

# Kubernetes Concepts

Starting kubernetes

1. Open terminal check the running docker service , if not running the run the docker service  
systemctl status docker  
systemctl start docker
2. We work with minikube, so start the minikube (it takes a lot of time)  
minikube start
3. Go to your directory, where you created the yaml file for various objects  
Cd kub-dir
4. Run your yaml file to create the object and see your running object  
Kubectl apply -f sp-pod.yml  
Kubectl get pods

you would get good content about services

<https://www.baeldung.com/kubernetes-find-service-url>

Important links by Amit Sharma

[https://github.com/amit17133129/freelance\\_projects/blob/main/minikube-install-aws.md](https://github.com/amit17133129/freelance_projects/blob/main/minikube-install-aws.md)

<https://github.com/Mirantis/cri-dockerd?tab=readme-ov-file>

Installation instructions for k8

<https://github.com/amolshete/Minikube-Installation>

Same video - <https://youtu.be/rO5a1TzsZGY>

Notes for kubernetes

<https://learning-ocean.com/tutorials/kubernetes/kubernetes-container-orchestration/>

Official documentation

[https://kubernetes.io/docs/reference/kubectl/generated/kubectl\\_explain/](https://kubernetes.io/docs/reference/kubectl/generated/kubectl_explain/)

K8 short names

<https://gist.github.com/piotrpersona/abc28ed2c251c71127bd7d15300f2ae5>

Very imp command -

minikube delete && minikube start

Commands told by sir

Run

create

Explain

Describe —

Get pods –

Delete pod —

Label pod –

## Twist

1. Agar particular tag ke saath label provide kar diya hai to uss tag ki value change nhi kar sakt, but –overwrite ke through possible hai ( he he he )
2. Run a pod with a label using - - (two dashes)
3. - - dry-run (don't execute the command, just gives the output) (just check the syntax of command).
4. - o for getting output (also use - - dry-run) > (redirect the configuration) your-file.yaml
5. Bina kisi container ke andar ke andar jake uss container par command chalana.  
Problem ki baat ye hoti hai ki ek pod ke andar multiple containers run karre hote hai.  
No problem yrr, simply you should know the pod-name and container-name  
Kubectl exec <pod-name> -c <container-name> <command-to-run>
6. For downloading ssh-client  
apt install openssh-server openssh-client
7. Chatting with other machine in same network  
netstat -nltp // Checking active internet connections  
Netcat -l -p 4444 // Listening to this port 4444  
Telnet localhost 4444 // Establishing telnet in some network with some port
8. Kisi service ko pod se connect karne ke liye, service ka selector should be equal to label of pod
9. Nodeport service ko active karne ke liye ye command chalana imp hai  
minikube service <nodePort-service-name> --url  
Then service web browser se accessible hogi
10. **Commands related to memory detail**  
Docker container stats <container id>  
// This command tells the memory utilisation and memory limit of a particular container  
Free -m // it gives the details about the ram

## Detailed Notes

### Kubectl label

```
# Update pod 'foo' with the label 'unhealthy' and the value 'true'  
kubectl label pods foo unhealthy=true
```

```
# Update pod 'foo' with the label 'status' and the value 'unhealthy', overwriting any existing  
value
```

```
kubectl label --overwrite pods foo status=unhealthy

# Update all pods in the namespace
kubectl label pods --all status=unhealthy

# Update a pod identified by the type and name in "pod.json"
kubectl label -f pod.json status=unhealthy

# Update pod 'foo' only if the resource is unchanged from version 1
kubectl label pods foo status=unhealthy --resource-version=1

# Update pod 'foo' by removing a label named 'bar' if it exists
# Does not require the --overwrite flag
kubectl label pods foo bar-
```

## Kubectl create

```
# Create a pod using the data in pod.json
kubectl create -f ./pod.json

# Create a pod based on the JSON passed into stdin
cat pod.json | kubectl create -f -

# Edit the data in registry.yaml in JSON then create the resource using the edited data
kubectl create -f registry.yaml --edit -o json
```

## Kubectl run

```
# Start a nginx pod
kubectl run nginx --image=nginx

# Start a hazelcast pod and let the container expose port 5701
kubectl run hazelcast --image=hazelcast/hazelcast --port=5701

# Start a hazelcast pod and set environment variables "DNS_DOMAIN=cluster" and
# "POD_NAMESPACE=default" in the container
kubectl run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMAIN=cluster"
--env="POD_NAMESPACE=default"

# Start a hazelcast pod and set labels "app=hazelcast" and "env=prod" in the container
kubectl run hazelcast --image=hazelcast/hazelcast --labels="app=hazelcast,env=prod"

# Dry run; print the corresponding API objects without creating them
```

```
kubectl run nginx --image=nginx --dry-run=client

# Start a nginx pod, but overload the spec with a partial set of values parsed from JSON
kubectl run nginx --image=nginx --overrides='{"apiVersion": "v1", "spec": { ... }}'

# Start a busybox pod and keep it in the foreground, don't restart it if it exits
kubectl run -i -t busybox --image=busybox --restart=Never

# Start the nginx pod using the default command, but use custom arguments (<arg1 .. argN>) for that command
kubectl run nginx --image=nginx -- <arg1> <arg2> ... <argN>

# Start the nginx pod using a different command and custom arguments
kubectl run nginx --image=nginx --command -- <cmd> <arg1> ... <argN>
```

## Kubectl delete

Kubectl delete pods –all (deleting all pods)

## Kubectl create

Kubectl create -f <path of your yaml file>

## Kubectl get

Kubectl get pods  
Kubectl get pods -o wide  
Kubectl get pods –show-labels

## Kubectl edit

Kubectl edit pod <pod-name>

## Kubectl exec

kubectl exec --stdin --tty shell-demo -- /bin/bash  
(env command to check the environment variables)  
Kubectl exec <pod-name> -it /bin/bash (pod ke andar ek hi container hai, thus no need to give the name of container)  
Kubectl exec <pod-name> -c <container-name> -it /bin/bash (see the container name from yaml file)

**Note : you have to be in the same namespace where the pode is created.**

## Kubectl expose

```
Kubectl expose pod <pod-name> - -port=8000 - -target-port=80 - -name <new-service-name>
```

// creating new service at a particular port (cluster ip concept)

```
Kubectl expose pod <pod-name> - -type=NodePort - -port=8000 - -target-port=80 - -name <new-service-name>
```

## Kubectl describe

```
Kubectl describe service <service-name>
```

```
Kubectl describe pod <pod-name>
```

## Yaml file

1. Firstly use explain command to get some reference

```
Kubectl explain <resource name>
```

```
Kubectl explain <resource name> - - recursive | less // this will help to make the yaml
```

2. Our first pod

```
apiVersion: v1 // these 2 sections are fixed as per (kubectl explain <resource name>  
Kind: Pod
```

```
Metadata: // data about data
```

```
    Name: my-first-pod
```

```
    Labels:
```

```
        Env: prod
```

```
Spec:
```

```
    Containers: // it's an array, you will specify all the containers
```

```
        - name: containername
```

```
        Image: something
```

- 3.

```
yaml file for service
```

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
    name: my-internal-service
```

```
spec:
```

```
    selector:
```

```

    app: secondpodlb // This should be present as label
  Ports:           // For cluster port
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP

```

4.

```

apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
spec:
  selector:
    app: my-app
  type: NodePort
  ports:
    - name: http
      port: 80
      targetPort: 80
      nodePort: 30036 // For nodeport
      protocol: TCP

```

## Flow of learning

1. Working with pods (creating, listing, delete)
2. Working with yaml files (short)
3. Difference between kubectl create and apply
4. Setting Environment variables
5. Entering inside the container
6. Executing commands in container via terminal and yaml file
7. Dealing with multiple containers
8. Concept of init containers
9. Cluster ip - services inside the pod
10. NodePod - service from outside pod
11. Replicatin controller - create/yaml-file/delete/edit (create,edit,replace - imperative object configuration - with create command) (for declarative - rc is running - just edit the yaml file then apply)
- 12.

## Kubernetes from docker

1. Docker container exec -it <container-id> <command>  
Kubectl exec <pod-name> -c <container-name> -it <command>
- 2.

Note: command is any linux command, if you want to go inside the terminal, just type /bin/bash.

## Examples

### 1. Running commands in container

```
apiVersion: v1
kind: Pod
metadata:
  name: myfirstpod
  labels:
    label1: harshal
    label2: gaurav
    label3: saurav
spec:
  containers:
    - name: firstcontainer
      image: coolgourav147/nginx-custom
      env:
        - name: myname
          value: Gaurav
        - name: city
          value: Mumbai
      args: ["sleep","30"] // This will execute the command when container will be 1st created
```

### 2. Making multiple containers

```
apiVersion: v1
kind: Pod
metadata:
  name: myfirstpod
  labels:
    label1: harshal
    label2: gaurav
```

```

label3: saurav
spec:
  containers:
    - name: firstcontainer
      image: coolgourav147/nginx-custom
      env:
        - name: myname
          value: Gaurav
        - name: city
          value: Jaipur
      args: ["sleep", "3600"]
    - name: secondcontainer // set up second container
      image: coolgourav147/nginx-custom

```

## Special commands

1. Kubectl get pods -o wide -w // load wide with watch option
- 2.

## Errors

1. Error in validating data  
You did not provide proper indentation
2. Label must contain alphanumeric character

## Replication Controller

11. Deleting replication controller without deleting the existing pods  
Kubectl delete rc - --cascade=false <rc-name>
12. Various ways of changing rc with imperative approach  
Kubectl get rc // keep watching this in another terminal  
Nano firstrc.yml // write the yaml file for rc  
Kubectl create -f firstrc.yml // create your rc  
Kubectl get rc // see the replicas in your rc  
Kubectl scale rc - --replicas=5 <rc-name> // scaling your rc  
Kubectl edit rc <rc-name> // this will open yaml file to do edit with vi editor only

- Vi firstrc.yaml // edit yaml file
- Kubectl replace -f firstrc.yaml // instead of create, we used replace
- 13. Various ways of changing rc with declarative approach
  - Vi first.yaml // create file
  - Kubectl apply -f firstrc.yaml // create rc
- 14. Vi firstrc.yaml // edit file
  - Kubectl apply -f firstrc.yaml // rc get configure
- 15. When a pod is part of a rc (through label) then there is a field in rc as “controlled by”.  
This shows that particular pod is controlled by a particular rc.
- 16. Rc ki yaml file ke andar selector wala field nhi hota
- 17. If there is already a pod running with specific label. If we created a rc which is specifying that specific label. Then what are the chances that , that particular pod would be owned by newly created rc.  
Answer is that , if the label matches between pod and rc then that particular pod would become part of newly created rc, if and only if that particular pod doesn't have any owner (controlled by).
- 18. Describe and Explain bahut confusing hai, Describe ke saath resource type + name bas aata hai, but explain ke saath only resource type aata hai (along with -recursive)
- 19. Replication controller ke saath agar selector field na de to vo metadata me present label ko as a default selector treat karta hai. Replication controller me selector (if present ) and template ka metadata same hona chahiye.
- 20.

## Deployment in K8

### Deployment Strategies

There are multiple strategies for deploying your application to your Kubernetes cluster, each with different advantages. The best one to use will depend on the situation.

**Ramped(Rolling)** -- A ramped or rolling deployment is the default deployment strategy with Kubernetes. New pods are brought online, and traffic is directed to them to ensure they are working as expected before old pods are removed. This is particularly useful for stateful applications, as Kubernetes keeps old pods alive for a grace period after redirecting traffic to the new pod to allow any open transactions to terminate. (default)

**Recreate** -- Unlike the other deployment strategies, a recreate strategy does involve downtime, as all pods are terminated before new pods are brought online. This avoids having two versions of a container running at the same time.

**Blue/Green** -- With a blue/green deployment, new pods are deployed alongside the existing pods. The new pods are tested before redirecting traffic to them. Although this strategy

requires double the resources, it makes it much easier to roll back in the event of a problem arising with the new deployment.

## **Hash in Deployment**

Deployment itself attaches hash to pods of replica set, this avoid mixing of pods between old replica set and new replica set.

# My Notes

# Kubernetes

# 21

## [ ] Pod creation with Replication Controller.

1. Write the yaml file for Replication Controller

→ Use commands

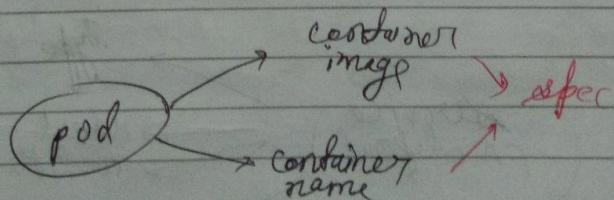
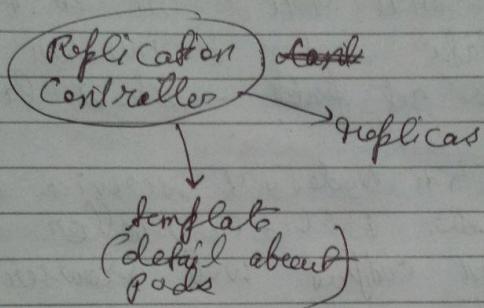
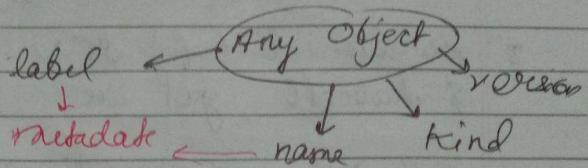
\$ kubectl explain rc

\$ kubectl explain rc --recursive

→ Inside yaml file, keep following details

→ It's a Replica Controller (give version v1 & kind as Replica Controller). Ref ~~the~~ sh name of Replica Controller as per your choice but label should be "app name = readingapp". This ~~RC~~ RC should create 5 replicas of Pod as per your desired name ~~by~~ but label should be "type = app". "First container" should be the name of container & "coolgourav147/nginx-custom" should be the image of container

→ You should be aware about the details that are needed to be given as per object of kubernetes.



// fingerprints → pod in big metadata  
(name, label) & spec (container -  
name and container-image) diff of  
100% 31142111

(3)

2:20 pm

21 05 2024

2. Create the Replica Controller using `apply` command.

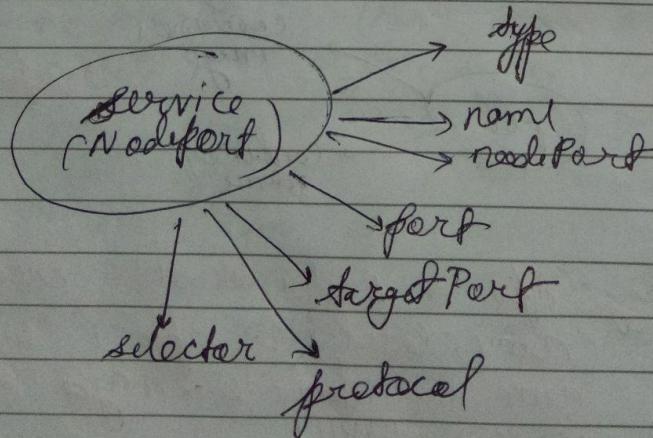
`$ kubectl apply -f myrc.yaml`

3. See the list of RC's

`$ kubectl get rc`

4. On notice created by this pods are lab  
Provide first name & 01 to Replica controller  
in metadata section & mention name will be  
`$ kubectl get pods --show-labels`

5. Sir of 5th NodePort service create one  
This is the Replica Controller get other all  
Pods on output web browser use curl  
and see



\$ kubectl apply -f myNodePortService.yml  
\$ minikube service <nodeport-service-name> --url

\$

6- Start Replica Controller in first Pod and  
delete one pod of Replica Controller now  
Pod creates are still

For testing above situation, one terminal  
window open the cmd & another terminal  
window for pods in watch mode & \$ kubectl  
(watch kubectl get pods). Other terminal  
window for \$ kubectl --show-labels pods and  
delete one & \$ ( \$ kubectl delete pod  
<pod-name> )

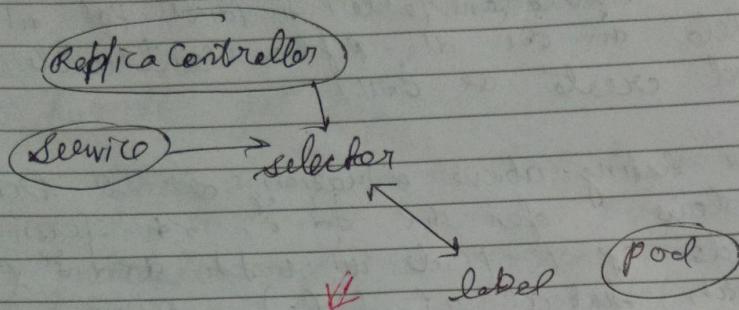
7- 3rd one Replication controller in watch  
and (continuous watch and wait) (\$ kubectl  
get rc -w).

At the same time, the pods get deleted  
and \$ ( \$ kubectl delete pod <pod-name> )  
Then we observe in Replication Controller  
in status (CURRENT & READY) it variations  
will be

~~And A Kubernetes object in its details  
will be its own self. One object in  
describe the find &  
\$ kubectl describe rc myrc.yaml~~

(S)

\$ kubectl get rc  
\$ kubectl describe rc <rc-name>



iff selector == label  
& pod part of service and belong list of rc

Then  
pod particular Replica Controller  
are part of that replica

8 If the first pod in (will be first) Replica controller will already have  $\delta$  label  
Remove old  $\delta$  and Replica controller  
will start to watch old  $\delta$

watching in another terminal window

\$  
\$ kubectl get rc <rc-name> -w  
\$ kubectl get pods --show-labels  
\$ kubectl label pod <pod-name>  
<label-key> -

As is dash sign

\$ kubectl delete -f myrc.yaml  
// this command does not delete the yaml file,  
// it just delete the object created by that  
// particular yaml file

(6)

## Deletion of Replica Controller

1. Sir Replication Controller will delete all  
and pods will watch and if at  
another window

\$ kubectl get pods -w

\$ kubectl delete rc <rc-name>

\$ kubectl get rc

2. Sir Replication controller object will  
delete and if there is no existing pod  
delete or else /

\$ kubectl get rc

\$ kubectl get delete rc --cascade=false  
<rc-name>

## Scaling of Replica Controller

1. Sir we scale it and it for Replication  
Controller at yaml file scale in value  
will change and it changes at  
apply or not & (Declarative Object  
Configuration)

\$ kubectl get rc

\$ vi myrc.yaml

\$ kubectl apply -f myrc.yaml

⑦

command

2. run Imperative today w/ Replica controller  
do scale and & /

\$ kubectl scale rc --replicas=5  
<rc-name>

\$ kubectl get rc

\$ kubectl get pods

3. run Imperative Object Configuration do  
through Replication Controller do scale and  
& /

\$ kubectl edit rc <rc-name>

\$ vi myrc.yaml

\$ kubectl replace -f & <rc-name> myrc.yaml

4. Vide #24 qz run driller & for static fairing  
multiple pod on label for all service (Replication  
Controller/ReplicaSet)  
# do selector to match need & of  
one pod for particular service on GKE  
own by all will

5. See told a way for each pod do other w/  
service own need &

\$ kubectl describe pod <pod-name>

in this pod has Replication controller  
under it find & it

the field add in which is  
controlled by: `ReplicationController/Spec.name`

↳ we also told, ~~that R.C is selector mention in the R.C file~~  
~~label is selector in the file~~  
~~selector and label different.~~  
~~it will pod R.C is under it (yaml), so you~~  
~~selector will effect on all the files~~

↳ we also told, ~~that R.C is selector mentioned in the R.C file~~  
~~pod in template is, yaml will get label is,~~  
~~as a selector treats and~~

### 6 Dry Run

yaml file on syntax check and if  
the dry run then  
\$ kubectl apply -f <name.yaml> --dry-run

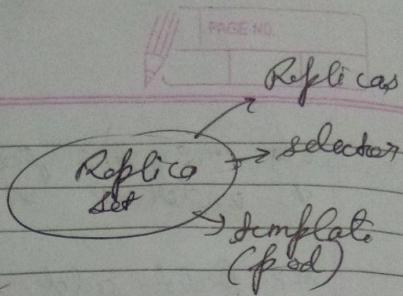
### 7 Syntax Help

If you forget the syntax of for particular  
type of object

\$ kubectl explain <object-type> --recursive  
it take the help of less

also use scrollable less to make content

## Replica Set



### 1. Special in Replica Set

Replica Set is nothing but Replication Controller in modified form &

Replica Controller only equality based selector can support more than but Replica Set  
Equality Based Selector in 2121-2121  
set based selector can support more of

### 2. Selector field is mandatory

Agar ReplicaSet के yaml file में Selector field नहीं होता तो error दिया जाएगा

### 3. Problem Statement -

Create a Replica Set, such that pod will  
at least app=myapp1 label present & all  
the Replica Set 3rd. particular pod will  
acquire the all /

spec :

selector :

matchLabels :

app: myapp1

### 4. Problem Statement

Create a Replica Set, such that pod will  
have any of the label → app=myapp1 OR app=myapp2

present & it is the Replica set for pod  
and acquire one or

spec:

selector:

matchExpressions:

- key: app

operator: In

values:

- myapp1

- myapp2

we can confirm above expression

\$ kubectl apply -f myrs.yaml

\$ kubectl get rs -n wide

// you will see SELECTOR

app in (myapp1, myapp2)

5. Replicaset will delete when 2nd Replica is deleted  
associated with pods delete if until

\$ kubectl get rs

\$ kubectl delete rs <rep-name>

## 6. Problem Statement

Create a Replica Set, which has 3 or more pods and  
acquires one with the same label as → app=

app = myapp1 or app = myapp2. but the label  
is all → type = backend

6

describe → object ~~name~~ type + name (See details)  
explain → object type  
1 PAGE NO.  

	See Syntax
--	------------

Just like,

$(\text{app} \\ \text{my app1}) \parallel (\text{app} \\ \text{my app2})$ ) && !.type backend

*spec:*

selektor:

## match Expressions

- key = app

opera  
values?

- myapp1
- myapp2

- key? type

operator: Not In

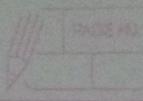
values;

- backend

Control AND  
operation এবং প্র  
সমস্যা

7. Mr suggestion of 25 at the Development Production Environment it stand-alone mode create  
dict in during 3rd file by default  
label provide and display

Otherwise 1321 environment at 31012 and service  
same label and selector use one ref slot  
at 31012 stand alone fed, launch another  
service in through control, if ref exist,  
without your information. Even ~~the~~ 321



(12)

unknown service is delete ~~start at 2012-01-01~~  
at 2012 stand alone pool get delete ~~at 2012-01-01~~

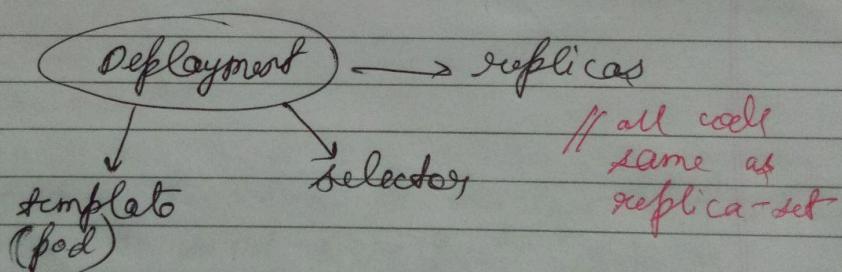


# Deployment Concept



1 Deployment Replica set, can be modified from existing to add new version deployment. deploy and time de-escalation can minimise downtime and previous version is easily rollback provide default &

$\rightarrow$  type  
// change  
 $\hookrightarrow$  next  
api version



2. If new deployment added & it, Kubernetes automatically adds selector and this will be  $\text{pod-template-hash} = 9766d747f$

3rd deployment (Replica set) in different version can differentiate the pod and it's (pod is automatically new label will attach to it)

services

19

\$ kubectl get svc -o wide //unless

\$ kubectl get rs -o wide

get 90% - 80%  
1/2 of deployment credit for 1st 20%  
1/2 of R&D in form of 1st 20%

\$ kubectl get pods --show-labels  
// pod & hash label attached

- 3. Use scale-up & scale-down cmd  $\rightarrow$   $\downarrow$  yaml file ~~cont~~ in Replicas cmd value cmd update cmd,
  - $\downarrow$  yaml file in apply cmd
  - $\downarrow$  pods cmd watch cmd (~~inside - w~~)

4. Sin ~~re~~ of Rollout word si introduced  
09/14/11

mid Deployment of new version deployment  
mid & roll-out

Ein Rollout der statistischen Modelle auf trigger

After following command use dir & ls and E /

~~✓ kubectl rollout status deployment~~

<deployment - name>

- 5 min Ready second

~~कृष्ण~~ विद्युति विद्युति विद्युति विद्युति विद्युति विद्युति

and it is ready EAT, then eat own ready

Second set-up 13 Dec 1983

Cards done

minReadySecond ~~set~~ & at entry  
Pod ~~read~~ alive else if 3rd fail  
rollBack / (start it)

#### 6. maxUnavailable

maxUnavailable: It will integer value 0, 1, 2, 3 etc  
It no. of pods mention and it will fail if  
unavailable else & will set new version in  
Pods deployment card & (start it)

Spec:

strategy:

rollingUpdate

maxSurge: 0

maxUnavailable: 1 → // 1st roll or 1st value

type: RollingUpdate

else, 3rd victim

new version deploy

rollBack /

#### 7. maxSurge

maxSurge denotes max & for existing Replicas  
in apart of EH Card extra Replicas  
create and 2nd &

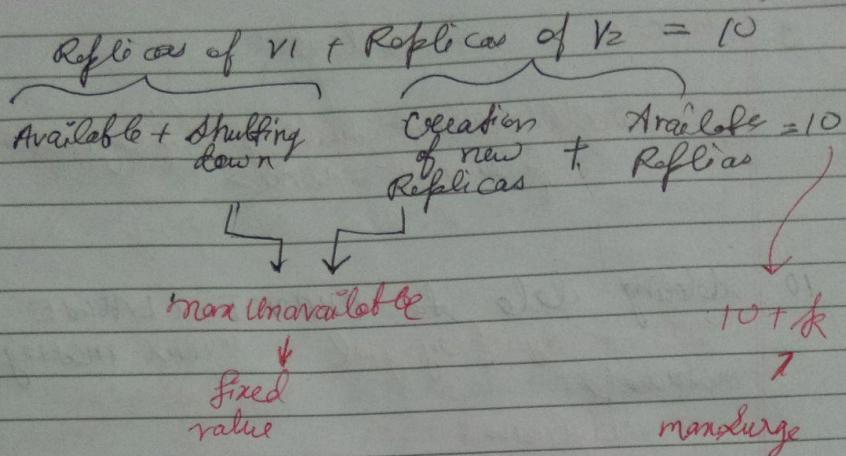
For eg

Replicas = 10

Replicas of  $v_1$  + Replicas of  $v_2$  = 10

(16)

Total  
Replicas



Suppose, max surge =  $k$

$$RHS \rightarrow RHS + k.$$

$\Rightarrow RHS \uparrow$  (increases)

$LHS \rightarrow$  Availability  $\uparrow$  (increases)

Hence

max surge set-off error  $\Rightarrow$  EHR  $\Rightarrow$  Unavailable  
Replicas increase  $\Rightarrow$  EHR  $\Rightarrow$  Availability and  
availability increase  $\Rightarrow$  EHR

\$ subject to  $\approx 0^\circ$  wide

8. What are the default values -

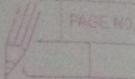
Strategy Type : Rolling Update (Availability more)

Min Ready Second : 0

max Unavailable : 25% of Replicas

max surge : 25% of Replicas

(7)



### 9. History of our rollout

\$ kubectl rollout history deploy <deployment-name>

### 10. Adding data to CHANGE-CAUSE

file = mydeploy.yaml "your message"

metadata:

annotations:

kubernetes.io/change-cause: "your message"

\$ kubectl apply -f mydeploy.yaml

\$ kubectl rollout history deploy <deployment-name>

### 11. Doing undo

\$ kubectl rollout undo deployment <deployment-name>

\$ kubectl rollout history deploy <deployment-name>

### 12. Get specific Revision of rollout history

\$ kubectl rollout history deploy <deployment-name>

\$ kubectl rollout undo --to-revision=2

deployment <deployment-name>

\$ kubectl rollout history deploy <deployment-name>

### 13 Pause/Resume deployment

\$ kubectl rollout status deployment <deploy-name>  
 // keep on watching the status of deployment  
 // in another terminal window

\$ vi mydeploy.yaml

\$ kubectl apply -f mydeploy.yaml

\$ vi mydeploy.yaml // change version of image

\$ kubectl apply -f mydeploy.yaml

\$ kubectl rollout status deployment <deploy-name>

\$ kubectl rollout pause deployment <deploy-name> // pausing deployment

\$

// keep on watching the deployment status

\$ kubectl rollout resume deployment

<deploy-name> // resume deployment

### 13 video #30

Question - what is the maximum value of  $\delta$

Revision 0 set at 21000  $\delta$ , at 21520

configure the next  $\delta$

## Deployment Recreate Strategy

1. Create a yaml file mydeploy having
  - Replicas
  - Selector
  - Scraps

and occurs

- apiVersion
- kind
- name
- label

2. giving strategy just after the Replicas

spec:

strategy:

Type: Recreate

\$ kubectl apply -f mydeploy.yaml

3. See Result of our Deployment

\$ kubectl get deploy

\$ kubectl get rs

\$ kubectl get pods

1. ~~3rd step~~ you file an edit on container image at least one of  $\delta$

~~but will get us~~

2. Development of during downtime in problem  $\delta$ , ~~so we have to recreate strategy~~ follow the standard  $\delta$

## giving Resource Request & Resource Limit (Pod)

1. In pod create and  $\delta$ , it have following
- container name
  - container image

& of course

- apiVersion
- kind
- name
- label

2. In Resource field of copy  $\delta$

~~kubectl explain --recursive pod | less~~

3. In Pod create and describe  $\delta$  and show  $\delta$  for pod field of resources utilize the  $\delta$

(2)

PAGE NO.

\$ kubectl describe pod <pod-name> | less

Sir says, this will field will tell you  
what the resource of the pod is going  
to use and how many resources

Then sir say Pod will delete the pod so

\$ kubectl delete pod <pod-name>

#### 4. Problem Statement

Create a pod which has memory 200 Mi  
allocate/available 100 & 500 milli-core  
CPU allocated/available 100

memory

spec:

containers:

resources:

requests:

memory: 100 Mi

CPU: 500.m / 0.1 // another way 0.1

\$ kubectl apply -f mypod.yaml

\$ kubectl describe pod <pod-name> | less

↑ you will see resources in request  
field

3. Sir says if pod del creates  $E_{1011}$ , it'll be if sufficient resources present else otherwise pod creates  $E_{1011}$  and status - pending show up  $E_{1011}$

### ~~\$ kubectl get pods~~

Sir says - If  $E_{1011}$  &  $E_{1012}$  pod don't create finally creates one in  $E_{1015}$  if Kubernetes cluster has enough resources pod attach on either  $E_{1011}$  or  $E_{1012}$  pod that fulfill its requirements

4. Sir says multi-container pod in case if Kubernetes scheduler pod del if creates  $E_{1011}$  or  $E_{1012}$  node if the resources available  $E_{1011}$  or  $E_{1012}$  pod in containers in resources all sum available  $E_{1011}$

Eg

a multi container pod (Pod1)

~~pod~~ Container1 needs 200 Mi memory  
Container2 needs 500 Mi memory

Scheduler Pod1 finds node if create  $E_{1011}$  or  $E_{1012}$  node if 700 Mi memory available (free  $E_{1011}$ )

## giving Resource Limit (Pod)

### 1 Problem Statement

create a pod for docker container  
 200 Mi ~~21~~ memory ch utilisation of  
 ch / also keep in mind, 321 particular  
 container ch ~~21~~ , Mi memory allocated/  
 available ~~21~~

Spec:

Containers:

Resources:

requests:

memory: 1 Mi

limit:

memory: 200 Mi

\$ kubectl get -f mypod.yaml

\$ kubectl get pods

\$ kubectl describe pod <pod-name>

2. after external worker node of the docker commands through Resources check and find it

\$ docker ps

\$ docker container list

\$ docker container stats <docker-image>

\$ free -m //linus command

# Kubernetes Study Phase 2

Video # 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50

## 1.What is namespace ?

Namespace ek **mechanism** hai, jisme hum kuch **clusters ko group** karne ke liye hum ek **virtual cluster environment** create karte hai. Iss environment mei hum kuch **conditions implement** kar sakte hai jese - resource limit set karna, access controls, variables set karna. Namespace ke through hum kubernetes me **multiple** virtual cluster environments create kar sakte hai jo ki ek doosre se **isolation** me kaam karte hai.

## 2.How to create a namespace ?

Creating a namespace with command

```
kubectl create namespace namespace1
```

Creating a namespace with yaml file

```
apiVersion: v1
kind: Namespace
metadata:
  name: namespace1
```

```
Kubectl apply -f my-ns.yml
```

Creating a pod by providing namespace name in yaml file

<complete yaml file for creating a pod>

Metadata:

```
  Namespace: <ns-name>
```

Creating a pod by providing namespace with command

```
Kubectl apply -f pod.yml --namespace <ns-name>
```

## 3.How to see list of pods running inside test ns (namespace)

```
Kubectl get pods -n test
```

## 4.See all the pods from all namespaces.

```
Kubectl get pods --all-namespaces
```

## 5.How to run kubectl command for other ns

```
Kubectl <command> -n <ns-name>.
```

Note - for creation of pod, we used --namespace while for executing kubectl command, we used -n.

## 6.Benefits of Namespace

- a. Preventing Conflicts between Multiple Teams (Isolation)
- b. Development stages
- c. Permissions (role based access control)
- d. Resource control
- e. Multiple resources with same name
- f. Create objects - ResourceQuota, ResourceLimit, Pods, Deployment, ReplicaController,

## 7.What are the namespaces that are created automatically

Here are the four default namespaces Kubernetes creates automatically:

- a. default—a default space for objects that do **not** have a specified namespace.
- b. kube-system—a default space for **Kubernetes system objects**, such as kube-dns and kube-proxy, and add-ons providing **cluster-level features**, such as web UI dashboards, ingresses, and **cluster-level logging**.
- c. kube-public—a default space for resources available to all users **without authentication**.
- d. kube-node-lease—a default space for objects related to **cluster scaling**.

## 8.How to set a namespace as default namespace

kubectl config set-context --current --namespace=<ns-name>

## 9.Write simply yaml file for pod creation

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  labels:
    app: myapp1
    type: frontend
spec:
  containers:
  - name: my-app-container
    image: coolgaurav147/nginx-custom
    imagePullPolicy: Never
  resources:
    requests:
      memory: "1Mi"
    limits:
      memory: "200Mi"
```

## **10. Perform a lab for accessing the service from same as well as different namespace.**

### **Steps**

1. Create a pod in default namespace having label as app=myapp and name as mydb.
  - a. Creating a pod as db (wrote yaml file)
2. Create a pod (name=myserver, namespace=default) and one more pod (name=testserver, namespace=test)
  - a. Create a pod as server (write yaml file)
3. Now create a service (name=mywebservice and selector as app=myapp)
  - a. Create a service via yaml file
4. Get the terminal access to container inside myserver pod and assess the service of the same namespace
  - a. get the terminal access to container of myserver
  - b. curl the service via service name
5. Now get the terminal access to container inside testserver pod and access the service of default namespace.
  - a. get the terminal access to container of testserver pod
  - b. curl the service via syntax - servicename.nsname.svc.cluster.local

### **yaml file for creating a pod**

```
apiVersion: v1
kind: Pod
metadata:
  name: myserver
  labels:
    app: myapp1
spec:
  containers:
  - name: my-contaninerk
    image: coolgourav147/nginx-custom
```

### **yaml file for creting a service**

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: webserver
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
```

## **11.How to delete a ns**

```
kubectl delete ns <ns-name>
```

## **12.What is ResourceQuota object in kubernetes**

ResourceQuota object ke through hum per **namespace** ke **aggregate resource consumption** par limit kar sakte hai by implementing **constraints**. We can limit the **quantity of objects** that can be created in a namespace by type, as well as the **total amount of compute resources** that may be consumed by resources in that namespace.

## **13.How to check the resource quota of current namespace**

```
kubectl describe ns <ns-name>
```

```
kubectl describe resourcequota <resourcequota-name>
```

## **14.Create a ResourceQuota which allows creation of only 2 pods**

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: myquota
spec:
  hard:
    pods:2
```

```
kubectl apply -f myquota.yml –namspac <ns-name>
```

Try to create 3 pods and see how many pod creation is allowing.

## **15.List the kubernetes objects for which we can set the ResourceQuota**

Prefer the following documentation for getting the list of objects for setting ResourceQuota  
<https://kubernetes.io/docs/concepts/policy/resource-quotas/>

## **16.Create a resourceQuota such that for cpu set request and limit as 0.5 and 1, for memory set request and limit as 500Mi and 1Gi**

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: myquota
spec:
  hard:
    request.cpu: 0.5
    request.memory: 500Mi
    limit.cpu: 1
    limit.memory: 1Gi
```

```
kubectl apply -f myquota.yml -n <ns-name>
```

<see the details about namespace and quota>

yaml file for creating a pod with some limits and request on cpu

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod-compute-check
  labels:
    app: mylabel
spec:
  containers:
  - name: my-app-container
    image: coolgaurav147/nginx-custom
    imagePullPolicy: Never
    resources:
      requests:
        cpu: 0.1
        memory: 250Mi
      limits:
        cpu: 500m
        memory: 500Mi
```

Try to create the pods in limit and over limit and check what is possible

**17.What happens if we do not provide the resource request with pod specification, we just only provides the request limits.**

At that time the value of resource limits would be assigned to pod request.

**18.What happens if we do not provide the resource limit with pod specification, we just only provides the request request.**

At that time, it would through the error during pod creation. There you had set the limits for the namespace. Thus it would try to give the whole resources to the pod, but pod creation would not be successful. Try it from your side

it is giving error - must specify limits.cpu and avoiding pod creation

**19.Create a resourceQuota which follows both object based and compute based approach.**

<in yaml file specify resource type with number and resource quantity>

<apply the yaml file to create the resource quota>

<describe the details of your namespace to see the quotas>

**20.What is LimitRange object in Kubernetes**

By default, containers can **consume any amount** of resources in **namespace**. When administrator **implements** LimitRanges in namespace, containers are **restricted** to use only **limited** amount of resources in namespace.

## 21.How to create a LimitRange object via yaml file

```
apiVersion: v1
kind: LimitRange
metadata:
  name: testLimit
  namespace: ns1
spec:
  limits:
    - default:      // This specifies resource limit for the container
      cpu: 200m
      memory: 500m
      type: Container // C should be capital
```

```
kubectl apply -f limit.yaml
```

### Now we can create the pod without giving the limits for resources

```
apiVersion: v1
kind: Pod
metadata:
  name: firstPod
spec:
  containers:
    - image: nginx
      name: firstcontainer
```

```
kubectl describe ns <ns-name>
```

```
kubectl describe pod <pod-name> // we will see the limits
```

## 22.For the above limitRanges, we don't provided the default request for cpu and memory. So again create a limitRanges and provide so.

<all same as previous limitRanges>

```
spec:
  limits:
    - default: <something>
      defaultRequest: // This specifies resource request for the container
        cpu: 100m
        memory: 250m
      type: <something>
```

```
kubectl describe ns <ns-name>
```

```
kubectl describe pod <pod-name> // we will see the limits and requests without setting up
```

## 23.Tell some scenarios during Limit range

If we provide resource parameters in pod specification ( we had provided maximum and minimum limit in limit range) then it should be within range then only it would allow the creation of pod.

sometime in pod specification, request is treated as minimum and limit is treated as maximum. Values should be within ranges as specified in LimitRange object.

in pod specification if we set only limit then value of request will become as the value of limit

in pod specification if we set only request then value of limit will not become equal to value of request.

#### **24.Create the yaml file for setting the maximum and minimum resources in LimitRanges**

```
apiVersion: v1
kind: LimitRange
metadata:
  name: testLimit
  namespace: ns1
spec:
  limits:
    - default:
        cpu: 200mi
        memory: 500mi
    defaultRequest:
        cpu: 100mi
        memory: 250mi
    min:      // pod request should not be less than this value
        cpu: 0.2
        memory: 200Mi
    max:      // pod limit should not be greater than this value
        cpu: 0.9
        memory: 700Mi
  type: Container
```

Create pod and try to put the value within range and out the range then check how it is working.

#### **25. What is the use of attribute “maxLimitRequestRatio” in the specification of LimitRange object.**

if

limit/Request <= 2 it means pod is allowed

limit/Request > 2 it means pod is not allowed

## **26.What is configMap**

Kubernetes me configMap ek **API object** hai jisse hum application ki **non-confidential configuration** ko **key-value** ke form me store karte hai. ConfigMap ke through hum **environment related configuration** ko container image se **decouple** karte hai, isse humara application **portable** ban jata hai.

## **27.Create a configMap**

### a.Using command line argument

```
kubectl create cm cm1 --from-literal=database_ip="x.x.x.x" --from-literal=user="user1"  
--from-literal=password="xxxxx"
```

### b.Using a file

```
mkdir configmap-dir // creating a directory  
vi application.properties // opening an editory to create properties file
```

```
database_ip="x.x.x.x"  
database_user="user1"  
database_password="xxxxxx"
```

```
kubectl create cm cm3 --from-file=application.properties // you can give multiple files  
kubectl get cm  
kubectl describe cm cm3
```

### c.Using a directory

```
kubectl create cm cm5 --from-file=configmap-dir/
```

## **28.Create a configMap using an environment file**

```
vi env.sh
```

```
variable1=value1  
variable2=value2  
variable3=value3  
variable4=value4
```

```
kubectl create cm cm6 --from-env-file=env.sh
```

## **29.What is difference between creating a cm (ConfigMap) by –from-env-file and –from-file.**

if we describe the above-created config map, then we can observe that content in the env file will be created as the individual key-value pair, so every variable has its own value pair. On the other hand if we create a config map using --from-file then it will create single key-value pair where the key will be file name and value will be the content of the file.

note: make sure in env file proper variable name. The invalid variable will be ignored while creating config map

### **30. How to create a yaml file for creating cm without writing the yaml file**

```
kubectl create cm cm1 --from-literal=database_ip="x.x.x.x" --from-literal=user="user1"  
--from-literal=paaword="xxxxx" --dry-run=client -o yaml > yaml1.yaml
```

```
kubectl apply yaml1.yaml
```

<execute the above command and write the yaml version of yaml>

#### yaml file for creating cm

```
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: my-configMap  
data:  
  key1: value1  
  key2: value2
```

### **31.What are the two approaches for giving values to key while creating a cm via yaml.**

-> giving key-value (single line)

<all same for yaml for creating config map>

```
data:  
  key1: value1  
  key2: value2  
  key3: value3
```

-> giving key-data (multiple lines)

```
data:  
  key1: |  
    data line 1  
    data line 2  
    data line 3  
  key2: |  
    data line 1  
    data line 2  
    data line 3
```

### **32. How to inject configMap into the container of pod**

#### giving reference of each variable inside container specification

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: <pod-name>  
spec:
```

```

containers:
- image: coolgourav147/nginx-custom
  name: firstcontainer
  env:
    - name: variablefromcm
      valueFrom:
        configMapKeyRef:
          key:<variable-name>
          name: <cm-name>
    - name: variablefromcm
      valueFrom:
        configMapKeyRef:
          key: <variable-name>
          name: <cm-name>

```

```

kubectl apply -f my-pod.yml
kubectl get pods
kubectl exec -it <pod-name> /bin/bash
# env
<it will show all the environment variables>

```

### **33. How to inject complete cm as environment variables**

<remaining pod specifications will be same>

```

spec:
  containers:
    - image: image:latest
      name: firstcontainer
      imagePullPolicy: Never // ye dekha dekha sa lagra hai
      envFrom:
        - configMapRef:
            name: <cm-name>

```

### **34. How to get configMap as a file in a pod**

For getting cm as a file we need to mount config map in a volume

```

apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: mypod
      image: redis
      volumeMounts:
        - name: foo
          mountPath: "/etc/foo"

```

```
readOnly: true
volumes:
- name: foo
  configMap:
    name: myconfigmap
```

For further notes prefer notion.

### 34. Tell some instruction about vim editor

shift + G => for going at bottom line  
gg => for going to top line  
/ <word> n => searching word in forward direction  
? <word> n => searching word in backward direction  
\* => search the current cursor word in forward direction  
# => search the current cursor word in backward direction  
:%s/<old-word>/<new-word>/g => replacing all the words  
u => for undoing changes  
ctrl + r => for redoing changes  
o => start writing from next line of cursor  
shift + o => start writing from previous line of cursor  
shift + i => start writing from starting of cursor line  
shift + a => start writing from ending of cursor line  
d => deleting the text by special selection  
:e! => undo to original file  
p => paste  
y => copy  
shift + v => select whole line  
:set nu => setting the line number  
:set nonu => removing the line number  
  
:<line-no> => go to particular line with its number  
ctrl + ww => swap to next file (firstly you have to open file in continuation)  
vim -d file1 file2 => comparing 2 files

i => entering into insertion mode  
v => entering into visual mode

### **study status**

I had created a new pod, that pod is having the environment variables as a config map when I will return, I will go inside the container and see the environment variables