - Connect to the Command-line access

Click on the three dots in RHS and select Connect , so we will get a page like below:

## Connect to the cluster

You can connect to your cluster via command-line or using a dashboard.

### Command-line access

Configure [kubectl](#) command line access by running the following command:

```
$ gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project teak-spot-394123
```

[RUN IN CLOUD SHELL](#)

### Cloud Console dashboard

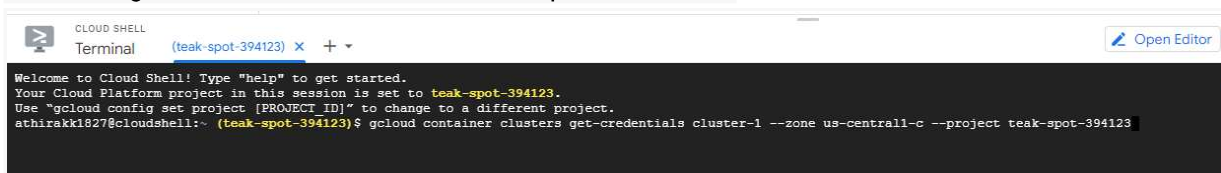
You can view the workloads running in your cluster in the Cloud Console [Workloads dashboard](#).

[OPEN WORKLOADS DASHBOARD](#)

OK

Copy the code displayed above in command-line access : “`gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project teak-spot-394123`” and click on “`RUN CLOUD SHELL`”.

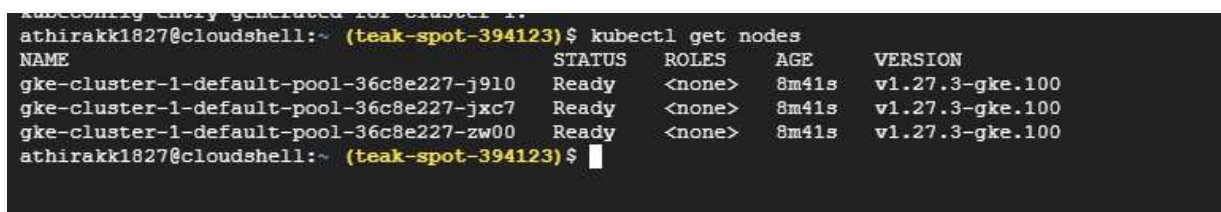
We will get a window like this and press enter.



```
Cloud Shell
Terminal (teak-spot-394123) x +
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to teak-spot-394123.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
athirakk1827@cloudshell:~ (teak-spot-394123) $ gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project teak-spot-394123
```

3 - Confirm all the nodes are in Ready state by running below command:

`kubectl get nodes`



```
athirakk1827@cloudshell:~ (teak-spot-394123) $ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
gke-cluster-1-default-pool-36c8e227-j910 Ready    <none>    8m41s  v1.27.3-gke.100
gke-cluster-1-default-pool-36c8e227-jxc7 Ready    <none>    8m41s  v1.27.3-gke.100
gke-cluster-1-default-pool-36c8e227-zw00 Ready    <none>    8m41s  v1.27.3-gke.100
athirakk1827@cloudshell:~ (teak-spot-394123) $
```

4 - Run the below command to create Namespace called facebook

`kubectl create ns facebook`

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ kubectl create ns facebook
namespace/facebook created
```

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ kubectl get ns
NAME          STATUS   AGE
default       Active   57m
facebook      Active   46s
gmp-public    Active   56m
gmp-system    Active   56m
kube-node-lease Active   57m
kube-public   Active   57m
kube-system   Active   57m
athirakk1827@cloudshell:~ (teak-spot-394123)$
```

5 - Below is the yaml file we are going to create for load balancer demo: Add this file in your Github repository and take the URL as well.

YAML File

=====

---

```
apiVersion: v1
kind: Pod
metadata:
  name: webserver
  namespace: facebook
  labels:
    role: web-service
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

---

```
apiVersion: v1
kind: Service
metadata:
  name: web-service
  namespace: facebook
  labels:
    role: web-service
spec:
  selector:
```

role: web-service  
type: LoadBalancer  
ports:  
- port: 80

And apply the below yaml file from github by using below command:

```
kubectl apply -f  
https://raw.githubusercontent.com/athlearn/kub/main/lb-facebook.yaml
```

```
athirakki1827@cloudshell:~ (teak-spot-394123)$ kubectl apply -f https://raw.githubusercontent.com/athlearn/kub/main/lb-facebook.yaml  
pod/webserver created  
service/web-service created  
athirakki1827@cloudshell:~ (teak-spot-394123)$
```

We can see from above output that , pod and service have been created

Run the below command to see how the load balancer service type is working.

```
watch -n 1 kubectl get all -n facebook -o wide
```

```
Every 1.0s: kubectl get all -n facebook -o wide  
NAME          READY   STATUS    RESTARTS   AGE   IP        NODE                                     NOMINATED NODE   READINESS GATES  
pod/webserver 1/1     Running   0           4m6s  10.4.2.7  gke-cluster-1-default-pool-36c8e227-jxc7 <none>           <none>  
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE   SELECTOR  
service/web-service LoadBalancer  10.8.15.76    34.134.69.179  80:30895/TCP     4m5s  role=web-service
```

So we're launching a pod, which will expose our application to the outside world which we can access through the external IP mentioned in the below screenshot.

Copy the IP and paste in the browser , here is the output.....

34.134.69.179

## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

