29th June

- What is Kubernetes?
- Understanding Kubernetes Architecture
- Components of Kubernetes Master
    - kube api server
    - kube scheduler
    - controller manager
    - ectd
- Components of Kubernetes Node
    - Kubelet
    - kube-proxy
    - container engine

Kubernetes

It is a container orchestration tool.

Docker swarm and k8 is container orchestration tools, whatever advantges of doker swram as same for k8.

K8 is a Google created this kubernetes.It is a opensource tool.Pre-requisite is docker.

k8s -- 8 letters between k and s

Kubernetes create, deploy and manage clusters.

Cluster: master+nodes combination is called.

By using kubernetes we form a cluster

K8S schedules, runs and managers isolated containers.

Convert isolated containers running on different hardware's into a cluster.

In AWS we have a service EKS (Elastic kuberneters service)

Can create cluster, manage clusters, and experience orchestration.

Features of kubernetes

----

1) Orchestration (clustering any no of containers on different hardware's)

2) Auto scaling- handling failures

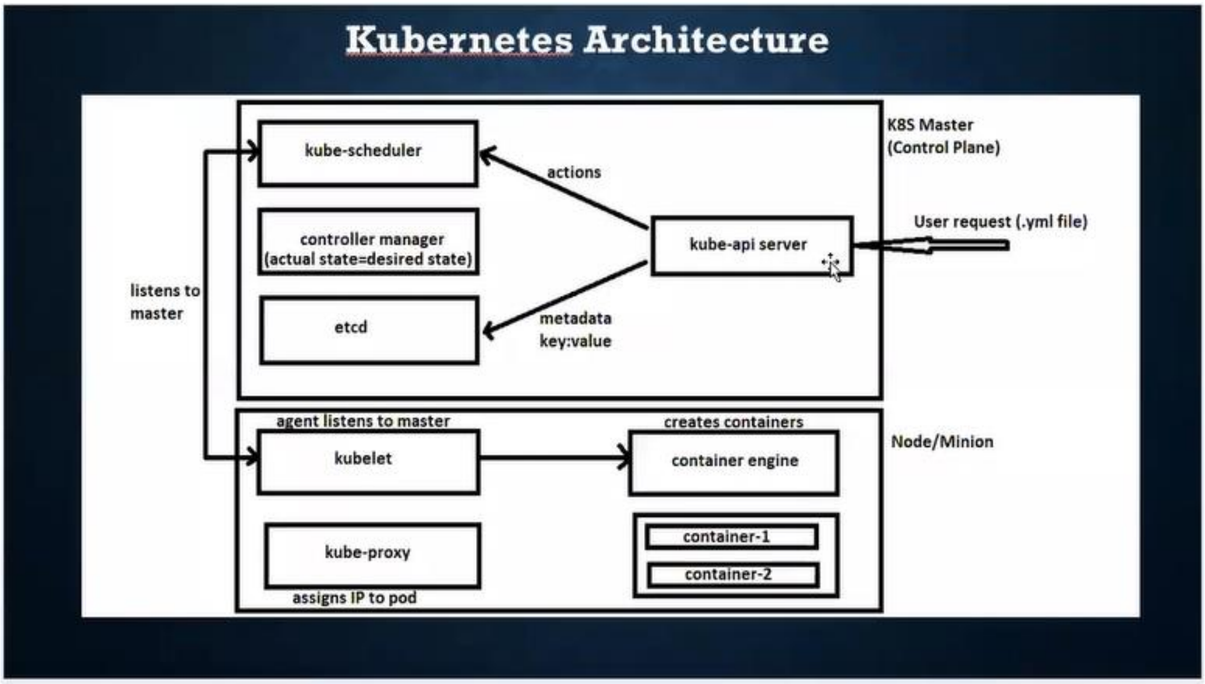
3) Auto healing –handling failures (new containers in place of crashed containers

similar to handling failover scenarios in docker swarm)

4) load balancing

5) rollback (going to previous versions)

Kubernetes Architecture



Above diagram

One master and one node there

Cluster is combination of 1 master and multiple nodes.

Pod is atomic unit of deployment in kubernetes.

(k8 is a orchestration toll cant contain container , the pod can create container in k8 to run application)

Every container is cretaed Inside pod only.

Pod created by k8

Container will create by Docker.

Pod consists of one or more docker containers.

Pod runs on node. Available in node only

Node is controlled by Kubernets master

Kubernetes does not understand containers.

Kubernetes can understand only pods.

In this diagram , we have one master and one node.

node is also called minion.

Kubernetes master is also called as control plane.

Only one master in k8, we don't have multiple masters

++

Con Orches Containerization

----------- ------------------

Dockerswarm ---------------------------> Docker

Kubernetes ------------------------------- Docker / XYZ

Master contains 4 components/ services

1) kube api server

2) kube scheduler

3) controller manager (acutal state = desired state)

4) ectd

definition files: As a devops engineer you create a yaml file (.yml) file is also called definition files.

What this yaml file contains?

1) No of nodes you want?

2) Each node should have how many pods

3) Each pod should contain how many containers, containers based on which image and name.

All the above information will be available in yaml file.

This file is also called manifest file.

This document should be provided to kuberneted master.

+++++++++

kube api server acts like a receptionist.

It receives the yaml file and pass the request to kube scheduler.

(we use yaml or commands)

++++++++

kube scheduler will take the action.

So kube scheduler will create pods and containers.

++++++++

What is etcd?

Etcd is also called cluster store.

It has the information of the complete cluster.

It is used to store the data of master, node and containers.

Data is stored in key-value pair.

Ex: pod name, howmany pods,container name,no. of containers ,image name

+++++++++++

What is controller manager?

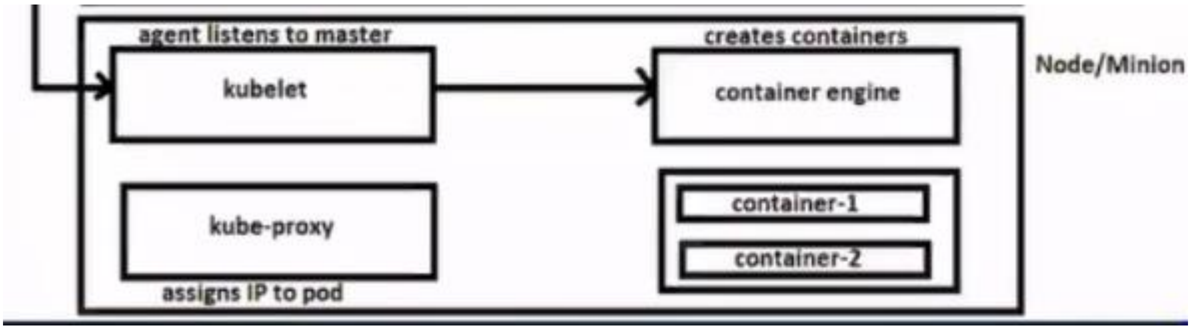
It is responsibile to make sure that the actual state is same as desired state.

desired state= definition file

desired state is requirement—yamlfile→no.of pods→make sure pods running→if pod down actual state is not same as desired satae bcz 4pods running →control manger identified bcz it always monitoring the infrastructure whether the actual state equal to desired satae or not→cm inform to kube scheduler it performs the action accordinglt to get back the pod is back to set actual state is desired state.

These four components together called as control plane.

++++++++++



Kubernetes node

Node container 3 components

1) kubelet

2) kube-proxy

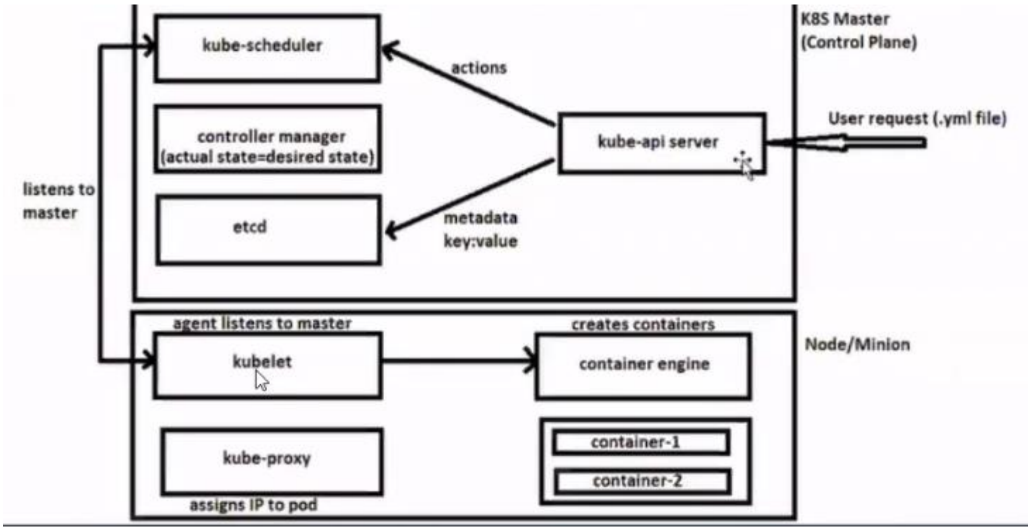
3) container engine(docker)

Kubelet -- is also called as agent, as it listens to kubernetes master.

kube-scheduler component communicates to kubelet.

kubelet communicates to container engine (docker) so that containers are created.

Note: Containers are created in pods



Imp: kubelet present in the node and kube scheduler present in the master machine these 2 components coordinate with each other to create the infrastructure.

So called as agent, which listens to kubernetes master.

kube proxy --

It will provide IP Address to pod.

Every pod has an ip.

+++++++++++++++++++++++++++

Kubernetes Terminology

In docker Swarm, Manager Machine takes the load.

In Kubernetes Manager is called as Master.

Kubernetes master does not take up the load because pods /containers will not run on the master. The complete load taken by the node.

It only distributes load to slaves/ nodes.

Nodes are also called Minion.

Minions combined together is called as cluster.

Smallest Object that kubernetes can create is pod.

(pod as same as container , they are similar so pod=conatainer)

Within the pod, we have the container.

Kubernetes commands are always triggered using kubectl.

Kubernetes introduced on June 2014 by Google.

To practice Kubernetes on AWS , we have a service EKS (Elastic Kubernetes Service)

To practice Kubernetes on Azure , we have a service AKS (Azure Kubernetes Service)

To practice Kubernetes on GCP , we have a service GKE (google Kubernetes engine)

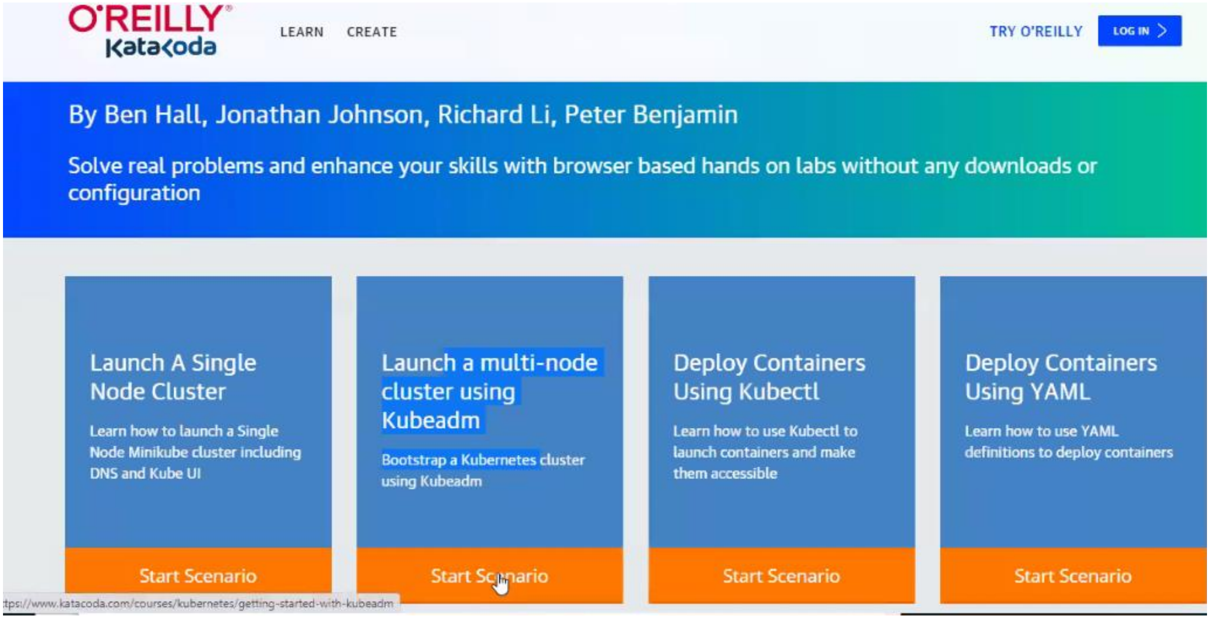
AWS, is expensive

Freeways to work on kubernetes is katakoda

Goto https://www.katacoda.com/

Learn --- --- Kubernetes Introduction -- Start Course

-- Launch Multinode cluster -- Start Scenario



By Ben Hall, Jonathan Johnson, Richard Li, Peter Benjamin

Solve real problems and enhance your skills with browser based hands on labs without any downloads or configuration

Launch A Single Node Cluster

Learn how to launch a Single Node Minikube cluster including DNS and Kube UI

Start Scenario

Launch a multi-node cluster using Kubeadm

Bootstrap a Kubernetes cluster using Kubeadm

Start Scenario

Deploy Containers Using Kubectl

Learn how to use Kubectl to launch containers and make them accessible

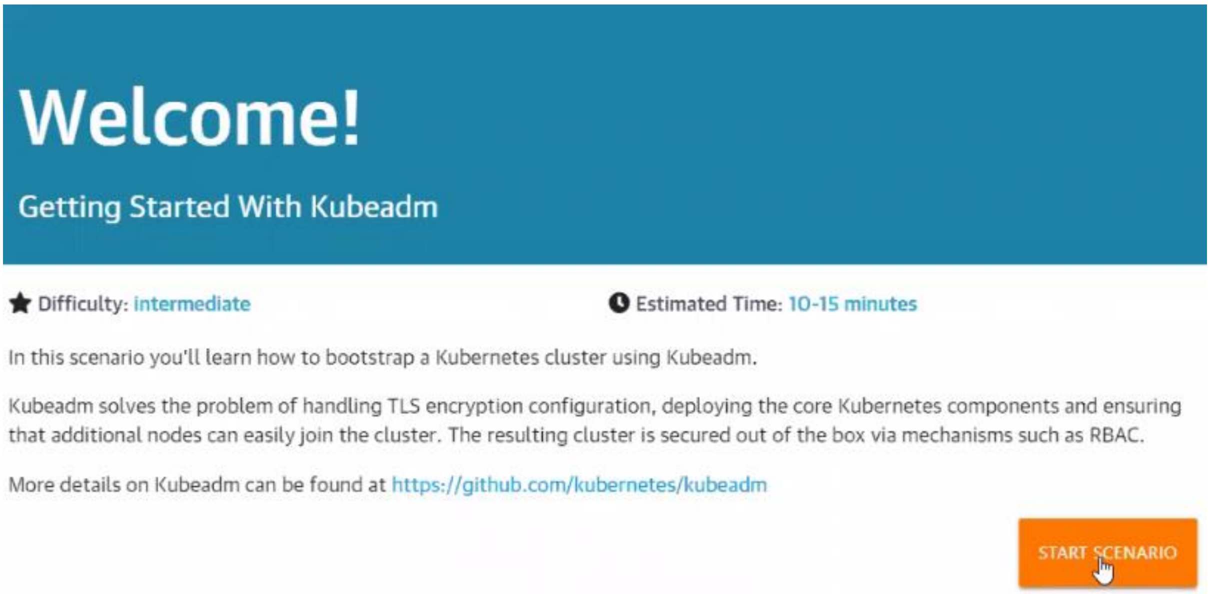
Start Scenario

Deploy Containers Using YAML

Learn how to use YAML definitions to deploy containers

Start Scenario

tps://www.katacoda.com/courses/kubernetes/getting-started-with-kubeadm



Welcome!

Getting Started With Kubeadm

★ Difficulty: intermediate

Estimated Time: 10-15 minutes

In this scenario you'll learn how to bootstrap a Kubernetes cluster using Kubeadm.

Kubeadm solves the problem of handling TLS encryption configuration, deploying the core Kubernetes components and ensuring that additional nodes can easily join the cluster. The resulting cluster is secured out of the box via mechanisms such as RBAC.

More details on Kubeadm can be found at https://github.com/kubernetes/kubeadm

Login using gmail

Step 1: Initialise Master

Run kubeadm init command (just click on it)

We need to copy configuration files to home directory and change ownership.

Run sudo cp command.

Continue

Step 2: Deploy Container networking Interface

Run the three commands

cat , kubectl apply, kubectl get pod

Continue

Step 3:

Run

kubeadm token list

kubeadm join (this will create slave)

Continue

Step 4:

Run

kbectl get nodes

You can see one controlplane and one node

We have one more site

https://labs.play-with-k8s.com/

using which we can practice Kubernetes.

But, both the options will be slow.

+++++++++++++++++++++++++++++++++

We learn kubernetes on GCP, as AWS is expensive.

Sign up to GCP account using gmail credentials. (Free trial comes with USD 300)

https://cloud.google.com/

Sign in using gmail

Click on console

You will enter into google cloud platform console

Navigation Menu --- Kubernetes Engine -- Clusters -- Create cluster -- Create

Observation: Cluster size is 3

By default, it creates 3 node cluster.

Master Machine is not provided as alinux server.

It is given as a service.

As it is a service, it never fail.

So, we do not need to worry about master.

To connect to the cluster

In GCP, Cloud Shell is the terminal, used to connect to the cluster.

kubectl get nodes (we can see the nodes)

After practice, Delete the cluster.

Next day, we can create the cluster again.

30th June

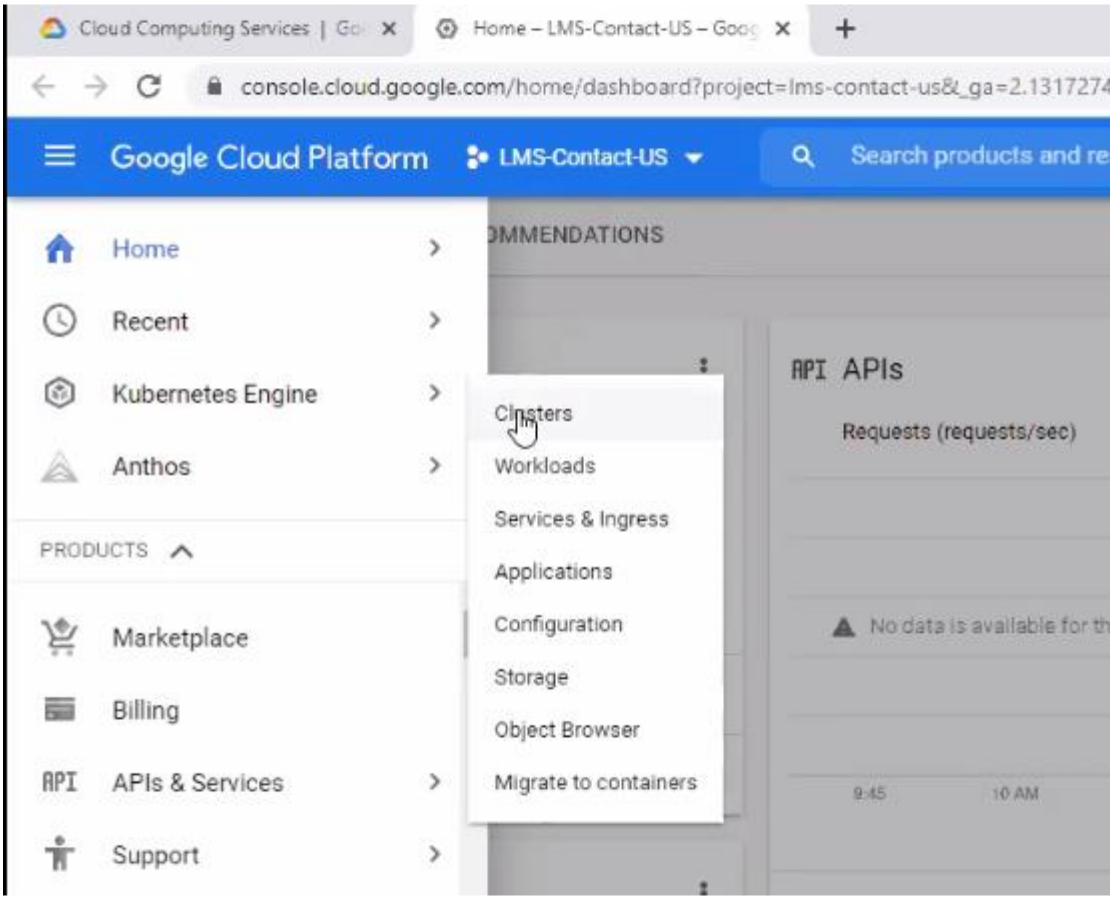
https://cloud.google.com/

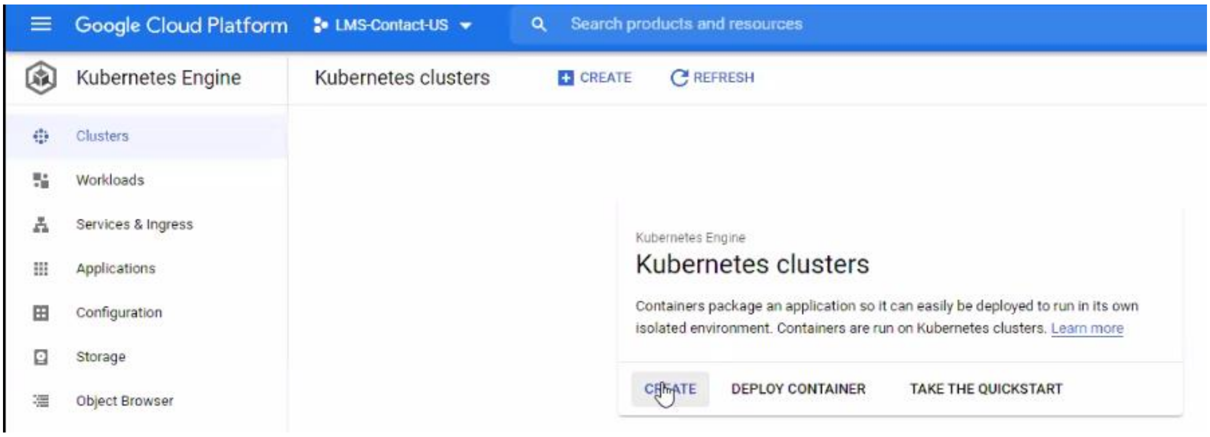
Sign in using gmail

Click on console

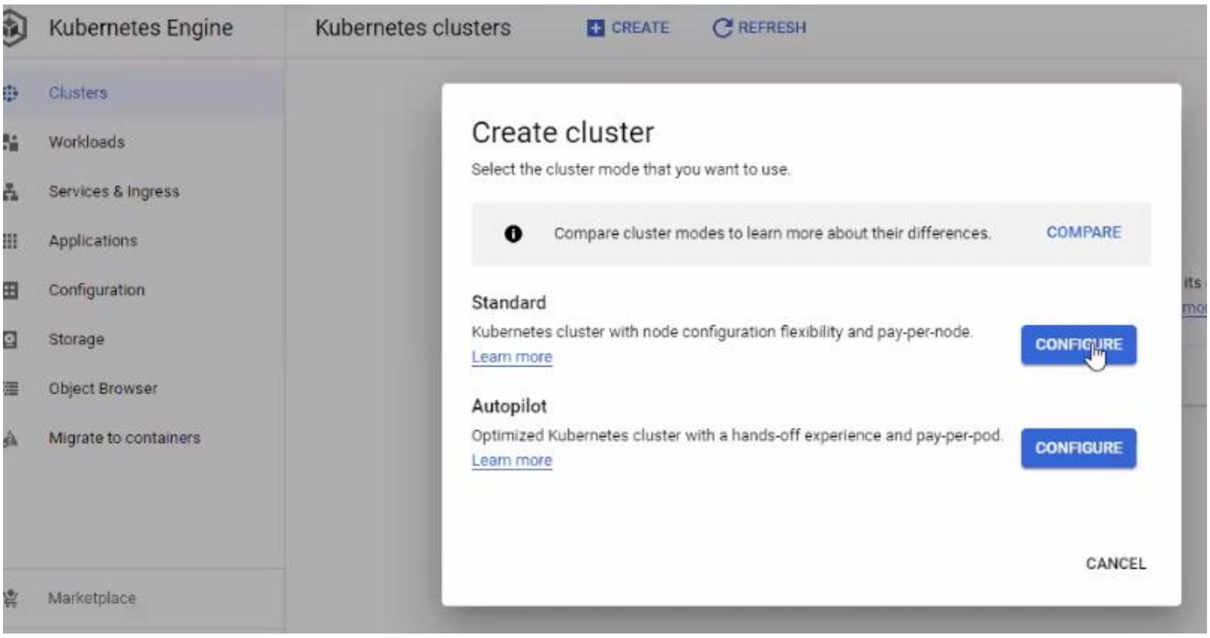
You will enter into google cloud platform console

Navigation Menu --- Kubernetes Engine -- Clusters -- Create cluster -- Create





Containers package an application so it can easily be deployed to run in its own isolated environment. Containers are run on Kubernetes clusters. Learn more



Create cluster

Select the cluster mode that you want to use.

Compare cluster modes to learn more about their differences. COMPARE

Standard

Kubernetes cluster with node configuration flexibility and pay-per-node. Learn more

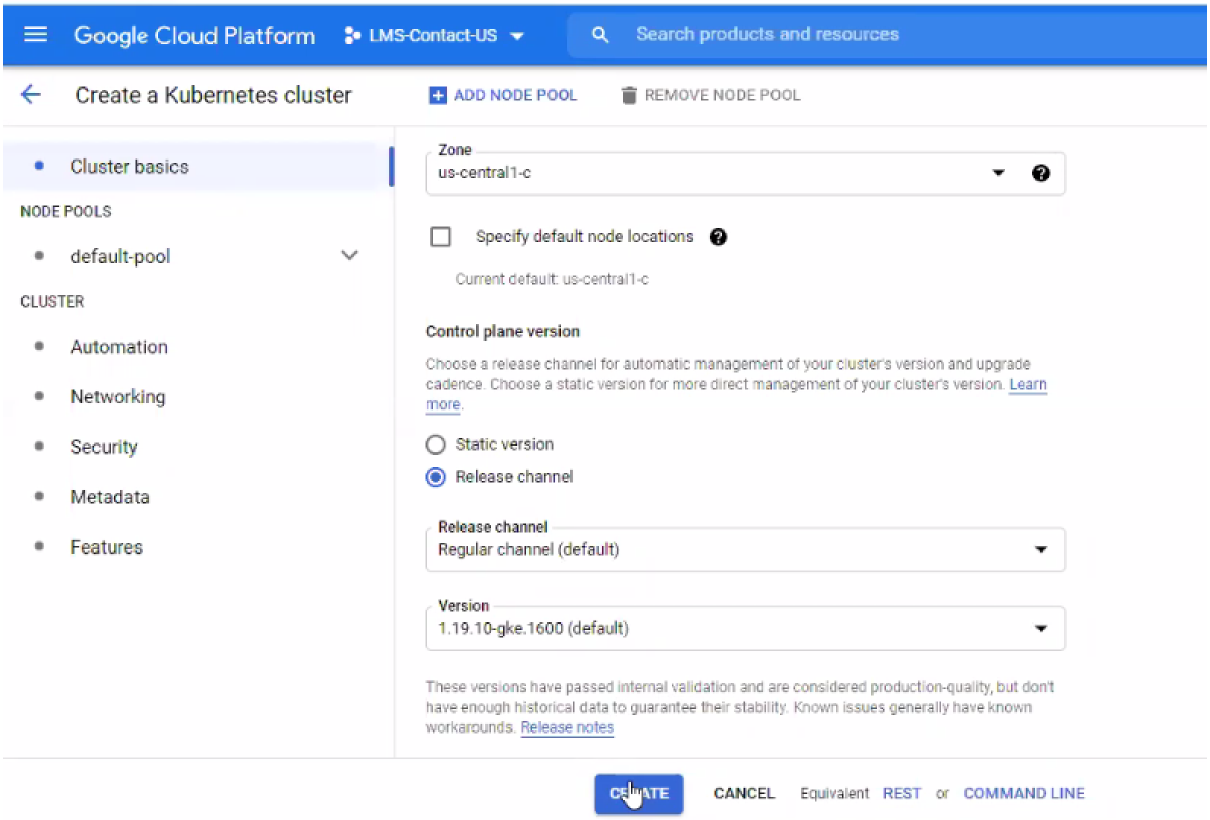
CONFIGURE

Autopilot

Optimized Kubernetes cluster with a hands-off experience and pay-per-pod. Learn more

CONFIGURE

CANCEL



Zone: us-central1-c

□ Specify default node locations

Current default: us-central1-c

Control plane version

Choose a release channel for automatic management of your cluster's version and upgrade cadence. Choose a static version for more direct management of your cluster's version. Learn more.

○ Static version

● Release channel

Release channel: Regular channel (default)

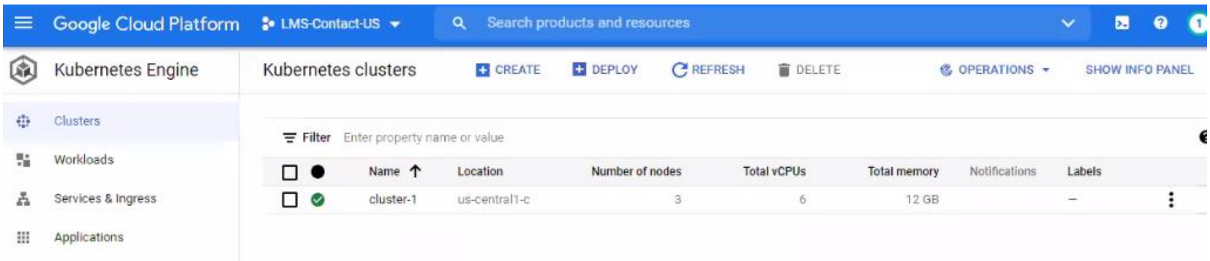
Version: 1.19.10-gke.1600 (default)

These versions have passed internal validation and are considered production-quality, but don't have enough historical data to guarantee their stability. Known issues generally have known workarounds. Release notes

CREATE CANCEL Equivalent REST or COMMAND LINE

Observation: Cluster size is 3

By default, it creates 3 node cluster.



☐	●	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
☐	✓	cluster-1	us-central1-c	3	6	12 GB		–

Master Machine is not provided as a linux server

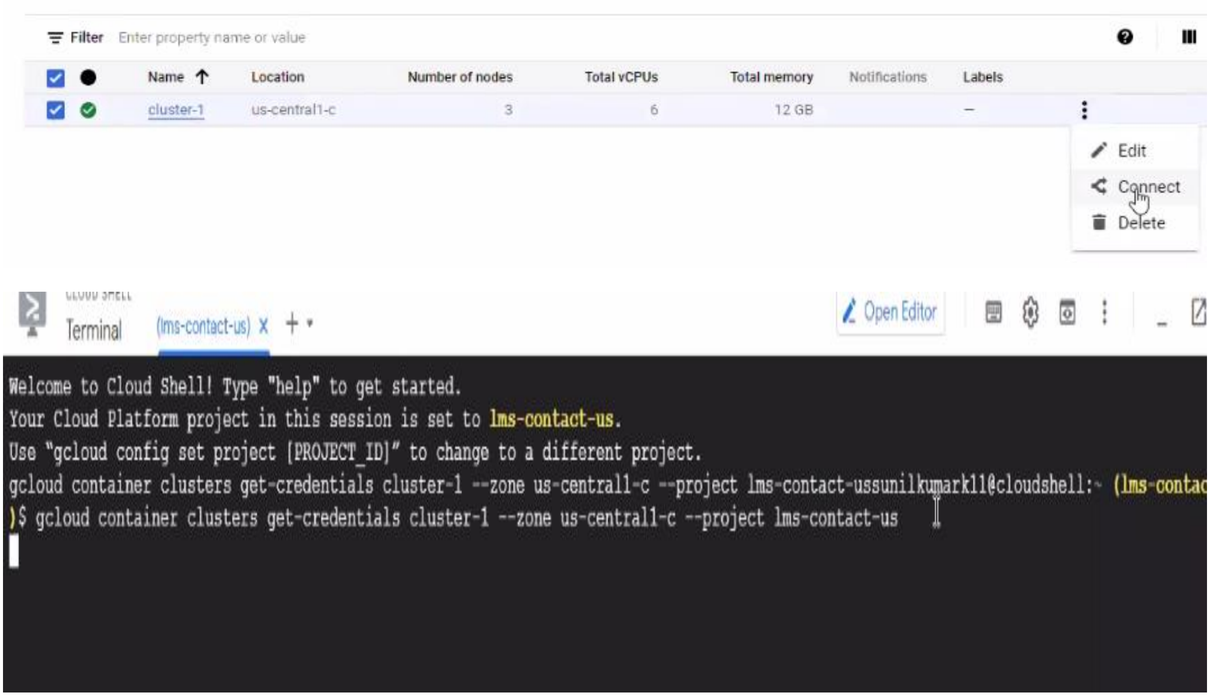
It is given as a service.

As it is a service, it never fail.

So, we do not need to worry about master.

To connect to the cluster

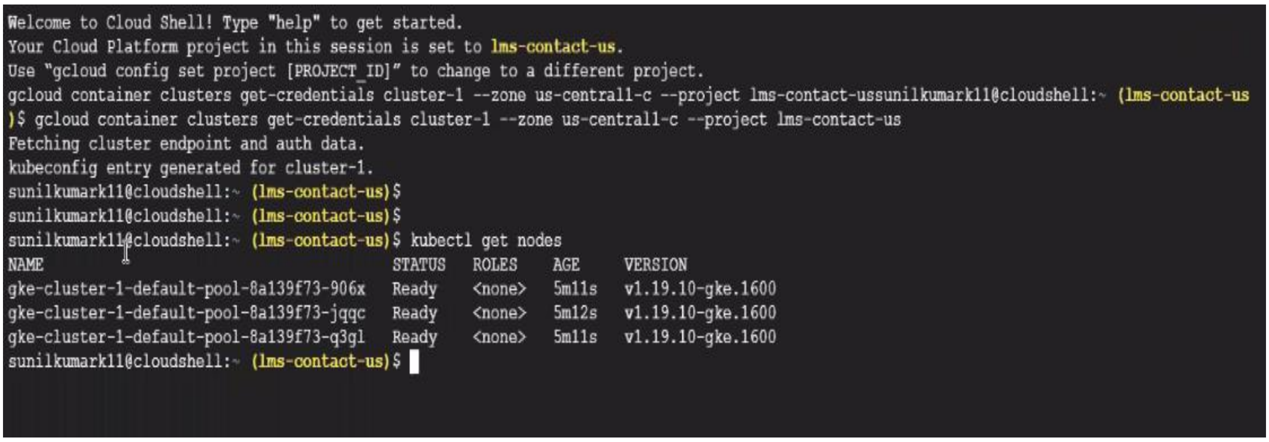
In GCP, Cloud Shell is the terminal, used to connect to the cluster.



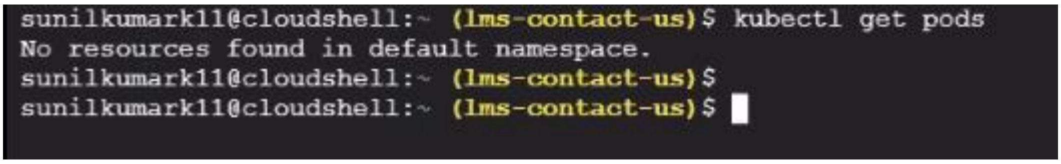
☑	●	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
☑	✓	cluster-1	us-central1-c	3	6	12 GB		–

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to lms-contact-us.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project lms-contact-ussunilkumark11@cloudshell:~ (lms-contac
)$ gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project lms-contact-us
```

kubectl get nodes (we can see the nodes)



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to lms-contact-us.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project lms-contact-ussunilkumark11@cloudshell:~ (lms-contact-us
)$ gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project lms-contact-us
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster-1.
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get nodes
NAME                                       STATUS   ROLES    AGE    VERSION
gke-cluster-1-default-pool-8a139f73-906x   Ready    <none>   5m11s  v1.19.10-gke.1600
gke-cluster-1-default-pool-8a139f73-jqqc   Ready    <none>   5m12s  v1.19.10-gke.1600
gke-cluster-1-default-pool-8a139f73-q3gl   Ready    <none>   5m11s  v1.19.10-gke.1600
sunilkumark11@cloudshell:~ (lms-contact-us)$
```



```
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get pods
No resources found in default namespace.
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$
```

After practice, Delete the cluster.

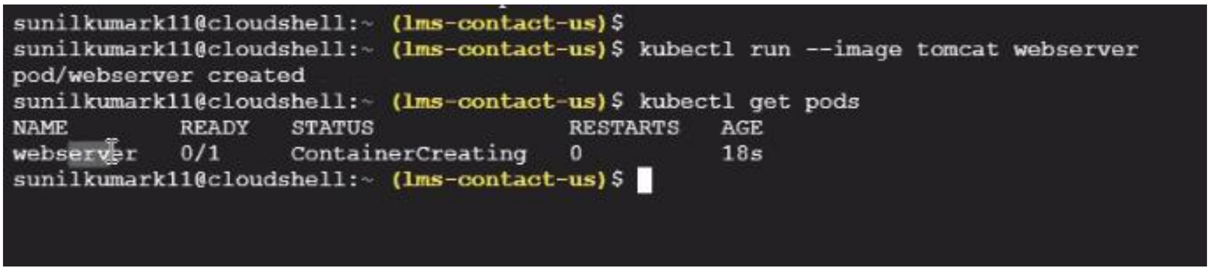
Next day, we can create the cluster again.

To Create pod

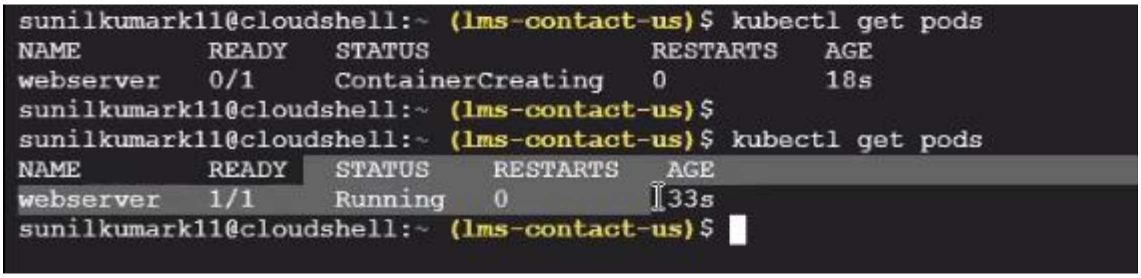
kubectl run --image tomcat webserver

pod will conatin a container so it needs an image then install tomcat

webserver= name of pod



```
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl run --image tomcat webserver
pod/webserver created
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get pods
NAME        READY   STATUS              RESTARTS   AGE
webserver   0/1     ContainerCreating   0          18s
sunilkumark11@cloudshell:~ (lms-contact-us)$
```



```
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get pods
NAME        READY   STATUS              RESTARTS   AGE
webserver   0/1     ContainerCreating   0          18s
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
webserver   1/1     Running   0          33s
sunilkumark11@cloudshell:~ (lms-contact-us)$
```

+++

We have connected to the cluster not connected to single node

Node inbuilt installed docker in k8s

Kuberntes uses various types of objects.

1 Pod: This is a layer of abstraction on top of a container. This is the smallest object that kubernetes can work on. In the pod, we have the container. kubectl commands will work on the pod and pod communicates there instructions to the container.

2. Service Object: This is used for port mapping and network load balancing.

3. NameSpace: This is used for creating partitions in the cluster. Pods running in a namespace cannot communicate with other pods running in other namespace.

4. Secrets: This is used for passing encrypted data to the pods.

5. ReplicaSet / Replication Controller: This is used for managing multiple replicas of a pod to perform activities like load balancing and auto scaling.

6. Deployment: This is used for performing all activities that a ReplicaSet can do. It can also handle rolling updates.

Create Cluster.

Open cloud shell terminal.

Command to create a pod

```
kubectl run --image tomcat webserver
```

(Webserver is pod name)

To see list of pods

----

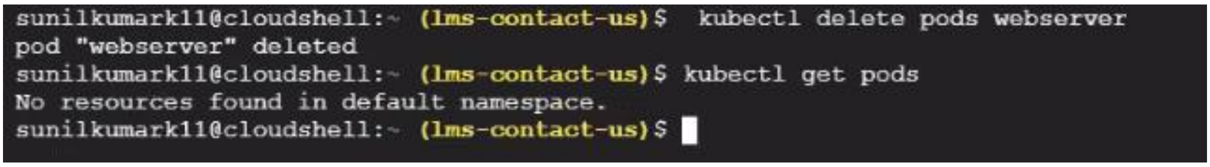
kubectl get pods

If we do not specify replicas, it creates only one replica.

To delete the pod

----

kubectl delete pods webserver

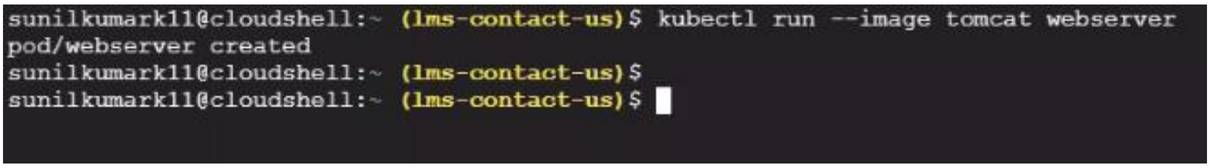


```
sunilkumark11@cloudshell:~ (lms-contact-us)$  kubectl delete pods webserver
pod "webserver" deleted
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get pods
No resources found in default namespace.
sunilkumark11@cloudshell:~ (lms-contact-us)$
```

Lets create pod again

----

kubectl run --image tomcat webserver



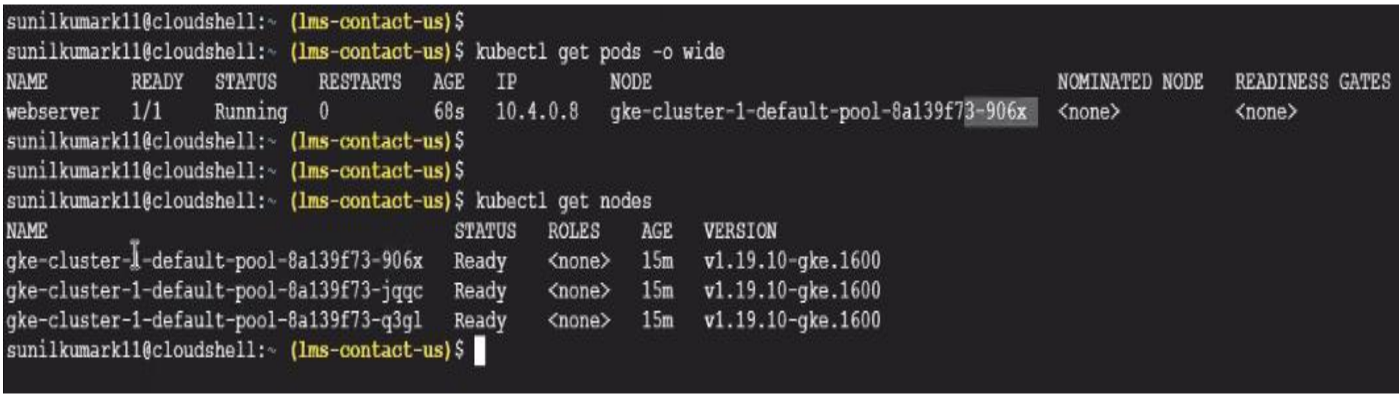
```
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl run --image tomcat webserver
pod/webserver created
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$
```

Pod gets created, inside pod container gets created.

To know on which node, this pod is running

kubectl get pods -o wide

(o - stands for output)



```
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP         NODE                                       NOMINATED NODE   READINESS GATES
webserver   1/1     Running   0          68s   10.4.0.8   gke-cluster-1-default-pool-8a139f73-906x   <none>           <none>
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$ kubectl get nodes
NAME                                       STATUS   ROLES    AGE   VERSION
gke-cluster-1-default-pool-8a139f73-906x   Ready    <none>   15m   v1.19.10-gke.1600
gke-cluster-1-default-pool-8a139f73-jqqc   Ready    <none>   15m   v1.19.10-gke.1600
gke-cluster-1-default-pool-8a139f73-q3gl   Ready    <none>   15m   v1.19.10-gke.1600
sunilkumark11@cloudshell:~ (lms-contact-us)$
```

Ending with-906x so it is running on node 1 as it is ending with same 906x

We cant control on which node the pod cab be run, its automatic selecting

---

Yml files

But, Kubernetes performs container orchestration by using definition files. Definition files are yml files

Definition file, will have 4 top level elements

1. apiVersion:

2. kind:

3. metadata:

4. spec:

apiVersion:

Depending on kubernetes object we want to create, there is corresponding code library we want to use.

apiVersion referes to code library

Kind	apiVersion
Pod	v1
Replication COntroller	v1
Service	v1
NameSpace	v1
Secrets	v1
RepliaSet	apps/v1
Deployment	apps/v1

kind:----------

Refers to kubernetes object which we want to create.

Ex: Pod, Replicaset, service etc

metadata:-----------

Additional information about the kubernets object

like name, labels etc

spec:------

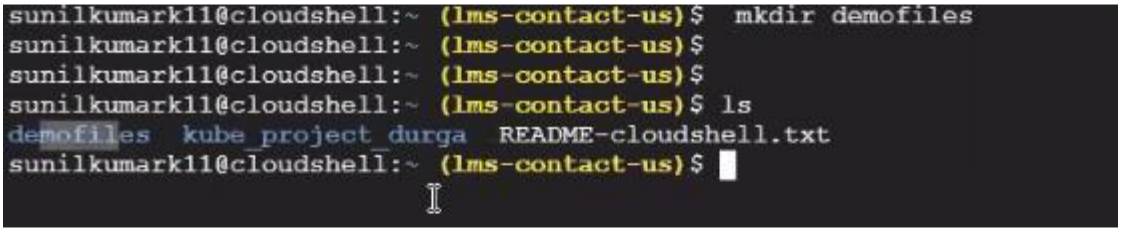
Contains docker container related information like image name, environment variables, port mapping etc.

+++

Connect to cluster by using cloud shell.

$ mkdir demofiles

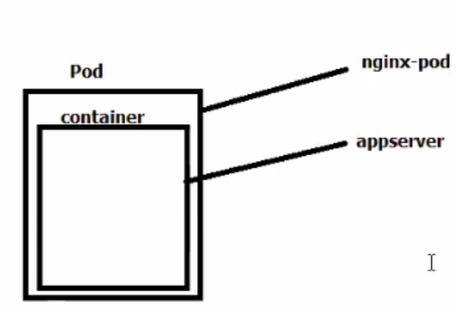
$ cd demofiles



```
sunilkumark11@cloudshell:~ (lms-contact-us)$  mkdir demofiles
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$
sunilkumark11@cloudshell:~ (lms-contact-us)$ ls
demofiles  kube_project_durga  README-cloudshell.txt
sunilkumark11@cloudshell:~ (lms-contact-us)$ 
```

Ex1: Create a pod definition file to start nginx in a pod (nginx container)

Name the pod as nginx-pod, name the container as appserver.



cat > pod-definition1.yml

vim pod-definition1.yml

---

apiVersion: v1

kind: Pod

metadata:-----------------additonal info so below providing name, label=author

name: nginx-pod

labels:

author: sunil------------------------any user defined value like can write client name,project

type: reverse-proxy-------------------

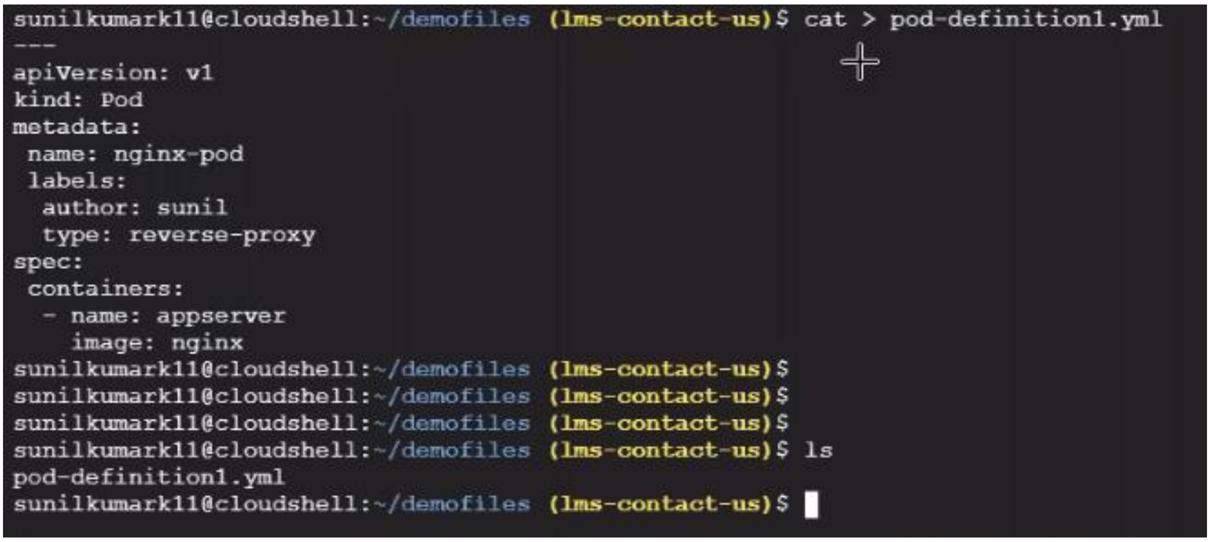
spec:---

containers:

- name: appserver

image: nginx

:wq



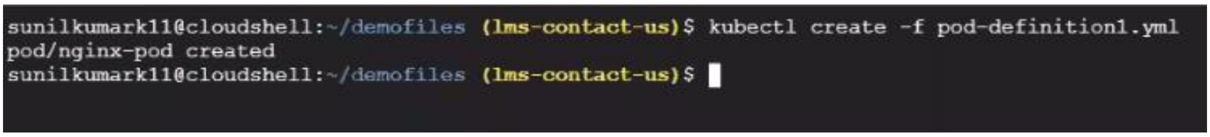
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ cat > pod-definition1.yml
---
apiVersion: v1
kind: Pod
metadata:
 name: nginx-pod
 labels:
  author: sunil
  type: reverse-proxy
spec:
 containers:
  - name: appserver
    image: nginx
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ ls
pod-definition1.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Ctril+d after copy yaml file

Command to run the definition file

----

kubectl create -f pod-definition1.yml



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition1.yml
pod/nginx-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Pod is created.

To get the list of pods

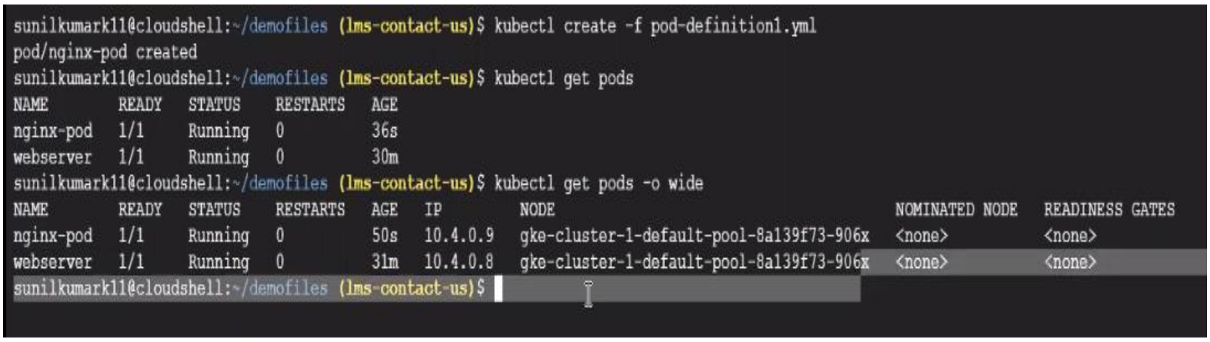
----

kubectl get pods

To get the list of pods along with IP address and which node the pod is running

----

kubectl get pods -o wide



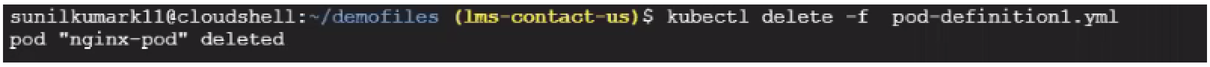
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition1.yml
pod/nginx-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          36s
webserver   1/1     Running   0          30m
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP         NODE                                      NOMINATED NODE   READINESS GATES
nginx-pod   1/1     Running   0          50s   10.4.0.9   gke-cluster-1-default-pool-8a139f73-906x   <none>           <none>
webserver   1/1     Running   0          31m   10.4.0.8   gke-cluster-1-default-pool-8a139f73-906x   <none>           <none>
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Both pods running on same node, we cant control to run on which node.

To delete the pod created from the above file

----

kubectl delete -f pod-definition1.yml



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl delete -f  pod-definition1.yml
pod "nginx-pod" deleted
```

+++++++++++++++++++++++++++++++++++

+++++++++++++++++++++++++++++++++++

ReplicationController:

This is an high level object used for handling multiple replicas of a specific pod.

Here we can perform load balancing and scaling.

ReplicationController uses keys like replicas, template" etc in the "spec" section.

In template section we can give metadata related to the pod and also use another spec section where we can give containers information.

Ex: Create a replication controller for creating 3 replicas of httpd

3 repicas means 3 pods

```
cat > replication-controller.yml
```

```
vim  replication-controller.yml
```

```
---
apiVersion: v1
kind: ReplicationController----------------not creating the pod here because replication controller
metadata:-------------------------meta data for ReplicationController
 name: httpd-rc-------------------name of replication controller
 labels:
  author: sunil--------------------any user defined info
spec:
 replicas: 3---------------3 pods ,provided in under spec section
 template:
  metadata:-----------------------additional info to template
   name: httpd-pod-------------------------template name of pod (pod name begins with httpd-pod followed by some alpha numeric characteristics)
   labels:-------------------to the pods
    author: sunil
  spec:-------------------spec for pods
```

containers:---------------------pod contains containers

- name: myhttpd

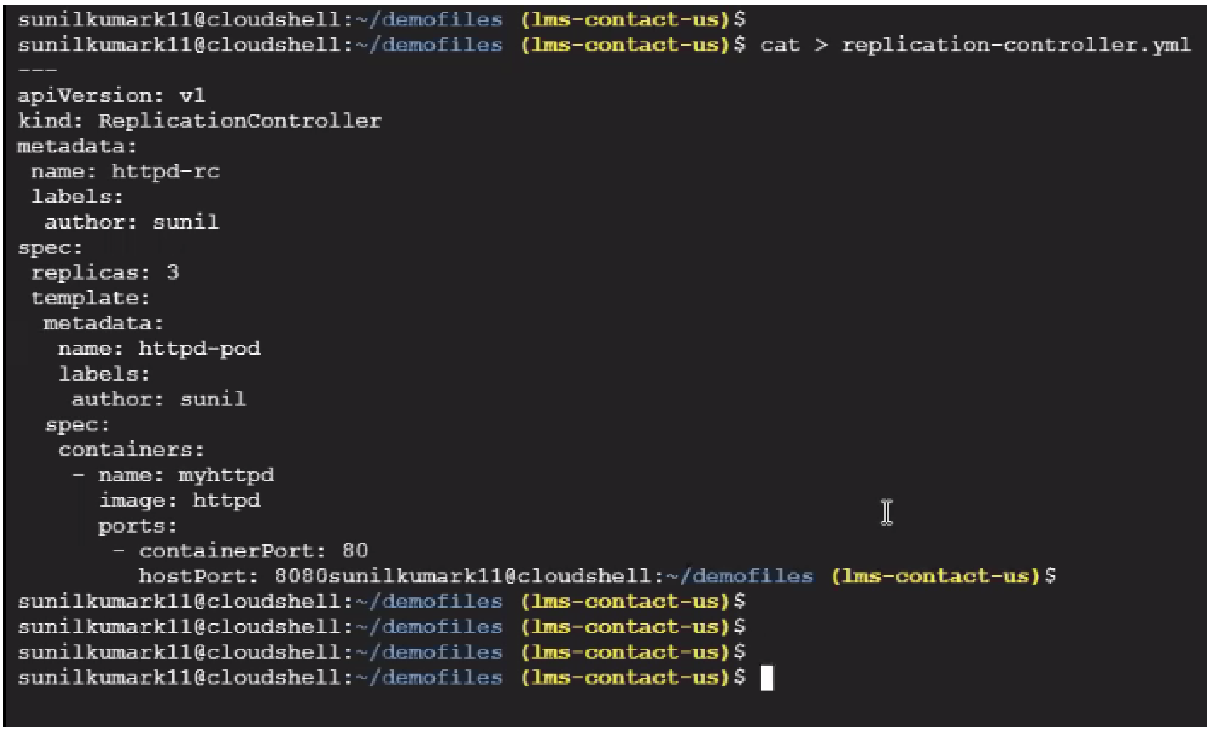
image: httpd

ports:

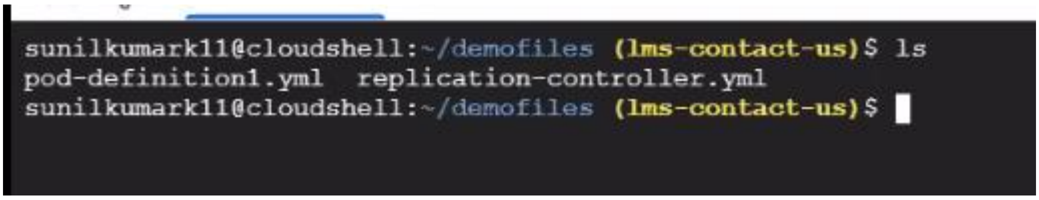
- containerPort: 80

hostPort: 8080

:wq

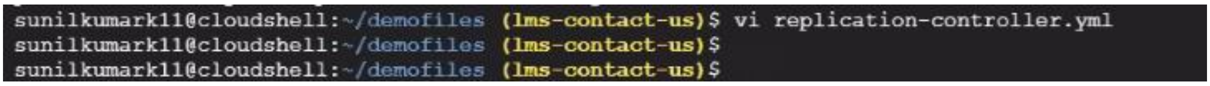


```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ cat > replication-controller.yml
---
apiVersion: v1
kind: ReplicationController
metadata:
 name: httpd-rc
 labels:
  author: sunil
spec:
 replicas: 3
 template:
  metadata:
   name: httpd-pod
   labels:
    author: sunil
  spec:
   containers:
    - name: myhttpd
      image: httpd
      ports:
       - containerPort: 80
         hostPort: 8080sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ ls
pod-definition1.yml  replication-controller.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

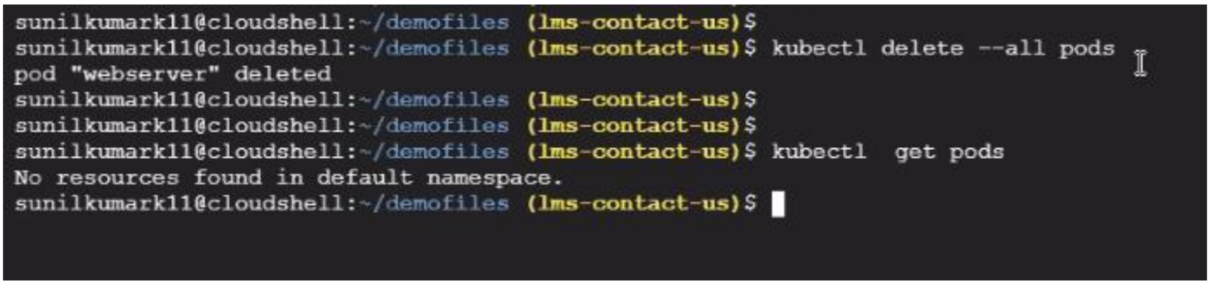
verify



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ vi replication-controller.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

kubectl delete --all pods   (To delete all the existing pods)

kubectl get pods (No pods available)



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl delete --all pods
pod "webserver" deleted
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
No resources found in default namespace.
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Open the port

In aws we open security group where as in gcp open port like below

Before creating the pods we will open ports then create pods.

Kubernetes ready to create ports but gcp evel restricted so we create the port by giving commands .in aws also same .Even if uyou mention port info in yaml file

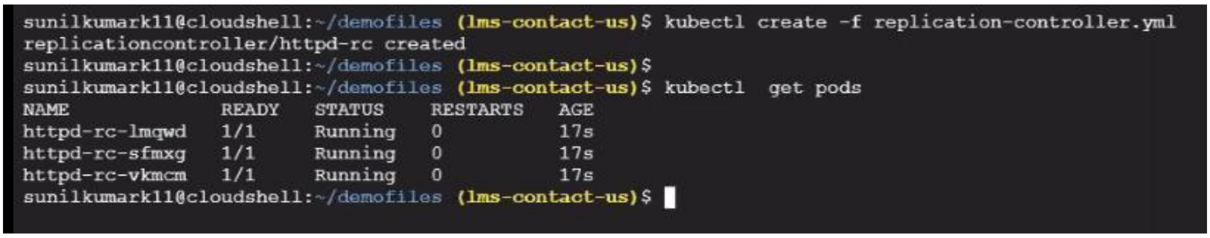
containers:----------------------pod contains containers

```
    - name: myhttpd
      image: httpd
      ports:
       - containerPort: 80
         hostPort: 8080
```

gcloud compute firewall-rules create rule21 --allow tcp:8080

kubectl create -f replication-controller.yml

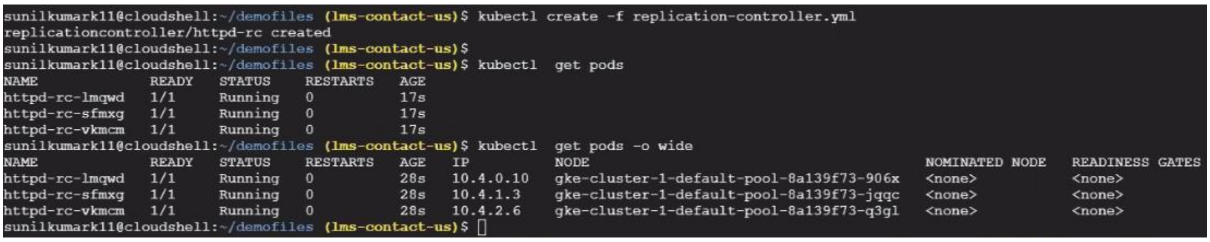
kubectl get pods (We should get 3 pods)



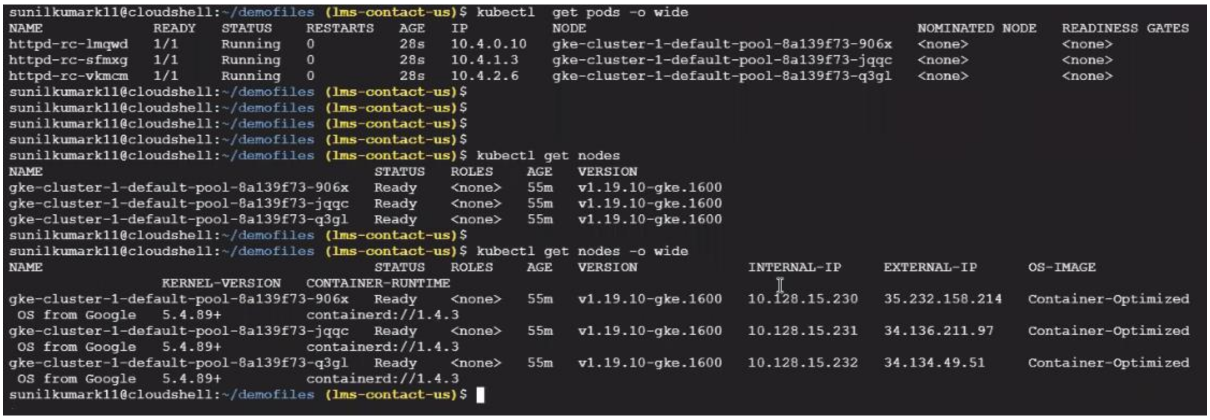
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f replication-controller.yml
replicationcontroller/httpd-rc created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME             READY   STATUS    RESTARTS   AGE
httpd-rc-lmqwd   1/1     Running   0          17s
httpd-rc-sfmxg   1/1     Running   0          17s
httpd-rc-vkmcm   1/1     Running   0          17s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

kubectl get pods -o wide (Observation , 3 pods are distributed in 3 nodes)

kubectl get nodes -o wide



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f replication-controller.yml
replicationcontroller/httpd-rc created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME             READY   STATUS    RESTARTS   AGE
httpd-rc-lmqwd   1/1     Running   0          17s
httpd-rc-sfmxg   1/1     Running   0          17s
httpd-rc-vkmcm   1/1     Running   0          17s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP          NODE                                       NOMINATED NODE   READINESS GATES
httpd-rc-lmqwd   1/1     Running   0          28s   10.4.0.10   gke-cluster-1-default-pool-8a139f73-906x   <none>           <none>
httpd-rc-sfmxg   1/1     Running   0          28s   10.4.1.3    gke-cluster-1-default-pool-8a139f73-jqqc   <none>           <none>
httpd-rc-vkmcm   1/1     Running   0          28s   10.4.2.6    gke-cluster-1-default-pool-8a139f73-q3gl   <none>           <none>
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ 
```

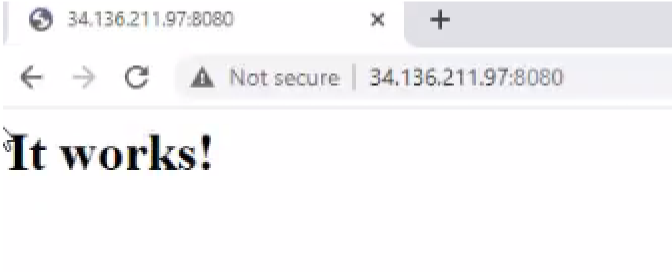


```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP          NODE                                       NOMINATED NODE   READINESS GATES
httpd-rc-lmqwd   1/1     Running   0          28s   10.4.0.10   gke-cluster-1-default-pool-8a139f73-906x   <none>           <none>
httpd-rc-sfmxg   1/1     Running   0          28s   10.4.1.3    gke-cluster-1-default-pool-8a139f73-jqqc   <none>           <none>
httpd-rc-vkmcm   1/1     Running   0          28s   10.4.2.6    gke-cluster-1-default-pool-8a139f73-q3gl   <none>           <none>
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get nodes
NAME                                       STATUS   ROLES    AGE   VERSION
gke-cluster-1-default-pool-8a139f73-906x   Ready    <none>   55m   v1.19.10-gke.1600
gke-cluster-1-default-pool-8a139f73-jqqc   Ready    <none>   55m   v1.19.10-gke.1600
gke-cluster-1-default-pool-8a139f73-q3gl   Ready    <none>   55m   v1.19.10-gke.1600
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get nodes -o wide
NAME                                       STATUS   ROLES    AGE   VERSION             INTERNAL-IP     EXTERNAL-IP      OS-IMAGE
                 KERNEL-VERSION   CONTAINER-RUNTIME
gke-cluster-1-default-pool-8a139f73-906x   Ready    <none>   55m   v1.19.10-gke.1600   10.128.15.230   35.232.158.214   Container-Optimized
 OS from Google   5.4.89+          containerd://1.4.3
gke-cluster-1-default-pool-8a139f73-jqqc   Ready    <none>   55m   v1.19.10-gke.1600   10.128.15.231   34.136.211.97    Container-Optimized
 OS from Google   5.4.89+          containerd://1.4.3
gke-cluster-1-default-pool-8a139f73-q3gl   Ready    <none>   55m   v1.19.10-gke.1600   10.128.15.232   34.134.49.51     Container-Optimized
 OS from Google   5.4.89+          containerd://1.4.3
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ 
```

Take external IP (Public IP) of any node

35.239.250.215:8080

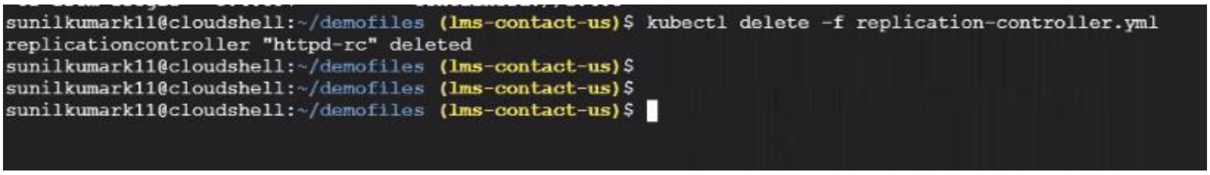
34.136.211.97:8080



It works!

To delete the replicas then delete the rplication controller then pods will be deleted

kubectl delete -f replication-controller.yml



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl delete -f replication-controller.yml
replicationcontroller "httpd-rc" deleted
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ 
```

++++++++++++++++++++++++++

Replica Set : sameas replication controller with added advantage

----

Pod is the smallest kubernetes object, which we worked on.

Next Level is replication controller.

ReplicaSet is similar to replication controller.

In replicatSet, we have an additional field in spec section called as "selector" field.

This selector uses a child element called "matchLabels" , where it will search for pods based on specific label

specific label name, and adds them to the cluster.

Ex: Create a replicaset file to start 4 tomcat replicas and then perform scaling

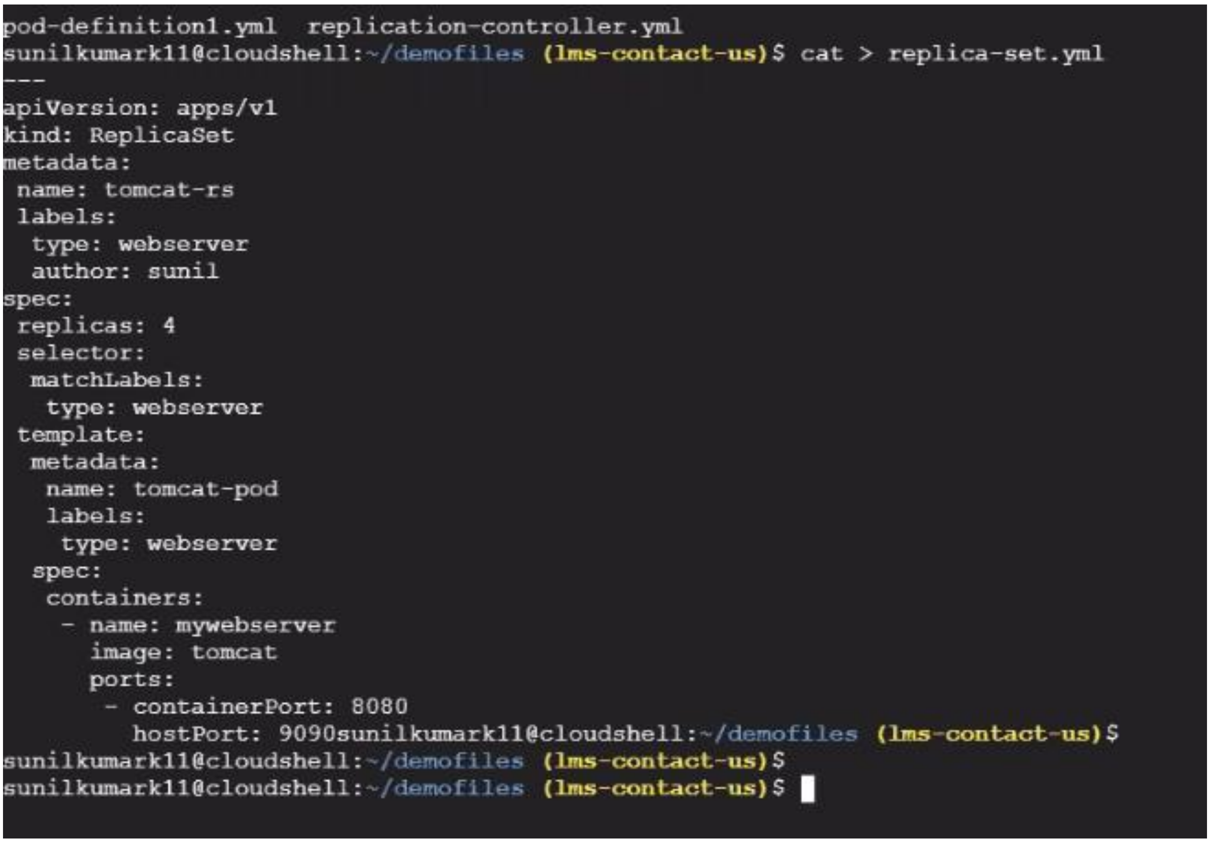
```
cat > replica-set.yml
```

```
vim replica-set.yml
```

```
---
apiVersion: apps/v1
kind: ReplicaSet--------------------------
metadata:
 name: tomcat-rs--------------name of ReplicaSet
 labels:
  type: webserver
  author: sunil
```

```
spec:
 replicas: 4
 selector:
  matchLabels:
   type: webserver
 template:
  metadata:
   name: tomcat-pod
   labels:
    type: webserver
  spec:
   containers:
    - name: mywebserver
      image: tomcat
      ports:
       - containerPort: 8080
         hostPort: 9090
```

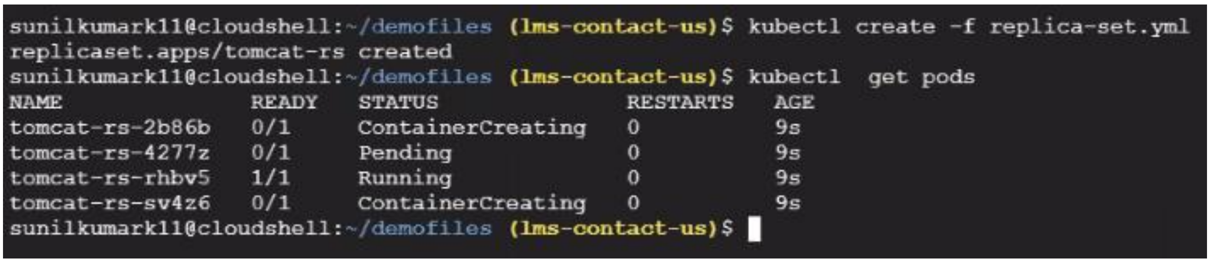
```
:wq
```



```
pod-definition1.yml  replication-controller.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ cat > replica-set.yml
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: tomcat-rs
 labels:
  type: webserver
  author: sunil
spec:
 replicas: 4
 selector:
  matchLabels:
   type: webserver
 template:
  metadata:
   name: tomcat-pod
   labels:
    type: webserver
  spec:
   containers:
    - name: mywebserver
      image: tomcat
      ports:
       - containerPort: 8080
         hostPort: 9090sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

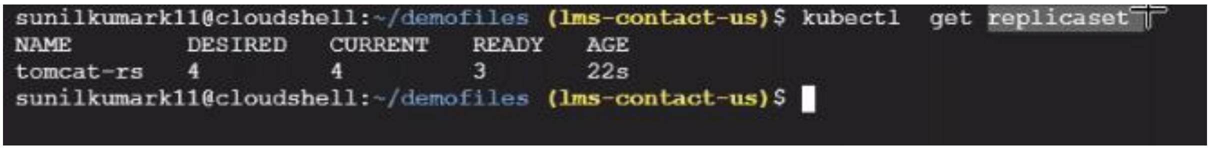
kubectl create -f replica-set.yml

kubectl get pods (We should get 4 pods)

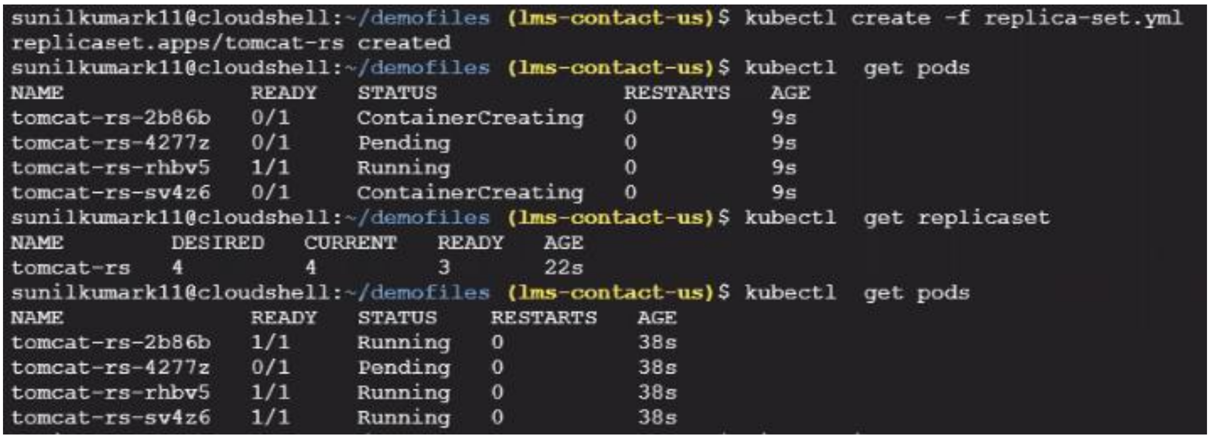


```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f replica-set.yml
replicaset.apps/tomcat-rs created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME              READY   STATUS              RESTARTS   AGE
tomcat-rs-2b86b   0/1     ContainerCreating   0          9s
tomcat-rs-4277z   0/1     Pending             0          9s
tomcat-rs-rhbv5   1/1     Running             0          9s
tomcat-rs-sv4z6   0/1     ContainerCreating   0          9s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

kubectl get replicaset



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get replicaset
NAME        DESIRED   CURRENT   READY   AGE
tomcat-rs   4         4         3       22s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f replica-set.yml
replicaset.apps/tomcat-rs created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME              READY   STATUS              RESTARTS   AGE
tomcat-rs-2b86b   0/1     ContainerCreating   0          9s
tomcat-rs-4277z   0/1     Pending             0          9s
tomcat-rs-rhbv5   1/1     Running             0          9s
tomcat-rs-sv4z6   0/1     ContainerCreating   0          9s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get replicaset
NAME        DESIRED   CURRENT   READY   AGE
tomcat-rs   4         4         3       22s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME              READY   STATUS    RESTARTS   AGE
tomcat-rs-2b86b   1/1     Running   0          38s
tomcat-rs-4277z   0/1     Pending   0          38s
tomcat-rs-rhbv5   1/1     Running   0          38s
tomcat-rs-sv4z6   1/1     Running   0          38s
```

Lets perform scaling from 4 pods to 6 pods

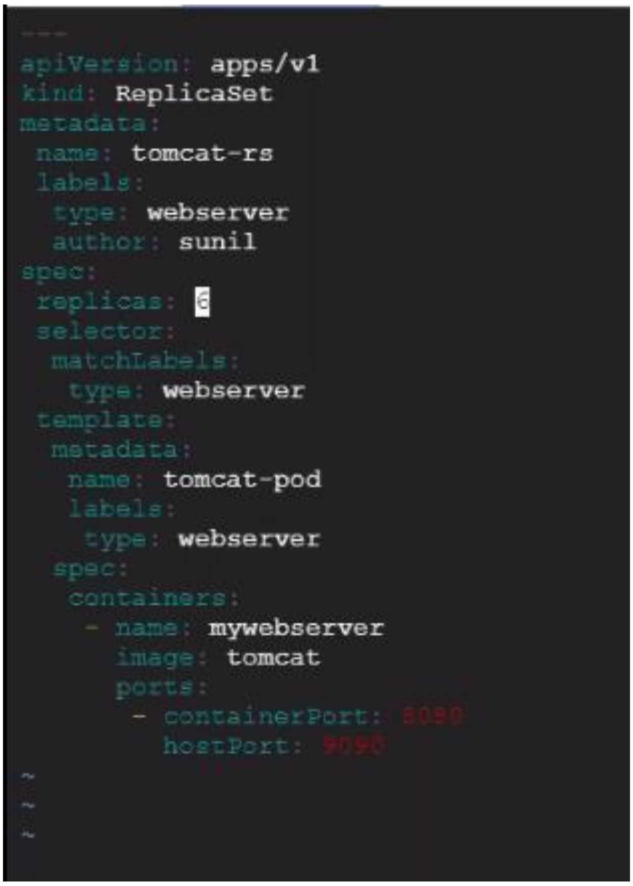
Option 1: We can open the definition file and make changes in the code from 4 to 6 in replicas field.

vim replica-set.yml

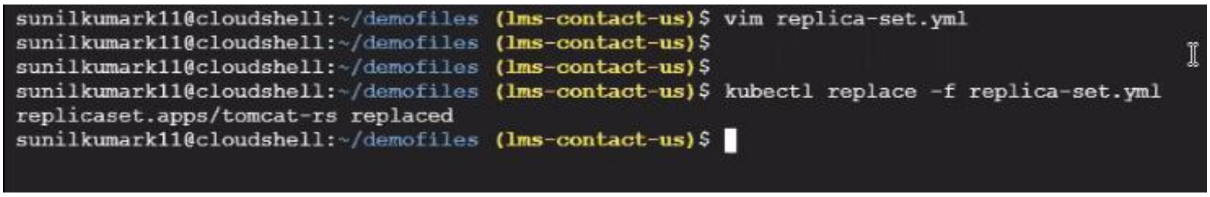
Now, we should not use create commands, we should use replace command.

kubectl replace -f replica-set.yml

kubectl get pods (We should get 6 pods)

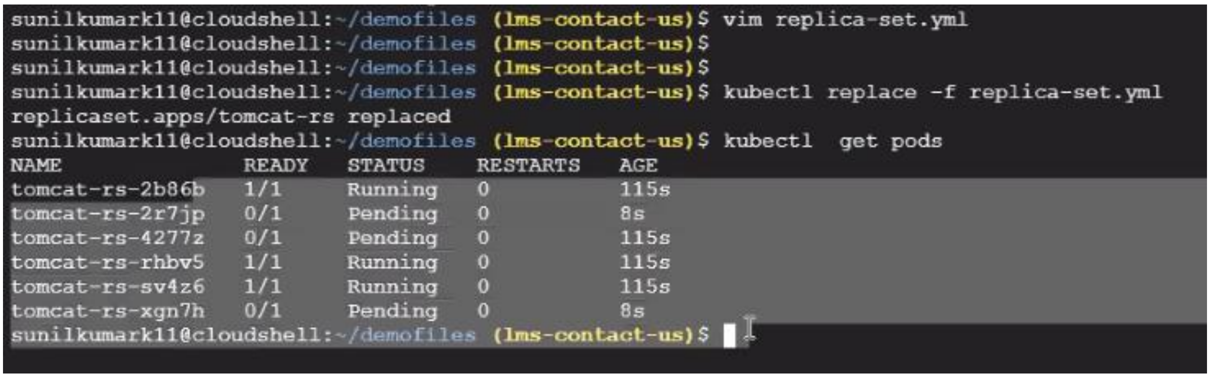


```
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: tomcat-rs
 labels:
  type: webserver
  author: sunil
spec:
 replicas: 6
 selector:
  matchLabels:
   type: webserver
 template:
  metadata:
   name: tomcat-pod
   labels:
    type: webserver
  spec:
   containers:
    - name: mywebserver
      image: tomcat
      ports:
       - containerPort: 8080
         hostPort: 9090
~
~
~
```



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ vim replica-set.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl replace -f replica-set.yml
replicaset.apps/tomcat-rs replaced
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Observe replication set replaced so we see 6 pods



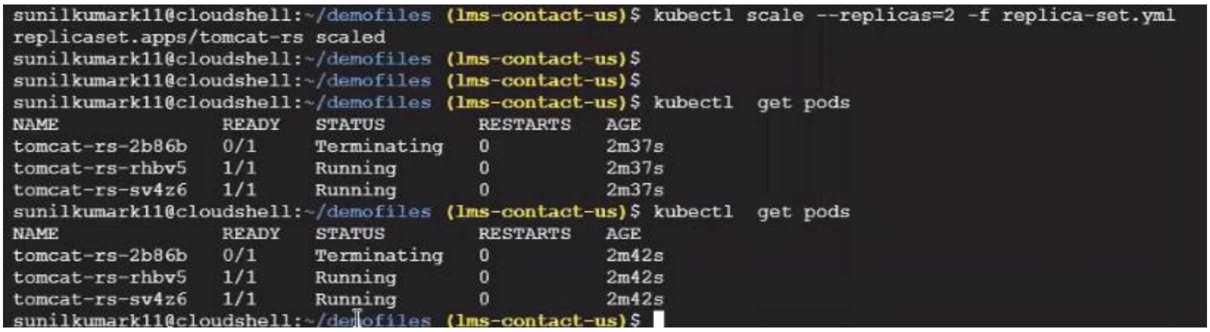
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ vim replica-set.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl replace -f replica-set.yml
replicaset.apps/tomcat-rs replaced
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME              READY   STATUS    RESTARTS   AGE
tomcat-rs-2b86b   1/1     Running   0          115s
tomcat-rs-2r7jp   0/1     Pending   0          8s
tomcat-rs-4277z   0/1     Pending   0          115s
tomcat-rs-rhbv5   1/1     Running   0          115s
tomcat-rs-sv4z6   1/1     Running   0          115s
tomcat-rs-xgn7h   0/1     Pending   0          8s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Option 2:

Now 6 replicas scale down to two by using commands

kubectl scale --replicas=2 -f replica-set.yml (for scale up and scale down command is same)

kubectl get pods (We should get 2 pods)



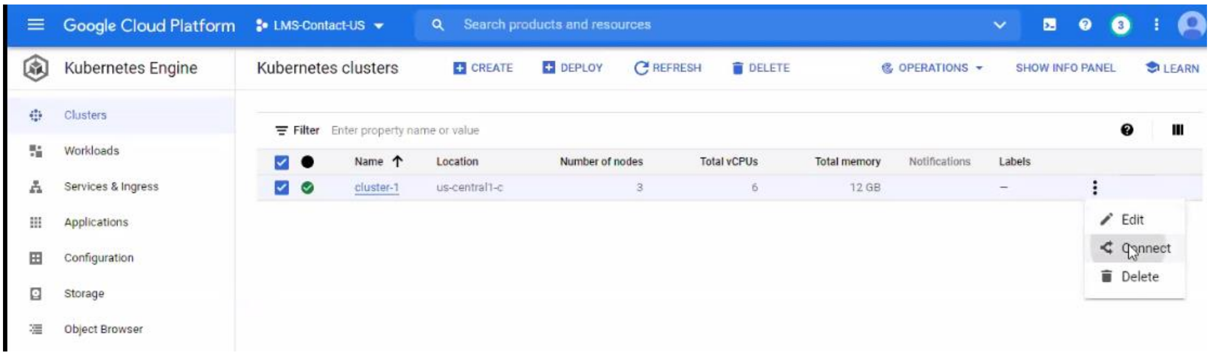
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl scale --replicas=2 -f replica-set.yml
replicaset.apps/tomcat-rs scaled
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME              READY   STATUS        RESTARTS   AGE
tomcat-rs-2b86b   0/1     Terminating   0          2m37s
tomcat-rs-rhbv5   1/1     Running       0          2m37s
tomcat-rs-sv4z6   1/1     Running       0          2m37s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl  get pods
NAME              READY   STATUS        RESTARTS   AGE
tomcat-rs-2b86b   0/1     Terminating   0          2m42s
tomcat-rs-rhbv5   1/1     Running       0          2m42s
tomcat-rs-sv4z6   1/1     Running       0          2m42s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Get 2 pods, another is terminating

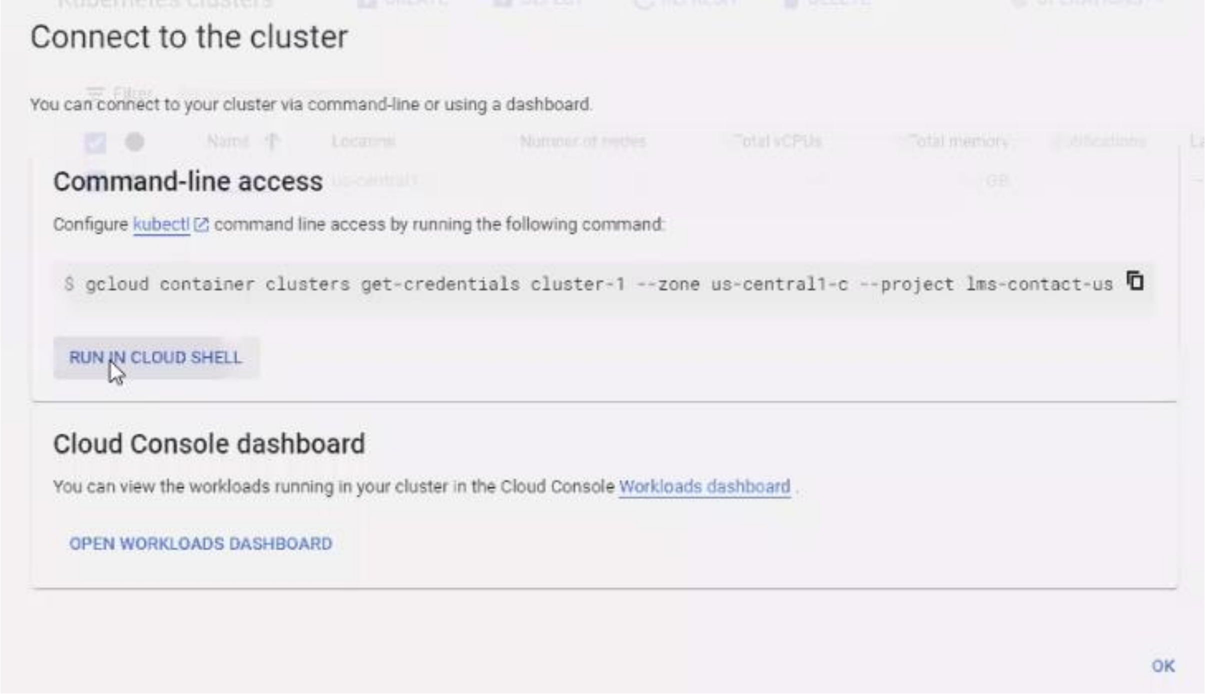
++

1st July

Create cluster and connect , run in cloud shell



Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
cluster-1	us-central1-c	3	6	12 GB		–



Connect to the cluster

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

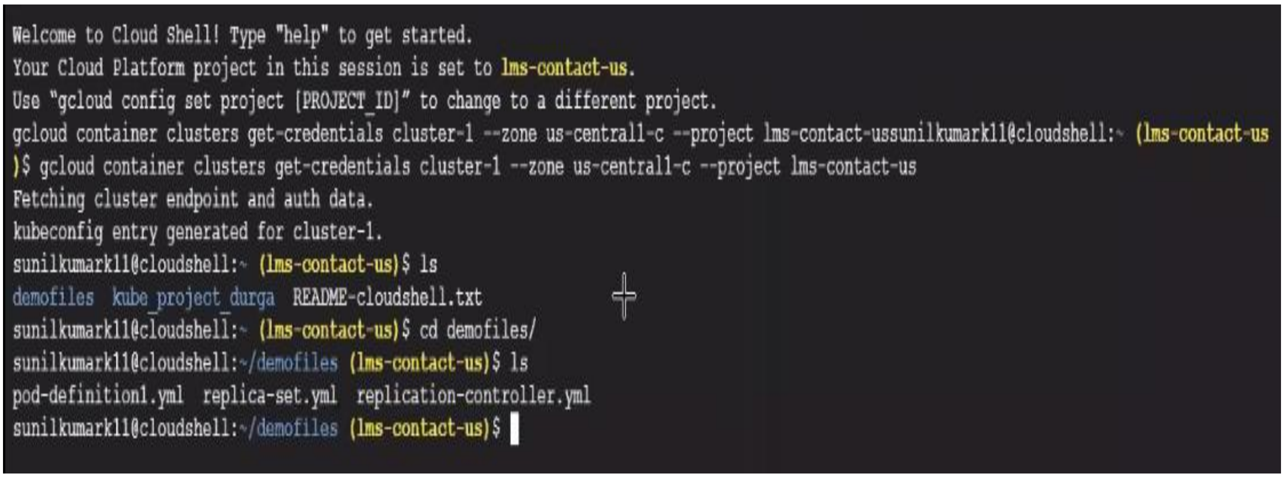
Command-line access

Configure kubectl command line access by running the following command:

```
$ gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project lms-contact-us
```

Cloud Console dashboard

You can view the workloads running in your cluster in the Cloud Console Workloads dashboard .



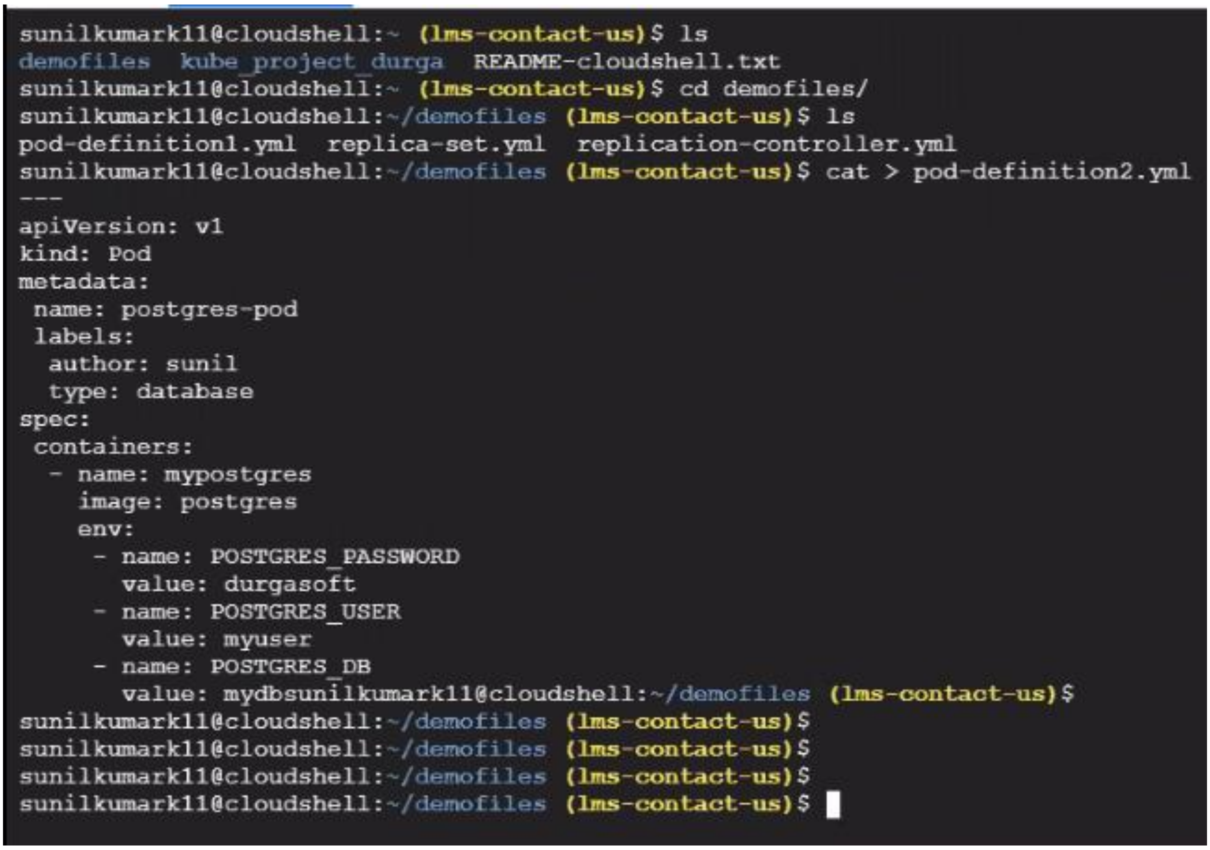
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to lms-contact-us.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project lms-contact-ussunilkumark11@cloudshell:~ (lms-contact-us
)$ gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project lms-contact-us
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster-1.
sunilkumark11@cloudshell:~ (lms-contact-us)$ ls
demofiles  kube_project_durga  README-cloudshell.txt
sunilkumark11@cloudshell:~ (lms-contact-us)$ cd demofiles/
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ ls
pod-definition1.yml  replica-set.yml  replication-controller.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Ex 2:

```
Cat > pod-definition2.yml
vim pod-definition2.yml
```

```
---
apiVersion: v1
kind: Pod
metadata:
 name: postgres-pod-------------------name of pod
 labels:
  author: sunil
  type: database
spec:-------------------------------------pod will have containers so in spec we mention containers.
 containers:
  - name: mypostgres
    image: postgres
    env:-----------------------------------------------------------environment variables
     - name: POSTGRES_PASSWORD
       value: durgasoft
     - name: POSTGRES_USER
       value: myuser
     - name: POSTGRES_DB
       value: mydb
```

```
:wq
```



```
sunilkumark11@cloudshell:~ (lms-contact-us)$ ls
demofiles  kube_project_durga  README-cloudshell.txt
sunilkumark11@cloudshell:~ (lms-contact-us)$ cd demofiles/
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ ls
pod-definition1.yml  replica-set.yml  replication-controller.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ cat > pod-definition2.yml
---
apiVersion: v1
kind: Pod
metadata:
 name: postgres-pod
 labels:
  author: sunil
  type: database
spec:
 containers:
  - name: mypostgres
    image: postgres
    env:
     - name: POSTGRES_PASSWORD
       value: durgasoft
     - name: POSTGRES_USER
       value: myuser
     - name: POSTGRES_DB
       value: mydbsunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ █
```

Command to run the definition file

---

kubectl create -f pod-definition2.yml

To get the list of pods

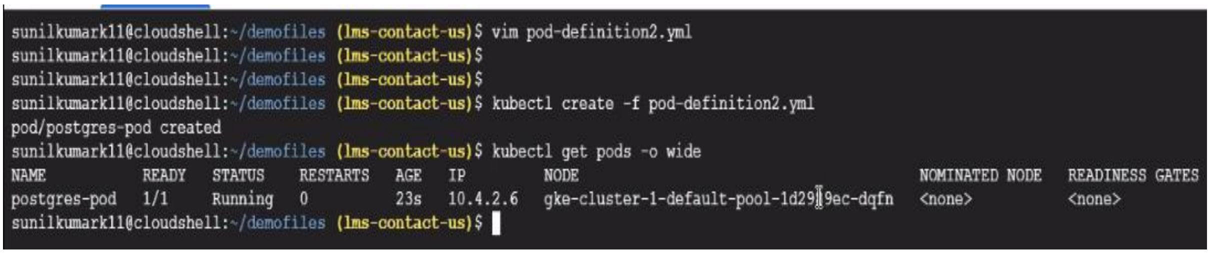
----

kubectl get pods

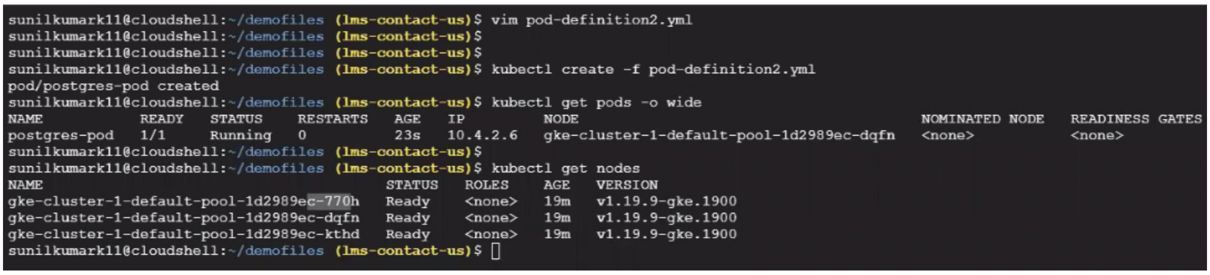
To get the list of pods along with IP address and which node the pod is running

----

kubectl get pods -o wide



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ vim pod-definition2.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition2.yml
pod/postgres-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP          NODE                                       NOMINATED NODE   READINESS GATES
postgres-pod   1/1     Running   0          23s   10.4.2.6    gke-cluster-1-default-pool-1d2989ec-dqfn   <none>           <none>
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

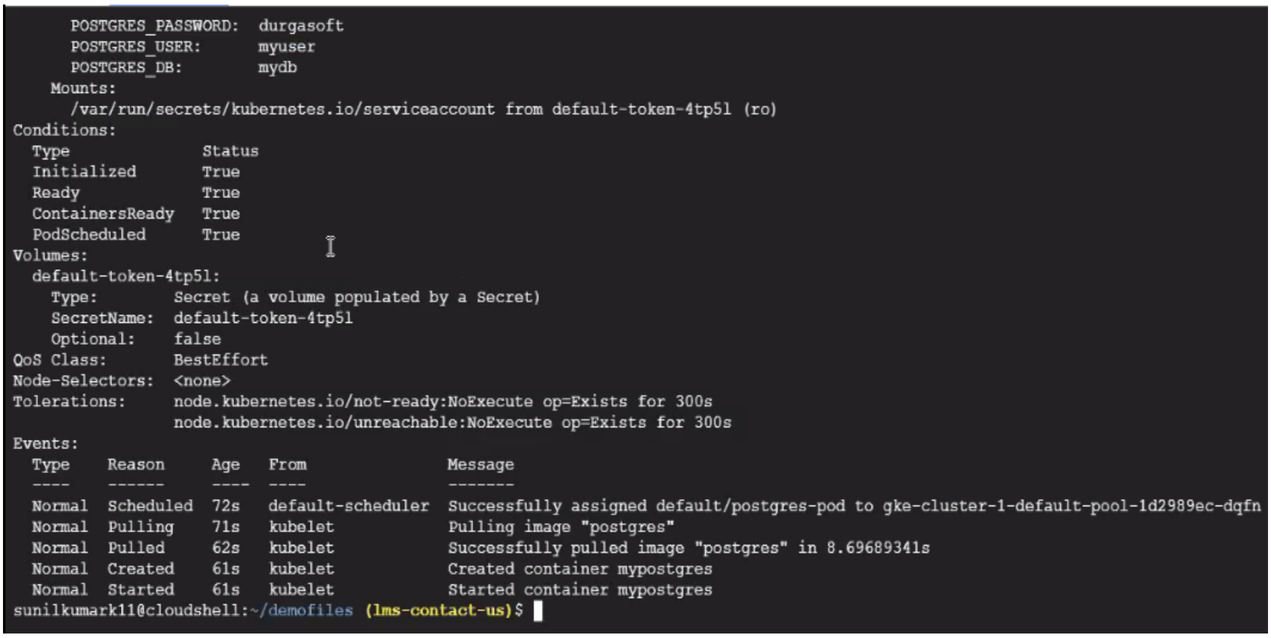


```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ vim pod-definition2.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition2.yml
pod/postgres-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP         NODE                                       NOMINATED NODE   READINESS GATES
postgres-pod   1/1     Running   0          23s   10.4.2.6   gke-cluster-1-default-pool-1d2989ec-dqfn   <none>           <none>
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get nodes
NAME                                       STATUS   ROLES    AGE   VERSION
gke-cluster-1-default-pool-1d2989ec-770h   Ready    <none>   19m   v1.19.9-gke.1900
gke-cluster-1-default-pool-1d2989ec-dqfn   Ready    <none>   19m   v1.19.9-gke.1900
gke-cluster-1-default-pool-1d2989ec-kthd   Ready    <none>   19m   v1.19.9-gke.1900
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

TO get more details about the pod

---

kubectl describe pods postgres-pod

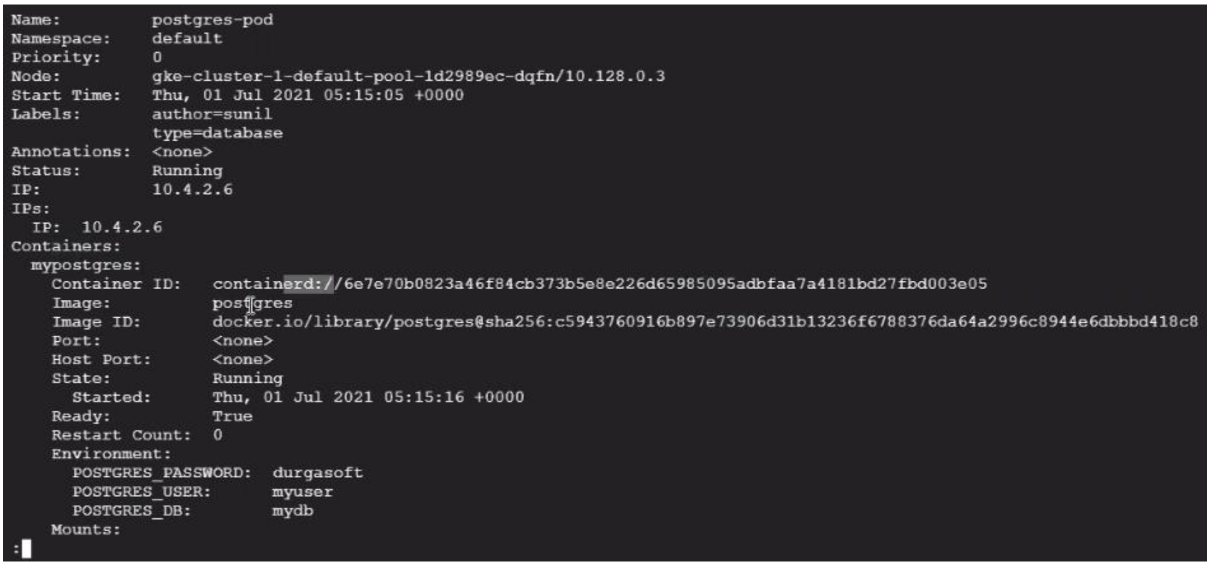


```
      POSTGRES_PASSWORD:  durgasoft
      POSTGRES_USER:      myuser
      POSTGRES_DB:        mydb
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-4tp5l (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-4tp5l:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-4tp5l
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  72s   default-scheduler  Successfully assigned default/postgres-pod to gke-cluster-1-default-pool-1d2989ec-dqfn
  Normal  Pulling    71s   kubelet            Pulling image "postgres"
  Normal  Pulled     62s   kubelet            Successfully pulled image "postgres" in 8.69689341s
  Normal  Created    61s   kubelet            Created container mypostgres
  Normal  Started    61s   kubelet            Started container mypostgres
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

We can't scroll up and see the info so we can give below command to see page wise.

or

kubectl describe pods postgres-pod | less



```
Name:         postgres-pod
Namespace:    default
Priority:     0
Node:         gke-cluster-1-default-pool-1d2989ec-dqfn/10.128.0.3
Start Time:   Thu, 01 Jul 2021 05:15:05 +0000
Labels:       author=sunil
              type=database
Annotations:  <none>
Status:       Running
IP:           10.4.2.6
IPs:
  IP:  10.4.2.6
Containers:
  mypostgres:
    Container ID:   containerd://6e7e70b0823a46f84cb373b5e8e226d65985095adbfaa7a4181bd27fbd003e05
    Image:          postgres
    Image ID:       docker.io/library/postgres@sha256:c5943760916b897e73906d31b13236f6788376da64a2996c8944e6dbbbd418c8
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 01 Jul 2021 05:15:16 +0000
    Ready:          True
    Restart Count:  0
    Environment:
      POSTGRES_PASSWORD:  durgasoft
      POSTGRES_USER:      myuser
      POSTGRES_DB:        mydb
    Mounts:
:
```

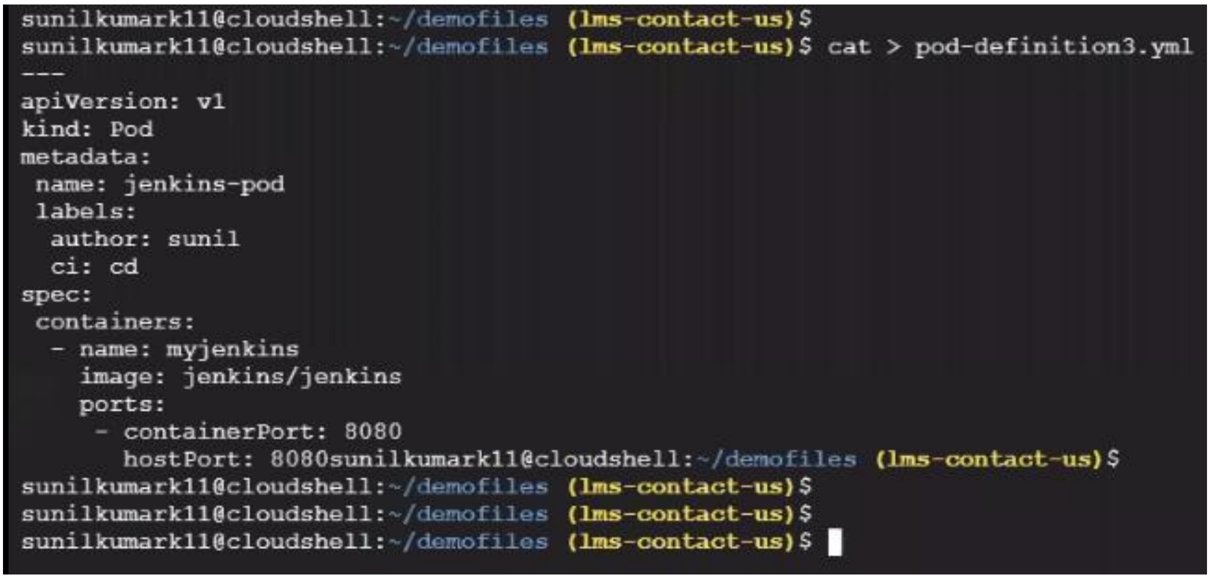
Ex3:

Cat > pod-definition3.yml

vim pod-definition3.yml

```
---
apiVersion: v1
kind: Pod
metadata:
 name: jenkins-pod
 labels:
  author: sunil
  ci: cd
spec:
 containers:
  - name: myjenkins
    image: jenkins/jenkins
    ports:
     - containerPort: 8080
       hostPort: 8080
```

:wq



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ cat > pod-definition3.yml
---
apiVersion: v1
kind: Pod
metadata:
 name: jenkins-pod
 labels:
  author: sunil
  ci: cd
spec:
 containers:
  - name: myjenkins
    image: jenkins/jenkins
    ports:
     - containerPort: 8080
       hostPort: 8080sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

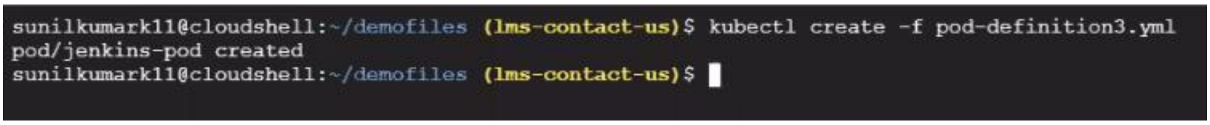
How to open the port? (in this ex , already opened ports so no need to open)

----

gcloud compute firewall-rules create rule35 --allow tcp:8080

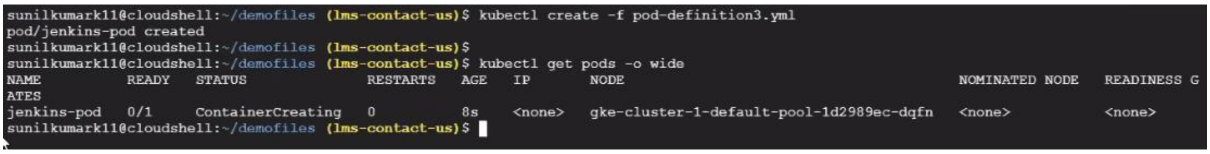
gcloud compute firewall-rules create rule2 --allow tcp:9090

kubectl create -f pod-definition3.yml



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition3.yml
pod/jenkins-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

kubectl get pods -o wide



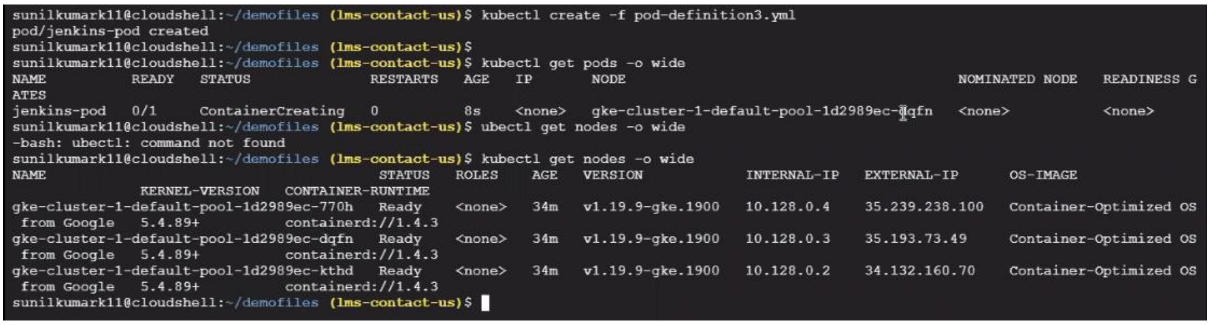
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition3.yml
pod/jenkins-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods -o wide
NAME          READY   STATUS              RESTARTS   AGE   IP       NODE                                       NOMINATED NODE   READINESS G
ATES
jenkins-pod   0/1     ContainerCreating   0          8s    <none>   gke-cluster-1-default-pool-1d2989ec-dqfn   <none>           <none>
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Take a note on the node in which the pod is running.

gke-cluster-1-default-pool-9fb99245-q1nm

TO get the list of nodes

kubectl get nodes -o wide



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition3.yml
pod/jenkins-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods -o wide
NAME          READY   STATUS              RESTARTS   AGE   IP       NODE                                         NOMINATED NODE   READINESS G
ATES
jenkins-pod   0/1     ContainerCreating   0          8s    <none>   gke-cluster-1-default-pool-1d2989ec-dqfn   <none>           <none>
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ ubectl get nodes -o wide
-bash: ubectl: command not found
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get nodes -o wide
NAME                                         STATUS   ROLES    AGE   VERSION             INTERNAL-IP   EXTERNAL-IP      OS-IMAGE
               KERNEL-VERSION   CONTAINER-RUNTIME
gke-cluster-1-default-pool-1d2989ec-770h   Ready    <none>   34m   v1.19.9-gke.1900   10.128.0.4    35.239.238.100   Container-Optimized OS
 from Google   5.4.89+          containerd://1.4.3
gke-cluster-1-default-pool-1d2989ec-dqfn   Ready    <none>   34m   v1.19.9-gke.1900   10.128.0.3    35.193.73.49     Container-Optimized OS
 from Google   5.4.89+          containerd://1.4.3
gke-cluster-1-default-pool-1d2989ec-kthd   Ready    <none>   34m   v1.19.9-gke.1900   10.128.0.2    34.132.160.70    Container-Optimized OS
 from Google   5.4.89+          containerd://1.4.3
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ 
```

Take the external IP of the node

35.223.183.189:8080

34.68.242.87:8080

Open browser (chrome)

35.223.183.189:8080 (we should get the jenkins page)

++

Deployment Object

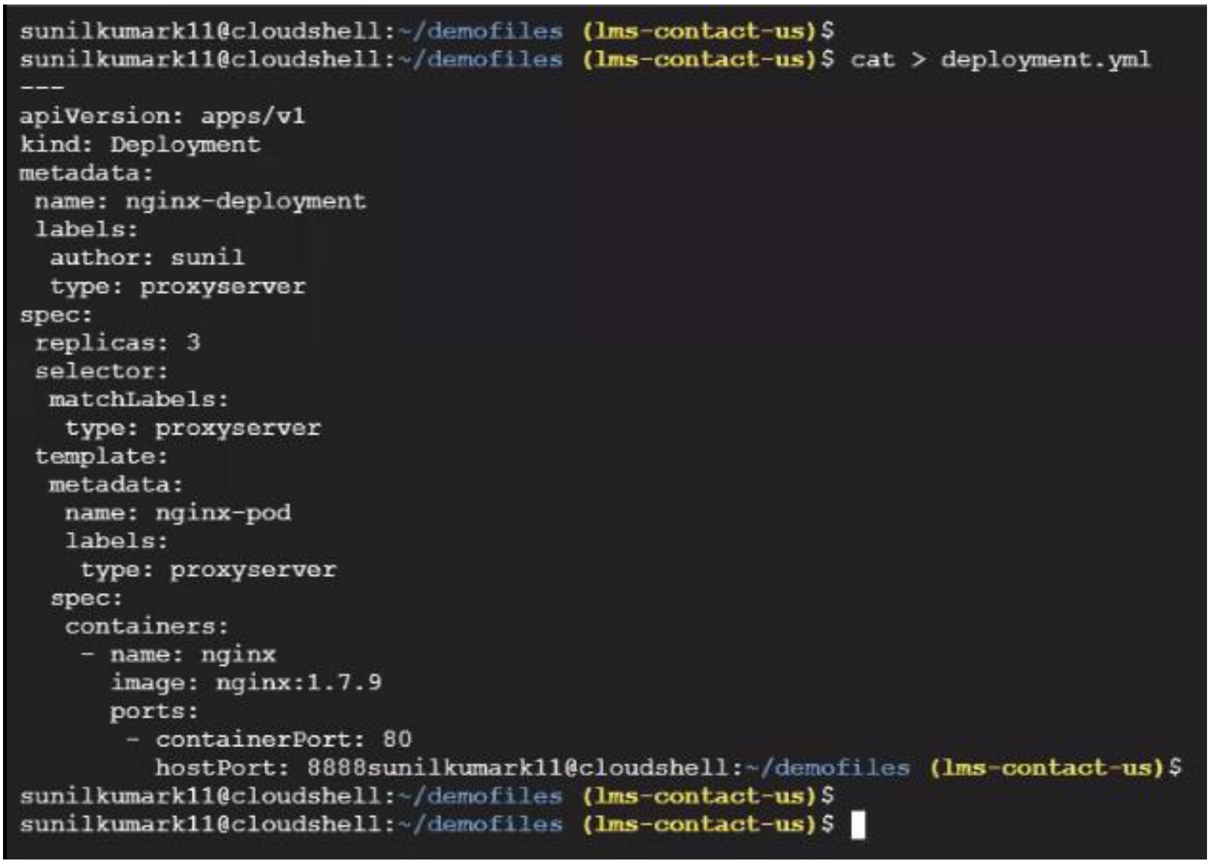
This is also an high level object which can be used for scalling, load balancing and perform rolling updates.

Create a deployment file to run nginx 1.7.9 with 3 replicas.

Later perform a rolling upgrade to nginx 1.9.1

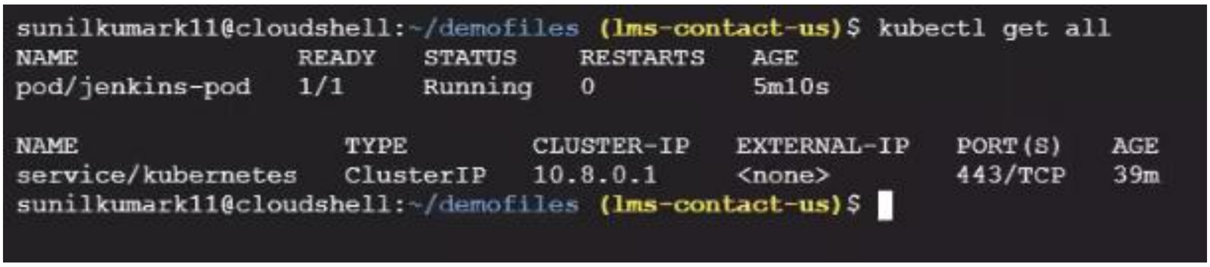
vim deployment.yml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 labels:
  author: sunil
  type: proxyserver
spec:
 replicas: 3
 selector:
  matchLabels:-------------------
  type: proxyserver
 template:
  metadata:
   name: nginx-pod
   labels:
    type: proxyserver
  spec:-----------------------------------technical details abt container
   containers:
    - name: nginx
      image: nginx:1.7.9
      ports:
       - containerPort: 80
         hostPort: 8888
:wq
```



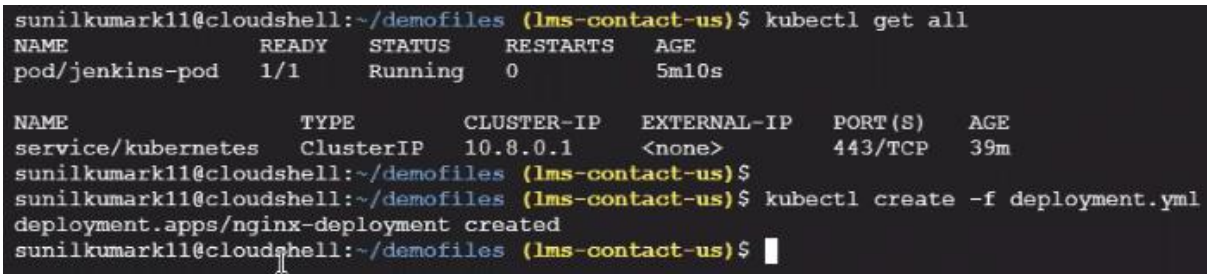
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ cat > deployment.yml
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 labels:
  author: sunil
  type: proxyserver
spec:
 replicas: 3
 selector:
  matchLabels:
   type: proxyserver
 template:
  metadata:
   name: nginx-pod
   labels:
    type: proxyserver
  spec:
   containers:
    - name: nginx
      image: nginx:1.7.9
      ports:
       - containerPort: 80
         hostPort: 8888sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

kubectl get all (we have one default service running)



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get all
NAME               READY   STATUS    RESTARTS   AGE
pod/jenkins-pod    1/1     Running   0          5m10s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.8.0.1     <none>        443/TCP   39m
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

kubectl create -f deployment.yml



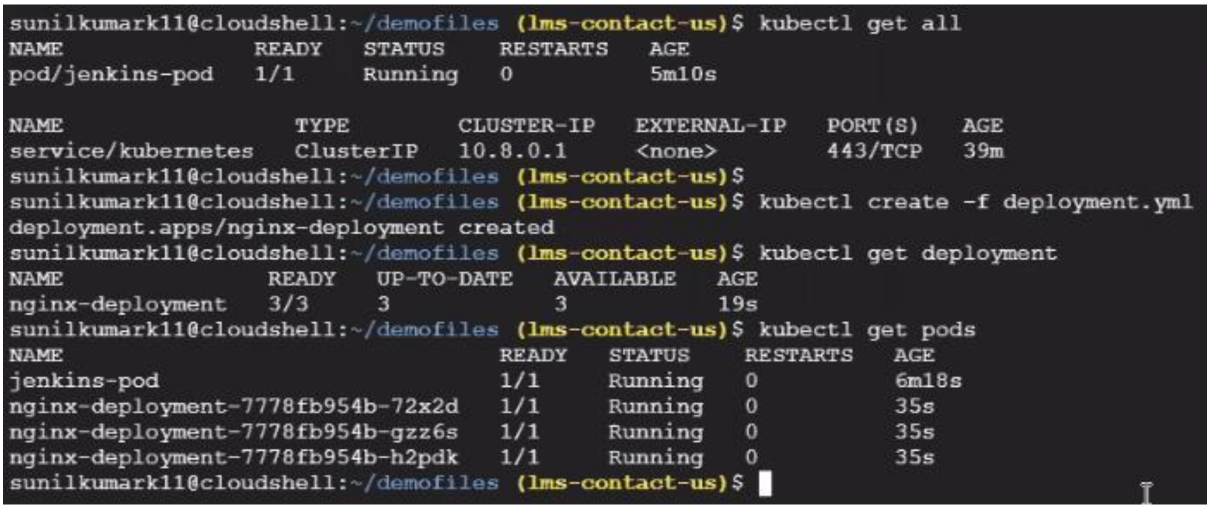
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get all
NAME               READY   STATUS    RESTARTS   AGE
pod/jenkins-pod    1/1     Running   0          5m10s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.8.0.1     <none>        443/TCP   39m
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f deployment.yml
deployment.apps/nginx-deployment created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

TO check, if the deployment is created or not

--

kubectl get deployment (we can see 1 deployment object)

kubectl get pods (we should get 3 pods)

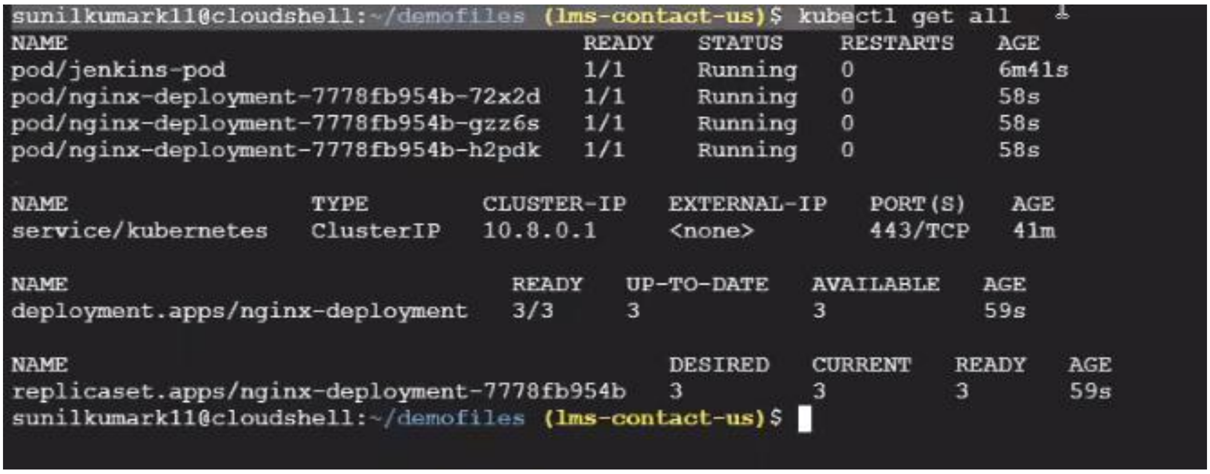


```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get all
NAME               READY   STATUS    RESTARTS   AGE
pod/jenkins-pod    1/1     Running   0          5m10s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.8.0.1     <none>        443/TCP   39m
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f deployment.yml
deployment.apps/nginx-deployment created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           19s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
jenkins-pod                         1/1     Running   0          6m18s
nginx-deployment-7778fb954b-72x2d   1/1     Running   0          35s
nginx-deployment-7778fb954b-gzz6s   1/1     Running   0          35s
nginx-deployment-7778fb954b-h2pdk   1/1     Running   0          35s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

We can anyways perform scaling, apart from that we can perform rolling updates.

kubectl get all (we get all the objects)

Take a note of the full name of the deployment object



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/jenkins-pod                         1/1     Running   0          6m41s
pod/nginx-deployment-7778fb954b-72x2d   1/1     Running   0          58s
pod/nginx-deployment-7778fb954b-gzz6s   1/1     Running   0          58s
pod/nginx-deployment-7778fb954b-h2pdk   1/1     Running   0          58s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.8.0.1     <none>        443/TCP   41m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           59s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-7778fb954b   3         3         3       59s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

When you create a deployment object then the replication set created automatically

Both created same time see age of both is 59 sec

replication set subset of deployment object

we know replication set performs scaling

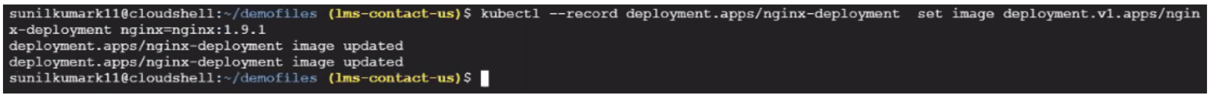
deployment.apps/nginx-deployment—full name of the deployment object

To perform rolling update

----

kubectl --record deployment.apps/nginx-deployment set image deployment.v1.apps/nginx-deployment nginx=nginx:1.9.1

(provide hiherversion upgrade; lower version-degarde in cmd)

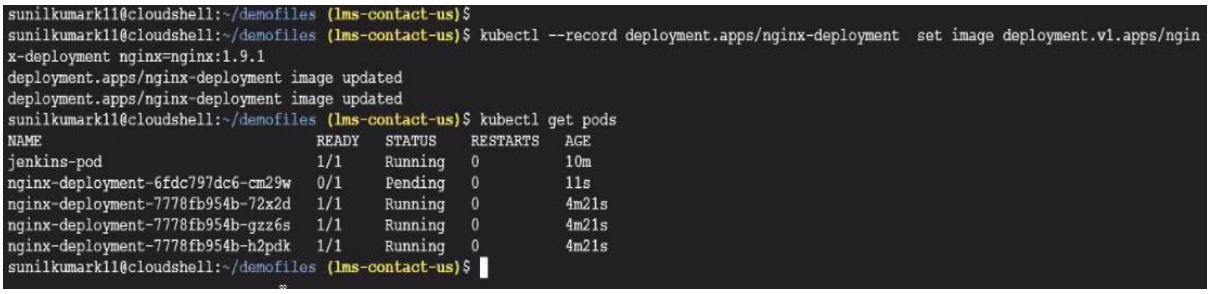


```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl --record deployment.apps/nginx-deployment  set image deployment.v1.apps/ngin
x-deployment nginx=nginx:1.9.1
deployment.apps/nginx-deployment image updated
deployment.apps/nginx-deployment image updated
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

We get a message (image updated)

kubectl get pods

To know more about pod



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl --record deployment.apps/nginx-deployment  set image deployment.v1.apps/ngin
x-deployment nginx=nginx:1.9.1
deployment.apps/nginx-deployment image updated
deployment.apps/nginx-deployment image updated
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
jenkins-pod                         1/1     Running   0          10m
nginx-deployment-6fdc797dc6-cm29w   0/1     Pending   0          11s
nginx-deployment-7778fb954b-72x2d   1/1     Running   0          4m21s
nginx-deployment-7778fb954b-gzz6s   1/1     Running   0          4m21s
nginx-deployment-7778fb954b-h2pdk   1/1     Running   0          4m21s
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Upgrading one after other as see in above

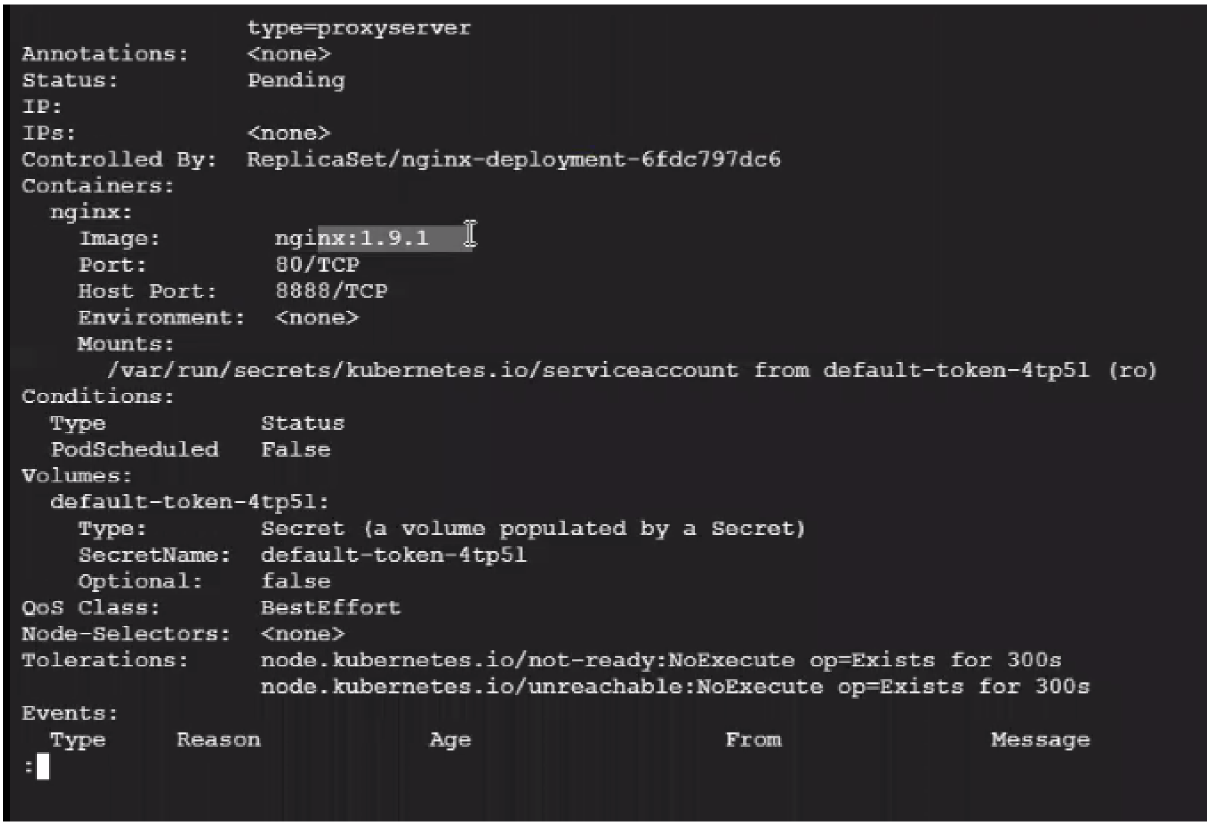
----

kubectl describe pods podname

take any pod name --nginx-deployment-6fdc797dc6-qrlqb

kubectl describe pods nginx-deployment-6fdc797dc6-qrlqb | less

we can see as Image: nginx:1.9.1



```
                type=proxyserver
Annotations:    <none>
Status:         Pending
IP:
IPs:            <none>
Controlled By:  ReplicaSet/nginx-deployment-6fdc797dc6
Containers:
  nginx:
    Image:        nginx:1.9.1
    Port:         80/TCP
    Host Port:    8888/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-4tp51 (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  default-token-4tp51:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-4tp51
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason          Age                 From              Message
:
```

:q

(It will take some time)

kubectl get pods

++

+++++++++++++++++++++++++++++++++++++++---------------------------

Service Object

3 nodes every node has ip address, to access we use ip

We access pod using service ip

This is used for network load balancing and port mapping.

Service Object uses 3 ports

1. Target port - Its is pod or container port

2. port - Refers to service port.

3. hostPort - Refers to host machine port to make it accessible from external network.

Service Objects are classified into 3 types

1. clusterIP: This is default type of service object used in kubermetes and it is used when we want the pods in the cluster to communicate with each other and not with external network.

2. nodePort: This is used, if we want to access the pods from an external network and it also performs network load balancing. ie Even if a pod is running on a specific slave, we can access it from other slave(node) in the cluster.

3. LoadBalancer: This is similar to nodePort. It is used for external connectivity of a pod and also network load balancing and it also assigns a public ip for all the nodes combined together.

++++++++++++++++++++++

```
vim pod-definition1.yml
```

We will be creating a service object for the labels used in pod-definition1.yml

```
kubectl create -f pod-definition1.yml
```

As we know by pod will be created using the above command.

We want to create service object for the above pod

Ex: Create a service definition file for port mapping on nginx pod

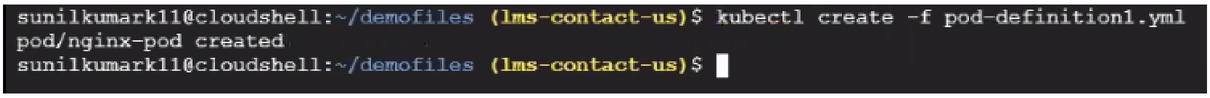
```
vim pod-definition1.yml
```

```
---
apiVersion: v1
kind: Pod
metadata:
 name: nginx-pod
 labels:
  author: sunil
  type: reverse-proxy
spec:
 containers:
  - name: appserver
    image: nginx

:wq
```

Using selector filed we can link with pod and service obj

Bylables we can link with to give same name.

kubectl create -f pod-definition1.yml



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f pod-definition1.yml
pod/nginx-pod created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Observation: Along with the pod, service object gets created.

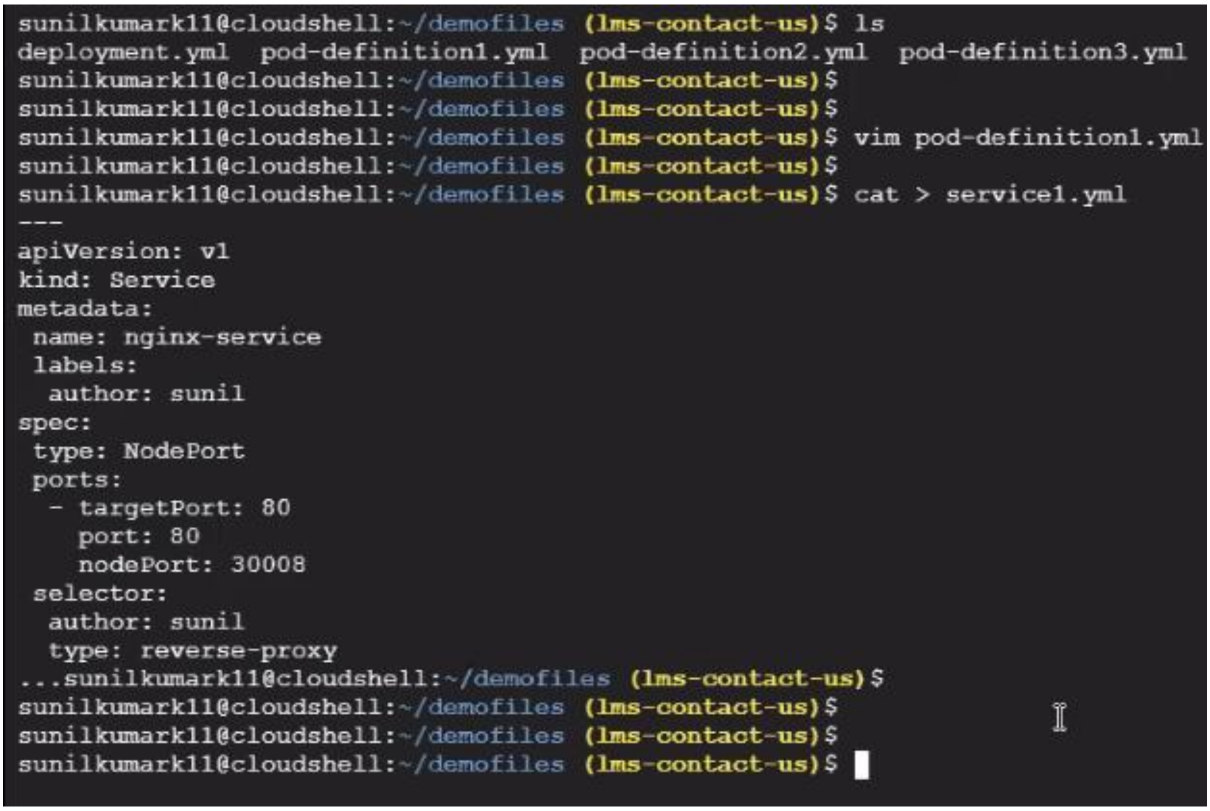
This service object is type clusterIP. Hence cannot be accessed from external network.

gcloud compute firewall-rules create rule3 --allow tcp:30008

vim service1.yml

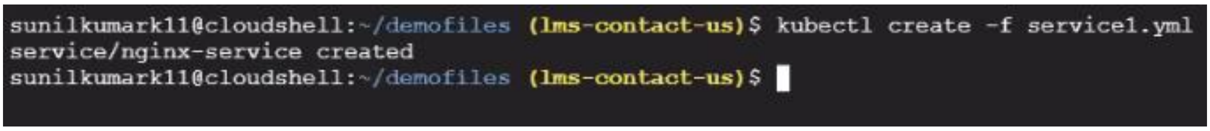
```
---
apiVersion: v1
kind: Service
metadata:
 name: nginx-service------------------service obj name
 labels:
  author: sunil
spec:
 type: NodePort
 ports:
  - targetPort: 80-----------------------container port
    port: 80
    nodePort: 30008
 selector:
  author: sunil
  type: reverse-proxy
...
```

:wq



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ ls
deployment.yml  pod-definition1.yml  pod-definition2.yml  pod-definition3.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ vim pod-definition1.yml
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ cat > service1.yml
---
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
 labels:
  author: sunil
spec:
 type: NodePort
 ports:
  - targetPort: 80
    port: 80
    nodePort: 30008
 selector:
  author: sunil
  type: reverse-proxy
...sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

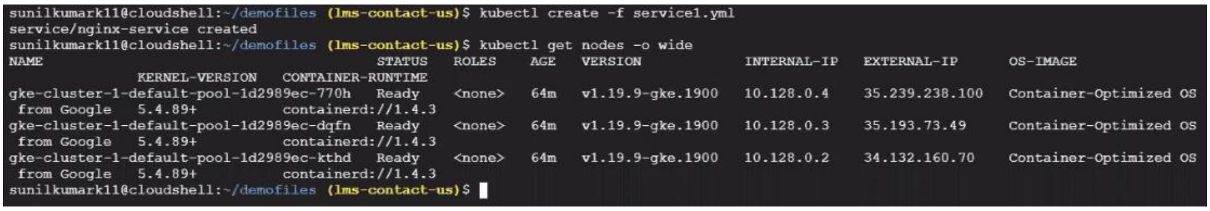
kubectl create -f service1.yml



```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f service1.yml
service/nginx-service created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

Now, the nginx pod is accessible externally

kubectl get nodes -o wide



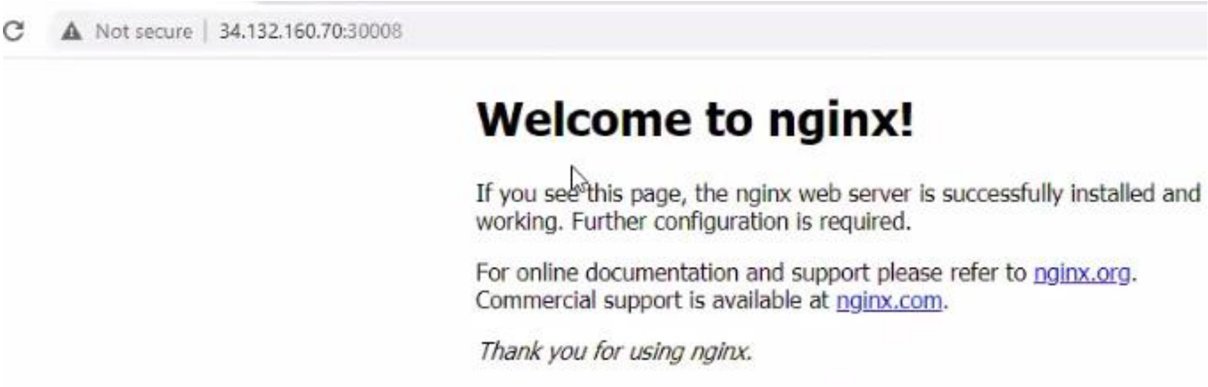
```
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl create -f service1.yml
service/nginx-service created
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$ kubectl get nodes -o wide
NAME                                        STATUS   ROLES    AGE   VERSION             INTERNAL-IP   EXTERNAL-IP      OS-IMAGE
            KERNEL-VERSION   CONTAINER-RUNTIME
gke-cluster-1-default-pool-1d2989ec-770h   Ready    <none>   64m   v1.19.9-gke.1900   10.128.0.4    35.239.238.100   Container-Optimized OS
 from Google   5.4.89+          containerd://1.4.3
gke-cluster-1-default-pool-1d2989ec-dqfn   Ready    <none>   64m   v1.19.9-gke.1900   10.128.0.3    35.193.73.49     Container-Optimized OS
 from Google   5.4.89+          containerd://1.4.3
gke-cluster-1-default-pool-1d2989ec-kthd   Ready    <none>   64m   v1.19.9-gke.1900   10.128.0.2    34.132.160.70    Container-Optimized OS
 from Google   5.4.89+          containerd://1.4.3
sunilkumark11@cloudshell:~/demofiles (lms-contact-us)$
```

As we have created nodePort, we should able to access from any node.

Take external_IP from anynode

34.66.234.81:30008   (We should be able to access nginx)

34.123.230.145:30008



Not secure | 34.132.160.70:30008

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.
Commercial support is available at nginx.com.

Thank you for using nginx.

If service object is not created, we used to identify in which node the pod is running, take that node IP, from that node IP, we used to access that application.

(Note: We need to open 30008 port in cluster)

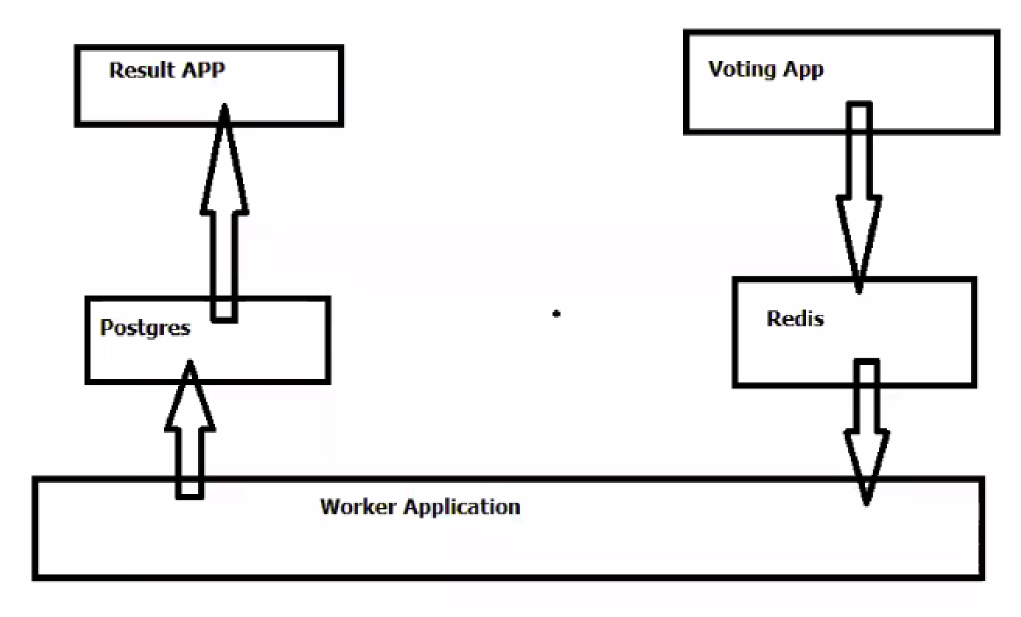
++++++++++++++++++++++

Kubernetes Project

This is a python based application which is used for accepting a vote (voting app).

This application accepts the vote and passes it to temporary db created using redis. From redis, the data is passed to worker application created using dotnet. Dotnet based application analyses the data and stores it in permanant database created using postgres.

From postgres database, results can be seen on an application created using node JS.



Have a look at the project Architecture.

Redis and postgres pod needs to assigned as cluster IP.

As cluster Ip is used for internal communication.

Voting App and Result App needs to be assigned as loadbalance type.

We need to create 5 definition files.

These 5 images related to this project is available in hub.docker.com

Using those images, we will create pods.

We need to create 5 pod definition files

We need to create 4 service files

We will be creating these definition files using pycharm.

```
vim voting-app-pod.yml



---
apiVersion: v1
kind: Pod
metadata:
  name: voting-app-pod
  labels:
    name: voting-app-pod
    app: demo-voting-app
spec:
  containers:
    - name: voting-app
      image: dockersamples/examplevotingapp_vote
      ports:
        - containerPort: 80
...



:wq



vim result-app-pod.yml


apiVersion: v1
kind: Pod
metadata:
  name: result-app-pod
  labels:
    name: result-app-pod
    app: demo-voting-app
spec:
 containers:
   - name: result-app
     image: dockersamples/examplevotingapp_result
     ports:
       - containerPort: 80
...
```

```
:wq
```

```
vim worker-app-pod.yml
```

```
---
apiVersion: v1
kind: Pod
metadata:
  name: worker-app-pod
  labels:
    name: worker-app-pod
    app: demo-voting-app
spec:
  containers:
   - name: worker-app
     image: dockersamples/examplevotingapp_worker
...
```

```
:wq
```

```
vim redis-pod.yml
```

```
---
apiVersion: v1
kind: Pod
metadata:
  name: redis-pod
  labels:
    name: redis-pod
    app: demo-voting-app
spec:
  containers:
   - name: redis
     image: redis
     ports:
      - containerPort: 6379
...
```

```
:wq
```

vim postgres-pod.yml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: postgres-pod
  labels:
    name: postgres-pod
    app: demo-voting-app
spec:
  containers:
    - name: postgres
      image: postgres:9.4
      ports:
        - containerPort: 5432
...
```

We are done with 5 pod definiton files.

We need to create 4 service definiton files.

vim redis-service.yml

```
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: redis-service
  labels:
    name: redis-service
    app: demo-voting-app
spec:
  ports:
    - port: 6379
      targetPort: 6379
  selector:
    name: redis-pod
    app: demo-voting-app
```

```
:wq
```

Note: As we have not specified the type of service object, by default it creates service object to type Cluster_IP

```
vim   result-app-service.yml
```

```
apiVersion: v1
kind: Service
metadata:
  name: result-service
  labels:
    name: result-service
    app: demo-voting-app
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
  selector:
    name: result-app-pod
    app: demo-voting-app
```

:wq

The above two service objects are of type load balancer. we can access it from the external network.

Open gitbash from the location where all the definition files are saved.

```
$ git init
$ git add .
$ git commit -m "a"
```

Open github ---> create new repository

Repository name - kuber_project

upload the files from local repository to remote repository using the two commands

$ git remote add XXXXX

$ git push XXXX

We should able to see the definition files in github repository (Total 9 files)

We need to download the 9 files into kubernetes cluster.

Login to GCP console

Create kubernetes cluster

Connect to the cluster

Get the repository URL in github.

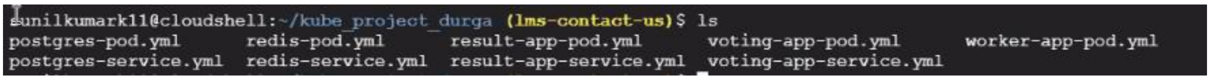
$ git clone rep_url

$ git clone https://github.com/sunildevops77/kube_project_durga.git

(Observation all the definition files will be downloaded)

$ cd kuber_project

$ ls   (we get the files)



```
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$ ls
postgres-pod.yml      redis-pod.yml      result-app-pod.yml      voting-app-pod.yml      worker-app-pod.yml
postgres-service.yml  redis-service.yml  result-app-service.yml  voting-app-service.yml
```

$ kubectl create -f voting-app-pod.yml

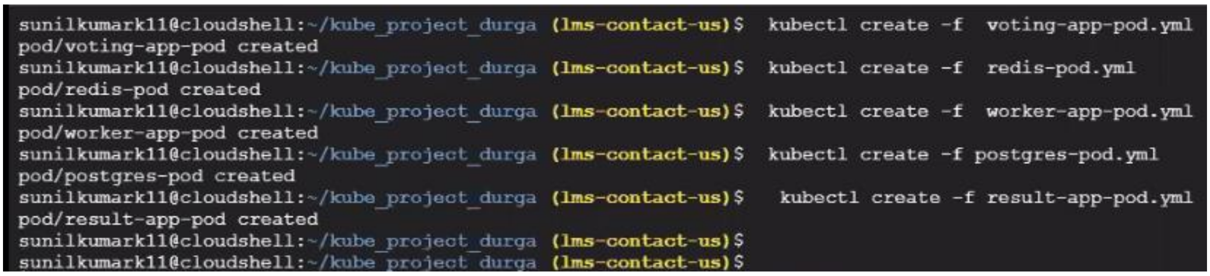
$ kubectl get pods (we should get one pod)

$ kubectl create -f redis-pod.yml

$ kubectl create -f worker-app-pod.yml

$ kubectl create -f postgres-pod.yml

$ kubectl create -f result-app-pod.yml



```
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$  kubectl create -f  voting-app-pod.yml
pod/voting-app-pod created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$  kubectl create -f  redis-pod.yml
pod/redis-pod created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$  kubectl create -f  worker-app-pod.yml
pod/worker-app-pod created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$  kubectl create -f postgres-pod.yml
pod/postgres-pod created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$   kubectl create -f result-app-pod.yml
pod/result-app-pod created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$
```

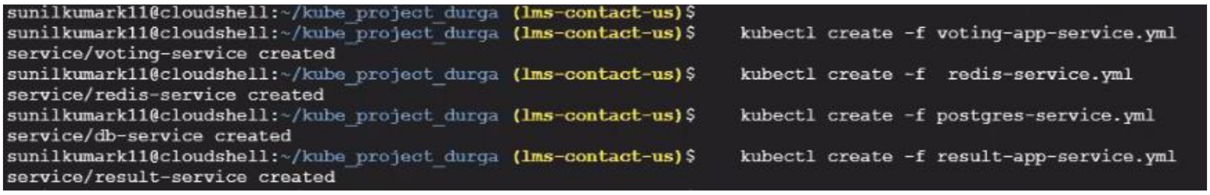
Now, we need to run service definition files

$ kubectl create -f voting-app-service.yml

$ kubectl create -f redis-service.yml

$ kubectl create -f postgres-service.yml

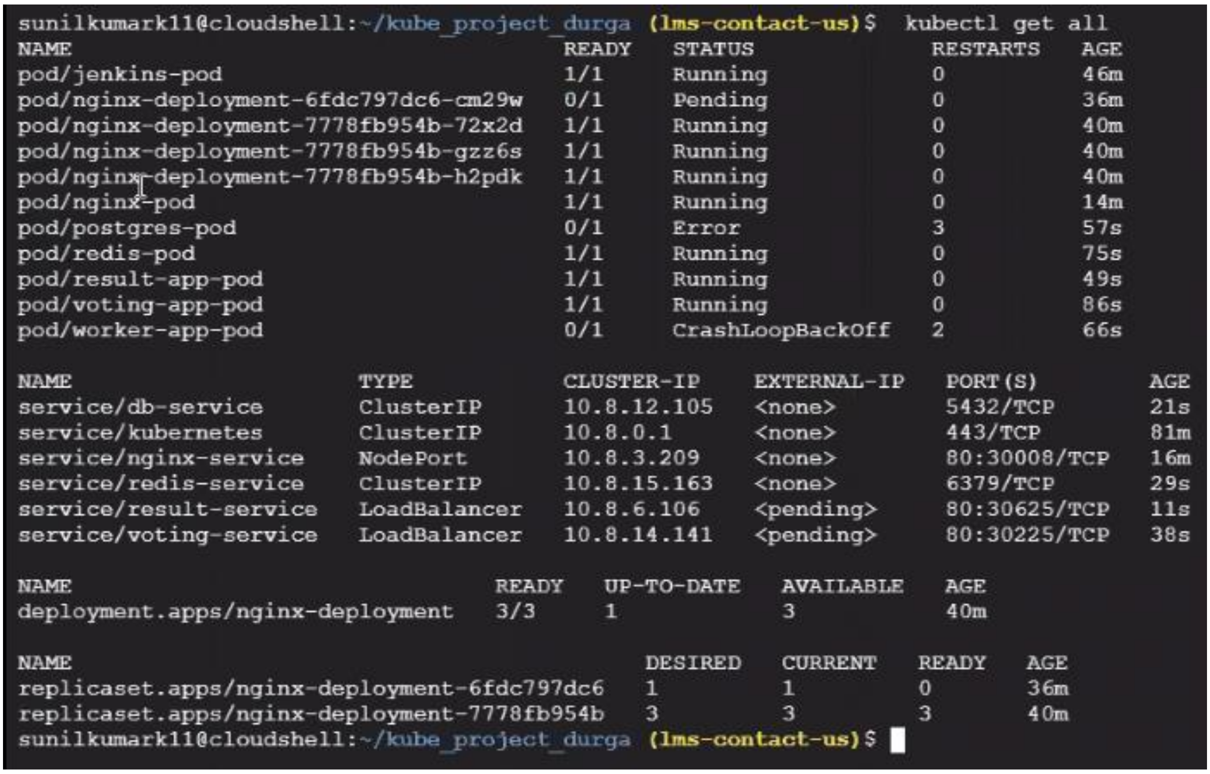
$ kubectl create -f result-app-service.yml



```
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$    kubectl create -f voting-app-service.yml
service/voting-service created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$    kubectl create -f  redis-service.yml
service/redis-service created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$    kubectl create -f postgres-service.yml
service/db-service created
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$    kubectl create -f result-app-service.yml
service/result-service created
```

To get all the information

$ kubectl get all



```
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$  kubectl get all
NAME                                    READY   STATUS             RESTARTS   AGE
pod/jenkins-pod                         1/1     Running            0          46m
pod/nginx-deployment-6fdc797dc6-cm29w   0/1     Pending            0          36m
pod/nginx-deployment-7778fb954b-72x2d   1/1     Running            0          40m
pod/nginx-deployment-7778fb954b-gzz6s   1/1     Running            0          40m
pod/nginx-deployment-7778fb954b-h2pdk   1/1     Running            0          40m
pod/nginx-pod                           1/1     Running            0          14m
pod/postgres-pod                        0/1     Error              3          57s
pod/redis-pod                           1/1     Running            0          75s
pod/result-app-pod                      1/1     Running            0          49s
pod/voting-app-pod                      1/1     Running            0          86s
pod/worker-app-pod                      0/1     CrashLoopBackOff   2          66s

NAME                      TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
service/db-service        ClusterIP      10.8.12.105   <none>        5432/TCP         21s
service/kubernetes        ClusterIP      10.8.0.1      <none>        443/TCP          81m
service/nginx-service     NodePort       10.8.3.209    <none>        80:30008/TCP     16m
service/redis-service     ClusterIP      10.8.15.163   <none>        6379/TCP         29s
service/result-service    LoadBalancer   10.8.6.106    <pending>     80:30625/TCP     11s
service/voting-service    LoadBalancer   10.8.14.141   <pending>     80:30225/TCP     38s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     1            3           40m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-6fdc797dc6   1         1         0       36m
replicaset.apps/nginx-deployment-7778fb954b   3         3         3       40m
sunilkumark11@cloudshell:~/kube_project_durga (lms-contact-us)$
```

We can see 5 pods and 4 services created.

Observation: worker pod also failed.

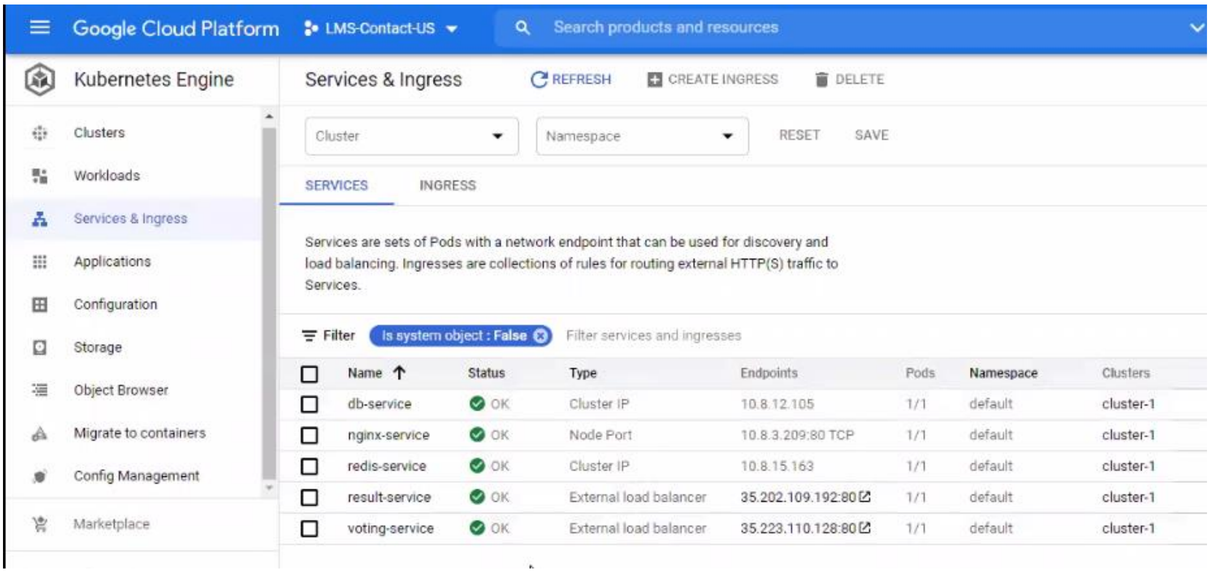
++++++++++++++++++++++++++++++++

These images are coming from community called dockersamples

Connection between workerpod and postgresPod is creating issues.

Go to service&ingress option in the kuberneted dashboard

We can see the four services, which are created.



Services are sets of Pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to Services.

☐	Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters
☐	db-service	✓ OK	Cluster IP	10.8.12.105	1/1	default	cluster-1
☐	nginx-service	✓ OK	Node Port	10.8.3.209:80 TCP	1/1	default	cluster-1
☐	redis-service	✓ OK	Cluster IP	10.8.15.163	1/1	default	cluster-1
☐	result-service	✓ OK	External load balancer	35.202.109.192:80	1/1	default	cluster-1
☐	voting-service	✓ OK	External load balancer	35.223.110.128:80	1/1	default	cluster-1

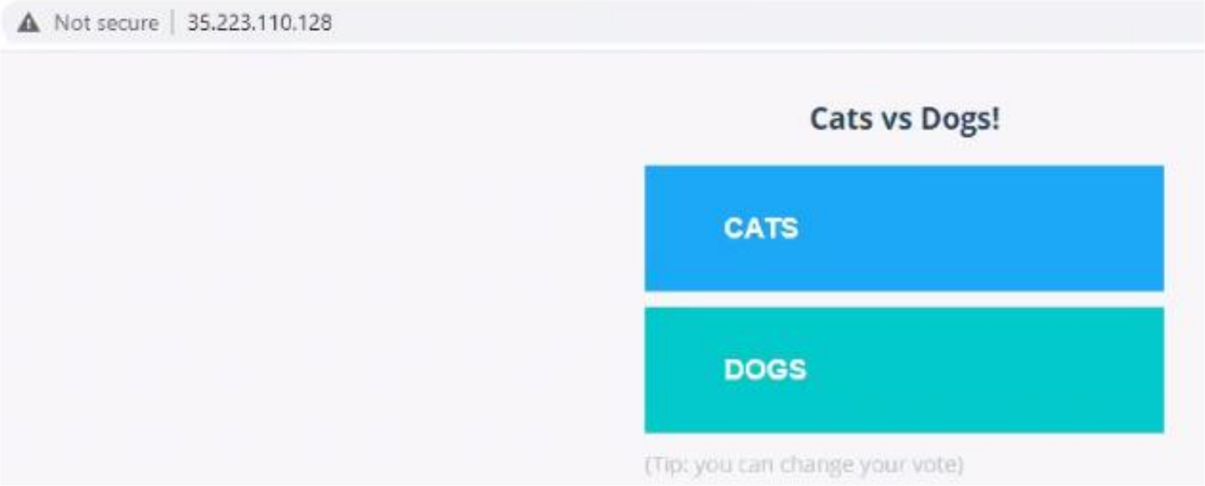
Note:voting app,result app have external ip addresses

Bcz they are type of load balancer so additional ip gets created

Any service obj of of load balancer so additional ip gets created

Click on endpoint (IP) of voting Application

Click in URL, we get Voting App (CATS / DOGS) -- This is python based App.



Click on endpoint (IP) of result Application

Click in URL, we get Result App

Whenever you want app to have public access. in that case, we should create service obj for the pod and then service obj should be type is load balancer so we can pickup the loadbalancer ip address to access the pod

++++++++++++++++++++++++++++++++++++++