Contents

[Important Note 1](#_Toc101485272)

[Useful commands on google cloud shell 2](#_Toc101485273)

[1. Check kubectl version 3](#_Toc101485274)

[2. Create Deployment 3](#_Toc101485275)

[3. Expose Deployment 3](#_Toc101485276)

[4. Get events 3](#_Toc101485277)

[5. Create pod in cluster 3](#_Toc101485278)

[6. Get Pods 3](#_Toc101485279)

[8. Get replica sets 4](#_Toc101485280)

[9. Get service 4](#_Toc101485281)

[10. Get deployment 4](#_Toc101485282)

[11. Explain pods 4](#_Toc101485283)

[12. Describe pods 4](#_Toc101485284)

[13. Delete pods 4](#_Toc101485285)

[14. Increase replica set 4](#_Toc101485286)

[15. How to do version upgrade 4](#_Toc101485287)

[16. Get event sorted by time 4](#_Toc101485288)

[17. Get component list of master node-> 4](#_Toc101485289)

[Concepts 5](#_Toc101485290)

[Kubectl 5](#_Toc101485291)

[Node 5](#_Toc101485292)

[Pods 5](#_Toc101485293)

[Replica Set 6](#_Toc101485294)

[Deployment 6](#_Toc101485295)

[Service 7](#_Toc101485296)

[GCP Region and zones 7](#_Toc101485297)

[Kubernetes architecture 8](#_Toc101485298)

# Important Note

GCP url -> [kushguptalikesu@gmail.com](mailto:kushguptalikesu@gmail.com). Free credit till 15-mar-2022

<https://console.cloud.google.com/>

<https://idemia.udemy.com/course/kubernetes-crash-course-for-java-developers/learn/lecture/16905386?start=0#notes>

To Make cluster size 0 ->

gcloud container clusters resize in28minutes-cluster --zone us-central1-c --project bustling-flux-344211 --num-nodes=0

gcloud container clusters resize NAME --zone ZONENAME --project PROJECTNAME --num-nodes=0

To make cluster size 3 again ->

gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-c --project bustling-flux-344211

gcloud container clusters resize in28minutes-cluster --zone us-central1-c --project bustling-flux-344211 --num-nodes=3

To login via glocud sdk shell on local->

Open gcloud shell

Type -> gcloud auth login

After login via [kushguptalikesu@gmail.com](mailto:kushguptalikesu@gmail.com) you can type same commands that we are typing on google cloud console web app.

Set project id as first step to ->

gcloud config set project bustling-flux-344211

and then to check current project is set properly ->

gcloud config list core/project

To run kubectl command kubectl needed to be installed on local.

commands list->

https://github.com/in28minutes/kubernetes-crash-course#commands-executed-during-the-course

# Useful commands on google cloud shell

For every command plural and singular both can be used. Like kubectl get pods or kubectl get pod both will work.

### Check kubectl version

kubectl version

### Create Deployment

kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE

It will deploy app to k8s cluster and return deployment id. It will create deployment, replica set and pod

### Expose Deployment

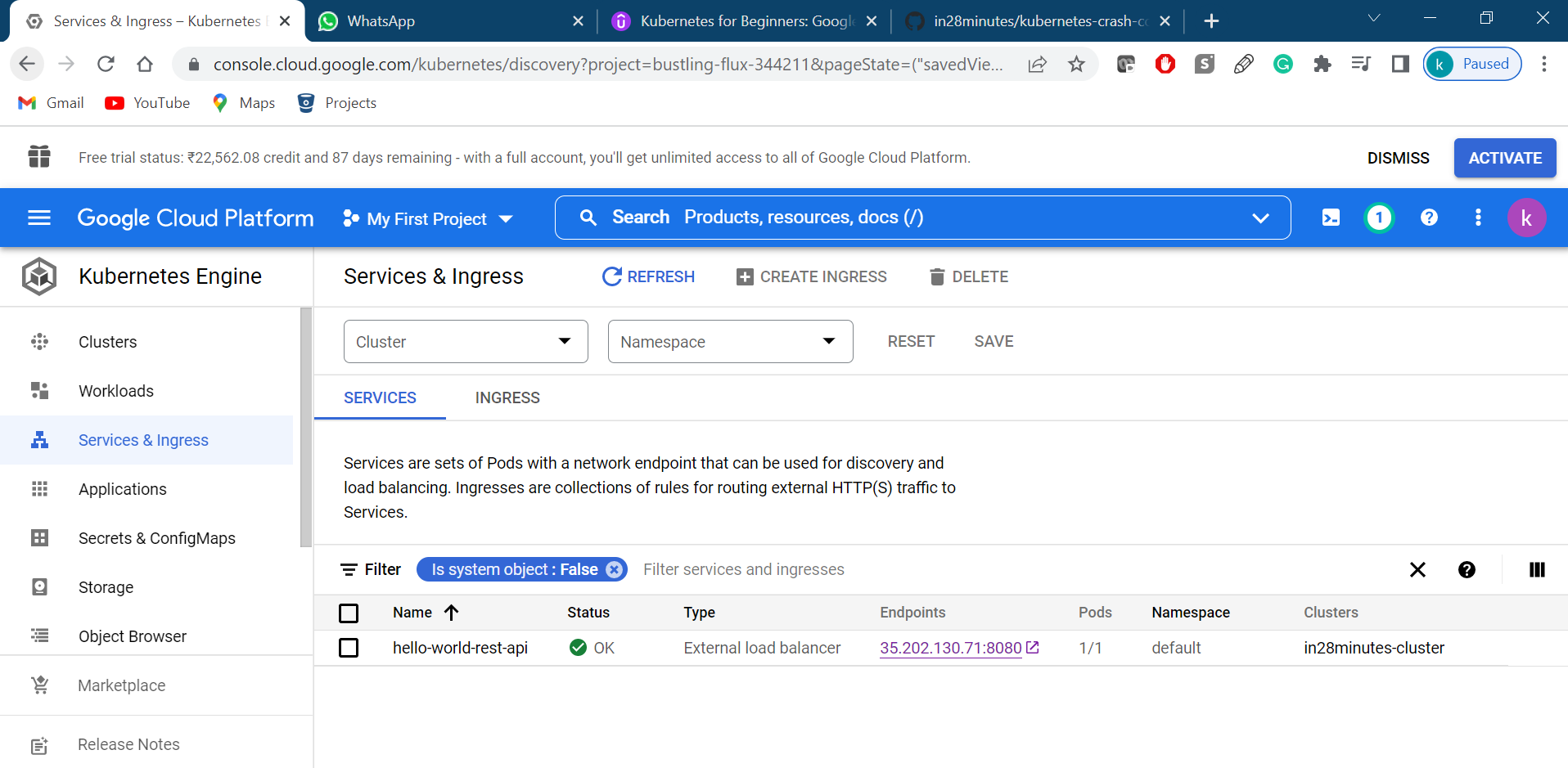
kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080

to be able to access application. We need to expose the deployment to the outside world. For that expose command is used. Where we specify port and type. It will create Kubernetes service.

Type=LoadBalancer will specify that service will be of load balancer of google cloud. If same command is run in aws then aws specific load balancer will be used. Which will take request and load balances pods.

Once the application is up and running go to services section in Kubernetes-> service & ingress section and verify health as ok and then click url mentioned in Endpoints to access app.

Example ->



<http://35.202.130.71:8080/hello-world>

here ip is the ip of load balancer service which we created and hence on add/delete of pods this ip remain same.

### Get events

kubectl get events will tell all the events happened. Like to deploy application pods ,replicas set created. Image pulled from docker hub. And finally service created and running.

### Create pod in cluster

kubectl run nginx1 –image=nginx

it create pod named nginx1 using docker image nginx from docker hub.

### Get Pods

kubectl get pods : give all pods details

kubectl get pods -o wide : give all pods details with extra info like ip.

1. Open Bash of running pods

kubectl exec -ti <podname> -n <namespace> -- bash

### Get replica sets

kubectl get rs : give all rs info

kubectl get -o wide rs

Here in output desired will tell number of desired pods to be running in a node.

Current will tell no of pods currently running.

And ready will tell no of pods ready to serve request.

### Get service

kubectl get service

### Get deployment

kubectl get deployment

### Explain pods

kubectl explain pod

give documentation of pod.

### Describe pods

Kubectl describe pod #podNameGotFromGetPodsCommand

From here we get all detail of pod like ip, namespace, containerId, ImageID, start time of pods, current status of pod.

### Delete pods

kubectl delete pod #podNameGotFromGetPodsCommand

### Increase replica set

kubectl scale deployment #deploymentNameGotFromGetDeploymentCommand --replicas=3

here now replica set will ensure that always 3 pods must be running in a node.

### How to do version upgrade

kubectl set image deployment #deploymentName #containerName=#ImageName

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

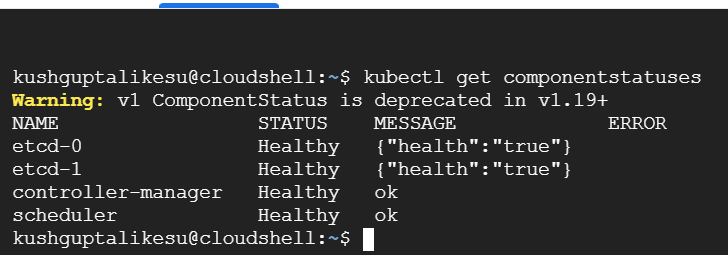
### Get event sorted by time

kubectl get events --sort-by=.metadata.creationTimestamp

### Get component list of master node->

kubectl get componentstatuses

it list master node components status



It show by default two instances of db is running.

# Concepts

Kubernetes also known as k8s as there are 8 letters between k and s

### Kubectl

Kube controller is the tool by which we can interact with k8s cluster. By it we can deploy application, increase no of instances of application etc. Kubectl commands remain same for all vendors whether it is AWS, GCP or Azure. It has client and server. Via client we interact and later kubectl uses server to contact with clusters.

### Node

They are virtual servers. Like amazon called them ec2, google call them compute engine. In Kubernetes they are called nodes.

In Kubernetes there are 2 types of nodes->

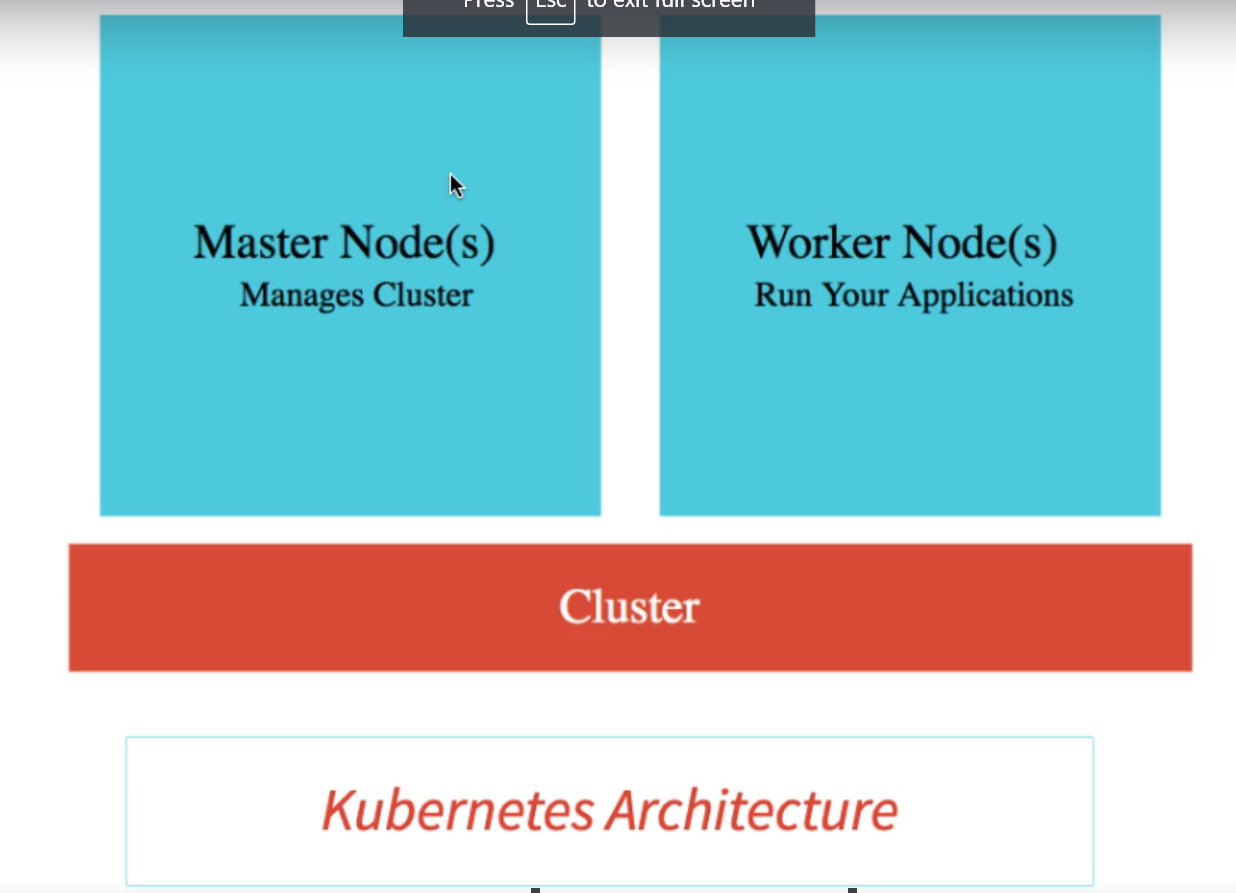
Master node and worker nodes.

Master node is like a manager who manages worker nodes and manages cluster.

Worker node do the job of running application.

In a cluster there are multiple nodes.

In a typical cluster there is only 1 master node and multiple worker nodes.



We can create cluster with google cloud. Kubernetes engine-> cluster-> create cluster-> select zone.

By default, cluster will contain 3 nodes.

In nodes section of newly created cluster, we can see nodes info. Like memory CPU zone etc. In every node some space is taken up by Kubernetes for internal work like managing nodes.

### Pods

Pod is a collection of containers that can run on a host. This resource is created by clients and scheduled onto hosts.

It is the smallest deployable unit in Kubernetes. In a single pod 0 to many containers might be running. By default, single container run inside pod.

Why pods? Why not k8s used container since in most of the cases single container is running in 1 pod. It is because k8s want to provide abstraction by calling it pod and hence we can also use vagrant in place of docker. So yaml file of k8s will remain unchanged. As it uses pod everywhere.

There are some use-cases where we might need to run 2 containers in single pod. Like sidecar where it wants to provide some support to main container application. All containers present in pod share resources and within same pod containers can talk to each other using local host.

**Every pod has 1 default hidden container also running named ‘pause’. We cannot see pause container directly.**

On a single node multiple pods might be running.

Multiple nodes together make a cluster.

Each pod has unique IP Address.

Every pod can be related to different application or same application.

### Replica Set

ReplicaSet ensures that a specified number of pod replicas are running at any given time.

kubectl get rs ->

Here in output desired will tell number of desired pods to be running.

Current will tell no of pods currently running.

Ready will tell no of pods ready to service request.

By-Default, value is 1.

So if we delete any pod and suppose replica set is 1. rs always monitor the pods and if any pod goes down, Then it will automatically create new pod for us.

Everytime a new pod deleted new pod will be created with different ip address.

To set rs to 3 of any deployment use below command ->

kubectl scale deployment hello-world-rest-api --replicas=3

by default single pod runs single container.

### Deployment

In ui of google cloud under Kubernetes section we can see under workloads section deployment we have created. We can use ui also to create everything. But prefer kubectl commands as they are cloud independent.

1. by deployment we can do updates in pods and rs.

2. when we create deployment It will deploy app to k8s cluster and return deployment id. It will create deployment, replica set and pod

3. when we expose deployment it creates service.

4. by deployment we can also do version upgrade of application without downtime.

kubectl set image deployment #deploymentName #containerName=#ImageName

if imageName does not exist. In such case kubectl will still make older version of application kept running. As newer one is not working.

If we type kubectl get rs -o wide -> it will display two rs. One with 3 pods and older image. 2nd with 1 pod and newer image.

Similarly pod will list 4 pods details out of which 1 will be of newer image and status for first 3 will be running and last one will be ErrImagePull and also ready will be 0 for last pod.

For detail type – kubectl describe pod #podNameOfNewer

Flow->

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

First k8s try to create 1 pod of newer image. If it succeed it will kill 1 pod of older version image. Then again it will create 2nd pod of newer image. If it succeed it will kill 2nd pod of older version.

At last we will have 3 pods of newer image running and 0 pods of older version running

If we type kubectl get rs -o wide. It will show 3 rs out of which 2 will be of desired value 0 and 1 will be of 3 pods which we have recently deployed of version 2. Two non running rs will be one of older version and one wrong dummy image which we have tried earlier.

kubectl get pods -o wide ->

No of pods will be 3 which will be of newer version.

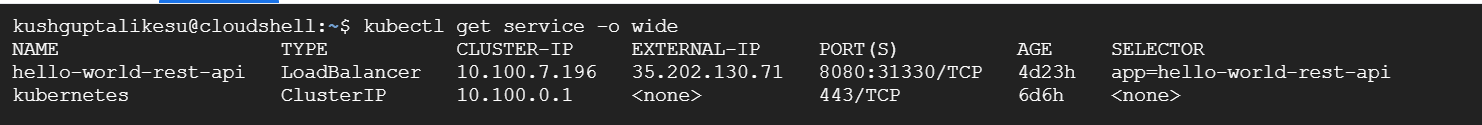
### Service

Consumer uses service to access application. Even if pods goes up and down, service ip address remain same. Hence end-user does not need to handle anything.

Service is created when we run expose deployment command. In our case ip of service is basically a load balancer and hence pods up and down does not impact its ip.

Type=LoadBalancer will specify that service will be of load balancer of google cloud. If same command is run in aws then aws specific load balancer i.e. ALB will be used. Which will take request and load balances pods.

Kubectl get services -> will display two services 1 will be of loadbalancer type and other is clusterIp of Kubernetes service. This 2nd service is not accessible outside.



### GCP Region and zones

1.region is the geographic unit. Like Mumbai.

2. zone is the isolated independent data center. Every region can have multiple zones. Like Mumbai has 3 zone in google cloud.

They provide availability, latency.

### Kubernetes architecture

In Kubernetes a nodes inside cluster are deployed in various zones. You cannot make one node in 1 region and 2nd node of cluster in other.

There are two type of nodes in k8s architecture ->

1. master node (control panel)

2. worker node or node

Master node in gke is managed by google cloud itself and we only manages worker nodes.

On Master node we does not deploy our own application it has different services which deal with cluster management. On worker nodes app is deployed.

Master Node components ->

1. Api server ->kube-api server provides api to interact with k8s. kubectl commands or GCP UI both invoke these api to perform operations on k8s cluster.
2. etcd-> it is a distributed database. And it contains desired state of cluster. Like no of replicas, load balancer policy, no of pods running, deployments, services etc. typically we make 3 replicas of etcd to make Kubernetes cluster always available.
3. Scheduler -> schedules the pods onto the nodes. It may take decision on basis CPU available, ram available etc.
4. Controller manager-> It check if as per desired config system is up and running. It manages overall health of the cluster.

Worker nodes or nodes->

1. kubelet-> also known as node agent. it is an agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.
2. Container runtime-> environment require to run the container. For example -> docker runtime. Any container following open container interface runtime specifications.
3. Pods -> group of 1 or more containers.
4. Kube proxy-> it is a network proxy. It help in exposing services.

If master nodes goes down, application still remain up and running. As application are running on worker nodes. Yes, we will not be able to do any changes in cluster running. But existing application will be kept running.