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

Important links by Amit Sharma
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/

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 <cantainer id>
    // This command tells the memory utilisation and memory limit of a 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)

Kubectl expose

Kuectl expose pod <pod-name> - -port=8000 - -target-port=80 - -name <new-service-name>
// creating new service at a particular port (cluster ip concept)
Kuectl 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
            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 // lood wide with watch option
2.

Errors

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

Replication Controller

11. Deleting replication controller without deleting the existing pods
    Kubectl delete rc - -cascase=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.yml // edit yaml file
    Kubectl replace -f firstrc.yml // instead of create, we used replace

13. Various ways of changing rc with declarative approach
    Vi first.yml // create file
    Kubectl apply -f firstrc.yml // create rc
14. Vi firstrc.yml // edit file
    Kubectl apply -f firstrc.yml // 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 partucular 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 (alogn with --recursive)
19. Replication controller ke saath agar selector field na de to vo metada me present label ko as a default selector treat karta hai. Replication controller me selector (if present) and template ka meteadaya 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.

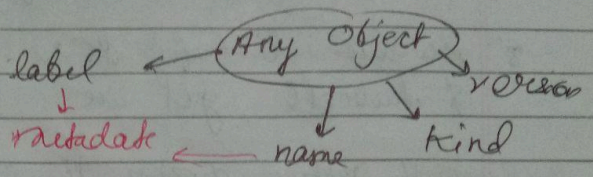
1. Write the yaml file for Rlication Controller

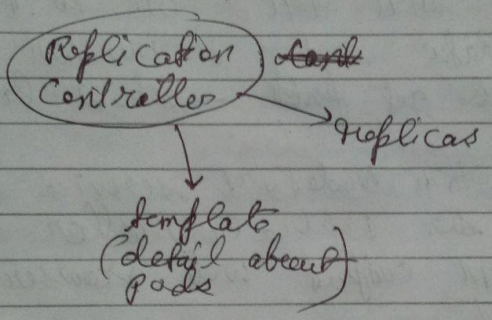
→ Use commands
$ Kubectl explain rc
$ kubectl explain rc --recursive

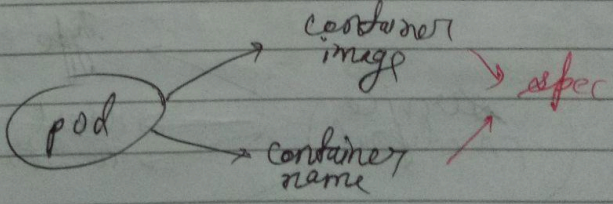
→ Inside yaml file, keep following details

→ It's a Replica controller (give version v1 & kind as Replica Controller). Ref ~~the~~ ~~it's~~ the name of Replica Controller as per your choice but label should be "app name = testingapp", This ~~Ref~~ RC should create 5 replicas of Pod as per your desired name ~~but~~ but label should be "type = app" "First container" should be the name of container & "~~coolgourav~~ 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.







// fingertips → pod में 2 चीज़ metadata (name, label) & spec (container-name and container-image) आती है 100% आएगी।

(3)

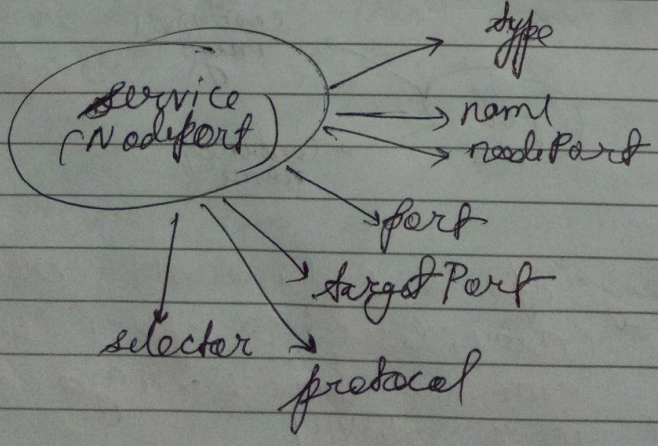
2:20 pm
21 05 2024

2. create the Replica Controller using apply command.
   $ kubectl apply -f myrc.yaml

3. See the list of RC's
   $ kubectl get rc

4. तुम notice करोगे कि सभी pods को वही label provide किया गया है, जो कि Replica Controller के metadata section में mention किया गया था।
   $ kubectl get pods --show-labels

5. अब हम Nodeport service create कर दी, जिससे कि Replica Controller द्वारा बनाए गए Pods का output web browser पर देख सकते हैं।



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

6. अगर Replica Controller के द्वारा Pod को delete कर दो तो Replica Controller new Pod create कर देगा।

For testing above situation, दो 2 terminal window open कर देते हैं, एक terminal window पर pods को watch करते हैं (watch kubectl get pods). Other terminal window पर --show-labels pods को delete करते हैं ($ kubectl delete pod <pod-name>)

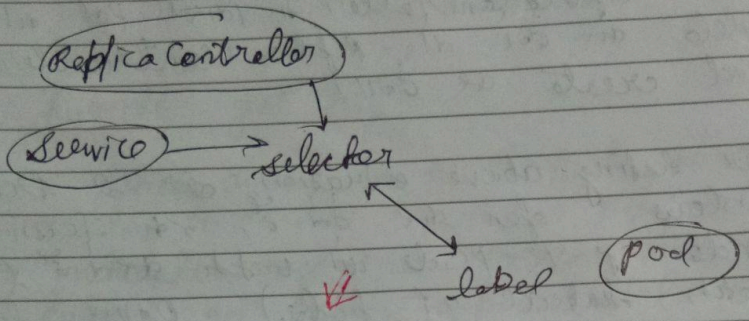
* अब Replication controller को watch करेंगे (continious watch करते रहेंगे) ($ kubectl get rc -w).

अब, जैसे ही, pods को delete करते हैं ($ kubectl delete pod <pod-name>) Then we observe कि Replication Controller के statics (CURRENT & READY) में variations आते हैं।

7. किसी भी kubernetes object की पूरी details जानने के लिए हम उस object को describe करते हैं
($ kubectl describe rc myrc.yml

(5)

($ kubectl get rc
$ kubectl describe rc <rc-name>)



if selector == label
&& pod और service दोनों belong करते हो

Then
pod perticular Replica Controller
का part बन जाएगा।

8 अगर किसी pod को (जो कि किसी Replica controller का already part है) label Remove करते हैं, तो and Replica Controller की statics को watch करते हैं

watching in another terminal window

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

This is dash sign

$ kubectl delete -f myrc.yml

//this command does not delete the yaml file, //it just delete the object created by that perticular yaml file

Deletion of Replica Controller

1. Sir Replication Controller को delete करते हैं and pods को watch करते हैं at another window

```
$ kubectl get pods -w
$ kubectl delete rc <rc-name>
$ kubectl get rc
```

2. Sir Replication controller object को delete करते हैं पर उनके existing pod delete न होय।

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

Scaling of Replica Controller

1. Sir बस इतना ही करते हैं कि Replication Controller की yaml file scale की value को कम-ज्यादा करते हैं & changes को apply करते हैं. (Declarative Object Configuration)

```
$ kubectl get rc
$ vi myrc.yml
$ kubectl apply -f myrc.yml
```

(7)

command

2. Sir Imperative ~~way~~ से Replica controller को scale करते हैं।

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

$ kubectl get rc

$ kubectl get pods

3. Sir Imperative Object Configuration के through Replication controller को scale करते हैं।

$ kubectl edit rc <rc-name>

$ vi myrc.yml

$ kubectl replace -f ~~<rc-name>~~ myrc.yml

4. Video #24 पर Sir बताते हैं कि अगर किसी लावारिस pod का label किसी service (Replication Controller/ReplicaSet/) के selector से match करता है तो वह pod उस particular service के द्वारा own किया जाएगा।

5. Sir told a way कि किसी pod को कौन सा service own करता है

$ kubectl describe pod <pod-name>

जब ~~कोई~~ कोई pod किसी Replication controller के under में आता है तो

यह field add हो जाता है
controlled By : ReplicationController/<rc-name>

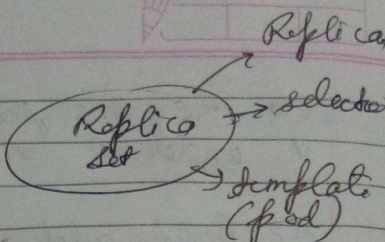
~~↳ Sir also told, अगर किसी R.C में secter mention नहीं है तो उस R.C का label ही selector का काम करता है. अगर selector and label different है तो जो pod R.C के under में बनेंगे, उनपर selecter का effect ज्यादा होगा।~~

↳ Sir also told, अगर किसी R.C में selector mentioned नहीं है तो उस R.C ~~के~~ में जो pod का template है, उसका जो भी label है, उसे as a selector treat करता है।

6 Dry Run
Yaml file का syntax check करने के लिए sir Dry Run करते हैं
$ kubectl apply -f <name.yml> --dry-run

7 Syntax Help
If you forget the syntax ~~of~~ for Particular type of object
$ kubectl explain <object-type> --recursive
~~// to take the help of less~~
~~cmd~~
// also use | less to make content scrollable.

Replica Set



1. **Special in Replica Set**
Replica Set is nothing but Replication Controller की modified form है।

Replica Controller only equality based selector को support करता था but Replica Set में Equality Based Selector के साथ-साथ set based selector की भी support करते हैं

2. **Selector field is mandatory**
अगर Replica Set की yaml file में selector field नहीं है तो यह error देगा। (Important)

3. **Problem Statement –**
Create a Replica Set, जिसमें कम से कम pod में atleast app=myapp1 label present है, तो यह Replica Set उस particular pod को acquire करे।

```
spec:
  selector:
    matchLabels:
      app: myapp1
```

4. **Problem Statement**
Create a Replica Set, जिसमें कम से कम pod में any of the label → app=myapp1 OR app=myapp2

present है तो वह Replica set उस pod को acquire करेगा

```
spec:
  selector:
    matchExpressions:
    - key: app
      operator: In
      values:
      - myapp1
      - myapp2
```

we can confirm above expression
$ kubectl apply -f myrs.yml
$ kubectl get rs -o wide
// you will see SELECTOR
app in (myapp1, myapp2)

5. Replica set को delete करने से Rep R.S से associated सारे pods delete हो जायेंगे
$ kubectl get rs
$ kubectl delete rs <rs-name>

6. Problem Statement
Creat a Replica Set, जो कि उन pods को acquire करे जिनमें ये label हो → app=myapp1 or app=myapp2. but ये label वाले हो → type = backend

(11) describe → object ~~name~~ type + name (for detail)
explain → object type (for syntax)

Just like

(app myapp1 || app myapp2) && ! type backend

```
spec:
  selector:
    matchExpressions:
    - key: app
      operator: In
      values:
      - myapp1
      - myapp2
    - key: type
      operator: NotIn
      values:
      -backend
```

→ इनका AND operation होता है ★★★

7 Sir suggestion दे रहे हैं कि Development/Production Environment में stand-alone Pode create करने के during उसमें ~~labe~~ by default label provide मत करना।

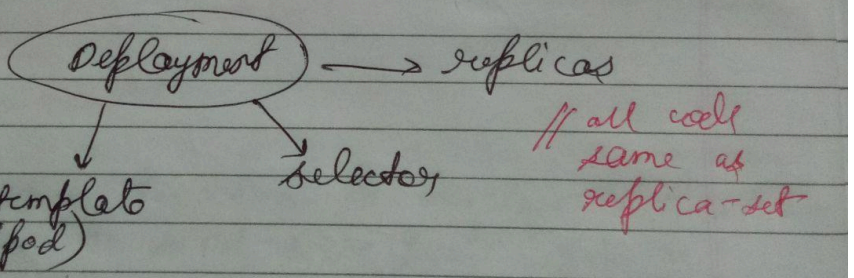
Otherwise उस environment में अगर कोई service same label का selector use कर रही होगी तो तुम्हारा stand alone Pod, किसी another service के through control हो रहा होगा, without your information. Even ~~dub~~ उस

unknown service को delete होने पर, साथ-साथ
उससे stand alone pod भी delete हो जाता
है।

Deployment Concept

1 Deployment Replica set ~~an~~ ਦਾ modified form ਹੈ ਜਿੱਥੇ ਅਸੀਂ pod ਦਾ new version deploy ਕਰਦੇ time downtime ਨੂੰ minimise ਕਰਦਾ ਹੈ and previous version ਤੇ easily Rollback provide ਕਰਦਾ ਹੈ।

// change → spec
↳ ~~spec~~ apiVersion



2. ਜਦੋਂ new deployment add ਹੁੰਦੀ ਹੈ, kubernetes automatically ~~ਤੇ~~ ~~ਦਾ~~ ਹਰ selector add ਕਰ ਦਿੰਦਾ ਹੈ → pod-template-hash = 9f66d7477

ਤਾਂਕਿ deployment (Replica set) ਨੂੰ different version ਨੂੰ ~~ਵਿਚ~~ differentiate ਕਰ ਸਕਦਾ ਹੈ।
(pod ਨੂੰ automatically new label ਵੀ attach ਹੋ ਜਾਂਦਾ ਹੈ)

services ↙

$ kubectl get svc -o wide //useless
$ kubectl get rs -o wide
//हमने deployment create किया है but
// उसे R.S. के form मे ही create
//करेंगे

$ kubectl get pods --show-labels
// pod में hash label attached होगा।

3. सर scale-up & scale down कर रहे हैं
↓ yaml file के Replicas की value को update करना,
↓ yaml file को apply करना।
↓ pods को watch करना। (~~-o wide~~ -w)

4. सर ने Rollout word से introduce कराया
जब Deployment की new version deployment करते हैं तो rollout trigger होता है। Rollout की statistic देखने के लिए following command use कर सकते हैं।
$ kubectl rollout status deployment
<deployment-name>

5 minReadySecond
जब हमको पता होता है कि हमारा pod कितनी देर में Ready होगा, then हम minReady second set-up करते हैं। जब हम

जितना time
minReadySecond ~~ज्यादा~~, set करते है तो हमारा pod ~~Read~~ active होने मे उतना time लगाएगा। (test it)

6. maxUnavailable
maxUnavailable मे जो integer value आती है, उसमे हम no. of pods mention करते है जो कि तब unavailable होते है जब हम new version के pods deployment करते है (test it)

```
spec:
  strategy:
    rollingUpdate
      maxSurge: 0
      maxUnavailable: 1
    type: RollingUpdate
```

→ // जितनी ज्यादा value होगी, उतने जल्दी new version deploy होगा।

7. maxSurge
maxSurge denote करता है कि existing Replicas के apart मे हम कितने extra Replicas create कर सकते है

For eg
Replicas = 10

Replicas of V_1 + Replicas of V_2 = 10

total replicas

Replicas of V1 + Replicas of V2 = 10

Available + shutting down

Creation of new Replicas + Available = 10

max Unavailable

fixed value

10 + k

max surge

Suppose, max Surge = k

RHS → RHS + k.

⇒ RHS ↑ (increases)

LHS → Availability ↑ (increases)

Hence

max surge set-up करने से LHS Available Replicas increase होते, so LHS system की availability increase होती।

$ kubectl get rs -o wide

8. What are the default values -

Strategy Type : RollingUpdate (Availability more)

Min. Ready Second : 0

max Unavailable : 25% of Replicas

max Surge : 25% of Replicas

9. History of our rollout

$ kubectl rollout history deployment
<deployment-name>

10 Adding data to CHANGE-CLAUSE
file = mydeploy.yaml "your message"
metadata:
annotations:
kubernetes.io/change-cause: "your message"

$ kubectl apply -f mydeploy.yml
$ kubectl rollout history deploy
<deployment-name>

11 Doing Undo
$ kubectl rollout undo deployment
<deployment-name>
$ kubectl rollout history deploy
<deployment-name>

12 Goto Particular Revision of rollout history

$ kubectl rollout history deploy <deploy-name>
$ kubectl rollout undo --to-revision=2
deployment <deploy-name>
$ kubectl rollout history deploy <deploy-name>

13 Pause/resume deployment

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

$ vi mydeploy.yml
$ kubectl apply -f mydeploy.yml
$ vi mydeploy.yml //change version of image
$ kubectl apply -f mydeploy.yml
$ kubectl rollout status deployment <deploy-name>

$ kubectl rollout pause deployment ~~mydep~~
<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 जो हम Revision को set कर सकते हैं, कैसे इसे configure कर सकते हैं।

Deployment Recreate Strategy

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

   and covers
   - apiversion
   - kind
   - name
   - label

2. Giving strategy just after the Replicas

```
spec:
  strategy:
    type: Recreate
```

$ kubectl apply -f mydeploy.yml

3. See Result of our Deployment
   $ kubectl get deploy
   $ kubectl get rs
   $ kubectl get pods

1. अब हम yaml file को edit करके container image को latest कर देते हैं।

$ kubectl get rs

5. Development के during downtime से problem नहीं, so यहाँ पर Recreate strategy को follow कर रहा है।

☐ Giving Resource Request ~~&~~ ~~Resource limit~~ (Pod)

1. अब हम pod create करते हैं, it have following
   - container name
   - container image

   & of course
   - api version
   - kind
   - name
   - label

2. अब Resource field को copy करते हैं।
   $ kubectl explain --recursive pod | less

3. अब Pod create करके describe करते हैं and show करते हैं कि pod कितना resource utilize कर रही है।

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

Sir says, कोई भी ऐसी field नहीं दिखा, जो यह ~~das~~ resource के बारे में कुछ describe करे।

Then sir उस Pod को delete कर देते हैं

```
$ kubectl delete pod <pod-name>
```

4. Problem Statement

Creat a pod जो कि जिसको लिए 200 Mi (memory) allocate/available होंगे & 500 milli- ~~cpu~~ CPU allocate/available होंगे

```
spec:
  containers:
    resources:
      requests:
        memory: 100Mi
        cpu: 500m / 0.1     // another way 0.1
```

```
$ kubectl apply -f mypod.yml
$ kubectl describe pod <pod-name> | less
```

// you will see resources in request field

5. Sir says कि pod वहीं create होगा, जहाँ पर sufficient resources present होंगे otherwise pod create नहीं होगा and status - pending show करेगा।

$ kubectl get pods

Sir says - यह अंजीब से pod को ~~create~~ finally create करने के लिए हम kubernetes cluster पर एक ऐसा pod attach कर सकते हैं जो कि इस pod की requirements को fulfill करे।

6. Sir says multi-container pod के case में kubernetes scheduler pod वहीं पर create करेगा जिस node ~~पर~~ ~~पर resources~~ पर pod के containers के resources का sum available होगा।

Eg

A multi container pod (Pod1)

~~pod~~ Container1 needs 200Mi memory

container2 needs 500 Mi memory

Scheduler pod1 उसी node में create करेगा जिस node में 700 mi memory available / free रहेगी

Giving Resource Limit (Pod)

1. Problem Statement,
create a pod ~~with for~~ जो एक container 200 Mi से ज्यादा memory की utilization न करे। Also keep in mind, उस particular container के लिए 1 Mi memory allocate/ available रहे।

```
spec:
  containers:
    resources:
      requests:
        memory: 1 Mi
      limit:
        memory: 200 Mi
```

```
$ kubectl apply -f myfod.yaml
$ kubectl get pods
$ kubectl describe pod <pod-name>
```

2. Agar external worker node से हो तो docker commands के through resources check करने होते हैं।

$ docker ps
$ docker container list
$ docker container stats <docker-image>
$ free -m //Linux 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 clusturs ko group karne ke liye hum ek virtual cluster envirment create karte hai. Iss envoromtment 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

Crating 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 commond 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 lob 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 mysever 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

<u>yaml file for creating a pod</u>

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

<u>yaml file for creting a service</u>

```
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 kubernete

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 –namspace <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>

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 tha 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.

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 Kubernete
By default, containers can consume any amount of resources in namespace. When administrator implements LimitRanges in namespace, containers are restricted to use only limited amound 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: 80mi
    memory: 250mi
  max:       // pod limit should not be greter than this value
    cpu: 700mi
    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
iimit/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=paaword="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: image:latest
    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 forther notes prefer notion.