Why kubernetes ⇒

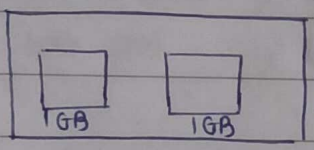
1) Auto restart of container

In this kubernetes help to auto restarting container, when container is crashed / exited state or if container is okay, but in under container web server is not working well then auto start these container.

2) healthcheck

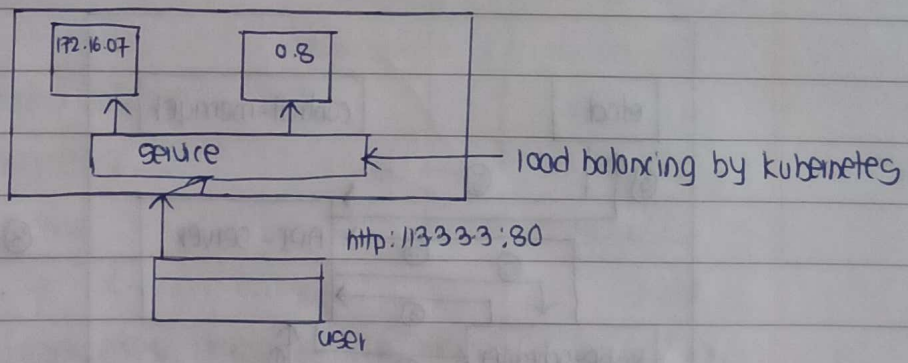
kubernetes also functionality of healthcheck, that helps checking web server working or not by kubernetes (if nginx is not working)

3) Autoscaling

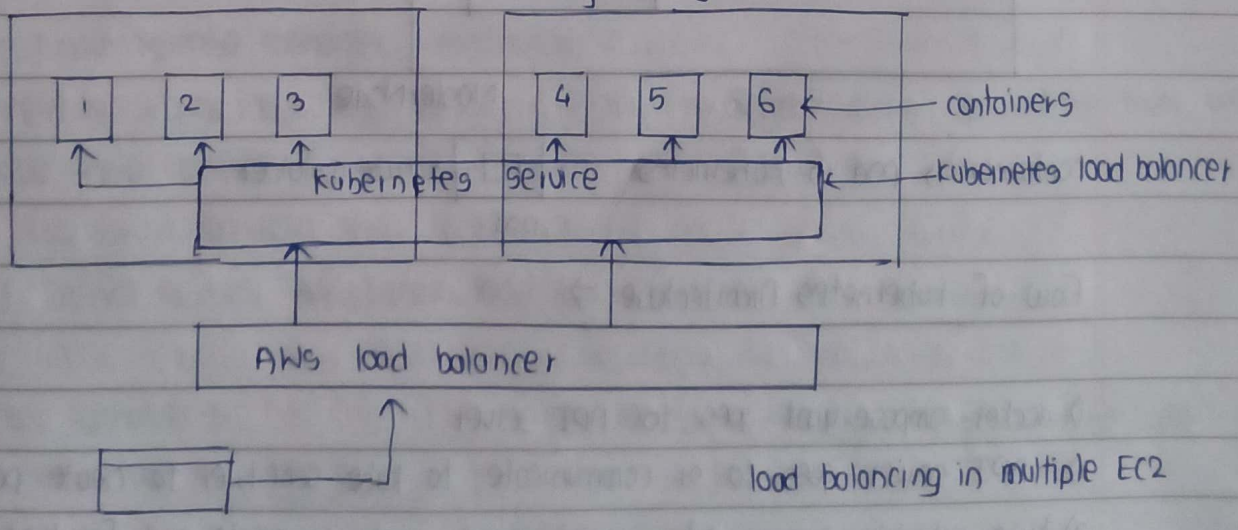


kubernetes Autoscaled container if container ram is fulled, then kubernetes scaling container

4) Load balancing



load balancing in single ec2



load balancing in multiple EC2

docker run --cpu 0.25 --memory 512 mb -d

↳ in docker

request 0.25 (min)

limit 0.5 (max) → in kubernetes

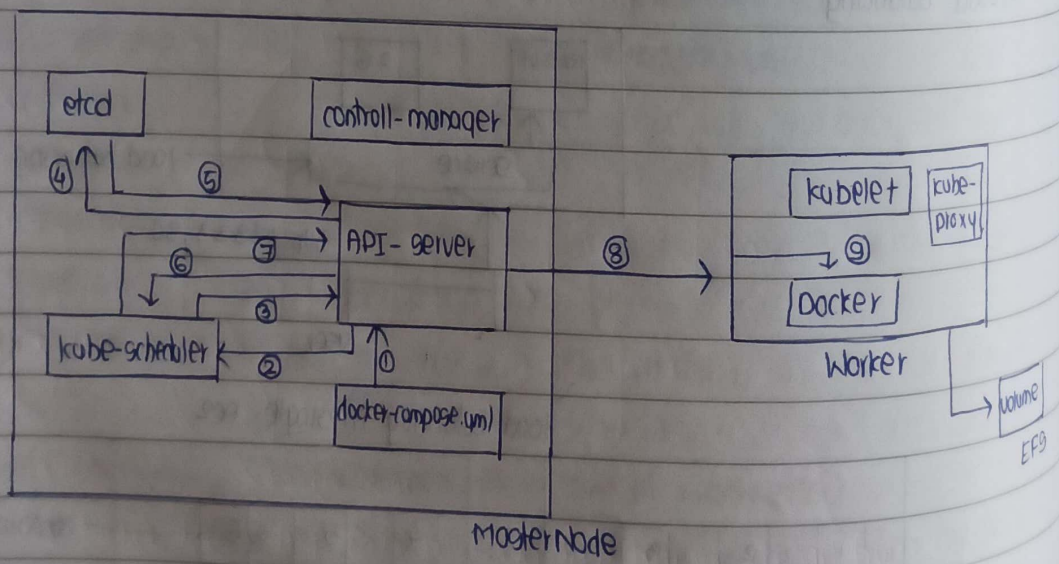
ECS → elastic container service

EKS → elastic kubernetes service

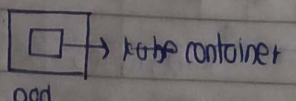
Why kubernetes ⇒

1) Auto start of container
2) healthcheck
3) Autoscaling
4) load balancing
5) Network
6) Auto - Healing
7) Auto Node Allocating
8) updates/new release / deployment
9) secreate / config (master node)

Kubernetes Architecture ⇒ Component of K8s ⇒



container → pod in kubernetes



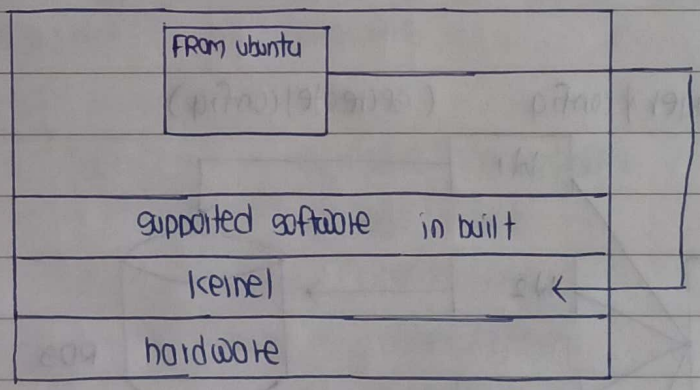
flow of kubernetes Architecture ⇒

1) docker-compose.yml goes to API server
2) API server goes to or communicate to kube-scheduler to create pod
3) kube-scheduler has no data in which worker empty to create pod for that communicate API server

4) API-server communicate to etcd and get data
5) etcd send data to API server.
6) API server data to kube-schedular.
7) kub-schedular decide in which worker create pod these send to API server
8) API server send to worker in worker through kubelet
9) kubelet create pod through Docker image file.
Hence our pod will create.

etcd -: no-sql non relational json
control-manager ⇒ control manager managed the pod, pod is crashed or distroyed or not responding this information sends the API server and then create new pod.

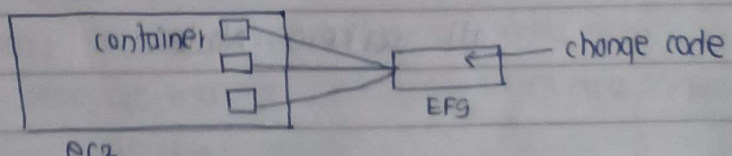
container is used host os and container os what is difference in os ?
⇒



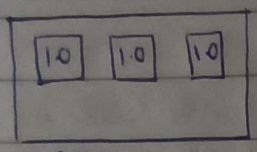
Day-16 kubernetes (why kubernetes, kubernetes objects)

8) Updates | new release manager | deployment.
if we have bug in our containers or any modification in containers then we release new update for new features. for that two ways
1) by volume
2) with help of kubernetes

1) by volume ⇒
disadvantage → get more time to update code

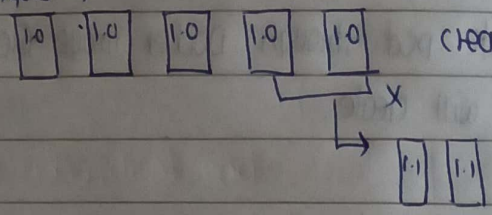


2) by kubernetes ⇒



ec2 change to 1.1

how kubernetes changes ⇒ थोड़ी थोड़ी करके सब करना rather than create करते



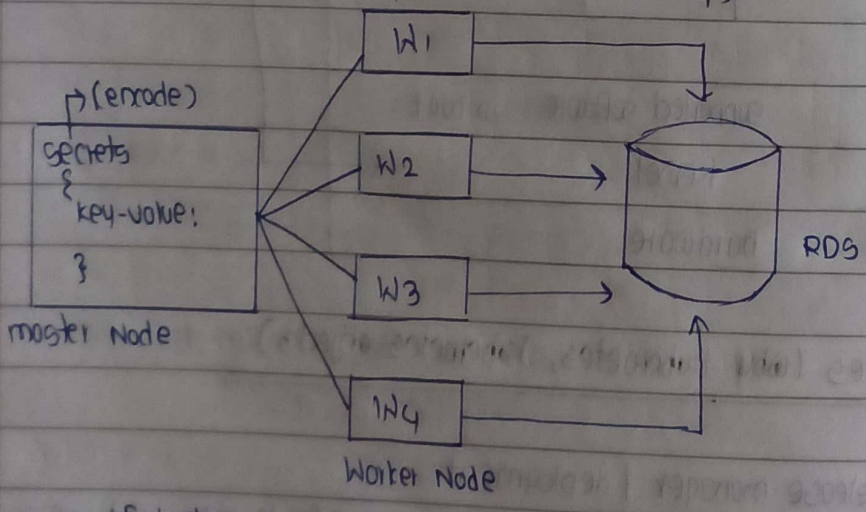
Roollback ⇒

If we new release manager and these are not working properly then rollback functionality is used in kubernetes that rollback to our previous up stable update's.

what is version ⇒

only changes in source code or software (software versions)

9) Password manager | config (secreate|config)



If hacker knows my RDS password then how we change from our containers at one time?

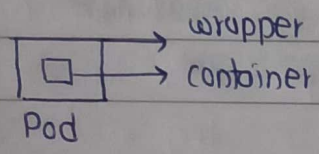
→ first change RDS password and then this password put in master node under that secreate ckl section key-value change Password then this password is send to worker node and changes automatically to all workers Node.

Also we can changes config in master node and this changes will change to all workers node

kubernetes objects ⇒

1) Pod
2) Service
3) volume
4) Network
5) configmap
6) secreate
7) Ingress

1) Pod ⇒



why pod ?

pod, under pod we can put any container, any container means different software created containers.

pod supported to all containers

Types of pods ⇒

1) Pod
2) replication controller
3) replica set
4) Deployment
5) statefull set
6) Daemon set

Pod → simple pod, wickness pod do not autostart containers.

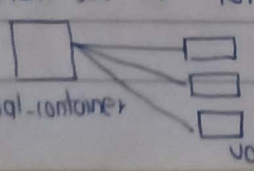
replication controller → if pod can crashed, then they create automatically new

replica set → they also start new pod, in short query we can see count or details of pod in list.

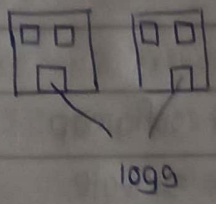
weakness in replication controller and replica set they can not support to update/release manager and rollback.

Deployment → Autostart, update/rollback, load balancing this will use in Industries

statefull set → for using for state level container mostly mysql they can remember (indexing) which data is store in which volume.

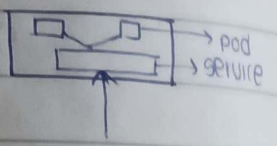


daemon set → in all worker, each worker create container and this used for logs



2) service

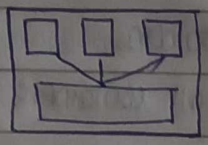
connectivity with from users/client and load balancing



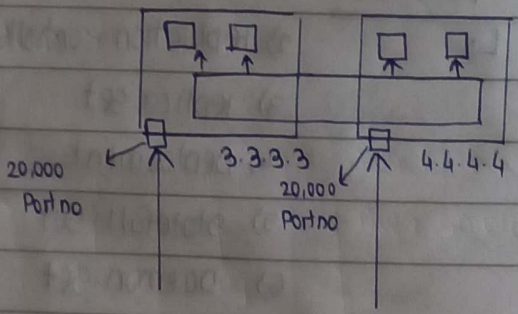
Types ⇒ 1) cluster IP
2) Node Port
3) Load balancer

cluster IP →

load balancing only under ec2, they are not connect to other wo



Node Port →



load balancer → most used (mostly used)

3) Ingress

for domain name assigning
for multiple services.

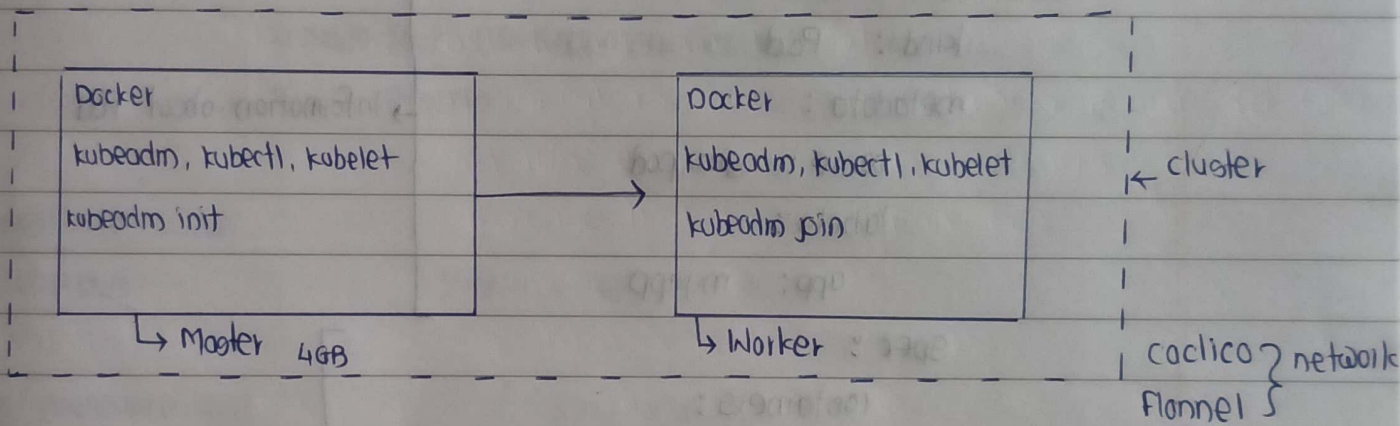
Q) In one pod, can we stored multiple containers?
→ Yes, this will not microservice architecture

```
apiVersion : apps/v1          → kubernetes files (manifest file)
kind       : Deployment       → kubernetes object
Metadata :                    → information about deployment
    name : nginxdeployment
    labels :
        apps: nginx           → To finding deployment
  spec :
    replicas : 2              → two create pod
    selectors :
        MatchLabels :
            app : nginx       → to find pod
    template :
        metadata :
            labels :
                app : nginx   → To finding pod
    Spec :
      containers :
        - name : nginx        → container (may be more)
          image : nginx
          ports :
            - containerPort : 80
```

.yml file

```
apiVersion :
kind : service
metadata :
    name : myservice
    labels :
        app : nginx
  Spec :
    selector :
        app : nginx           → mapped to pod
  Ports :
    - protocol : TCP      targetport : 80
```

Day-19 kubernetes Installation.



if worker is crashed then used 1) Autoscaling
2) lambda
3) Shell scripting Aws cli

If master node crashed → used two master node another is u called slave.

launch two instances
↳ 1) masternode } Ubuntu (masternode - t2.small)
2) workernode }

copy token from masternode when create. worker node →
installing calcicos ↳remove / join single line sudo Paste line join (single line)
kubectl get nodes
Adding 6443 Port no on both node.
kubectl get nodes (after worker node)
kubectl get pods

Day-20 kubernetes (pod creation)

start master and worker node.

```
kubectl get nodes          # to seeing nodes
kubectl get pods -n kube-system    # to seeing pods in master
mkdir nginxpod
cd nginxpod
nano nginxpod.yml
```

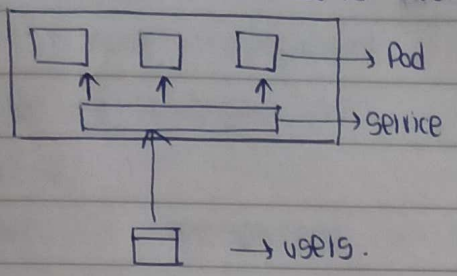
```
apiVersion: v1
kind: Pod
metadata:                          → Infomation about Pod
  name: nginxpod
  labels:
    app: myapp
spec:
  containers:                      → Infomation about containers
    - name: nginxcontainer
      image: nginx
      ports:
        - containerPort: 80
```

```
kubectl apply -f nginxpod.yml
kubectl get pods

sudo crictl ps     # to seeing containers
kubectl get pods -o wide  # infomation about pod
                    pod ip   node ip
kubectl describe pod podname    # same as describe command in docker
kubectl logs pod_name           # Add 10250 port no in worker node
kubectl exec -it nginxpod -- /bin/bash    # entering in contain pod
ls
exit
```

Day-21 kubernetes (NodePort, LoadBalancer)

To seeing website to outside world used ~~serve~~ service. website from browser.



Type of service ⇒ 1) cluster IP
2) Node Port (Debugging or Testing)
3) LoadBalancer

```
kubectl get nodes
cd nginxpod
ls
nano myservice.yml
        apiversion: v1
        kind: Service
        metadata:
            name: ~~myservice~~ mynodeportservice
        spec:
            type: ~~Nodeport~~ NodePort
            selector:
                app: myapp
            ports:
              - protocol: TCP
                port: 80
                targetPort: 80
                nodePort: 30001
kubectl apply -f myservice.yml
kubectl get services
kubectl get pods
```

Add 30001 port no. in worker node.

copy IP and check.

kubectl exec -it nginxpod -- /bin/bash

cd /usr/share/nginx/html/

ls

touch ravi.html

echo "ravi " > ravi.html

cd nginxpod/

#for loadbalancing

cp myservice.yml loadbalservice.yml

nano loadbalservice.yml

↳ kind: ~~Load~~ service

metadata:

name: loadbalservice

spec:

type: LoadBalancer

#remove nodePort

kubectl apply -f loadbalservice.yml

kubectl get services

load balancers aws

↳ create ALB

Internet-facing

listener

create Target group

TGP

http loadservice Port (31284 something)

select worker node

include as pending below

create

create load balancer

sg of load balancer - Add 80

Add Port no in worker node 30000 - 32000

copy DNS of LB and check.

kubectl get all # for all services

cp nginxpod.yml newnginxpod.yml

nano newnginxpod.yml

↳ name: newnginxpod

kubectl apply -f newnginx.yml

kubegtl get all

kubectl exec -it newnginxpod -- /bin/bash

cd /usr/share/nginx/html

rm index.html

~~nano~~ touch index.html

echo "naya hai" > index.html

check !

Delete load balancer.

Stop ec2

Day-22 kubernetes (ReplicationController, Replica set)

disadvantage of Pod - they do not create automatically if crashed.

ReplicationController

↳ 1) Pod recreation (self healing)

2) Scaling (manual scaling)

3) Replicas

Protocol

start nodes

kubectl get nodes

kubectl get pod

kubectl get services

kubectl delete all --all # To remove all

kubectl get all

kubectl logs pod/newnginxpod

first delete services and then delete pod

```
kubectl delete pod newnginxpod --force    # forcefully delete pod
kubectl get pod
kubectl get all
mkdir replicationcontrollerwala
cd replicationcontrollerwala
nano mynginx.yml
```

```
apiVersion : v1
kind : ReplicationController                    → Pod related
metadata :
    name: myrc
spec :                                          → How many pod
    replicas : 3                                  label selector
    selector :
        env: dev
    template :                                  → label defined
        metadata:
            labels :
                env: dev
        spec :                                  → containers
            containers:
                - name : nginxcontroller
                  image : nginx
                  ports:
                      - containerPort : 80
```

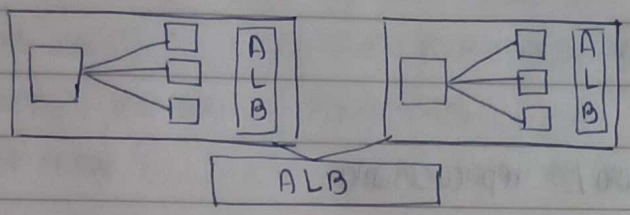
```
ls
kubectl apply -f mynginx.yml
kubectl get rc          # To get list of replicationcontroller
kubectl get pods
kubectl delete pod firstpod --force
kubectl get pod
kubectl scale rc myrc -- replicas = 4
kubectl get rc
```

kubectl get pod --watch # To seeing pod in live

kubectl delete all --all --force

Senario- two master node with diff cluster, can LB send request to workrer node.

Yes.



cd replicationcontrollerdola

ls

kubectl apply -f mynginx.yml

kubectl get replicationcontrollers

kubectl describe rc myrc

kubectl get rc -o wide

kubectl get pod -l -o wide

ls

cp myngnix.yml youginx.yml

cp myngnix.yml ourinx.yml

nano youringinx.yml

↳ name: myrc1

selector:

env: prod

nano ourinx.yml

↳ name: myrc2

selector:

env: fb testing

kubectl apply -f youringinx.yml

kubectl apply -f ourinx.yml

kubectl get rc -o wide

kubectl get pod -l env=dev } # seeing pods with help of key & value

kubectl get pod -l env=prod }

kubectl get pod -l env # seeing pods with help of key

Replication Controller

↳ Problem we do not see double pods with key

↳ Solution Replicas set

kubectl delete all --all --force

3] Replica set -

cp -r replicationcontroller/ replicaset/

cd replicaset/

nano mynginx.yml

```
apiVersion: apps/v1
kind: Replicaset
metadata:
    name: myrc
```

Same changes in youringinx.yml & ouringinx.yml

kubectl apply -f mynginx.yml

kubectl apply -f youringinx.yml

kubectl apply -f ouringinx.yml

kubectl get rs # to replicas set

kubectl get rs -o wide

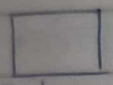
kubectl get pods -l env

kubectl get pods -l env=dev

~~kubectl get pods -l env in {dev, testing}~~

kubectl get pods --selector 'env in (dev, testing)'

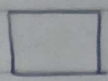
Day-23 kubernetes (Deployment (updates))



Pod
↳ disadvantage
1) do not recreate
2) not multiple pod with same name

ReplicationController
↳ replicas
pod recreate, manual scaling
disadvantage
↳ we do not see same label pod

Replicaset
↳ see same label pod
disadvantage
↳ new version by manually not rolling update



Deployment
↳ Perfect for app service

Deployment :-

- deployment is internally replicaset, internally replicationcontroller, internally Pod.
- deployment solved problem of - rolling updates,
  - Give batch wise update
  - Rollback functionality

```
mkdir deploymentdemo
nano mynginx.yml
    ↳ apiVersion: apps/v1
      kind: Deployment
      metadata:
          name: mydeployment
      spec:
          replicas: 2
          selector:
              matchLabels:
                  env: dev
kubectl apply -f mynginx.yml
kubectl get deployment
kubectl get all
```

for update

↳ 1) by command

2) change in yml file (change image name)

kubectl set image deployment mydeployment nginxcontainer = httpd

↳ update through command

kubectl rollout status deployment mydeployment

↳ update of status

kubectl get deployment mydeployment -o yaml

↳ kubernetes yml file (update with our new updates)

kubectl apply -f mynginx.yml

↳ 3rd update

kubectl get deployment mydeployment -o yaml

Revision

1 - nginx

2 - httpd

3 - nginx

4 - httpd also possible

kubectly rollout history deployment mydeployment

↳ history of updates

kubectl rollout undo deploy mydeployment --to-revision=2

↳ Rollback

kubectl rollout history deploy mydeployment

kubectl annotate deployments.apps mydeployment kubernetes.io/change-cause

= "version4"

↳ Putting names to updates (only applicable for previous update)

kubectl set image deploy/mydeployment nginxcontainer = php

kubectl annotate deployments.apps mydeployment kubernetes.io/change-cause = "version5"

kubectl rollout history deploy mydeployment

```
spec:
  replicas: 2
  strategy:    type: RollingUpdate
    RollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
```

maxSurge → maxsurge means maximum this amount of pod are created other than desired capacity, then if 1 is created it replace that live pod
अर्थात् Pod तयार होताना मग replace होतात.

maxUnavailable → If we get 1, then 1 pod will delete in live and then 1 pod create replacing that pod.
अर्थात् Pod delete होताना मग त्याजाग्यावर दुसरा Pod तयार होतात.

```
kubectl apply -f mynginx.yml
kubectl rollout history deploy mydeployment
kubectl get deploy --watch
kubectl get deploy/mydeployment --watch
```

Configure Pod and Containers :-

↳ 1) liveness (Pod is live or not) (Pod crashed or not)
2) Readiness Probes (Pod is handled request)

```
under spec containers:
        livenessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 10
          periodSeconds: 5
```

kubectl delete all --all --force

```
kubectl apply -f mynginx.yml
kubectl get deploy mydeployment -o wide
kubectl get pods                    # Test delete index.html page from any pod
kubectl exec -it pod_name -- /bin/bash
cd /usr/share/nginx/html ; rm index.html
# check pod is create or not
```

Day-24 kubernetes (health Probe, namespace)

(IQ) health Probe ⇒

↳ 1) liveness (Pod restart)

2) ReadinessProbe (Pod is standby)

3) startup Probe

1) liveness Probe means if request is not handled then terminated that pod and create new pod (restart)

2) ReadinessProbe means if request is not handled by such pod then service is not request to such pod, after some if this pod will healthy then again service will send request to such pod.

3) Startup Probe means it used to determine whether on application within container has sucessfully started or not.

3 ways to health check ⇒ 1) httpGet (most used)

2) tcpsocket

3) command

socket is used to real time data (for messaging)

↳ ex. whatsapp

http ⇒ request-response, Pull technology (connect — request - response - disconnect)

tcp ⇒ send/receive, Push technology (socket)

ReadinessProbe →

mkd mkdir healthProbe

nano livenessProbe.yml

nano readinessprobe.yml

↳ under containers:

livenessProbe:

httpGet:

path: /index.html

port: 80

```
        intialDelaySeconds: 10
        periodSeconds : 5
        failureThreshold : 3
      readinessProbe:
        httpGet:
          path: /index.html
          port: 80
        intialDelaySeconds: 10
        periodSeconds: 5
        failureThreshold: 3
kubectl delete all --all --force
kubectl apply -f readinessprobe.yml
kubectl get all
kubectl get pods
kubect exec -it anypodname -- /bin/bash
cd /usr/share/nginx/html
rm index.html
exit
kubectl get pods --watch        # check
kubectl get deploy -o wide
kubectl exec -it anypodname -- /bin/bash
cd /usr/share/nginx/html
touch index.html
echo "Bye" > index.html
exit
kubectl get pods --watch
kubectl get deploy                  # ready all pods

StartupProbe →
    Add after readinessProble      startupProbe:
nano startupprobe.yml                exec:
                                       command:
                                         - cat
                                         - /usr/share/nginx/html/index.html
```

- initialDelaySeconds: 10
- periodSeconds: 5

kubectl apply -f startupprobe.yml
kubectl get deploy mydeploy -o wide
kubectly get pod --watch

Namespace ⇒

- Application wise Grouping in label app: fb
env: dev

In small wise grouping used Label Namespce is used for Grouping.
In bigger size grouping we used Namespace.

facebook
env: dev
env: test
env: prod

insta
env: dev
env: test
env: prod

Namespace used for 1) isolation
↳ 2) Resource management
3) Organization

Day-25 kubernetes (namespace, volume)

kubectl get namespace # All namespace
kubectl get pods -n kube-system # seeing specific namespace pods
or kubectl get pods --namespace kube-system
kubectl create namespace mynamespace # creating namespace with command
kubectl describe namespace mynamespace
mkdir namespace
nano namespace.yml
kubectl apply -f namespace.yml -n mynamespace # Applying with namespace
kubectl get pod -n namespace mynamespace # seeing pod under namespace
kubectl config set-context --current --namespace mynamespace
mynamespace or default namespace change

kubectl get pod

kubectl config set-context --current --namespace default

Put again namespace as default

kubectl get pods

kubectl get pods -n ~~ner~~ mynamespace

kubectl delete pods -n mynamespace

kubectl delete namespace mynamespace # Deleting namespace

Namespace with the help of yml file.

nano customnamespace.yml

↳ apiVersion: v1

kind: Namespace

metadata:

name: aoplanamespace

kubectl apply -f customnamespace.yml

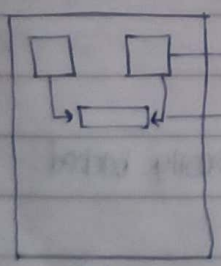
kubectl get namespace

If we delete namespace forcefully then some pod are remaining without no namespace this pod are called "orphan pod"

Volumes ⇒

1) emptyDir
2) hostPath
3) nfs (used)
4) persistent volume and persistent volume chain (PV and PVC) (most used)
5) configmap
6) secreate

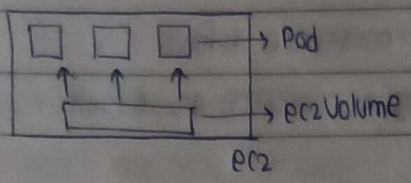
emptyDir ⇒



two containers are used same volume

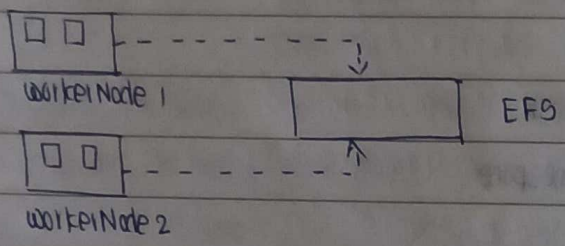
disadvantage- if pod are crashed, volume are also delete.

2) hostpath -

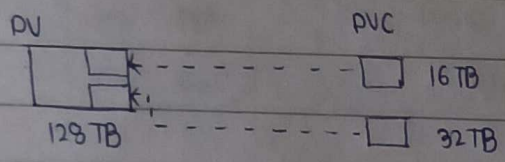


disadvantage - They do not share volume for another worker node.

3) NFS - (Network file system)



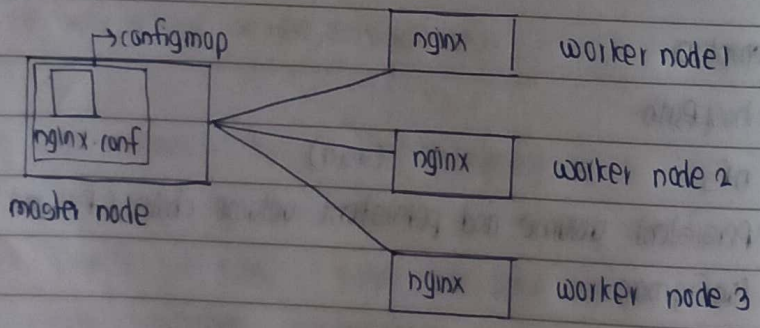
4) PV and PVC -



PV handled by Admin.

They allocated volume to us.

5) Configmap -



In Real time if we need to change configuration file code / database password or any data which applies to all pods which connected to master node, then we used configmap.

configmap used key value pair.

disadvantage - They stored key value in plain text - easily hacked.

6) secrete -

same as configmap but diff is they stored with encoding

```
mkdir volume
cd volume
mkdir emptydir.yaml
nano empty.yml
```

↳

```
spec:
  containers:
    - name: nginxcontainer
      image: nginx
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: myvolume
    - name: httpdcontainer
      image: httpd
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: "/usr/local/apache2/htdocs
          name: myvolume
  volumes:
    - name: myvolume
      emptyDir: {}
```

```
kubectl apply -f empty.yml
kubectl get pods
```

2) HostPath -

mkdir hostpath

nano hostpath-file.yml

↳

```
spec:
    containers:
        - name: nginxcontainer
          image: nginx
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: "/usr/share/nginx/html"
              name: myvolume
    volumes:
      - name: myvolume
        hostPath:
            path: /data
            type: DirectoryOrCreate
```

kubectl apply -f hostpath-file.yml

kubectl get pods

check volume in worker node.

3) NFS -

create efs

↳ create file system myefs

copy command using NFS client

copy efs security ID and paste to

Paste in security ID search box — enter — inbound rule

~~custom TCP~~ NFS - Anywhere 0000 save.

mkdir NFS cd NFS

mkdir efswala

NFS 9 -

1) EFS
   ↳ create file system
      myefs
         ↳ create
      Attach
      copy NFS client command
2) worker node
      ↳ mkdir myvolume

   efs ⇒ security group
   copy worker nod security group
   goes

   f170-ec2
      ↳ worker node

3) efs security group मध्ये worker node चा security group टाका
      NFS custom 2049 Paste worker node sg → save.
   copy using nfs client command
   Paste in worker node at last remove efs → add ~/myvolume
   sudo apt-get update
   sudo apt-get install nfs-common
   Paste in here command
4) master node
      ↳ nano nfsfile.yml
            ↳ volumes:
                - name: myvolume
                  nfs:
                    server: DNS name
                    path: /test
5) worker node
      ↳ cd myvolume
         ↳ mkdir test
           sudo mkdir test
           sudo nano index.html
                 ↳ <h1> </h1>
6) master node
      ↳ kubectl delete all --all --force
        kubectl apply -f nfsfile.yml
        kubectl get pods
        kubectl describe rs mypod
        kubectl exec -it mypod-2568b -- /bin/bash
        cd ~~bin~~ /usr/share/nginx/html
        cat index.html

pv and pvc with efs

1) create efs
   ↳ my-efs create.
2) master node
   ↳ nano pvfile.yml
      ↳ nfs:
         server: efs-DNS
         path : /test
   efs चा security group आणि worker node चा security group id.
3) worker-node
   ↳ ls
   mkdir mydir
   paste command using nfs client at last remove efs ~/mydir/
   sudo apt-get update
   sudo apt-get install nfs-common
   cd mydir
   sudo mkdir test
   cd test
   sudo nano index.html
      <h1> </h1>
4) master node
   ↳ kubectl apply -f pvfile.yml
      nano pvcfile.yml
      kubectl apply -f pvcfile.yml
      kubectl get pv
      kubectl get pvc
      nano pod.yml
      kubectl apply -f pod.yml
      kubectl get pods
      kubectl exec -it pod-name -- /bin/bash
      cd /usr/share/nginx/html
      cat index.html

sudo apt-get install nfs

sudo apt-get install nfs-common

Paste command which copy from efs - using nfs client.

mkdir nfsfile

nano nfsfile.yml

↳

```
volumes:
  - name: myvolume
    nfs:
      sever: efs-dns-paste
      path: /nfsfile
```

kubectl apply -f nfsfile.yml

kubectl get pods

Create dockerfile in which installed nfs, push in dockerHub, use this image in nfs volume.

Day-26 kubernetes (configmap, secrete)

configmap:

data of configmap stored in master node.

configmap stored key value

configmap creation

↳ 1) by command

2) by file

mkdir configmap

nano mysqlweb.yml

↳

```
apiVersion:
kind: pod
metadata: mysqlweb
spec:
  containers:
    - name: mysqlcontainer
      image: mysql:8.0
      env:
        - name: MYSQL-ROOT-PASSWORD
```

```
valueFrom:
    configMapKeyRef:
        name: mysql-config
        key: mysql-password
```

kubectl create configmap mysql-config --from-literal mysql-password = Pass@123
kubectl get configmap
kubectl get configmap mysql-config
kubectl apply -f mysqlpod.yml
kubectl get pods
by file
kubectl create configmap configmap2 --from-file default.conf
kubectl get configmap
nano default.conf
nano nginxconfig.yml

```
apiVersion: v1
kind: Pod
metadata:
    name: nginxpod
spec:
    containers:
        - name: nginx container
          image: nginx:latest
          volume mounts:
              - name: nginxconfigvolume
                mouthPath: /etc/nginx/conf.d/default.conf
                subPath: default.conf
    volumes:
        - name: nginxconfigvolume
          configMap:
              name: configmap2
```

secrete :

↳ 1) generic (password stored)

2) docker config (username & password from dockerhub)

3) TLS

↳ 1) crt (certificate)] SSL certificate

2) key (private key)]

1) Generic :

kubectl create secrete generic mygenericsecrete --from-literal mysql-password = Pass@123

kubectl get secrete

kubectl describe secrete mysecrete -o yaml

nano genericsecrete.yml

↳

```
apiVersion: v1
kind: Pod
metadata:
    name: secretewala
spec:
   containers:
      - name: mygenicsecretecontainer
        image: mysql:8.0
        env:
           - name: MYSQL_ROOT_PASSWORD
             valueFrom:
                secreteKeyRef:
                   name: mygenericsecrete
                   key: mysql-password
```

kubectl apply -f genericsecrete.yml

kubectl get pods.

kubectl secrete mygenericsecrete -o yaml

2) docker config :-

kubectl create secrete docker-registry mydockersecrete --docker-username= devopsbaby
--docker-password= personal access token --docker-server = dockerhub.com
-- docker-email = email

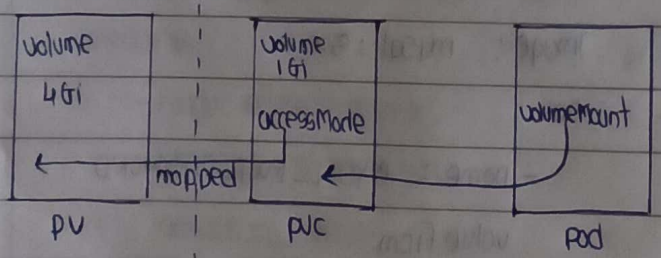
yml file

```
spec:
  containers:
    - name:
      img:
  imagePullSecrete:
      mydockersecrete
```

kubectl get secretes
kubectl describe secrete mydockersecrete -o yaml

Day-27 kubernetes. (PV & PVC , Daemonset, Job)

pv - persistent volume
pvc - persistent volume claim

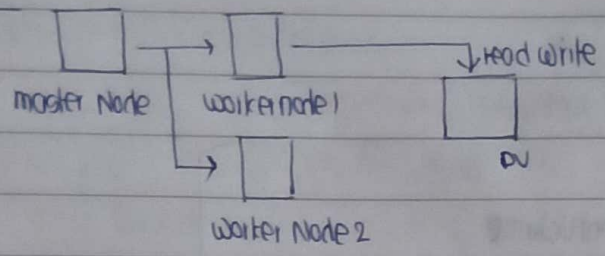


handled by someone
another.

Access Mode

1) ~~Read only~~ ReadWriteOnce
2) ReadOnlyMany
3) ReadWriteMany

1) Read Write once :



In readwriteonce, at a time only one worker node can perform read, write operation on persistent volume.

2) Read only Many :

In read only many all worker node can read at a time

3) Read Write Many :

In read write many all worker node can access to read and write to persistent volume

one PV can mapped with many PVC.

Q How mapped PVC to PV ?

↳ 1) both have same access mode

2) PVC volume is needed, this need is fullfill by PV

means if PVC wants 1 Gi storage, then PV have equal to qater than 1 Gi storage.

If PVC deleted then data from PV

↳ Reclaim Policy

↳ 1) Retain

2) delete.

Storage :

↳

EBS	EFS	S3	RDS
OS software	source code	image,	- user Generated Text
	EC2	video	content
		- user content Data	
- Application Content Data upc		- use in internet - Global	

mkdir pvpvc1a

cd pvpvc1a/

nano pvfile.yml

```
apiVersion: v1
kind: PersistentVolume
metadata:
    name: mypv
spec:
  capacity:
    storage: 1Gi
  accessMode:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  hostPath:
      path: /mydata
```

nano pvcfile.yml

```
kind: PersistentVolumeClaim
metadata:
    name: mypvc
spec:
  accessMode:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

nano nginxpod.yml

```
apiVersion: v1
kind: Pod
metadata:
    name: nginxpod
spec:
  containers:
      - name: mynginxcontainer
```

```
    image: nginx
    volumeMounts:
      - mountPath: /usr/share/nginx/html
        name: myvol
  volumes:
    - name: myvol
      persistentVolumeClaim:
        claimName: mypvc
```

kubectl apply -f pvcvolo.yml

kubectl apply -f pvcfile.yml

kubectl apply -f nginxpod.yml

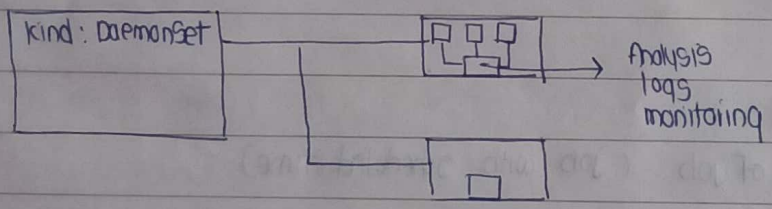
kubectl get pv

kubectl get pvc

kubectl get pods

kubectl describe pod nginxpod

Daemonset :



taint

tolerations

nano daemonsetfile.yml

```
apiVersion: apps/v1
kind: Daemonset
metadata:
  name: daemonvolo
  labels:
    app: promi
spec:
  selector:
    matchLabels:
      app: promi
```

```
template:
  metadata:
    labels:
      app: prom
  spec:
    containers:
      - name: daemonwala
        image: influxdb
        ports:
          - containerPort: 80
```

kubectl get daemonset

kubectl get ds

kubectl apply -f daemonsetfile.yaml

job :

↳ type of pod.

- specific work.
- backoflimit : 4 (4 times restart)

cronjob -

↳ scheduling of job (job with scheduled time)

* * * * * → (cron Expression)

↳ min (0-59) ↳ hour (0-23) ↳ day of month (1-31)

↳ month (1-31)

↳ day of week (0-6) like sunday monday

ex. 0 3 * * 1

↳ monday

↳ all month

↳ all day of month

↳ 3 AM

↳ 0 min

3 AM monday 3 AM all month

nano jobwala.yml

```
apiVersion: batch/v1
kind: Job
metadata:
    name: hellojob
spec:
  template:
    metadata:
      labels:
        app: hello
    spec:
      containers:
        - name: hello
          image: busybox
          command: ["echo", "hello, k8s"]
      restartPolicy : Never
  backoffLimit: 4
```

kubectl apply -f jobwala.yml

kubectl get jobs

kubectl describe pod jobwala.yml

Day - 28 Kubernetes (CronJob, Job, StatefulSet Pod)

CronJob :-

nano cronjob-file.yml

↳ apiVersion : batch/v1

In worker node

↳ df -h

free -m

sudo sh -c 'echo 1 > /proc/sys/vm/drop_caches'

↳ To removing buffer cache.

nano cronjobfile.yml

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hellojob
spec:
  schedule: "*/2 * * * *"
  jobTemplate:
    spec:
      template:
        metadata:
          labels:
            app: hello
        spec:
          containers:
            - name: hello
              image: busybox
              command:
                - echo
                - "hello, k8s"
          restartPolicy: Never
```

kubectl get apply -f ↓ cronjobfile.yml

kubectl get cron-job

kubectl get jobs

kubectl get pods

kubectl delete cronjob hellojob

When only certain Amount of job complete.

↳ spec:

completions: 5

parallelism: 2

kubectl apply -f jobfile.yml

To modify (increase volume) of EBS

modify volume

sudo growpart /dev/xvda 1

sudo resize2fs /dev/root

df -h

Statefulset Pod :-

mysql replicas: 3

↳ created 3 pod with different data

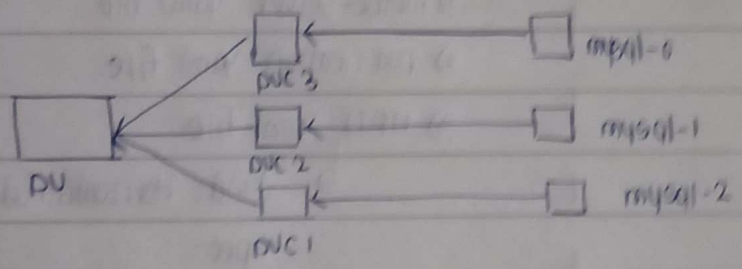
naming

mysql-0 —— volume 1

mysql-1 —— volume 2 } used different volume

mysql-2 —— volume 3

whereas in deployment used multiple pod used common volume.



mkdir statefulset

cd statefulset

df -h

nano pvfile.yaml

Day-29 kubernetes (Horizontal Autoscaler)

Autoscaling :

↳ Automatically increase size of pod when traffic or ram usage is increase.

HPA - horizontal Pod Autoscaler.

Aws Autoscaling	kubernetes Autoscaling
↳ ec2	↳ Pod
cloudwatch	metrics-server / datadog / Prometheus / Graphana
t2.micro 1 RAM 1 CPU → 75%	Pod yaml file → resources:
	requests:
	cpu: 100m
	limit: 500 → when 75%
	↓
	metrics-server
	1) metric-server yaml file
	2) pod creation yaml file
	3) HPA yaml file
	↳ kind: HorizontalPodAutoscaler
	spec:
	minReplica: 1
	max Replica: 10
	utilization:
	Threshold: 75%

```
mkdir autoscaling
cd autoscaling
nano components.yml
kubectl apply -f components.yml
kubectl get pods -n kube-system
kubectl edit deploy metrics-server -n kube-system
                    ↳ To editing any service.
kubectl get pods -n kube-system
```

nano mynginx.yml

↳ resources:

requests:

cpu: "100m"

limits:

cpu: "500m"

kubectl apply -f mynginx.yml

kubectl get pods

nano hpafile.yml

kubectl apply -f hpafile.yml

kubectl get hpa

kubectl get pods

↳ Apply here Autoscaling Test

Test by applying load in Pod

kubectl run it trafficpod --image=busybox -- bin/sh -c "while true; do wget -q -O - http://onepodip; done"

kubectl get pods -o wide

↳ copy ip from here.

check pod will increase or not

kubectl get pods