PAGE NO. :
DATE : 5/Jan/21

SESSION: 66 KUBERNETES

- Containers: CRIO & Docker.
- Kubernetes is a container management program. which replaces the human errors. It also launches new containers as per the usage. aka Autoscaling
- Multinode cluster

PAGE NO.:
DATE: 7/Jan/21

SESSION: 67 K8S

- To deploy an app we need an infrastructure and this infra needs to be managed.
  ∴ For management we use K8s.
  (also provides the infra)
- K8s is a program that manages the containers launched by Docker or Cri-O.

#hub.docker.com
search: nimal13
↳ nimal13/apache-webserver-php
(Download) or (pull)

#docker run -it --name nimalos1
nimal13/apache-webserver-php
(will work if it is not pre-installed first it will download it & then start launch it)

curl http://<URL>>

- K8s monitors the container and as soon as some container dies / offline it will automatically launches another container. ∴ known as container management tool.

#minikube (search web)
(for win) (copy the path)
↳ This is the program that helps us for K8s.

command prompt: (win)
cd C:\P_1\K_1\min_.

PAGE NO. :
DATE : / /

~~Kubernetes~~

#minikube.exe start --driver=virtualbox

↳ will download K8s VM from internet & set up on the top of VM. with all the config.

.... --kubernetes-version=v1.20.0

• After downloading it will launch an OS named minikube with all the configurations pre configured.

win

download URL.

k8s: w/docs/tasks/tools/install-kubectl/

on the same location

C:\Program Files\K8s\minikube\

↳ It will install kubectl

minikube status

die

kubectl.exe get pods

kubectl.exe run mywebl

--image = <<vimalB->>

kubectl.exe get pods

kubectl.exe delete pod mywebl

↳ It will automatically launch pod.

PAGE NO. :
DATE : / /

kubectl.exe create deployment mynwebl
--image=<<vimal13>>
↳ This time the pod will be under the supervision of deployment controller.

kubectl.exe get pods
kubectl.exe delete pod mynwebl-55-----2t
kubectl.exe get pods
↳ Now K8s will automatically create & launch a new pod for us bcz (of create deployment cmd)
DC (deployment controller).
kubectl.exe describe pods
↳ gives info abt the pods

• This K8s server is configured by minikube & it also provides a web UI.
minikube dashboard
↳ Launches web UI.
kubectl.exe get deployments
kubectl.exe expose deployments mynwebl --port 80 --type=NodePort
↳ Similar to PATing in docker.

minikube service mynwebl --url
↳ gives URL for mynwebl

{ Login: docker
passwd: tcuser } minikube OS on VM

PAGE NO. :
DATE : / /

#kubectl scale deployment mywebs --replicas=4

↳ create 4 LB services.

PAGE NO. :
DATE : 8/Jan/21

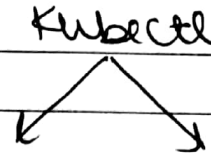
SESSION : 68 K8S

- Server Program is K8S or Master.
- Program through which the user interface is API or KubeAPI.

win

```
#minikube.exe status
#minikube.exe start
```

- EKS is the K8S cloud services.




```
#kubectl.exe get pods
#kubectl.exe get deployments
#kubectl.exe delete all --all
```
↳ delete all the pods, deployments

minikube

```
$ docker info
$ docker ps
```

win

```
#kubectl.exe run myos1 --image=<<vimal1357
#kubectl.exe get pods
```

```
$ docker ps | grep vimal13
```
↳ Launched

PAGE NO. :
DATE : / /

#kubectl explain pods mypod1

#cd Desktop

#mkdir CKA_ws

#notepad *pod.yml

{kind=pod
name=mypod1
apiversion: v1
kind: pod
name: mypod1
spec:
containers:
- name: myc1
image: redis
- name: myc2
image: mysql

apiversion: v1
kind: Pod
metadata:
name: "mypod1"
spec:
containers:
-
- name: "myc1"
image: "redis"

{{ saveAs: "pod.yml" }}

PAGE NO. :
DATE : / /

#

- Environment variables
  - Environment variables....
    - Path
      - Edit
        - New
          - C:\Program Files\Kubernetes\Minikube
            - OK
              - Apply

- Now Open a new ~~te~~ terminal.

#kubectl.exe get pods.
#kubectl delete pods myos1
- Only deletes myos1.

#kubectl create -f pod.yml
- Runs a yaml file

#kubectl get pods
- myos1

#kubectl description pods myos1
- Containers: ~~myos1~~ myc1

PAGE NO.
DATE 12/Jan/21

SESSION: 71 K8S

#minikube start
docker run -it --name os3 --image <<nimal13>>
↳ NO outside connectivity
docker run -it -p 1234:80 --name os4
--image <<nimal13>>
↳ outside connectivity to port 80.
(Node_ip: 1234)

- In K8S we will provide the ip of minikube to the client.

#kubectl run nebos1 --image=<<nimal13>>
#kubectl expose pod nebos1 --port=80
--type=NodePort
↳ expose the pod to outside
#kubectl get services
↳ will give the port no. to ~~kube~~ which we need to connect. for eg: 30180

#URL: 192.168.99.100:30180
minikube ip
↳ It'll work.

- Replication controller: It is a type RC. When a pod goes down RC will auto. automatically launch it.
#kubectl get rc

- When we restart a container it may change its ip so we tag / label them.

PAGE NO. :
DATE : / /

kubectl create -f pod.yml
kubectl describe pod mypod1
↳ No label attached.
kubectl delete pod mypod1

notepad pod.yml
metadata:
name: "mypod1"
labels:
app: web

kubectl create -f pod.yml
↳ This time a label will be attached
{ app = web }

kubectl get pods -L app
↳ display all pods with label app.

notepad rc.yml
apiVersion: v1
kind: ReplicationController
metadata:
name: "myrc1"
spec:
selector:
app: web
template:
metadata:
name: "mypod1"
labels:
app: web
spec:
containers:

PAGE NO. :
DATE : / /

- name: "myc1"
  image: "<<image13>>"

kubectl get rc
↳ No resources found
kubectl create -f rc.yml
kubectl get rc
↳ myrc1 - created
↳ It won't create any pod.
But if a pod goes down. it will relaunch it.
kubectl delete pod mypod1
kubectl get pods
↳ automatically mypod1 is launched
kubectl describe ~~pod~~ rc ~~mypod1~~ myrc1

kubectl expose rc myrc1 --port=80
--type=NodePort
kubectl get services
↳ ~~80:31896/TCP~~ 80:31896/TCP

URL: 192.168.99.100:31896
↳ Connected to a pod managed by rc.

- Desire state = 3
↳ It will always keep 3 pods running if any pod goes down. It will automatically launch them.

PAGE NO. :
DATE : / /

kubectl get rc
↳ Desired = 1

rc.yml
spec:
replicas: 5
selector:
—
—

kubectl create -f rc.yml
↳ Error: rc exists.

kubectl apply -f rc.yml

kubectl get rc
kubectl get pods -l app

- Label is must in replication controller.

PAGE NO.:
DATE: 13/Jan/21

SESSION: 72 K8S

minikube status

- Types (services)
  - ClusterIP
  - NodePort
  - External (Load Balancer)

- Reverse Proxy: When the client sends request to frontend server and frontend in turn transfers the request to the backend servers for the information.

- Frontend port is also known as service port or port. (Port = 1234)
- Backend port is also known as Target port. (Port = 80)

kubectl get svc
  (service)

notepad svc.yml

```
apiVersion: v1
kind: Service
spec:
  metadata:
    name: "mylb"
  selector:
    app: web
  ports:
    - targetPort: 80
      port: 8080
```

kubectl apply -f svc.yml

PAGE NO. :
DATE : / /

- It will create the service for all the pods having the label web. or selector.

kubectl apply -f pod.yml

kubectl get pods -l app
kubectl describe svc mylb1
↳ The pod will automatically will be discovered by mylb1.

minikube # curl http://<ip>:8080

minikube
$ curl 10.103.10.133:8080

svc.yml
spec:
  type: ClusterIP (By default)
  -type: NodePort

ports:
  - targetPort: 80
    port: 8080
    nodePort: 30000

#kubectl apply -f svc.yml
kubectl get svc
↳ Type: NodePort
PORT: 8080 : 30000
↑ svc port (Port) ↑ Port of node

PAGE NO. :
DATE : / /

kubectl describe svc mylb

curl http://192.168.99.100:30000
↳ Connectivity

- ClusterIP : No Outside Connectivity
  Client is in the cluster
- NodePort : Outside Connectivity

kubectl describe svc mylb
↳ Annotations : selector :
↳ Provides extra info.

PAGE NO. :
DATE : 19/Jan/21

SESSION : 75 K8S

minikube start
kubectl get pods
kubectl run mydb --image=mysql:5.7
↳ STATUS : ERROR
↳ pod won't run
k describe pod mydb
k logs mydb
• The error is bec of password.

k run myos1 --image=vimal13

k exec -it myos1 bash
↳ It will attach the pod to interactive terminal.

myos1
ifconfig
↳ ip of pod myos1
vi /root/.bashrc
↳ Permanent the env variable
a = 5
# echo $a

kubectl run myos2 --image=vimal13
--env=a=10
↳ It will also create an env variable.

k run mydb --image=mysql:5.7
--env=MYSQL_ROOT_PASSWORD=redhat
--env=MYSQL_DATABASE=wpdb
.....

PAGE NO. :
DATE : / /

: 5.1.1-php7.3-apache

kubectl run mywp --image=wordpress:~~dbi~~

k— expose pod mywp --type=NodePort --port=80

k— get svc

// Webpage url: ip:port

PAGE NO.:
DATE 20/Jan/21

SESSION: 76 K8S

#kubectl expose svc
↳ myjmpl is exp. NodePort.

kubectl describe mydb
↳ Env; root passwd is stored in clear text.

• K8S ~~security~~ secret
↳ It doesn't encrypts ~~do~~ or locks our data.

#kubctl get secrets
kubectl delete all --all
notepad pod.yml

```
metadata:
    name: "my-db"
    labels:
        app: mydb
spec:
    containers:
    - name: "mydb"
      image: "mysql:5.7"
```

kubectl create -f pod.yml
↳ crashes bec of absence of env. variable.

pod.yml

```
image: "mysql:5.7"
env:
- name: MYSQL_ROOT_PASSWORD
  value: redhat
```

PAGE NO. :
DATE : / /

```
- name: MYSQL_USER
  value: vimal
- name: MYSQL_PASS
  value: redhat
- name: MYSQL_DATABASE
  value: wpdb
```

kubectl apply -f pod.yml
↳ failed!!
↳ we can't edit in running a pod.
∴ we need other component of k8s
↳ Deployment.

kubectl delete all --all
kubectl create -f pod.yml
↳ success!!
kubectl describe pod mydb
↳ env: (pass is visible)
∴ we need the concept of secret.

notepad s.yml

```
apiVersion: v1
kind: secret
metadata:
    name: "mysecret"
data:
    p: redhat
```

kubeal create -f s.yml
↳ fails!!

- algo : base64
- encode is dif. than encryption.
- encoded data can be decoded
  ↳ different format.
- Once plain text is encrypted then it cannot be decrypted. or lock it without the password.

web: base64 encode (search on google)
  ↳ first encode the password in base64

s.yml
data:
  p: <encoded passwd>

kubectl apply -f s.yml
kubectl get secrets
kubectl describe secrets mysecret
  ↳ It won't reveal the data.

pod.yml
- name: MYSQL_ROOT_PASSWORD
  valueFrom:
    secretKeyRef:
      name: mysecret
      key: p

valueFrom:
  secretKeyRef:
    name: mysecret
    key: p

indent →

PAGE NO. :
DATE : / /

kubectl dump pod mydb
kubectl create -f pod.yml
kubectl describe pods mydb
↳ Now the password is not visible.
kubectl get secrets mysecret -o yaml

kubectl create secret -h
↳ 3 types of secrets we can create
generic redhat
kubectl create secret mys --from-literal=pl=A

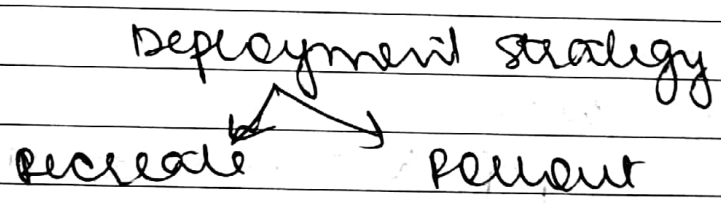
PAGE NO.:
DATE: 22/Jan/2021

SESSION: 77 K8S

- Replica set (RS) is similar to Replication Controller (RC) but with more new features.

kind: Deployment

- We can create or destroy the replicas as per our need known as scale in and scale out.; feature of RC & RS.
- Recreate is a strategy used for 100% up-time. Delete old pod and create newer one.
- Rollout when the pod with older version is terminated and the pod with newer version is started. It is also known as Ramped.



- Deployment is a feature of K8S for automatic rolling strategy.
- Blue/Green deployment is also another strategy.
- We do not override the version of containers. We create a new version and they only recreate when they see a newer version. If we override it they won't update it.

PAGE NO. :
DATE : / /

- RC & RS is the only one create replicas. Deployment manages RC & RS It deploys with a strategy. By default, strategy is Rolling upgrade

$kubectl create deployment -h
k- create deployment myd --image=vimal13
- It will also create a RS as well as deployment and also a Pod.
- The pod's failing part is managed by RS

k- describe deployment myd
  ↳ strategy Type: RollingUpdate
k- scale deployment myd --replicas=3
  ↳ Create 3 replica (scale out)
k- get deployment
k- delete deployment myd

VM Host
vim Dockerfile
FROM vimal13....
COPY index.php /var/www/html/index.html
docker build -t vimal13/apache-webserver-php:v1 .
docker login (if you have account)
docker push <img name>
  ↳ uploads the img to docker hub.

K8S
kubectl create deployment myd --image=vimal13:v1

#kubectl get pods
↳ new pod is launched.
#k — scale deploy myd --replicas=5
#k — get svc
↳ for external connectivity
#k — expose deployment myd --port=80
--type=NodePort
#k — get svc
↳ 80:31574/TCP
5 pods are running under this svc
curl 192.168.99.100:31574
↳ Node IP or #minikube ip
or minikube ssh
↳ It WORKS!!
*. New version is launched
kubectl create deployment myd
--image=vimal13:lwv2.
curl 192.168.99.110:31574
↳ lwv2 is shown
OR
kubectl set image deployment myd
apache-websserver-php = vimal13/apache-
webserver-php:lwv2
↳ It will quickly deploy rolling
upgrade strategy.

kubectl rollout status deployment myd
kubectl rollout history deployment myd
kubectl describe deployment myd

PAGE NO. :
DATE : / /

#kubectl rollout undo deployment myd
↳ It will go back to the older version.

PAGE NO.:
DATE 26/jan/2024

SESSION: 80 K8S

```
# notepad pod.yml
spec:
  containers:
    - name: "myrs"
      image: nimal13
```

```
labels:
  app: frontend
  team: team1
  region: IN
  env: prod
```

- labels are the tags given to the pods. selectors are used for the management of pods using labels. RC/RS uses selectors
- selectors always check for labels.

Selectors

- Equality-Based (matchLabels)
- set-Based (matchExpression)

```
# kubectl apply -f pod.yml
# k ---- get pods --show-labels   # with labels
# k ---- get pods -l "env=prod"
                  (or --selector
```

~~# k get pods -l "region=IN"~~

set-based

```
# k- get pods -l "region in (IN)"
# k- get pods -l "region in (IN,US)"
# k- get pods -l 'region in (IN,US),
                  team in (team1)
                  app in (backend)"
```

Or "region notin (IN, US)"

- The dif bet RC & RS is that RS uses set-based selection as well as equality based selection. This is the only difference.

~~pod.yml~~
apiVersion:
v1 → main components (Pod, RC, etc)
apps/v1 → newer comp (RS
~~Pod~~

pod.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
name: "mymembers"
~~labels~~:
spec:
replicas: 3
selector:
~~matchLabels~~
matchLabels:
env: prod

For pod
template:
metadata:
name:
labels:

spec ~~spec~~:
Cont: ~~containers~~:

PAGE NO. :
DATE : / /

kubectl create all --all

kubectl get rs
kubectl apply -f pod.yml
kubectl get rs
kubectl get pods . --show-labels

pod.yml

selector:
matchExpression:
- {key: team, operator: In, value: [team]}
- {key: tier, operator: In, value: [frontend, backend]}

kubectl describe rs myrepos

- If a pod is deleted . RS will automatically launch another

PAGE NO. :
DATE 27 Jan 21

SESSION: 81 K8S

storage

PVC (Persistent V. claim) — PV (P— .V—) — SC (storage class)

#kubectl get pvc
k—— get pv
k—— get sc

#docker run -it --name os11 nimal13
#cd /var/www/html

- The storage that is provided by docker is temporary or ~~example~~ ephemeral.
- ~~k8s do not provide persistent storage,~~
- Using plugins we can provide permanent storage to the containers. or persistent volume (PV)

mkdir /data1 ⇒ # inside node/ docker host.

docker run -it --name os12 ~~nimal13~~
-v /data1:/var/www/html nimal13
Volume1 mounts /data1 to /var/www/html
∴ data is persistent

storage

SAN (Raw storage) (EBS in AWS)

NAS (Preformatted storage) (S3 in AWS)

PAGE NO. :
DATE : / /

- Docker only provides persistent storage from host. We can't take storage from AWS, ebs, S3, etc. Therefore, we need to switch to container management tools for NAS, SAN, storages.
- K8S has various plugins for storage.

web: k8s storage providers/plugins

k run pod p1 --image=vimal13
k exec -it p1 bash
cd /
ls
cat > hi.txt
↳ shows k8s has storage but it is ~~exposable~~ ephemeral.
- If we delete the pod, all the data is lost.

- Per PVC we need to create one PV.

PAGE NO.:
DATE: 28 Jan 21

SESSION: 82 K8S

#Copy

#minikube start
notepad pv-deploy.yml
apiVersion: apps/v1
kind: Deployment
metadata:
spec: spec
template:
metadata:
spec:

#kubectl apply -f pv-deploy.yml
↳ create a deployment (prev created)
kubectl expose deploy myd --port=80
--type=NodePort
↳ for outside connectivity

cmd curl 192.168.99.100:30518
URL : http://

k-- exec -it myd-6 -- v6 -- bash
cd /var/www/html
cat > index.php
hello ww!!

PAGE NO. :
DATE : / /

notepad pvc.yml
apiVersion: v1
kind: ~~PVC~~ PersistentVolumeClaim
metadata:
~~storage: 10Gi~~
name: "mypvc"
spec:
resources:
requests:
storage: 10Gi
accessModes:
- ReadWriteOnce
storageClassName: "" #
↳ For static PV

k— apply -f pvc.yml
k— get pvc
k— describe pvc mypvc
↳ status Pending
↳ we don't have a block or pv.
notepad pv.yml
apiVersion: v1
kind: PersistentVolume
metadata:
name: "mypv"
spec:
capacity:
storage: 10Gi
accessModes:
- ReadWriteOnce
storageClassName: "" #

PAGE NO.:
DATE: / /

minikube
$ sudo mkdir /mydata

kubectl cmd
pv.yml
accessmode:
—
hostPath:
path: "/mydata"

#k- apply -f pv.yml
#k get pv
#k get pvc
↳ created & Bounded to mypv.

pv-deploy.yml
template:
spec:
volumes: → -name: "mypod-pvc"
persistentVolumeClaim:
claimName: "mypvc"
containers:
- =
volumeMounts:
- mountPath: /var/www/html
name: mypodpvc

k- apply -f pv-deploy.yml
k- describe pods

#k — exec -it mypd — — → bash
cat > /var/www/html/index.html
first page.

L. It will be backuped in the host dir (/mydata)

kubectl edit pv mypv
↳ edit dynamically or onfly.

reclaim
PVC Policy: recycle

- As soon as we save the file they automatically re-run the code and changes are visible.

PAGE NO.:
DATE 29/07/21

SESSION: 83 K8s

- Storage class: controller from K8s helps create dynamic PV.

minikube dashboard start
  - Enabled addons: default-storage class

k- get sc
pvc.yml
  spec:
    storageClassName: "standard"
k- apply -f pvc.yml
  - New PV is also created dynamically

- hostPath is a plugin that will take the storage from the path/dir and provides to the pod.
- If we want to use storage from NAS (Network attached storage) we use different plugin.

pv-nfs.yml
  spec:
    nfs:
      path:

OS1 for NFS

mkdir /mydata
vim /etc/exports
• /mydata 192.168.99.100 (rw, no_root_squash)
↳ minikube ip → NOSPACE
or
/mydata * (rw, no_root_squash)
systemctl start nfs-server
exportfs -v

• mini
$ ping 192.168.0.129
↳ NFS OS ip
$ sudo mount 192.168.0.129:/mydata /mnt/data
↳ for testing
$ sudo mkdir /mnt/data
$ sudo m—
↳ Not permitted!!!

• NFS has 3 versions (3, 4, 4.2). If it don't match from both the sides. It fails.

NFS OS
cat /var/log/messages
↳ port 60487
↳ insecure port
• ports < 1024 secure ports
ports > 1024 insecure

vim /etc/exports
(rw, no_root_squash, insecure)
systemctl restart nfs-server

PAGE NO.:
DATE: / /

misc
$ sudo mount ----
$ df -h
↳ It is mounted.
$ sudo umount /mnt/data } only for testing.
$ df -h

pgs
pvc.yml
spec:
storageClassName: "" **

* kube- delete pvc --all
↳ PV is also deleted

* k- apply -f pvc.yml
↳ Pending
↳ Bec PV is not available

* pv.yml (nfs)
spec:
<dll> hostPath
nfs: (NFS OS ip)
server: "192.168.0.129"
path: "/mydata"

* k- apply -f pv-nfs.yml
* k- get pvc
↳ Bounded (success!!)

~~pvc deploy.yml~~

* k- apply -f pv-deploy.yml
↳ Pod created with NFS storage

PAGE NO.:
DATE: / /

k- describe deploy myd
k- exec -it myd- -- --bash
df -h
↳ 192.168.0.129:/mydata
/var/www/html

cat > index.html
nfs test data
* This data will be backuped in NFS storage.

NFS
cd /mydata
cat index.html

PVS
k- delete pod --all
↳ Data won't be lost.

k- expose deploy myd --port=80
--type=NodePort

k- get svc
cmd: curl 192.168.99.100:32357
↳ minikube

k- scale deploy myd --replicas=3
k- get deploy

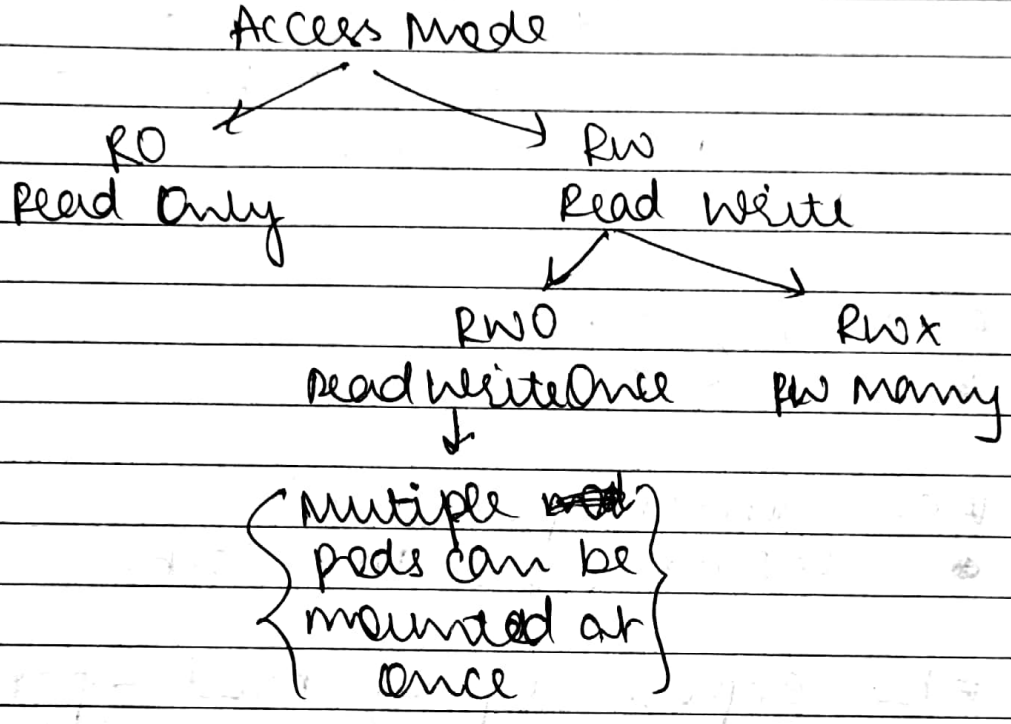
cmd: curl 192.168.99.100:32357
↳ All the pods will get same data.

k- exec -it myd- -- --bash
cat > /var/www/html/index.html
hi test

and :

↳ If we change the data. It will be updated in all the pods. And will show latest data.

↳ All the pods in deploy are mounted on NFS dir.



PAGE NO.:
DATE: 2/Feb/21

SESSION: 85 K8S

- Container Run Time Interface (CRI)
- It is like a box where we abstract the containers.
- Kubelet is a program will implement CRI and will launch container/pod.
- If we have one worker Node then we have SPOF (single point of failure)
  ∴ Use multiple worker nodes.

- Kube-Controller Manager (kube cm)
  It if a worker node goes down it relaunches them. It is similar to RC/RS but accountable for nodes.
- Kube-Master = Kube-API server + Kube-Controller Manager + Kube-Scheduler
- K8S cluster = Kube-Master + Worker Nodes

- Kubelet is an Agent Program.

Both
Master nodes
#su -root
yum install docker -y
systemctl enable docker --now

- OS to OS docker name changes both are -ce version.
- Kubeadm is an official cmd.

PAGE NO. :
DATE : / /

yum install kubeadm
↳ NO package avail!

~~daged~~
- yum list in drive.
- configure yum for kubeadm

yum repolist
yum install kubeadm -y
↳ ~~Not found~~ fail!
yum install kubeadm --disableexcludes=kubernetes
↳ (drive)
kubeadm kubelet kubectl
systemctl enable kubelet --now
——— status ———
↳ Not running; Trying to activate

kubeadm config images pull
docker images
↳ All the necessary imgs are downloaded

kubeadm init
↳ setup the master
↳ FAILED!!

kubeadm init --pod-network-cidr=10.240.0.0/16

- Docker is using cgroupfs driver.
But in K8S systemd driver is supported.

cd /etc/docker
vi daemon.json

PAGE NO. :
DATE : / /

{
"exec-opts": ["native.cgroupdriver=systemd"]
}

systemctl restart docker.
docker info | grep Driver
↳ cgroup: systemd

yum install iproute-tc -y
→ vim /etc/sysctl.d/k8s.conf
→ drop caches
kubeadm init --pod-network-cidr=10.240.0.0/16
--ignore-preflight-errors=NumCPU
--ignore-preflight-errors=Mem

↳ SUCCESS!!!
↳ THIS SYSTEM IS CONFIGURED AS MASTER

kubectl get pods
↳ Fail
↳ User cmd; they should know master ip & port number of API.

* copy & paste 3 cmd. (in drive)

kubectl get pods
↳ worked.

systemctl status kubectl
↳ started

PAGE NO. :
DATE : / /

docker ps
↳ many containers are launched.

kubectl get nodes
↳ One master node configured
↳ Status: NotReady

k-node1
sudo su - root
yum install docker -y
systemctl enable docker --now
yum install kubelet
↳ Conf yum. similar to master.
• We have to install an agent (kubectl) to connect node with master (+ kubeadm).

-y
yum install kubelet kubeadm kubectl
--disable — — .

systemctl enable kubelet --now
Syst— status kubelet

kubeadm join
↳ fail.

master
kubeadm token list
↳ lists token command
kubeadm token create --print-join-
↳ give the token.
↳ copy this entire and to worker.

PAGE NO. :
DATE : / /

node1

<paste the token>

same setup for docker driver.

~~Run the token~~

yum install iproute-tc -y

~~systemctl~~

sysctl -a | grep bridge-bridge-nf-call
↳ this facility is ~~not~~ disabled.
We need to enable it.

vim /etc/sysctl.d/k8s.conf
net.bridge.b— —
——— ——— (k8s doc)
— ———

TOken

master

kubectl get nodes
↳ master node.

~~kubectl~~ kubectl create deploy myd
--image=httpd

kubectl get deploy

——— pods
↳ state: Pending

kubectl describe pods pod id
↳ Events:
↳ Failed Scheduling

kubectl apply -f <URI flannel>

Pod runs successfully!!

PAGE NO. :
DATE : / /

kubectl expose deploy myd --port=80
--type=NodePort.

kubectl get svc.

kubectl get pods --all-namespace

SESSION: 86 K8S

PAGE NO :
DATE : 3Feb24

master
kubectl get pods
——— nodes
kubectl get pods -o wide
↳ tells the nodes of pod.
kubectl delete all --all

N2 # yum install docker -y
SET UP KUBEADM LAB.
sysctl --system
↳ restarts the service (final step).

- Namespace: In companies, they usually create one cluster and various teams using it simultaneously. So the problem of security arises. So K8S offers a feature ~~in~~ ~~toolkits~~ that creates a seperate ws/room for each team/user. These teams/users are known as tenants ∴ Feature is known as Multi-Tenancy. And the room/ws is known as Namespace.

master
kubectl get namespace
↳ default is one ~~name~~ namespace

kubectl create namespace lwns

kubectl get pods --namespace=lwns
↳ lists pods in lwns. ↳ or -n lwns

PAGE NO. :
DATE : / /

t_ create deployment myd1 --image=httpd -n lwns

kubectl delete namespace lwns
↳ If we delete a ns all its data will also be deleted.

kubectl get pods -n kube-system
kubectl create deployment myd --image=vimal13

- SDN is a software by which we can create a virtual network (switches, bridges)
- bridge utils is a soft. we can create software bridges, switches.

brctl show
yum install bridge-utils

docker network ls
↳ shows the virtual created by docker

brctl show

docker network inspect bridge
↳ switch
↳ gives total os connected to it

ifconfig -a
↳ worker node do not have any virtual network card.

PAGE NO. :
DATE : / /

brctl show
↳ we have 2 switches.

- K8S always create a switch with the help of docker, cni0.

brctl show
↳ cni0

k- create deploy myd --im
↳ creates a containers
k- get pod -o wide
↳ They created a pod and connected it to the switch and created an interface.
docker ps
↳ 1 container was launched.
- 1 virtual NIC is also created
↳ ifconfig -a

- The network is isolated. means if anyone from other laptop can't connect to them.

- If want to connect to containers in different network. we need to create two containers (C1, C4) then C1 will connect to C11 through switch and C11 will act as a router and send the packet to physical nic and then

it will send the packet to outer network. Similarly, we can create C2 & C12. where C12 will act as router and C2 is our destination container.
This concept is known as Overlay.
To implement this concept we have to use tunneling
- GRE tunneling
- VXLAN
- Virtual Extended LAN

- Flannel is the one who launches the containers C1 & C2 which acts as routers. They also have a DHCP and provides ip to all the containers.

- Kube-proxy: All the nodes will work as a proxy.

PAGE NO.:
DATE: 5 Feb 21

SESSION: 87 K8S

- In K8S we have SPOF w.r.t. master. That's why we also have HA master.
- openshift is a tool used to manage K8S.

cmd

kubectl get pods --server 54.86.177.222:6443
(Pub-master)
↳ it stuck (error from server)
↳ We have to provide user & passwd.

- 6443 is the by-default port for kube API-server.

master

cat /etc/kubernetes/admin.conf
users:
- name: kubernetes-admin
client-key-data: ——— (14.1)
↳ Password
↳ Password is big therefore we provide the file.
- Save as "admin.yml"

cmd

cd Desktop/CKA
kubectl get pods --server 54.86.177.222:6443 --kubeconfig admin.yml
↳ Failed

- We need to update the public ip of master in the file
↳ Server: https://54.86.177.222:6443

kubectl get pods --kubeconfig admin.yml

PAGE NO. :
DATE : / /

- ~~#kubectl admin.conf~~

↳ won't work for any ~~user~~ id other than the registered ones.

- If we create a cluster using minikube ~~it~~ the admin.conf file is already created in it. Therefore, ~~all~~ this and never gave issues.

master

kubectl get pods -n kube-system

↳ flannel is running

- In networking, if we create an isolated LAN with one switch only known as VLAN. This one switch will act as ~~a~~ two-three (depends) different switches.

- ~~Flannel~~ is a way through which we can create a illusion that two different network are same through the concept of tunneling and the ~~as~~ process is known as ip masquerading. The ~~tunnel~~ is known as ~~used~~ overlay network.

- Flannel is like a vpc and we give subnets to all the worker node.

node1

cat /var/run/flannel/subnet.env
↳ FLANNEL_NETWORK = 10.240.0.0/16
FLANNEL_SUBNET = 10.240.1.1/24
FLANNEL_IPMASQ = true
↳ IP masquerading is true,
Similar to VPC & subnet concept.

node2

cat /var/run/ ——— /subnet.env
↳ FLANNEL_SUBNET = 10.240.2.1/24

kubectl delete deploy --all
kubectl create deploy
kubectl get pods -o wide
↳ IP.
↳ One is launched in node2
kubectl scale deploy nyd --replicas=3
kubectl get pods -o wide
↳ 2 launched in n-1 and 1 in n-2
But bec. of flannel they think they are launched in same network.

master

kubectl get pods -n kube-system
↳ ~~etcd-ip~~ etcd-ip
↳ inside this pod there is one etcd named db which stores the info of all the secrets, networks and major management details

PAGE NO.:
DATE: 9Feb21

SESSION: 90 K8S

- Analogy
  - cluster == Hotel
    - namespace == rooms

- In K8S, IAM is known as Role
- Eg: ~~Sonu~~ U1(user) → deploy, pod svc, list, create
  - all power role, admin role.

- Inside these Roles we define the Privileges/Permissions.
- RBAC: Role Based Access Controlling

Masters

kubectl get nodes
kubectl cluster-info
  - gives info of the cluster.
    As well as port for api-servers

- Control-plane = apt server + K.M. + Scheduler Master.

kubectl create -h

- Authentication = ~~x~~ ssh, certificate Based ✓ Auth (in K8s)
- Key based auth is we have to use. Or ~~ssl~~ cert-based. They are the auth in between the programs.

- We create a ~~cert~~ cert (key) in our local system and send it to the master. They review the cert and stamp it and send back to us as verified. This key is ~~[illegible]~~ a private key.
- Certificate is known as CSR. After the master has reviewed it it known as CRT.

cd /etc/kubernetes/pki/
↳ ca.crt & ca.key
(critical file; we can sign the cert ~~[illegible]~~ with this file)

IAM
↳ Identity → Certificate Based
↳ ~~[illegible]~~ Access → Roles

W.S
① create private key
② CSR (certificate Signing Request)
③ send to master to sign it (CRT)
④ master will send back CRT.

OS-PART
~~[illegible]~~ mkdir /kubews
cd . -out nimal.key
openssl genrsa 1024
(algo-rsa) (size 1024 bytes)
↳ -out (same)
↳ name : nimal.key

PAGE NO. :
DATE : / /

openssl req -new -key vimal.key
-out vimal.csr

↳ creates a CSR (request)
↳ : IN
↳ : Raj
↳ : Jaipur
↳ : LW
↳ : Tech
↳ : Vimal
↳ : vimal@lw.com
↳ : enter
↳ : enter

* This file is encoded.

~~scp vimal.csr~~
copy vimal.csr to the master.

* (using putty copy & paste the file contents > . /pki.

master. # openssl x509 -req -in vimal.csr
(standard) -CA ca.crt
-CAkey ca.key
-CAcreateserial -out vimal.crt

↳ ca.srl : No such file or dir.
(-CA createserial (one time cmd))
*. vimal.crt is created
vim vimal.crt
↳ encoded file
<copy & paste vimal.crt in real OS >

PAGE NO. :
DATE : / /

ls
↳ nimal.crt nimal.csr nimal.key

URL Kubectl for rhel8
↳ Create yum repo in local.

yum install kubectl -y
kubectl version

kubectl config view

Master
cd /root/.kube/
ls
↳ config
(Contains all the info abt cluster)

OR. # kubectl config --kubeconfig nimal.kubeconfig set-cluster awskubecluster --server <master_ip_public>

Master
OR (kubectl cluster info ; ip)
https://15.4.8.6:6443

- We are using https (encrypted) (.: .CRT)
  URL : click on lock (URL)
  ↳ Details
- All the clients connecting through https server also need ca.crt (Master)
- .: we need to copy ca.crt file to local

PAGE NO.:
DATE: / /

.... :6443 ~~config~~ --certificate-authority=ca.crt

vim vimal.kubeconfig
↳ contains details of kube cluster.

kubectl config view
↳ failed.

kubectl config view --kubeconfig vimal.kubeconfig
↳ works.

kubectl get pods
↳ failed.

kubectl get pods --kubeconfig vimal.kubeconfig
↳ error. (we don't know user & pass)

~~# vimal.kubeconfig~~

kubectl config --kubeconfig vimal.kc set-credentials vimal --client-certificate vimal.crt --client-key vimal.key

vim vimal.kc

kubectl get pods --kubeconfig vimal.kc
↳ same error
Bec. contexts: null

~~# kubectl config~~

PAGE NO.:
DATE 10 Feb 21

SESSION 91 K8S

- IAM
  - Auth (certificate Based)
    - Access

- Auth (K8S)
  - User/passwd
  - User/TOKen
  - User/Certificate
  - User/TOKen (sso)

} Methods

OS. Level
cd /kubeus
Kubect get pods --kubeconfig vimal.kubeconfig

- By vimal.kubeconfig file we come to know which user to use
  - we can have more than one cluster, user, certificates in one file (vimal.kubeconfig)

- ~~###~~ we need to set the context.
  - which cluster-user combination we have to use.

kubeal config -h
kubeal config get-clusters --kubeconfig vimal.kc
  - gives details of clusters

PAGE NO. :
DATE : / /

kubectl config set-context awskubecluster
--user=vimal --kubeconfig
vimal.kubeconfig

#

vim vimal.kubeconfig

kubectl config set-context vimal@awskubecluster
--user= vimal --cluster awskubecluster
--kubeconfig vimal.kubeconfig

—— get-context ——.

→ For minikube
→ vimal@local (AuthInfo)
→ minikube@local (cluster)
→ vimal@local@ minikube@local (name)

kubectl get pods --kubeconfig vimal.kc
↳ again Failed!
vim vimal.kubeconfig
↳ current-context: "vimal@awskubeconfig"
or through cmd.
kube config use-contexts vimal@—
--kubeconfig vimal.kubeconfig

kubectl get pods --kubeconfig
vimal.kubeconfig
↳ unable to connect to
Server x509: cert.
(Error)

PAGE NO. :
DATE : / /

master
cd /etc/kubernetes/
cd pki
↳ ca.crt ca.key
apiserver.key apiserver.crt

vim apiserver.crt
↳ encoded
Only 2 ip are allowed.

- In cloud, we have one pub & one priv. ip. But our cluster is created using priv ip and we give pub ip to the api server

openssl x509 -in apiserver.crt -text

In doc
rm apiserver.*
kubeadm init phase certs all
--apiserver-advertise-address=0.0.0.0
--apiserver-cert-extra-sans=<pub_ip>

openssl x509 -in apiserver.crt -text
↳ SAN has pub_ip

docker ps | grep api
docker rm -f `docker ps -q -f 'name=k8s_kube-apiserver*'`

systemctl restart kubelet.

os: rhel
kubectl get pods --kubeconfig vimal.kubeconfig
↳ (Forbidden)

- K8S works on the principle of least privileges.
  ∴ we need to grant vimal (user) the ~~pp~~ access.

master
kubectl create ns tech

os: rhel
vim vimal.kubeconfig
~~#kubectl config~~
~~#kubectl get pods --kubeconfig vimalte --namespace tech~~

master
kubectl get roles -n tech
kubectl create role vimal-tech --resource=pods --verb=get,list -n tech

kubectl get roles -n tech

~~os: rhel~~
~~# kubectl get pods --kub~~
kubectl get rolebinding -n tech
kubectl create rolebinding vimal-tech-rb --role vimal-tech --user vimal -n tech

kubectl get rolebinding -n tech

kubectl describe rolebinding
vimal-tech-rb -n tech

kubectl describle role vimal-tech
-n tech

OR, then
kubectl get pods --kubeconfig
vimal.kc --namespace tech

master:
kubectl edit role vimal-tech -n tech

PAGE NO.:
DATE: 12 Feb 21

SESSION: 92 K8S

- Server program for K8S : kube API
- port : 6443
- kubectl behind the scene goes to the API server.

URL: kube api references
↳ API overview

master

curl https://<ip_api_master_pvt>:6443
↳ ca.crt req.

curl --cacert ca.crt https://172.31.91.29:6443
↳ anonymous forbidden

curl --cacert ca.crt https://—:6443/version
↳ anonymous can also connect.

curl —/api/v1/namespaces/default/pods
↳ anonymous forbidden

- For curl we need key, crt all in dif files.

vim /root/.kube/config
<copy client-certificate-data>
to admin.crt in pki dir.
{*y$ → copied data from cursor till last}
{*p → paste data}

<copy client-key-data>
to admin.key in pki dir

PAGE NO. :
DATE : / /

curl ——— --cert admin.crt --key admin.key --cacert [redacted].crt ———
↳ PEM client certificate issue

cat admin.crt | [redacted] base64 -d > [redacted] admin-final.crt

cat admin.key | base64 -d > admin-final.key

Curl --cert admin-final.crt --key admin-final.key ———
↳ WORKED!!!
↳ Lists all the pods & other details.

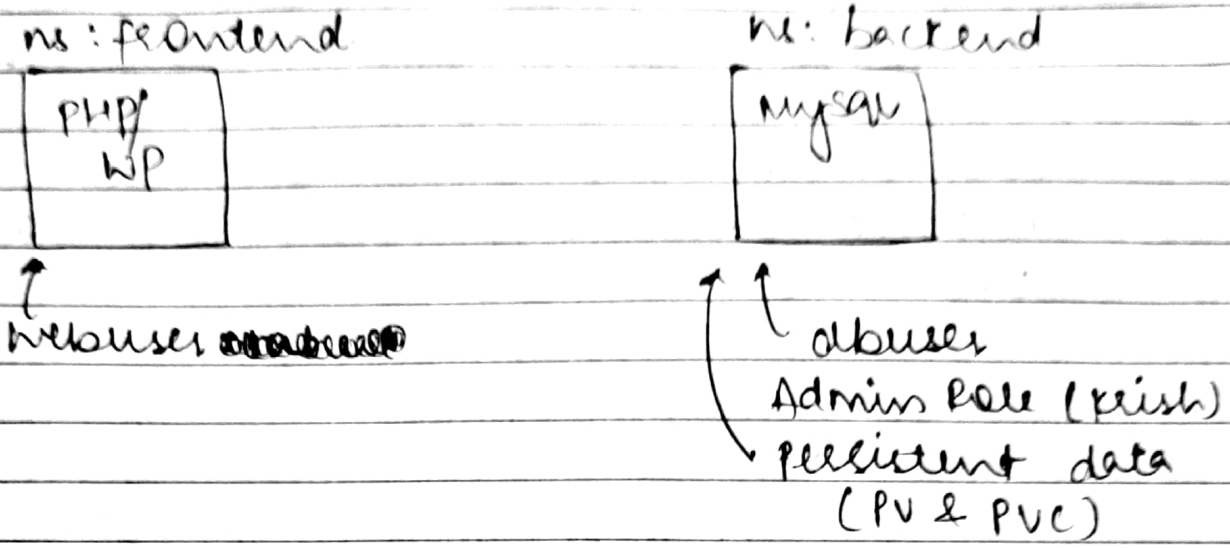
curl ——— /tech/pods
↳ no item running

curl ——— :6443/[redacted]
↳ lists all the api name

PAGE NO
DATE 18/Feb/21

SESSION: 93 K8S

- Cluster Roles



kube → Ns scope (pod, deploy)
kube → Cluster scope (PV, PVC)

clusterRoleBind: when we bind a cluster scope service to a user.

Master

```
#sudo su -
# cd /etc/kubernetes/pki
# openssl genrsa -out sadmin.key 1024
# openssl req -new -key sadmin.key
    -out sadmin.csr -subj "/CN=sadmin/O=store"
# openssl x509 -req -in sadmin.csr -CA
  ca.crt -CAkey ca.key -CA createserial
    -out sadmin.crt -days 365

# cp /root/.kube/config sadmin.kubeconfig
```

PAGE NO. :
DATE : / /

vim sadmin.kubeconfig
↳ ca.crt is inside it

users:
- name: sadmin
  user:

- context:
  cluster: kz
- server:
  <pub ip>

Change

* change context
↳ sadmin@kubernetes

• Delete admin details.
~~client certificate data~~
• copy sadmin.crt
↓
cat sadmin.crt | base64 -w0 (zero)
(encodes into one single line)

• client-certificate-data: ~~copy~~ <paste .crt>

cat sadmin.key | base64 -w0

• client-key-data: <paste sadmin.key>

copy sadmin.kubeconfig
↓
paste in vim.

• Now vim vs the file is located

#kubectl get pods --sadmin

cp sadmin.kubeconfig /home/ec2-user
chmod 664 sadmin.kubeconfig

PAGE NO.:
DATE: / /

min.

kubectl get pv --kubeconfig
sadmin.kubeconfig

↳ Forbidden (error)

• Now we need to create a cluster role and bind it.

kubectl api-resources --kubeconfig
sadmin.kubeconfig

↳ lists all the resources.

master

kubectl create clusterrole mysadmin
--verb=get,list,watch,create --resource=pv

kubectl create clusterrolebinding
myclusteradmin --clusterrole=mysadmin
--user=admin

min

kubectl get pv --kubeconfig
sadmin.kubeconfig

PAGE NO.:
DATE: 19Feb21

SESSION: 94 K8S

- Copy sadmin.kubeconfig in RHEL OS.

rhel

pwd (/root)
mkdir .kube
ls — config (main file)
cp /root/sadmin.kubeconfig config

- Now we don't always to mention the kubeconfig file.

#kubectl delete all --all
(also delete ns)

- Taint : We can't launch any pod/svc in master node. If Taint = None, that means user can launch the pods in it.

mkdir /ws
cd /ws
kubectl explain pod --recursive
↳ gives the details/explaination of the resource

kubectl run mypod --image=nginx --dry-run=client -o yaml
↳ They don't implement the cmd, just checks the syntax
↳ -o yaml converts the cmd to yml file.

PAGE NO. :
DATE : / /

- Kube-scheduler is the program that decides where to launch the pod. We can decide ~~~~ in which node we want to create the pod by using selectors.

kubectl label node <ip-172-31-87-29>.ec2.internal region=US

vim mypod.yml

```
spec:
  nodeSelector:
    region: US
```

kubectl apply -f mypod.yml
↳ Now ~~~~ the pod will be launched in the node with label region=US.

kubectl describe node <ip> <worker node> | grep Taint
↳ Taints: <none>

kubectl taint node <worker2 ip> ~~~~ mytype=veyimp:NoSchedule
(NO space)

kubectl label node <worker2 ip> region=IN

(mypod1)

- Now if we launch the pod in region IN it will not be launched with status Pending because of Taint.

kubectl edit pod mypod1
  - region: US (change)
  - FAIL
  - Re launch the pod mypod1 with region=US.

- Tolerance: when we launch the pod in a tainted node.

kubectl edit pod mypod1
  - Tolerations:
    - effect: NoSchedule (change)
      key: mytype
      value: xyzing
      operator: Equal } (change)
      (rm rest lines)

- If we taint a node then the older pods won't be affected.

- NoSchedule: It won't launch any pod without any tolerance but won't terminate the older pods.

PAGE NO. :
DATE : / /

- NoExecute: It will also terminate the older pods with no tolerance.
  Used for maintainance.
- ~~Abstract~~

PAGE NO. :
DATE : 22/Feb/21

SESSION : 97

launch the instance with customised ami

- copy and from deine.

- static pod : launched from node when master is down. Not managed by master. They end with hyphen & node in which it is launched.

/etc/kubernetes/manifestes

- Daemon set.

PAGE NO.:
DATE: 24Feb21

SESSION: 98 K8S

- Package manager in k8s is Helm
  also known as chart

- We install Helm in user / workstation
  we can also use master as user
  (or windows)

URL: helm install
(direct link)

Helm (Linux and 64)

V3 V2

V3 → don't need the "program"

V2 requires Tiller program

master
wget <link>
tar -xvzf helm-v3.5.2-linux-amd64.tar.gz
cp linux-amd64/helm /usr/bin/

helm version
↳ v3.5.2
mkdir cd /ws
mkdir myweb

*. folder_name == package_name
cd myweb
vim Chart.yaml ← same for all
(Chart.yaml)

PAGE NO. :
DATE : / /

apiversion: v1
name: myweb
version: 0.1.0
appversion: 1.1
description: this is apache server pkg.

helm install myweb mywebl
myweb → name

helm delete myweb

[myweb]# mkdir templates
cd /templates
#
kubectl create deployment myd
--image=vimal13:v1 --dry-run
-o yaml > deployment.yaml

[ws] # helm install myweb mywebl
↳ It will also launch a deploy.

[template] # kubectl expose deploy myd
--port=80 --type=NodePort --dry-run
-o yaml > svc.yaml.

#kubectl delete

PAGE NO. :
DATE : / /

helm upgrade myweb1
le requires 2 arg, failed!!

~~# Rever upgrade~~

- we need to update myweb1/Chart.yaml
version : 0.2

helm upgrade myweb myweb1

helm history myweb
helm rollback myweb1

[myweb]# vim values.~~yaml~~
~~replicaCount: 3~~
replicaCount: 3

vim template/deployment.yaml
spec:
~~replicas: {{ .Values.replicaCount }}~~
replicas: {{ .Values.replicaCount }}

helm delete myweb
helm install myweb myweb1/
--set replicaCount=4

helm repo list
le no repo.

PAGE NO. :
DATE : / /

helm repo add bitnami
https://charts.bitnami.com/bitnami

PAGE NO.:
DATE: 25 Feb 24

SESSION: 99 K8S

master
kubectl delete all --all

- Configmap: when we mount the config files to a storage, and the storage is mounted to all the pods.

configmap
kubectl get cm
kubectl create deploy myd --image=ivimal
kubectl exec -it myd -- --bash
ps -aux
cd /etc/httpd/conf.d
netstat -tnlp
↳ port 80 running

vim ~~web~~ web.conf
Listen 85
↳ load inside the pod
kubectl create configmap mywebconf
--from-file=web.conf

kubectl describe cm mywebconf

kubectl edit cm mywebconf
↳ now config files can be easily edited.

kubectl edit deploy myd
↳ now we attach cm to deploy

PAGE NO. :
DATE : / /

template:
  spec:
    ~~containers~~:
    volumes:
      - name: config-volume
        configMap:
          name: mywebconf

    containers:
      volumeMounts:
        - name: myvolume
          mountPath: /etc/httpd/conf.d

• Now the pod will be relaunched.

kubectl exec -it myd-—— --bash
  # netstat -tnlp
    ↳ 80 port is also active.
      Hence verified.

• CoreDNS: Light weight DNS product

URL: coredns.io
  ↳ coredns_1.8.3_linux_amd64
    ↳ copy link

wget <link>
tar -xvzf <file-name>

PAGE NO. :
DATE : / /

- DNS → port 53 /UDP

vim /etc/mydb
1.1.1.1 vimal
2.2.2.2 hello.iw.com
172.31.44.31 my

~~#./coredns <same dir as software~~

# vim mycoredns.conf
. {
hosts /etc/mydb
}
↳ conf for coredns

./coredns -conf mycoredns.conf

netstat -unlp | grep 53
↳ started (udp *53)

model
~~paste~~ vim /etc/resolv.conf
↳ this file is used to make the system a dns client.
nameserver 172.31.46.246
↳ dns-server-ip
aka master

ping vimal
↳ worked.

search (keyword) : It will by default add the domain name in dns name.

PAGE NO. :
DATE : / /

kubectl get pods -n kube-system
↳ coredns - ——— —

kubectl describe deploy coredns
-n kube-system

k delete all --all
k create ns tech
k create ns hr
k create deployment myd1 --image=vimal13
-n tech --replica=2
k expose deploy myd1 --port=80 -n tech
k get svc -n tech

——— myd2 — -n hr ———
——— myd2 ——— -n hr

- In k8s they give the same dns as the hostname of ~~podd~~ svc.

PAGE NO.
DATE 2Mar21

SESSION: 105 K8S

win
minikube start

ALB: Application Load Balancer is the LB between the microservices.
Ex: nginx
↳ also works as a proxy.

1. Launch nginx ~~as~~ ingress controller
∴ launch a pod that will work as the controller.

win
minikube addons enable ingress
kubectl get pods -n kube-system
↳ ingress-nginx-controller
↳ running
kubectl get ingress

shell
mkdir /msk
cd
vim Dockerfile
FROM vimal13/— index.php
RUN echo "search app" > /var/www/html/

docker build -t vimal13/searchapp:v1 .
docker
vim D—
RUN echo "mail app" > /var/www/ht/ ind

docker build —— mailapp:v1 .
docker push vimal13/searchapp:v1
—— mailapp:v1

PAGE NO. :
DATE : / /

kubectl create deployment maild --image=mail:v1

kubectl scale deploy maild --replicas=2

~~k~~ create ——— searchd ———
--image = ——search:v1

~~k~~ —— scale deploy searchd
--replicas=3

kubectl expose deploy maild
--port=80 --type=NodePort

——————— searchd
--port=80 ———————

kubectl get svc
↳ maild → 203 (ip)
↳ searchd → 251 (ip)

minikube ssh
ifconfig
↳ (99.100 ip of minikube and ie)

curl 192.168.99.100 : 30186
↳ port of maild

~~kubectl get svc -n kube-system~~

curl 192.168.99.100:80
↳ check

PAGE NO. :
DATE : / /

- In ingress controller we need hostname it won't allow with ip.

windows
c drive → windows → system32 → drivers → etc → hosts
(open with notepad.)

192.168.99.100 www.lw.com

curl www.lw.com

- Mandatory to give hostname for layer 7 protocol (network layer)

~~kubectl create ingress~~
notepad ingress.yml

```
apiVersion : networking.k8s.io/v1
kind : Ingress
metadata :
    name : "myingress"
spec :
    rules :
      - host : www.lw.com
        http :
          paths :
            - path : /search
              pathType : Prefix
              backend :
                service :
                  name : searchd
```

pathType : Prefix → pathType : Prefix

PAGE NO. :
DATE : / /

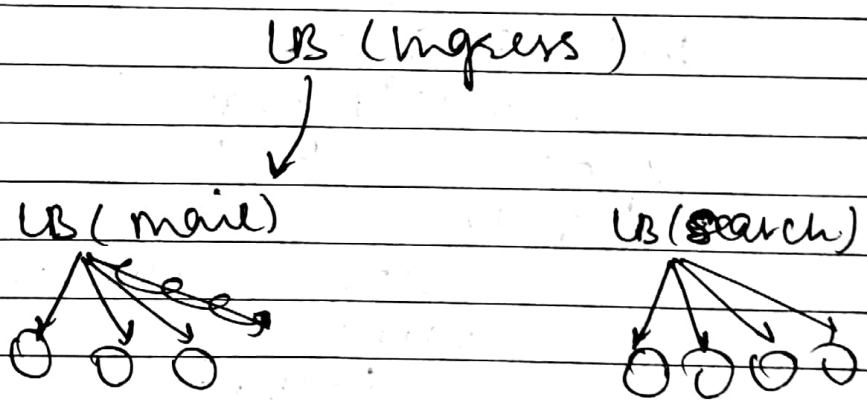
```
port:
  number: 80

- path: /mail
  pathType: Prefix
  backend:
    service:
      name: mail
      port:
        number: 80
```

kubectl apply -f ingress.yml

kubectl get ingress

curl http://www.lw.com/mail
↳ /mail not found

kubectl describe ingress myingress



Note Imp: In the docker containers we need to create the path folder in /var/www/html.

PAGE NO. :
DATE : / /

eg : http://www.cw.com/maid

Then in maid
The webpage should be in
/var/www/html/maid/
index.php

OR

- we can remove the target by rewriting the url.
means : change /maid to / while sending the request.

* vim cwingress.yml
metadata:
name:
annotations:
nginx.ingress.kubernetes.io/rewrite -target

nginx.ingress-kubernetes.io/rewrite-target: /$1

- we can also use annotation when our container is in cloud (aws, azure, gcp).

* kubectl apply -f cwingress.yml

curl http://www.cw.com:80/maid

PAGE NO.:
DATE: 3 May 21

SESSION: 106 K8S

- Network policy: Guard which allows or ~~denies~~ drops the packets before sending it through the SVC (LB).
  - Based on the SG (inbound traffic)

- In K8S all the network are done by overlay network.
  - Only used for overlay.
- Flannel doesn't allow firewall & security to our packets.
  - ∴ We need to switch it to CNI plugin.
  - Calico is CNI also provides N/P.
  - Weave also provides overlay as well as Network policy.

win end
ssh -i <key> -l ec2-user <pub ip>
master # sudo su -

kubeadm init ——.
  - master is configured

n1 # run kubeadm join cmd --node-name node1

master # cp / paste cmd given above
kubectl get nodes
  - shows up.

web : weave

kubectl apply -f ~~weave~~ weave.project"

ps -aux | grep kubelet

cd /etc/cni/net.d
↳ 10-weave.conflist
↳ weave config.

cd /opt/cni/bin
↳ all network related cmd are here.

ps -aux | grep weave

kubectl create deployment myd
--image=vimal13/apache
--replicas=3

kubectl expose myd --port=80
--type=NodePort

kubectl get networkpolicy

~~kubectl create~~

vim np.yml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: myfirewall
~~spec:~~

kubectl apply -f np.yml

vim np.yml

spec:
  policyType:
  - Ingress
  ingress:
  - {}

allows all the ports/ip

kubectl apply -f np.yml

vim np.yml
  podSelector: {}

apply -f np.yml

create deploy myd1 -image=httpd
expose 80 NodePort myd1

vim np.yml

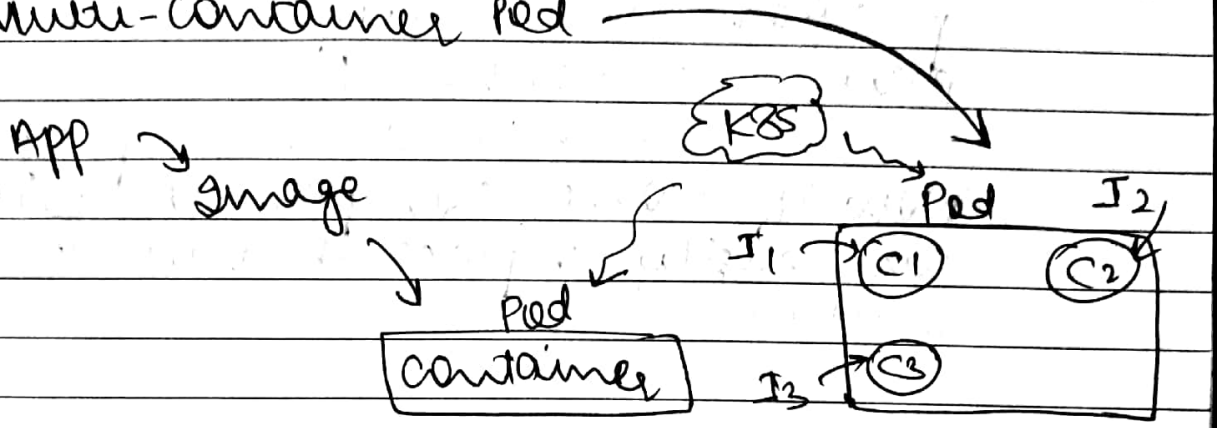
podSelector:
  matchLabels:
    app: myd1
ingress:
- ports:
  - protocol: TCP
    port: 85

apply -f np.yml

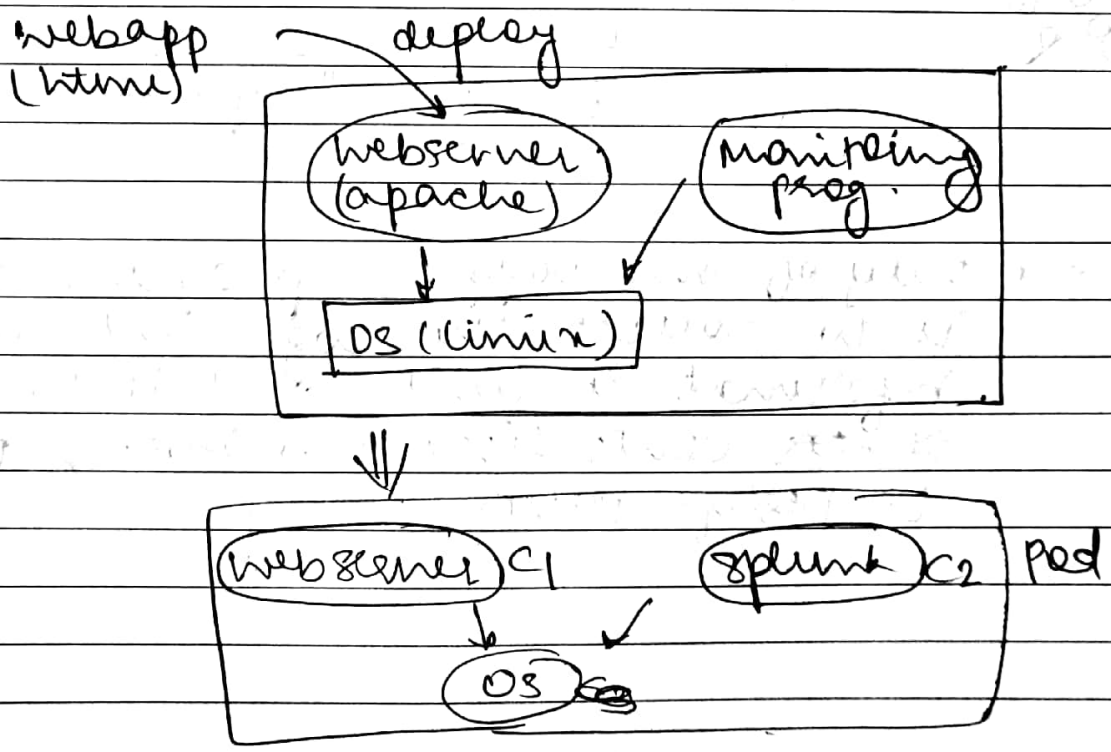
PAGE NO. :
DATE : 19/Mar/21

SESSION : 117 K8S

- Multi-container Pod



- Till now we have only seen single container pods.
- Monitoring Tool : Splunk, ELK, EFK



- In the $ environment either everything works or nothing.

PAGE NO. :
DATE : / /

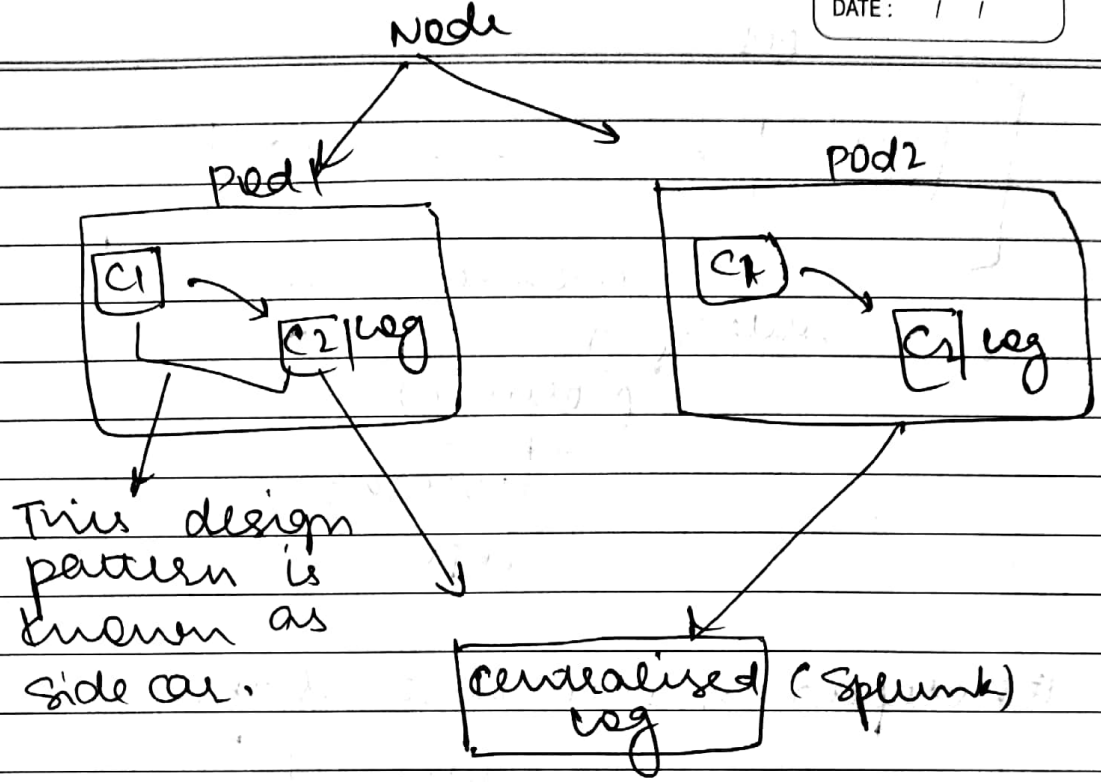
Therefore we put every service/ program in a different container.
C1 → webserver
C2 → monitoring program
Now if some component fails we can easily replace it with a new container.

notepad pod-basic.yml

multicontainer pod

```
spec:
  containers:
    - name: "webC1"
      image: "-apache-php"
    - name: "webC2"
      image: "splunk"
```

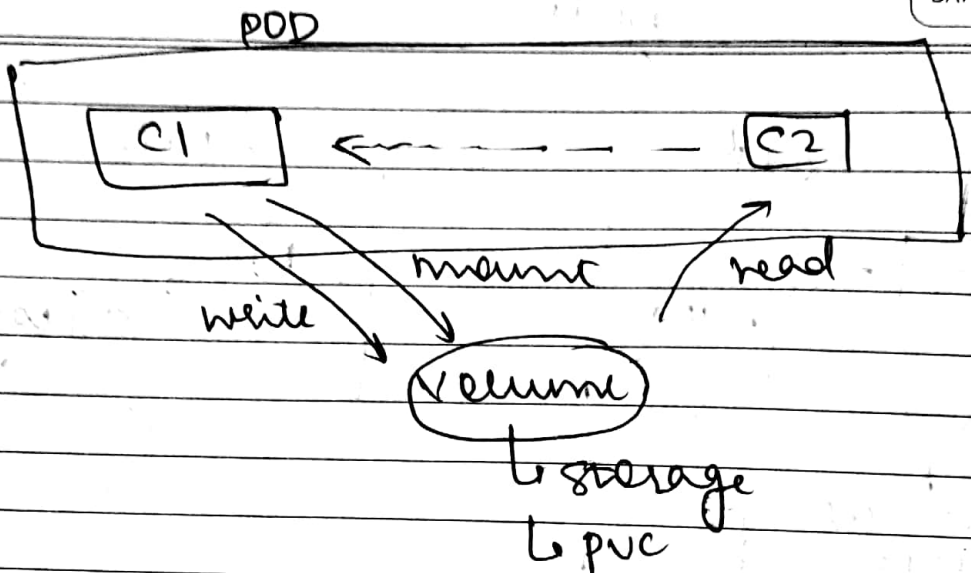
- Duty of logging program/agent is to collect the log and then reformat it and send it to the centralised monitoring program.
  Ls Design Pattern : Adapter



- Instead of storing the logs in local container they send it to the centralised log.
  This design pattern is adapter

- Here C2 collects the logs from C1 and performs some operations before sending the data to the log. ∴ It can also act as a proxy.

notepad multi-sidecar.yml (provided string)



notepad multi-share-vol.yml
↳ shared

kubectl exec -it mc1 --bash
↳ by default they will always pick the first container
or
kubectl exec -it mc1 -c 1st --bash

- Here C2 is writing the logs and C1 is reading.

- Multi_IPC
↳ Inter Process communication
Docker :: Process

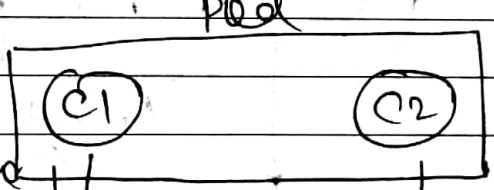
kubectl logs mc2 -c 1st
~~1st~~ 2nd

PAGE NO. :
DATE : / /

How dif containers in same pod communicate
- storage
- memory
- network

kubectl apply -f multi-adaptor.yml

- Init container
  - C1 & C2 should start at the same time



But what happens is if C1 starts prior then we might face data logs or If C2 starts prior it will fail because it doesn't have any data to consume.

- Init Container is a concept if one container is priority then the other containers keep on waiting till all its processes are completed successfully.

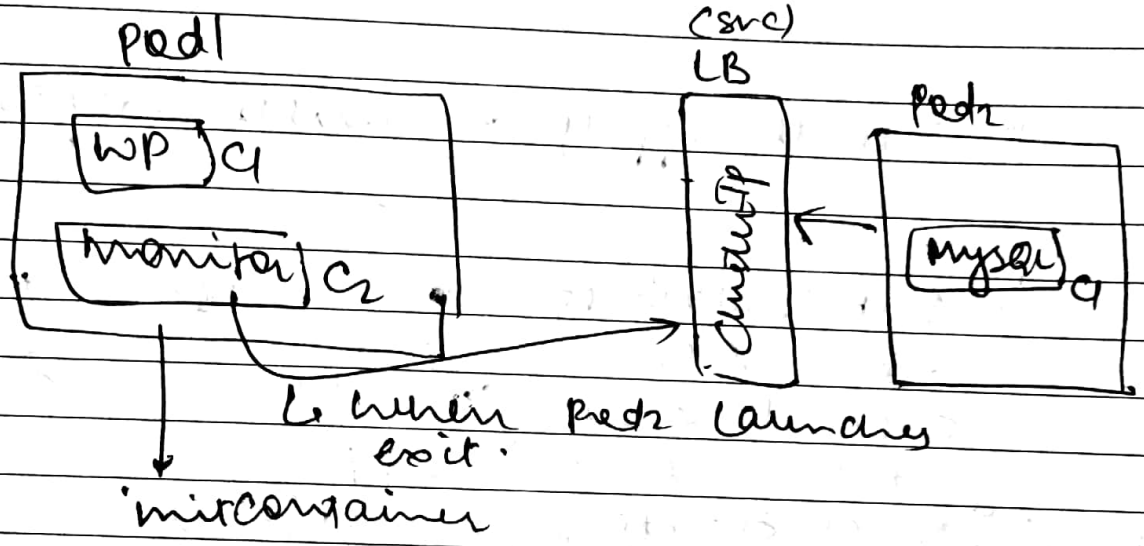
```
spec:
  initContainer:
    - name:
    - name:
  Container:
```

PAGE NO. :
DATE : / /

- Example wordpress & mysql

↑ init container

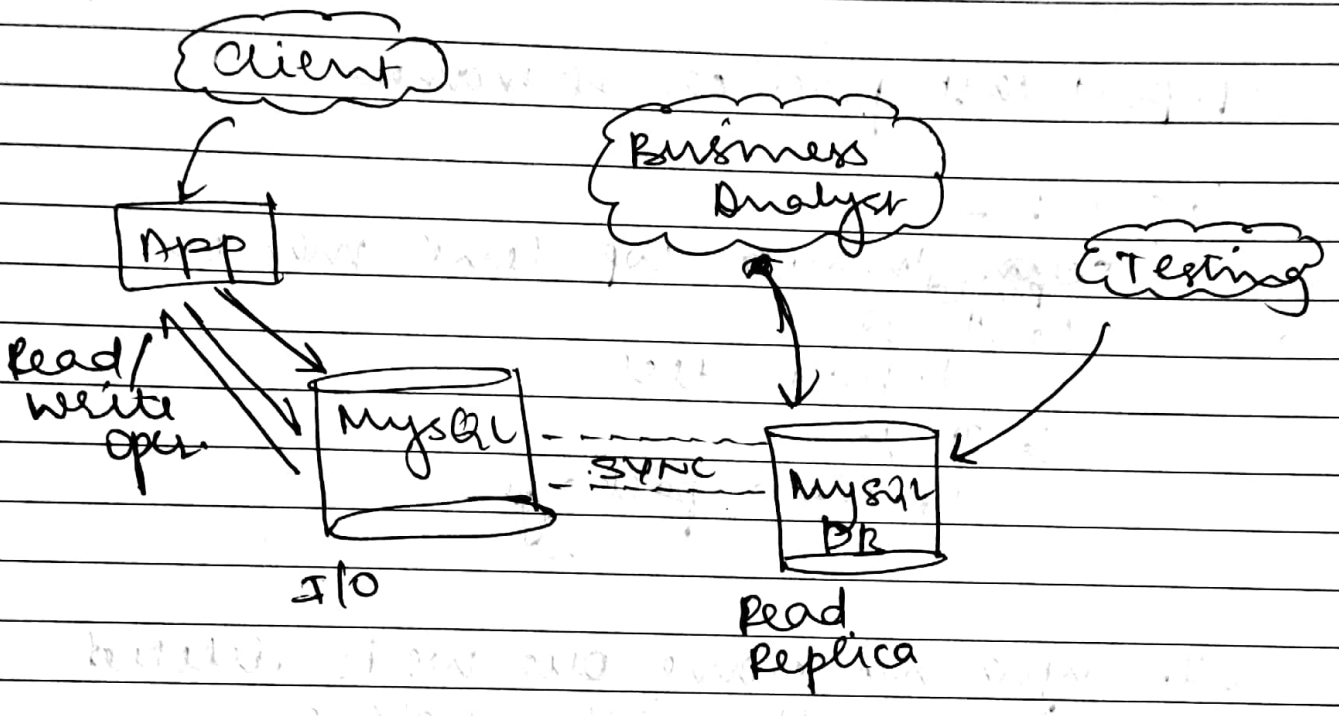
Bec without P.E our data will be lost.

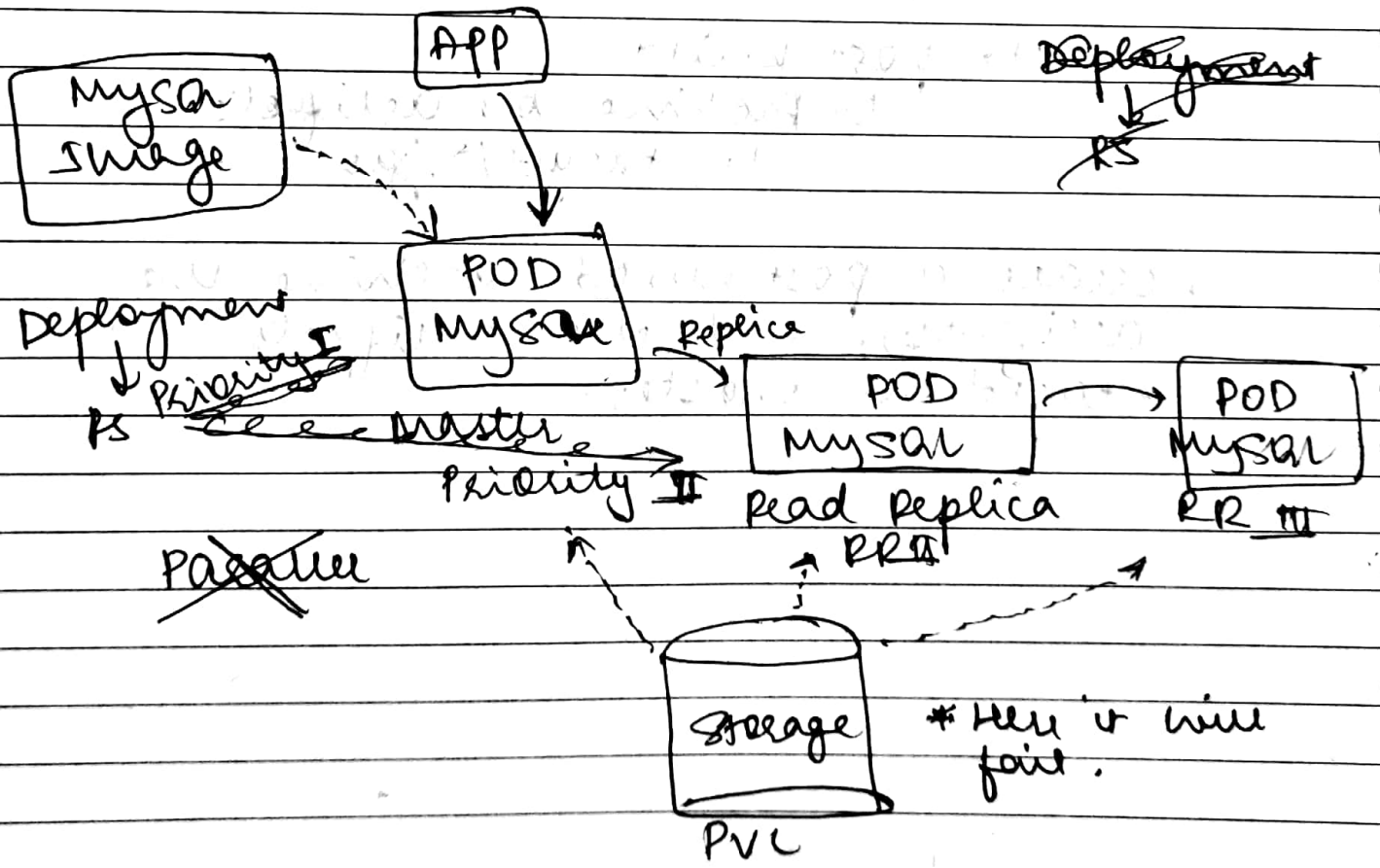


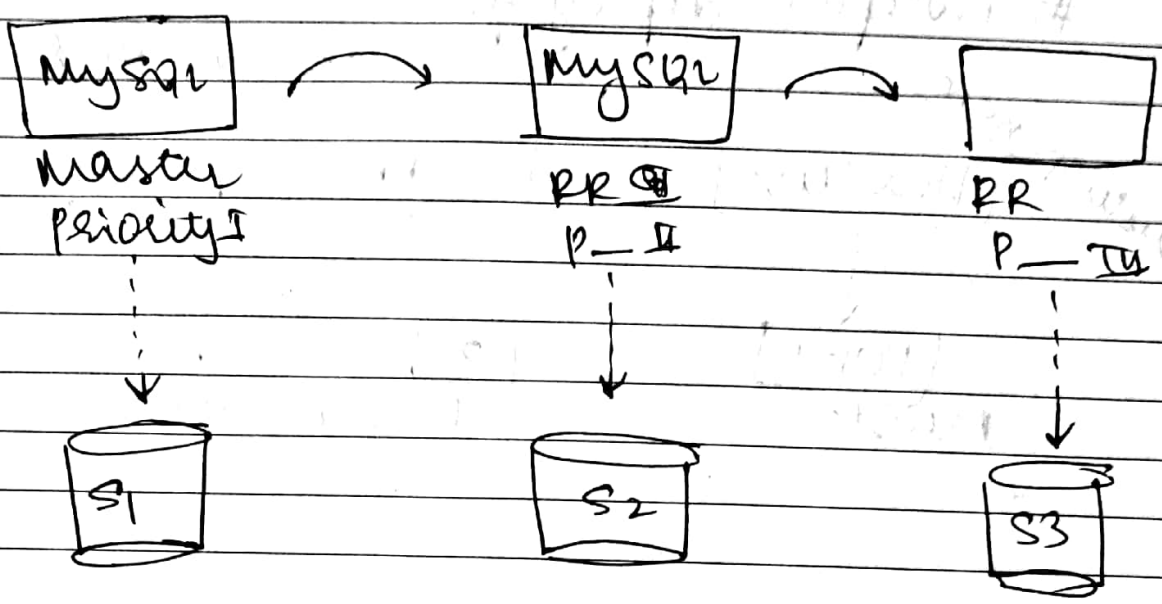
PAGE NO. :
DATE 25/Mar/21

SESSION: 124 K8S

- Storage == Bottleneck
  - Bec it's the slowest.





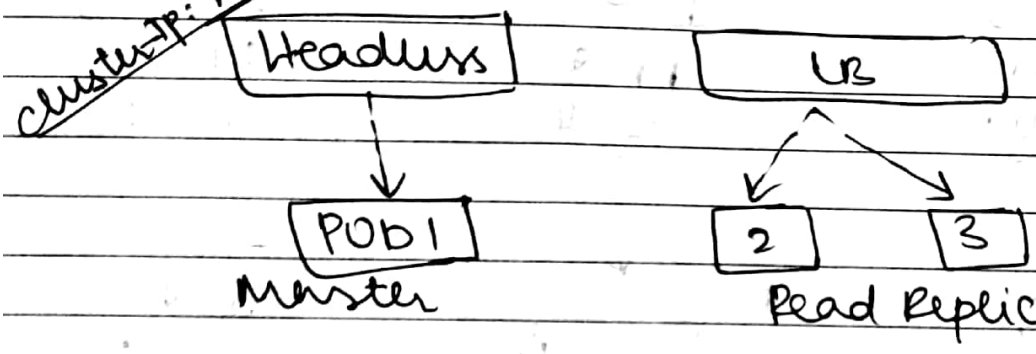


- In this kind of setup we can't use deployment.
- There is new kind in K8S for this type of setups.
  kind = StatefulSets
- PVC → VCT
  ↳ Volume claim template
- Headless Svc : when the ip of clusterIP is none.
  That means, it does not have any ip.

PAGE NO. :
DATE : / /

~~# notepad deploy.yml~~ ~~we can edit deploy code~~

sf_svc.yml



sf_configmap.yml
= 2 cm → 1. Master
— → 2. replica

sf_ss.yml
kind: StatefulSet
- here instead of pvc we use vct. That will give seperate/ shared storage to all the pods.

PAGE NO. :
DATE : 14/4/21

Session: 142 K8s

Job/ Cronjob

mini ~~cube~~ start

- life of a pod is ∞. Bcz generally we configure a server on the pods and their life is ∞ until we close or terminate them. Their restart policy is always.

~~Cronjob~~
~~Program~~ }
~~Test~~

- MR is the algo especially for batch processing. These are known as jobs

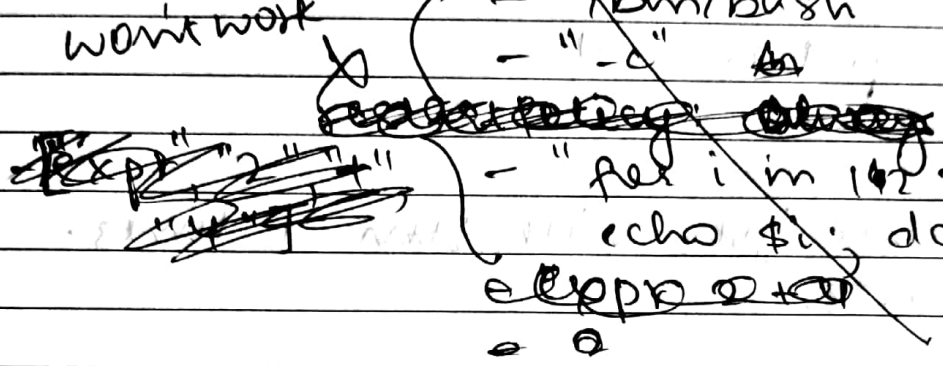
kubectl get job
to resources (job) with the speciality of deployment & name
restart policy = never.

kubectl get cronjob

kubectl create deploy myd --image=centos:7 --dry-run -o yaml > j.yml

notepad j.yml

containers:
- image: centos:7
  name: centos
  command: ["expr", "2", "+", "4"]
  - "/bin/bash"
  - "-c"
  - "for i in 1 2 3 4 5 6; do
    echo $i; done"



- Here, after the cmd are run, the container will restart because of its restart policy.
- Job is more suitable for Big Data where all the data is distributed.
  This feature is not available in deploy.
  - whether we want to run the jobs in parallel, series, etc.

∴ For batch processes instead of deploy we use job.

job.yml
apiVersion: batch/v1
kind: job
metadata:
  = } all same as deploy.}
  * no concept of replication in job
  * no selector
restartPolicy: Never

kubectl apply -f job.yml

- To run the jobs in series.

spec:
  completions: 4

Now, this job will run 4 times in series.

completions: 4
parallelism: 2

Now 2 jobs will run in parallel.

- kind: CronJob
  apiVersion: batch/v1beta1
  kind: CronJob
  metadata:

spec:
  schedule:
  jobTemplate:
  spec:
    completion:
    parallelism:

spec:
  schedule: "* * * * *"
  completions: 6
  2

This will also schedule the job.

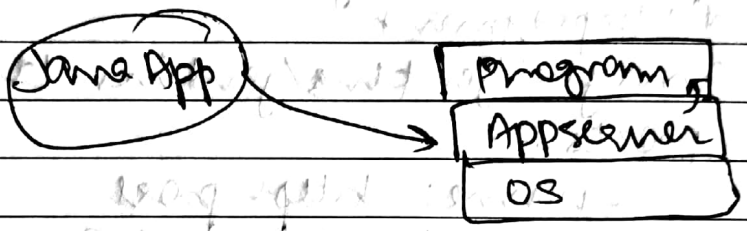
PAGE NO. :
DATE : 16/Apr/24

Session: 145 K8S

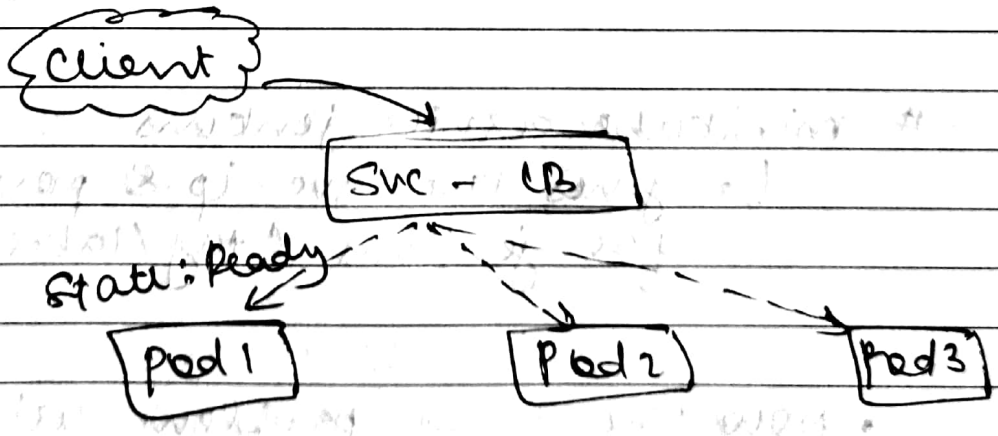
Jenkins over K8S

↳ Readiness Probes
↳ Liveness Probes

minikube start



- Sometimes, the app server becomes ~~bse~~ ready but the java app is so heavy it needs ~~so~~ further ~~son~~ time to be fully started. But the client can't use it till the app is completely ready.
  i.e pod is ready to use but the app inside it is not.



↳ starts the process in the pod. ; LB sees state: Ready and sends the traffic to the pod.

PAGE NO. :
DATE : / /

L But sometimes, the app is not fully configured / launched.
Here, we are stuck. And client has to wait for some time to fully load over the app.

notepad jenkins.yml
kind : Deployment
image : jenkins/jenkins : lts
ports :
- name : http-port
containerport : 8080
- name : jnlp-port
containerPort : 5000
- - -
- - -
- - -
~~kind : Svc~~ kind : service

kubectl apply -f jenkins.yml

minikube service jenkins
L gives the svc ip & port for jenkins (tag / label)

- Now the same problem arises. Jenkins app requires some more extra time to fully launch the server.

PAGE NO. :
DATE : / /

kubectl describe pod jenkins
↳ Conditions:
Ready: True
{ Acc to K8S, the pod is started. But they can't go inside the pod & check if the svc has really started or not }

- Readiness Probes.

notepad jenkins.yml
containers:
- name:
image:
readinessProbe:
httpGet:
path: /login
port: 8080

- Now it will only make the state ready until the app is fully launched.

- Liveness probe : It is more oriented towards the programming.
If backend of the app is not ready then we use liveness probe.

PAGE NO.:
DATE: 27/09/21

Session: 152 K8s

Security Context

- By default, all the processes in the containers are run with root user.

```
# kubectl create deploy myd --image=——
# kubectl exec -it deploy myd -- sh
  sh:# id
      ↳ root ; uid = 0
  # ps -aux
      ↳ id : root

# notepad security-context.yml
  ↳ spec:
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
      ——
      =
      =
      containers:
        =
        =
          securityContext:
            allowPrivilegeEscalation: false
```

- Due to this we do not have certain permissions but we want to change that. So while ~~create~~ launching the container we can add the capabilities

```
securityContext:
  capabilities:
    add: ["NET_ADMIN", "SYS_TIME"]
```

- Limit Request : If we want certain requirements for a pod to be launched. Lets say we req. 4Gb RAM so scheduler will search for the node having 4Gb RAM free.
- We can also set a limit for the pods. Max 2Gb RAM it can use.

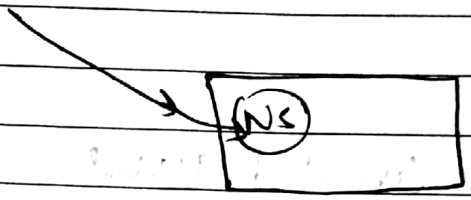
```
resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
```

#kubectl get limitrange

```
kind: LimitRange
==
spec:
  limits:
  - default:
      cpu: 1
    defaultRequest:
      cpu: 0.5
    type: Container
```

PAGE NO. :
DATE : / /

Quota (RBAC)



- User can only work in one namespace only. But there is no constraint over the no of resources used by the user in that namespace. This is RBAC
But if we want to create the constraints we can create quota.

k- create namespace ns1
k get quota -namespace ns1

```
kind : ResourceQuota
metadata:
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu:
    limits.memory:
```

k- create quota --hard=cpu=1

PAGE NO.:
DATE: / /

Metric Server

- Helps in providing all the logs & metrics of all the pods / nodes.
- agent they launch is cAdvisor.
- ★. code available on github.
- In minikube we have addons.

minikube addons enable metrics-server.

PAGE NO.:
DATE: 29/Apr/21

Session: 154 K8S

K8S with Cri-O

Ap1 : Ubuntu Server: 20.04
↳ Launch instance
↳ No. of instances: 3
↳ All Traffic

Multi-Node Cluster
↳ Cri-O

ubuntu →
sudo su - root
all cmd are in prive

download keys
apt update
Install crio
Start svc.

crictl images

set repo.
install K8S
start master.

kubectl get nodes
↳ master

• Everything will be same expect the runtime is CRI-O.

PAGE NO.:
DATE: / /

- Instead of flannel we use calico as overlay network.

etcd

↳ It is just a database
Whatever we have created in k8s all the info is stored in a database → etcd (3rd party db)

- It is launched inside a pod etcd-minikube (in minikube)

- If we have a planning of migration/upgradation we can take the backup of etcd. Then paste it in the new device all the previous work won't be lost.

kubectl get pods -n kube-system
kubectl exec -it etcd-minikube -n kube-system -- sh
sh # etcdctl

- etcd is a secured pod we need server.crt, ca.crt, server.key
cd /var/lib/
etcdctl put

PAGE NO. :
DATE : / /

sh# etcdctl put --key= ---- city jaip.

↳ In drive

sh# etcdctl snapshot save

HPA → horizontal pod autoscaler
~~create~~ checks & resources.

kubectl get hpa

- We can create replicas on the basis of limits.