Kubernetes K8s

- Kubernetes is an orchestation engine and opensource platform for managing containerized applications.
- Kubernetes do container deployment, scaling & descaling of containers & container load balancing

Q. so Kubernetes is a replacment of docker?

Ans. No Kubernetes not a replacment of docker but we can consider Kubernetes is replacment for docker swarm

Kubernetes is significantly more complex than swarm and required more work to deploy. and it vers strong as compare to docker swarm clusters.

- Kubernetes Born in google, written in Go-language and then Donated to CNCF (cloud native computing foundation) in 2014
- Befor Kubernetes available for market or other It resorces Kuberneties internaly used by google

Q what version of Docker you ar using

Ans

Q what version of Kubernetes you are using.

Ans

- when ever in interview thy ask about version dont have to tell latest version atleast Previous two of the current version
- Kubernetes v1.0 was released in July 21, 2015

Kubernetes Features

Automated Scheduling

Kubernetes provide advanced scheduler to launch container on cluster node base on there resorce requirment.

self healing capablities

Kubernetes are capable for rescheduling replacing and restarting the containers which are died.

automated rollout and roolbacks

- rollout in kubernetes refer to the process of updating a deployment of a new version of configration, you might want to roll out a new version of your application to introduce new feature, bugs fix or configration chunges
- rollbacks reverting a deployment to a previous state.

If new version of your application has issue or you have to revert to a stable state.

Horizontal Scaling & load Balancing

Kubernets can scale up and scale down the appli-cation as per the requirment with simple comma-nd, using gui or automaticly based on cpu usage.

storage Orchestation

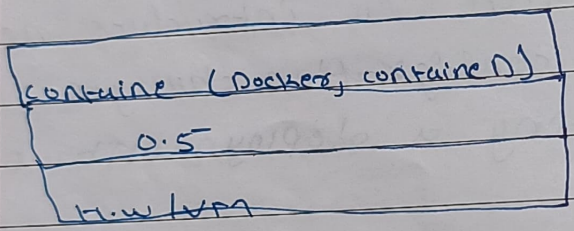
with kubernets you can mount the storage system of your choice. you can either opt for local storage or choose a public cloud provider such as GCP or aws (EBC, EFS, NFS)

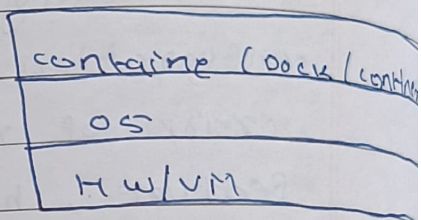
Kubernetes Architecture

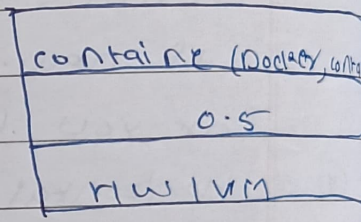
Kubernetes implements a cluster computing background. everything work from inside a kubernetes cluster

The cluster hosted by one node acting as 'master' of the cluster and other node as a worker node which do actual containerization.

three master is recommended at least for high availability & fault tolerance







Q if I make image built by docker by using Docker so can it image compatible with other container like container-D or CRIO like that

Ans yes, there is one standerd define OCI (open container initiative)

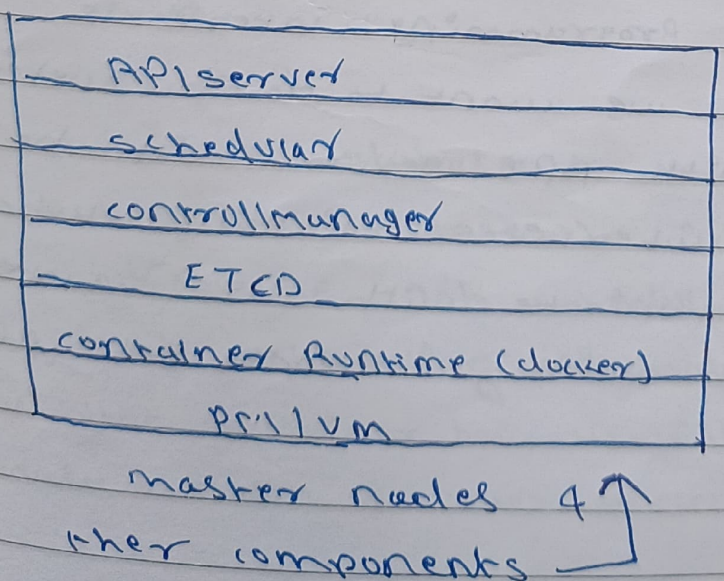
Docker internally using container-D
kubernets suppor for docker via dockershim
kubernets not direcly suppor docker it need one more software called dockershim so it's a lots of overhead to kubernetes

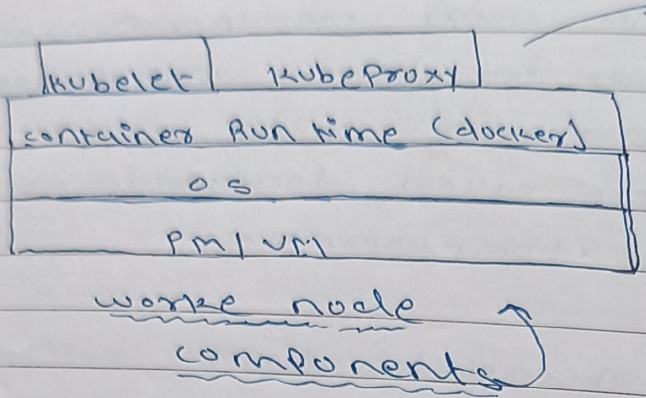
so why not kubernetets direcly support container-D becoz docker internally use container-D and to support docker kubernetes need dockershim its lots of overhead for kubernetes just to suppor docker. so thats why they deprecated.

kubernetes is deprecating docker as a container runtime after V1.20
depercation means you can use it but they dont give officealy support it.

But Docker is still Powerfull becoz container-D is not having very good cli & api to manage images.

control plan or master :-
both are same so some are the components which are the parts of master nodes. which additionally required.





→ It's both also part of master but when we explain architecture we count it in worker

@ Master Node

- The master node is responsible for management of kubernets cluster.
- It is mainly the entry point for all administrative tasks.
- It handles the orchestration of worker nodes.

→ Pods :-

- Pods is the collection of one or more containers if you want to deploy container we have to create Pod we do not directly deploy container. Pod represents running process in k8s, Pod will have one or more contair Pod alway run on nods.

components of master

① → API server

(Application Programming interface)

what ever we want to do in cluster we have to contact with API server with help of kubctl (kubernetes cli using kubctl with that we can communicate with k8s cluster) like kubctl get nodes, kubctl get pods.

Q How kubectl communicate with your cluster when you fier command on kubctl what process run on background

Ans kubectl communicate with kubernetes API server to interact with server

1) configration File (~/.kube/config):
- The kube/config file contain information about cluster user & context
- It include details like cluster API server address user credential and namespace.
- file path is ~/.kube/config

2) API server interaction:
- when you run a kubectl command it read configration from specified or default kube/config file
- It get information about cluster user determine which cluster it need to communicate with.
- when it get request from kubctl so kubernetes authenticate the user & outherise it and then validate the request and then process
- if you dont have the authority or do not authenticate you request get fail.

What is ETCD ?

It is kind of database to store all cluster datas of kubernetes, it store information of kubernetes cluster like job scheduling information about pods, job scheduling information.
Kis Pod me kitne container hai, us ki IP kya hai
Store metadata & status of cluster. ② everything store in key value Pair

• if we do not specify where over pods have to be scheduled so a shedular watches from newly created pods that no pods assigned to to the nodes so shedular creat pods to those nodes
• schedular become responsible for finding best node for that pod for run on

Now then after schedular come in picture

→ Scheduler :- scheduler is responsible for deciding in which nodes pods will scheduled in the cluster

know we know kubelet which is agent software which present on every nodes

so scheduler communicate with kubelet (by API server) & identity that it's node have enough memory or CPU so I can scheduled to this node

control manager :-

- if K8s on cloud so it will be cloud-control-Manager
- if K8s on non-cloud so it will be kube-controll-manager.

control manager
Atual = desired

components of control Manager

① Node-controller
② Route-controller
③ service-controller
④ volume-controller

→ kube-proxy (present in every node)

- kube-proxy enable the kubernetes service by maintaining network rule on host.
- kube-proxy maintain network rule on nodes this network rule allow network communication to your pods from inside or outside of your cluster
① Assighn IP address to pods ② give unique ip to each pods

→ kubelet (present on every node) (static pods also managed by kubelet)

health check of ~~node~~ pods and send heart beat to the master ① Agent running on the node ② work on ~~Pods~~ Port no 1025 ③ send success / fail report to the master

self manage K8s cluster / Bare Metal K8s cluster
minikube --> single Node K8s cluster (for POC)
kubeadm --> using kubeadm we can provision multinode K8s cluster
kubespary --> It is ansible way of configuring multinode K8s clusters.

Managed K8s cluster
EKS - Elastic Kubernets service (AWS cloud)
AKS - Azure Kubernetes service (Azure cloud)
GKE - Google Kubernetes Engine (GCP cloud)
IKE - IBM Kubernetes Engine (IBM cloud)

when ever client required there application on Baremetal instead of any cloud due to security reason that time we use selfmanaged / Bare metal K8s cluster

As per cloud there is best practice that the worker node should be in diffrent availability zone.

Kubernetes are applicable to handle every container like docker, container-D or Rocket
Docker have ther managment tool or orchestration tool which is Docker swarm which is competeter of Kubernetes But there is Kubernetes are used beoz of there feachures. docker swarm only managed docker container But Kubernetes are managed ever container like docker, container-D & rocket.
Kubernetes Provide Scalibility
Kubernetes Provide High Availability

- Kubernetes schedules run and manage isolated container which are running on virtual / physical / cloud machines
- All top cloud Provider support Kubernetes.

Ek
~~one~~ Pode me Ek container kte hai but ek se zyado container liy ja sakte hai

Ek se zyado Pod ek container me create kiy ja sakte hai

Cluster → Node → POD → container → applic
can be many Node
can be many POD
can be many containe
can be many applic

- Pods have IP address (container dont have IP addre
Agar Pod kharab hua or we can say they use pod
ya fail hua For any reason IP only which pods IP)
so it can not be recreate uski
ga jaga dusra banega uski
IP alag hogi

etc/kubernetes/admin.conf ← have all kubeconfig information

API server listning on :6443

practicle

after installation

as we know we can install qubectl in any system so install it in your powershell / local matchin

so if you install qubectl in your cluster or local matchin but you can not access your kubernetes cluster. why? becaus you have the qube config in your system where your have qubectl

so copy the content of qube config file from

ε vi /etc/kubernetes/admin.conf ← from master node

and then mkdir ~/.kub ~~config~~ in directory make file vi ~/.kub/config

and past the content in the file

if you want to connect or communicate with your kubernetes (you just want to install kubectl and the kub config file in that serve)

no mater your matchin is from other cluster or other operating system like windows or other account (vpc peering) just required kubctl & kubeconfig file or it may be on your maste, worke, seperat EC2 or any where jus need to install kubctl and kubconfig file

some concepts of k8s :- (kubectl get all -n kube-system --namespace — to see all the resorces or objects)

Pods
service
ReplicationControllers
Replicasets
DaemonSets
Deployments
Statefullset
configMaps
Secrets
storageclass
Peristent Volumes
Namespace

Kubernetes namespace :-

- You can thin tha a Namespace as a virtual cluster inside your Kubernetes cluster (it just a logical grouping)
- You can have multiple namespace inside a single Kubernetes cluster and they all are logically isolated from each other
- So the namespace help you and your tems with organization, security and performance also.

By default we have four namespaces

1. Default :- if we creating any Kubernetes resorces without specifying namespace it created in Default namespaces.

2. Kube-system :- it have all thing related to the Kubernetes system like api server, etcd, Kube-proxy, wave-net, schedular that all Pods in Kub-system namespace.

So Any deployment to this namespace are plying dangerous game and can be cause irreparable damage.

3. Kube-Public :- it is readable by everyone but the namespace is reserved for system usage

4. Kube-node-lease :-

Using namespace for isolation

name space are used for isolation as we know, But there is couple of woye to use it.

It is used for to split projects into seperate environ-ment.

for example one project have multiple teams product team payment team, they are working there own application so we should give seperate name space to the teams so they can deploy application there respective namespaces so it will be logicaly isolated from other

- namespace usefull for preventing cross-project contamination since namespace provide independent environ-ment.
- I can set the security & resorces on namespaces

- kubectl creat namespace test
- kubectl get namespace

for create/update/Delete/Read any k8s resorces we use kubectl (CLI)

we can do it in two ways

- Imperative Approch --> using commands
- Declerative Approch --> we can declare and then create/update/delete in form of manifest files.

Does kubernetes support JSON also?

Yes, But 99% we are use yaml file.

kubectl api-resources
So this command give all api & there versions

Q what is resource quota. How it help us ?
Ans with the communication of production team there leader or architec ~~we~~ that how many Pods you expected or how many resorces you required on the basis of that we set the Quota.

→ K8s APIS / objects / Resorces / workLoads

POD
ReplicationController
ReplicaSet
Daemonset
Deployment
statefullset

- communication
Service
clusterIP
NodePort
LoadBalencer

- Storage / volume
Persistent volume
Persisten Volumeclaim
storage class

- RBAC (Role Buck Access control)
Role
RoleBinding
clusterRole
clusterRole Binding

- Scheduling & maintenence
Node Selector
Node Affinity
POD Affinity / Anfinity
Traints & tolerations
Drain
cardon
uncardon

POD :-

A pod always run on a Node.

Pods is smallest building block or basic unit of deploying in Kubernetes.

In a Kubernetes cluster, a Pod present running process

Inside a Pod, you can have one or more containers

what is common in the container which is running in same Pods?

Those containers all share a uniq network IP, storage, networks and any other specification applied to the Pod.

How containers in a same Pods or application ot there container can communicate ~~which~~ with each other?

So Pods have IP address not container so network is common, so application or container use local host to communicate with each other

what if the application have there diffrent ports?

Lets take example container-1 is configure to use Port 8080 and container-2 configure to use port 9090, each container expose it's services on it's specified Port, so it is possible that they can communicate with each other using local host no matter they have there diffren Pods but thy can communicate with localhost

If we demand to ~~~~be Kubernetes that 1 Pod with two container so is the container are on diffrent nodes?

No, if you want two container in same Pod so both containe are on same nodes.

Each Pod have its unique IP Address within the cluster

Pod model types :

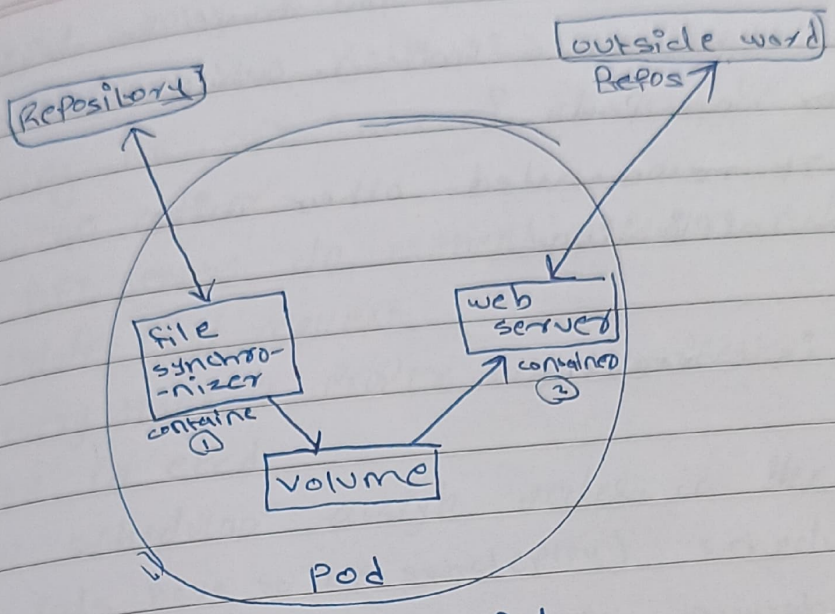
Most often when you deploy Pod to a kubernete cluster it will contain a single container. But there are a instance when you might need to deploy a Pod with multiple containers.

① single - container Pods

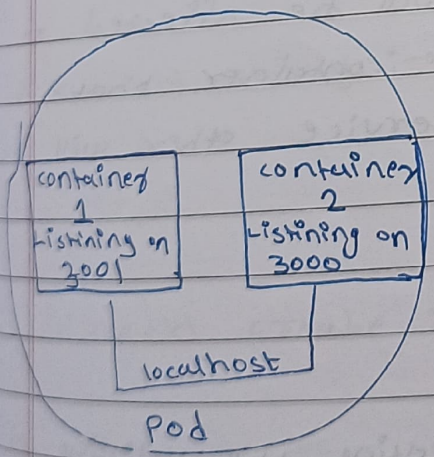
- it contain a single application container
Resorces of Pods such as CPU and memory are managed at container level within the Pod.
- It is the most popular

② Multicontainer Pods

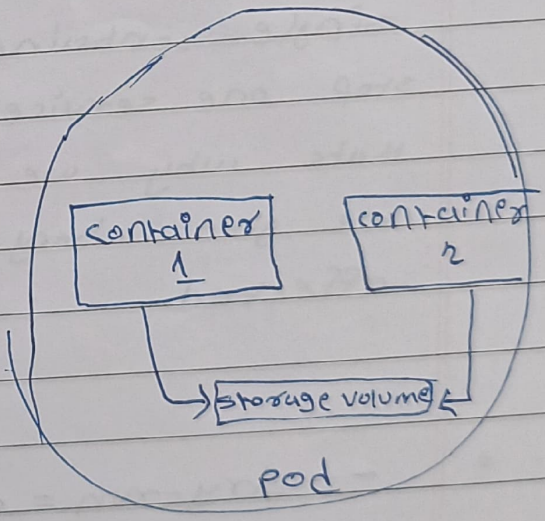
- it contain a multiple containers that collaborate and share the same network namespace & storage volume
- container within the same Pod share the same resorces
- useful for application where multiple Process need to run togethe
it may be one is primery container and another one is utility (helper) container that help or enhance the application function



Multi-container Pod



Shared Network space



Shared storage volume

- At the namespace level we can setup the resource Quta (resorcequta applyed on namespace level)
- or in container container level we defined resources so we set or defined & set resorces on container level

How process happens

kubectl apply -f file.yaml — so what is happining is with the help of manifest file we are requesting to api server to create resorce in my kubernetes cluster
then API server do outhentication and otherization & validate that request
if every thing is proper in manifest it store information in etcd
then schedular assigh nodes to the Pods (which node Pods have to be ~~sch~~deploy) scheduler check any node ~~has~~ have memory & cpu avalable which required ove container it's going to deploy on that node.
and how scheduler get information about node that it have resorces cpu & memory is free or not it communicate with cubelet the agent which running on the node

kubectl apply :-

kubectl create :-

kubectl update :-

kubectl delete :-

Service

A service is responsible for making our pods expose inside the network or expos outside the cluste (internet)

Q how whos service identi identify on which pods it's have to send the trafic? How it will identify?

Ans service identify on the basis of labels and selectors

There is diffirent type of services there

- clusterIP — only within the cluster
- NodePort
- Headless
- loadBalencer (external load Balencer / cloud load Balencer)

Q what is diffrent between docker service & K8s service?

Ans

- clusterIP : when we create service type clusterIP that expose the service on cluster internal IP
  - service reach able only within the cluster
  - this is the default type
  - when we create a service
  - service is just a logical concept the real work is done by the "kube-proxy" Pod that is running on each node

Port :- service Port
Target Port :- container Port

service identify the Pod on basis of label & selector
we have to use it very carefully
beoz when if we create one Pod or by mistake we
give table which is associated with service so it get
the trafic of Pods which associated with service which is
associated on basis of label.
so label of service Pod should be same But when
we create new Pod so label is diffrent if we want
Pods dont have associated with service.

Now object name within same name space should be
unique name : nodejsPod so we can use same name
with diffrent namespace but within same namespace we cant.

→ NodePort : Nodeport range between 3000 to 32767
if we do not specify it take by default between
if you wont access your container from outside of
your cluste you can done by nodepod.

Q what is cluster capacity? How much resorces (cpu/memory)
u have available in the clusters? (all factor are
the depend how many Pods can be deploy)

An we deploy Pods each Pods consume some memory
or RAM so becoz of that ove Kubelet is not
functioning well, so how many Pods we can deploy
based on over resorces, we can deploy

Q How many nodes on your cluster ?
Q what is the capacity of each nodes ?
Q How many namespaces

we can say we have 20 Nodes of cluster
Each Node 128 GB RAM
64 core CPU
800 Pods
↳ everything required on use case

→ Pod lifecycle

- make Pod request to API server using local Pod defination file (manifest file)
- The API server save the info of POD in ETCD
- The schedular find the unschedul job & schedul to the node.
- The entier lifecycle state of the Pod is store in ETCD

we have diffrent diffrent status like Pending, CrashLoopBackUP, Running ImagePullBackoff / ErrorImagePull, createconfig-Error.

Pending → may be due to insufficient resorces
crashLoopBackUp → it due to cmd issue or application issue so kubernets try to multiple restart to up the container so we have to communicate with developer team (Pod scheduled BUt container not started) need to redeploy

- Kubectl describe Pod <Pod name>
- Kubectl logs <podname> -c <containername>

ImagePullBackOff / ErrorImagePull :- not able to pull image may be it is private image & you are not authenticating, wrong registory details.

CreateContainerConfigError :-

configmap -

secretes -

you are refering configmap & secretes in your pod manifest but you are not having configmap & secretes in your cluster

Pod concepts

Pod is ephemral and wont be rescheduled to a new node once it dies.

You should not create pod directly we have to use diffrent object.

Pod should be crected and manage by some controllers

Replication controller

Replicaset

Daemonset

Deployment

statefullset.

Orchestration is the purpose we a using kubernetes so if we deploy pods directly orchestration does not happen.

Static Pod :- (manage by kubelet not part of k8s control plane)

over etcd, apiserver, controller-manager, & schedular are running as pod so it dont have any controller Replication controller or replicaset or any other controll-er

static pods file store in /etc/kubernetes/manifes
if you want to change static pods path you can go
cd /var/lib/config
cd /var/lib/kubelete
vi config.yml
so that Pod is known as static pod.

- static Pod is directly managed by kubelet
- The kubelet (agent on every node) is responsible for each static Pod and start it if it is fail
- static Pod does not have any associated replica controller, Daemonset, Replication set.
- kublet service itself watches it and restarts it when it ~~create~~ crashes.

The main purpose of static Pod is by using kubelet to supervise the individual control plane components

practicle for example if you delete api server
kubectl delete Pod kub-apiserver -n kube-system
then if you do
kubectl get Pod -n kube-system
so you see it back up & running
so kubelet. get it back.

Q can we create static Pod in any node?
Ans Yes we can
Q I want static Pod on master or in slave how can I create?
Ans yes we can create on slave also.
① go to slave node ② Place your yaml file in /etc/kubernetes/manifest ③ then it is deploy the Pod as static Pod. ④ then check you Pod is running on default as static ⑤ if you delete kubelet restart it

Q method of delete Pods

① ~~by Pod~~ ① by Pod name ② delet. yaml file so Pod get deleted ③ also delete by label name

→ Re request

cpu : 100m → m = milicore 1cup = 1000m

memory: 200Mi → Mi = megabite

when we request the resorces with over Pod then the kubelet checks that over node have the resorces or not, if not then Pods are going to be in Pending state

- If you using Resorce Request concept, it cluster have that resorces so that over Pods are going to schedule if you dont use request concept if kubectl allow to scheduled so over running Pods get affected.

→ Labels and Selectors :-

Labels : when one thin in k8s need to find another thing in k8s it use labels.

eg: labels

app: nginx

role: web

env: dev

Selectors: selector use the label to find object match with same label

eg selector

env = dev

app != dp

ReplicationControllers (rc) namespace = true ;-

- Replication controller is one of the features of Kubernetes, which manage pod lifecycle.
- it is responsible for making sure that the specified number of pod replicas are running at any point of time
- Replication controller is enable you to easily create multiple Pod
- It make sure that the number of Pod always exist
- If Pod does crash, the replication controller replace it.
- Replication controllers and Pods are associated with labels.
- Creating "RC" with count of 1 ensure that a POD is always available.
- I can scale up & scale down number of replicas on basis of my load. at any point of time.

when we creating Replication there is first we defing over replica name & All then in ~~template~~ specification.
then one more specfication which is under templet which is about Pod specification

Replication controller and Pods are in same file they are ~~like~~ like Parent & child.

→ if you delete the replication like
kubectl delete rc <replication controller name> -n <name
so Pods is also deleted which manage by this replication controller
and if you deleted the yaml. file so all the service, replica, Pods all get deleted. what ever in the file

Replica set is the difference selector
in Replication controller - equality base selector
in Replicaset set base selector & one more difference is api version remaining all are same

Q If you give the name : Pod1 but you have replica 3 so what is Pod name is two Pods have same name?
No every Pod have different name if you have replica 3 so Pods name are like your replica name & some alphabet if replication controller name: nodejsreplica
so Pods name like
Pod/nodejsreplica-32509
Pod/nodejsreplica-18902
Pod/nodejsreplica-43210

ReplicaSet :- (rs)

Replicaset is the advanced version of repli-cation controller
Replicaset also manage the Pod life cycle.
It will create and manage pods and we can scale in and scale out the Pod replicas.
Kind: ReplicaSet
apiversion: apps/v1
Replicaset suppor equality-based selector as well as set-based selector
selector, Now see How it use

In Replication controller	In Replical sets
selector name: dev-pod	selector matchedlabels name: dev-pod
	selector match Expressions

some typical use of Daemonset
① Running cluster storage daemon on every node.
② running log collection daemon on every node
③ running a node monitoring daemon on every node.

Daemonset (DS) :-

- A Pod which we deploy through Daemonset so copy of that Pod is deploy on every node
- if ~~Pod~~ node is added in cluster so pod is added on that cluster also
- if you delete daemonset so what ever ~~Pod~~ Pods run by Daemonset all get deleted
- if node delete daemonset pos also delete with node

Example - kube-Proxy, agent base application kube-proxy are also depoly by the Daemo-nSet

number of nodes = number of pod

agent base applications are deployed through Daemonset like log agent, monitoring agents (Prometh

Deployment :- (Rollout trigger when we change image or ports not when im change

In kubernetes, Deployment is the reco-mmended way to deploy Pod or RS, simply because of the advanced features it come with.

There is some key benifits :-

- Deploy a RS
- Update Pods
- Rollback to older Deployment version
- Scale deployment up or down
- ~~Pows~~ Pause and resume the deployment
- use status of the deployment to determine state of replicas.
- clean up older replicasets that you dont need anymore
- It provide zero down time deployment means without down application i can update containers.

in replicaset if i want to update my image nginx:1.3.1 to nginx 1.3.2 so will chang in manifest & then delete the pod & redeploy so it get downtime which is solve or which is not in deployment

In Kubernetes Deployment have the two type of strategies

① Recreate (downtime)
② RollingUpdate (default) No downtime

- Recreate: (delete old the after deploy new)
  In this type of very simple deployment, all the ~~old~~ old Pods are killed at once and get replaced all at once with the new ones. and we can roll back. we have the pods and applications running in containers you want to change or update your pods specification by using Recreate deployment stratigies
  what happen in that one you apply recreate stratigies it first ~~delete~~ delete old pod and then create new. in between creat or delete over application face downtime

- Rolling deployment:
  It is the default deployment strategy in Kubernetes, It work slowly (it get settle down the new pod wait 30 to 40sec & then after terminating old one) one by one replacing Pod of privious version of replication with Pods of new version.
  without any cluster downtime
  It create new pods first and wait to settle down the pod then old pods traffic rout to the new pods then it going to terminate or delete the old pods so there is zero downtime in Rolling update deploymen.

→ what is maxsurge & maxUnavailable in Deployment strategies?
In Rollingupdate deployment there are three components are there
1) max Unavailable 2) maxsurge 3) minReadyseconds

kubectl rollout undo deployment <deployment name> -n test-ns

kubectl rollout undo deployment <deployment name> --to-version =3

strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 1
    maxUnavailable: 1
    minReadySeconds: 10

maxunavailable :
when we are updating maximum (01) one Pod will be deleting

maxsurge :

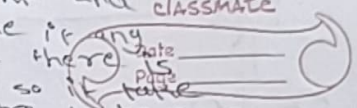
minReadyseconds:

Q what happen if the process is running out of CPU?

Ans

Q if the Process running out of memory ?

→ we can create the new pods with deployment and route the traffic to new one and delete old one if any thing happen we also roleback to old one but there two set of pod with deployment are created so it take resorces at that time on the node, make sure to have more number of CPU & Memory at that time.



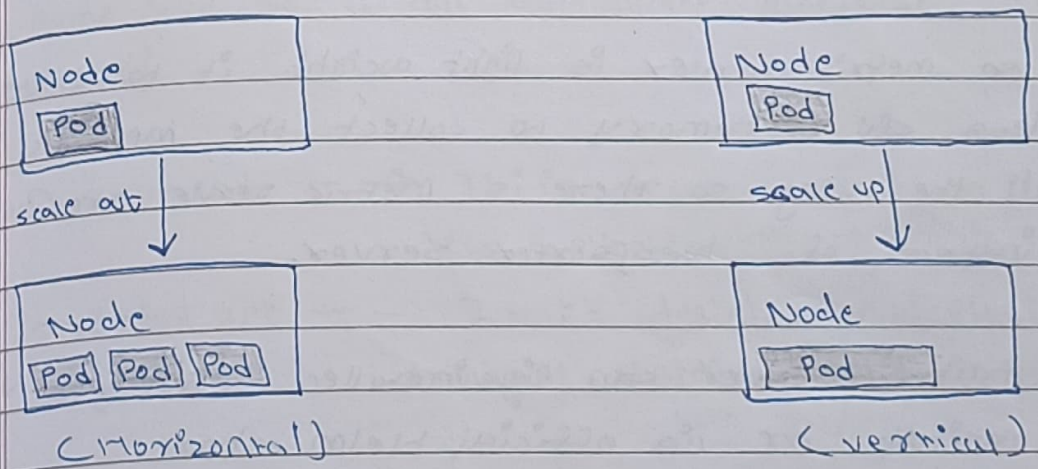
blue green deployment :-

Blue green deployment is an application release model that gradually transfer user trafic from previous version of an app or microservice to new release.

→ we can create another target group and chany the target group to new cluster then after delete old one but there is two diffren cluster are there

K8s Horizontal Pod Autoscaler (HPA)

K8s Vertical Pod Autoscaler (VPA)



HPA (Horizontal Pod Autoscaler) :-

- ~~HPA are used metrics API for communication or for collecting data it use the metrics API~~
- and metric server communicate with Kubelet to collect the data.
- The Horizontal Pod Auto Scaler Automatically scale the number of Pods in replication controller, deployment replicaset based on observed CPU utilization and memory utilization.
- Now let see what is metric server

we can define in Horizontal Pod scaling yaml resource memory 80% or also CPU 90% so request ~~reach~~ sch. higher and utilization reach to 90% so it scale Pods.

Kubernetes metric server :-

The kubernetes metric server collec resource metrics from the kubelet running on each worker node.

- heapserter :-

same like metricserver there is heapserter but it is depricated. it also do some thing but it is depricated. why? beoz it tak lots of CPU & memory

So its completly depricated from kubernetes 1.13

So metric server is light weight it take very less CPU & memory to collect the metrics & all the thing so there is metric serve are using instead of heapserter server.

- matric server can be installed through YAML manifest or via official Helm chart.
- kubernetes metric server intract with API server to gather information about the cluster state, about pods, nodes & other relevent information.
But it required permission to intract with API server
- without metric server HPA it wont work because it request data. or information

~~is it mandatory to take your cluster out sid~~

If your application is running on the pod can you take your data base outside of your cluster. or can you take your database in same cluster as a in other pod? database can be RDS or other is it mandatory to take database as a pod.
all depend over usecase, lets see your database which is mongodb is running on the server which IP is 172.31.5.68 so if over kubernetes & RDS are on same vpc so it can communicate otherwise Peering required.

Deployment → It is recommneded to use state less application

statefull set → It is design for deploying & managing statefull Applications.

But we can use Deployment and use volume for statefull applications.

Kubernetes Volumes

- A container file system live only as long as container does, so when container terminates and restart file system changes are lost
- Volume in Kubernetes are very easy to manage
- It basicly directory which are mounted to a Pod
- Kubernetes support many types of volume
- A Pod can use any number of volumes type simultaniously.
- Volume are especially important for stateful application such as databases.
- Two different Pod can use the one sing volume
- Kubernetes supports several types of volumes
  - EBS
  - azure file
  - hostPath
  - configmap
  - emptyDir
  - nfs
- we can attach more then one volume to the Pod.
- So if my pods get deleted then I ~~may~~ loss my data so we have use volumes.

```
volume
  - name: mongodbhostvol
    hostPath:
      path: /tmp/mongodb
```

hostPath use volume within same node

→ hostPath :—

- if volume and Pods container are on same node so when Pod get delete it use same volume after recreate so there is no loss of data
- so what if Pod are reschedule on different host what happen? so in that case we have to use some external storage or database. which is out of over cluster like EBS, Azure file

with help of hostPath we use volume with same node (if over Pod down and then up on other node so it can not communicate with host path beco2 over volum is on other node & Pod schedule on other there but on the node)

so thats why we use external storages like NFS EBS AzureFileDisk.

1) Take one EC2 which we mount to the cluster as a volume
2) Take it in same vpc (avoid vpc peering)
3) open port of NFS
4)

If we use volume directly so k8s do not store the data about volume how many volume is ther or if all pods is deleted how I know which volum is that pod is using so thats why we have to use PV & PVC

Date
Page

→ Persistent Volume Claim :- (ns level)

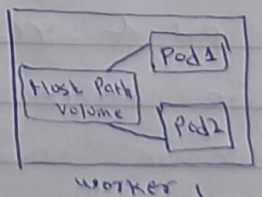
- Persistent Volume claim (PVC) is a request for storage (volume) by user
- while creating Pod the way we desined how much memory & cpu we need so same while we creating PVC we specify specific size & access modes. How much size & how or what type of access we need.

→ Persistent Volume :- (cluster level resorce)

Persistent Volume is a piec of storage (Host-Path, NFS, EBS...etc)
Persistent volume provide an API for user & Administator that show detail of storage provided from how it consumed.

Concept :-

When we are using hostPath we know it work within node if Pods and volume are on same node so it work But if Pod get down and then reschedule in other node so it can not communicate with host Path which is on other node so there is concept come Persistent Volume which is outside of your cluster persistent volume may be (EBS, S3, EFS, NFS or etc) even over clusrer get down data will be safe & accessable.



if we are using storage class so we only write PVC
so PV is created by storageclass

Persistent Volumeclaims & Persistent Volumes are the cluster level resorce not the namespace level resorce.

PVC and PV should be in same namespace
Storage class (Provisioner) if we have two or more storage class so we can choose one as a default storage class

Persistance Volume — cluster level every name space Pods can use
Persistance Volume claim — claim from same name space where the Pod we wont
PVC & Pod should be same namespace

→ there is some condition are there to map your PV to PVC
- we know your PV have there access mode which should be - ReadWriteOnce , - ReadOnly
- same your PVC claming with other Access mode
- if PVC or PV both have diffrent access mode it should not map. both have same access mode your PV & your PVC

Q If we have 2Gi volume & claming 1Gi if it have same access mode it maped no issue. But remaining 1Gi is mapped with other?

Ans No, remaning PV is not mapped with other

one PV = one PVC (it is

Pod.yml

```
volume
-name : db
 persistant volume claim:
  claimname : mangopvc
```

PVC with pod manifest we refer

PVC we refer in Pod manifest

The PVC and the Pod where we refer the PVC should be same namespace.

There is two type of persistent volume

1) static volumes → volume created by admins (Devops Eng) manually
2) Dynamic Volumes → creating volume dynamically using storage class.

static :- Volume creted by admins

if you claming volume or you are having the volume is it get fine in that case if you created volume or some one claming so it get it

Dynamic :- But what happen in Dynamic there is storage class we configure when someone claming the volume or volume is not available it automaticly take volume available the storage class is like a driver which is automaticaly get available the volume to the claim.

if storage class configured only we creat PVC so PV is created by storage class

storage class is nothing but one of the k8s resource

in EKS there is some default storage class are there

in EKS there is EBS are defaul storage class

in AKS there is Azure file & Azure disk is default

if more then one storage class we mark one is a default (In open source we have to configure storage class by the manifest file where we define over if its nfs so nfs server pre-IP as mention & it is over default storage class) in EKS the

→ To be in mind while mapping & creating EBS is pre-configured.

1. PV & PVC access mode will be same
2. claming capacity of Gi is less or equal to capacity of PV
3. namespace not mandatory

If there is multiple storage class we set one of them as default and if we want to use other we have to mention there name in PVC

Q what is accessmode How its work?
Q what is Reclaimpolicy How its work?

Accessmode

Readwriteonce → if Pods are on same node they can access the volum with Readwriteonce
RWO
if thy are on diffrent nod it is not for that.

Read write many → multiple pod if thy are on
RWX diffrent nodes also can read write on the volume

Read onlymany → multiple pod if they are on
ROX diffrent nodes only read th volume.

Read write Once Pod → only one pod can read & write only.

RECLAIM Policies :-

Delete : if we delete PVC the associated PV get deleted

Recycle : if we delete PVC the associated PV get recycle
if you delete PVC so ~~del~~ PV delete there data PV is there & if any PVC claim it they can use it or associate it

Retain : if you delete PVC there is PV avail-able or there data also available or it can be associated with other PVC if any on claims it

environment?

ans→ no same image but should be able to use for development staging & Prod they can refer/use environment variables. classmate Date Page

store the secret and refer it in Pod env

Configmap and Secrets

configmap are for non-sensitive configration data like environment variables, configration files or many other configration data your application needed (instead of ~~env~~ hardcoding in environt we refer it by name where in configmap we put)

Secrets are similar to config map but use or design for storing sensitive data like Password API keys, or many confidential data

- You should store config (like db connec" detail, Passw, API Key) outside application itself.

Configmap :

- store data in key-value pair
- easy to manage configrations accross different environments.
- store non-sensetive configration information
- store like environment variables, commandline argument application needed configration files.

Example :

on one of the Pod you have web application running in k8s that connects to database, so instead of hardcoading database connection details we can use configmap,

```
kind: configmap
metadata: db-config
    data:
        url:
        username:
        password:
```

configmap as a Volume

```
kind: configmap
metadata:
    name: javawebapptomcat
data
    tomcat-user.xml
        <  ---------- >
        <  ---------- >
        <  ---------- >
```

We put information like that in configmap and when you have any changes in your dp you just change

a volume so over file will mount to the container.

Date
Page

the configmap and Pods can refer it

```
envFrom:
  - configMapRef:
      name: db-config
```

```
volumes
  - name: conf
    configmap
    name:
```

we give it during Pod deployment Ym file so when Pod deploy the Pods have that configmap details

② ~~Sir in which location developer can refer it?~~

Secretes

Example scenario

→ Image pull from Priv-Registory (Docker with

But in EKS cluster we use ECR so IAM role

Eg: containerRegistryReadOnly → EKS Node

③ I want to route my traffic to unhealthy pods
③ I want my application restarted if it get unhealthy
③ to talked this type of issue there is concept liv & Read Pro

Q. what is Probe?
Ans - It's nothing but kind of health check

→ Liveness Probe & Readyness Probe (work on conta level)

Perform healthcheck by Liveness & Readyness Pr

Probes are kind of healthcheck there are two type of Probe Liveness & Readyness Probe
both are the healthcheck use to control health of an application running insides Pod's container.
but How different it with each other (Both action done by kubelet)

→ Liveness Probe :

- suppose that a pod is running over application inside the container but dueto some reason lets say memory leak, CPU usage, application deadlock etc so over application not responding to a request & stuck in error state.

- so using liveness probe we can check the healt of the application so for some reason the liveness Probe fails (health check fail) it restart the container liveness prob fail when your container not running may be your application having issue in some configration so it's not starting so we can find it by it liveness is fail it over application having is

→ Readyness Prob :

- Readyness Probe are use to detect that container are ready to accept the traffic or not
- Readiness Probes are essential for ensuring that application within a Pod are up & running before they start receiving traffic.

↳ check on github

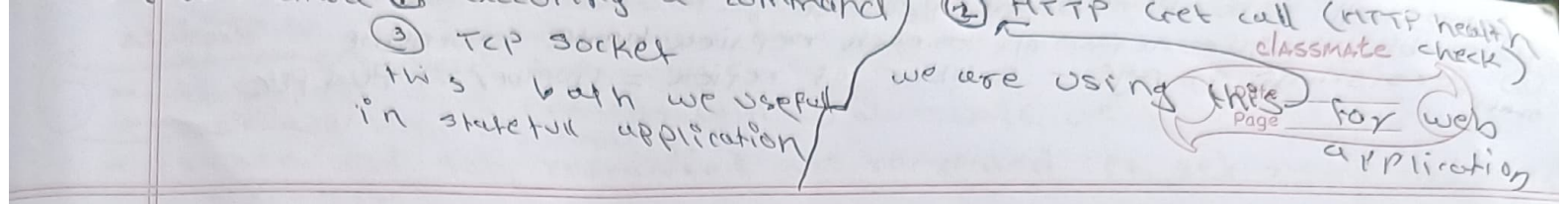
Name	Ready	
Pod1	1/1	— Its mean in pod, 1 container 4 1 ready
Pod2	1/3	— it's mea in pod 3 containe 41 ready two remaining or no-ready.

→ ther are diffrent methods for readyness &p

- if you hove ~~statefull~~ application it should be http healthcheck
- if you have statefull application it should be tcp or customcommand because database does not have application

Ex when your application in the containers a running but in some case if you have ~~two pods~~ i one pod and within it you have two container

- one container and there application is worning fine
- second container is also running but the a -cation is not responding whatever probl it may be memory leak, cpu usage, applic deadlock
- But if you ~~get~~ see "kubectl get pod" pod and there container is running but application not accesable when you sen traffic or traffic come to that application t trafic route to the one container it's fi but when it rout second container you got error 404 or 401 like that
- so there is liveness Prob & readyness brob come to the Picture.
- whenever your container is not healthy it restart the containers.



when liveness probe and readiness probe are fail it show result like

NAME	~~REA~~	READY	STATUS
maven-web-app-83214		0/1	Running

Q when kubectl restar the container if liveness probe yet fail

Ans when liveness fail 3 times which is bydefault (which we can change) kubectl restart the container

→ Statefulset

we know we are deploying over pod with Deployment which is the recomended way becoz of there advanced feature -

But know there is one more object called statefulset which is recomended to deploy statefull applications like database, Jenkin nexus, kafka cluster.

we can deploy statefull application by deployment by attaching volume but there is diffrent is deployment & statefullset know lets see what

when we use deployment all the application Pods are using same volume and data across all of then will be same. so it usually use for stateless application.

Statefull sets Deployment provide :-
① stable, unique network - each replica have hostname
② persistent storage - number of repica = number of PV & PVC
stable

But when we go for the statefullset each replica of statefullset have there own storage

- statefullset is kubernetes resource.
- use to manage statefull application.
- statefullset is also a controller but not like Deployments
- it does not creat replicaset rather itself create the pod with unique naming convention

eg:- if you create statefulset with name nongo it will create pod and multiple replica there name will increment like mongo-0 mongo-1, mongo-2, etc.

- Every replica of statefullset will have it's own state and each of the pods will be creating it's own PVC (Persistent volume claim)

- so statefullset with 3 replica will create 3 pod each having it's own volume so total 3 PVCs. So 3PVC & 3PV
- If we do it with deployment it create 3 replica but all replica use same volumes.

- if one pod is deleted then the statefullset recreate it with same host name & with same storage and network identifier.

- when you creating pod by Replicaset, Replication controller & deployment it creat replica at same ti

so we get good performance.
- If we get any problem with ① database we have ② other so we get high availability.
- when pod get restarted or recreated it get stable network & volumes.

But when you create by statefullsets it creat one after one, when one create completly & another one creat

- when we deleting & updating statefullset it delete in reverse order mean 2 1 & 0
- when we creating it order like 0, 1 & 2
-

→ service "headless service"

- when we using service Nodeport & clusterIP it act like loadbalancer
- one time rout trafic to one pod & another time on different port.
- till now we know service type clusterIP & nodePort which do loadbalencing by clusterIP
- But in some case we dont want loadbalancing we have to contact to specific pod in between the group of pods there the headless service come into picture.

- common use of headless service is when you individual pod with specific roles so you want directly communicate with that specific pod you dont wont to go through loadbalencer so that the headless service use

spec:
clusterIP: None

when we use headless service each pod of that statefull set are discover by DNS name

In ~~persistent~~ statefull set yml file there is VolumeclaimTemple we dont wont to creat PVC & refer in yml in statefull set volumeclaim temple is there they create there PVC which is equal to number of replica.

Scheduling

scheduling are done by shedular ; which node pods will scheduled ~~but~~ but what basis scheduler will decide

Ans it sheduled on resorces avalable on the nodes & what are pods requested.

- K8s scheduler select right node by check nodes capacity for CPU and RAM and co -ring it to the pods resorce request.

- But there is some scenarios when you want your pods end up on specific pod nodes

- If you wat to run Pods on same matchi
- If you dont want to run pods on same n
- If want Pods run on that perticular matchi which have high CPUs.

1. NodeSelector
2. NodeAffinity
3. PODAffinity / POD AntiAffinity
4. Taints & Tolerations
5. Cardon
6. Drain
7. Uncardon

kubectl label node <IP> name=workerone

① Nodeselector :- (we have to give label to over node first)

```
nodeSelector:
    name: workerone
```

we have to label over node and then specify in over container spec so pod get schedule on that node only.

But if we have name: workerone label only one node so what happen when that node get down due to insufficient memory, diskPressure, or any reason.

So for that we give same labels to group of nodes if any node get down so ~~kube~~ scheduler schedule the ~~AD~~ the Pods on other node which have same labels.

② Node Affinity :

Node Affinity is also use for selecting a node on basis of label

It is like advance as compare to Nodeselector it have the same nodeselector & nodelabel phenomenon But using node Affinity we specify certain rules to select nodes on which Pods can be scheduled.

→ Soft Rules (PreferedRules)

- Prefered during scheduling & ignore during execution means when we use soft Rules it prefered first the node which have same labe if deploy on that node
- But if there is no node free so it sheduled the Pods on other node also which have diffrent labbe or we can say which label do not matched the affinity

Hard Rule (Required Rules)
requiredDuring schedulingIgnoreDuring Execution

PODAffinity / PODAntiAffinity

It work based on labels on the Pods which already scheduled on the ~~Pods~~ nodes.

when one Pod is running on the node then I want one more Pod which I want to deploy on same node so that time we use this PODAffinity rule

when I dont want the Pod on perticular node so this time we use PODAntiAffinity.

- Pod Affinity allow you to schedule a Pod on a set of nodes based on labels present on the nodes. However in certen scenarios we might want to schedule certain pods together or we want make sure that certain pods are never schedule together this can be

achived by PodAffinity and PodAntiAffinity respe-
-ctively.

NodAffinity — when we want over pod schedule on that perticular node

PodAffinity — when we want over pod schedul on nod wher over pod which already run (which we specify) scheduled on that node.

eg:- if Pod1 are running on nodeA, so if we want that Pod2 also will be scheduled on nodeA (on same node) so we use Pod Affinity

Nod Anti Affinity - when we dont want that over pod to be schedule on that Perticular ~~Pod~~ Node

PodAntiAffinity - when we dont want over pod schedule on same node
it we have 2 replica or over pod will schedule on same node & nod get down so what happen?
that time we use PodAntiAffinity so over node get down over applic get access by other pod which running on diffrent node.

→ Taints & Tolerations :-

Node affinity is property of Pods that attract them to set a node But Taints are apposit, they allow a node to repel a set of Pods.
↓
Reject

Taints is property of nodes that allow you to ~~appeal~~ repel (Reject) a set of Pods

Kubectl taint nodes <nodeName> <key>=<value>:<effect>

there is some possible effects

1) Noschedule - Doesnt schedule a Pod
nodes with this taint are not consider for scheduling new Pods.
Pods without toleration For this taint will not be schedule on node with the "Noschedule" taint

If you have some already existing Pod which is running on that node so new Pod not be schedule in that node and if you delete the running Pod which is already running so after delete they do not schedule on that node.

2) PreferNoschedule :
It Node with this taint are not Pretered for scheduling of new Pods but if no other nodes are available Pods will be scheduled on nodes with this taints.

3) NoExecute :

there is maintenance done by step by steps.

→ Node Maintenance / cluster maintenance

① cordon — the node → it will make the node as unschedulable
scheduler do not schedule on that node
but exsisting pods continue to run.

kubectl cordon <Node>

② Drain — the node → kubectl drain <IP>
Evacuate existing Pods from the node by kubectl drain command & rescheduled existing Pod to other available nodes.

③ Perform Maintenance :- once node is drained, you can perform maintainance (upgrading, hardware change like that)

④ Uncordon — the node → kubectl uncordon <Node IP>
After maintenance you can make node as schedulable again.

→ ResourceQuotas (k8s object at ns level)

It is a k8s object, using resourceQuota you can limit resource consumption at namespace level.

- How many pods can be created
- How many memory can be used

- when several users or team uses share cluster with fixed number of node so there is concern that one team could use more then its fair share of resorces.

- if you deploy resorceQuotas on your namespace you have to defined reques & limits in your Pod manifest it is mandatory.

→ ~~to~~ limit-Rang :- (k8s object at namespace level)

we defined the limit range but there is one k8s object limitRang which we configered

if we do not defind Limts & Requ of Rsorces in Pod manifest so it take by detaut which we configure in limit Rang object

Network Policies :-

If you want to control traffic at the IP or Port level so you might consider using K8s Network Policies for Perticular applications in your cluster

It help to defind How your Pod allow to communicate with various entities

Prerequisites

Network Policies are implimented by the Network -Plugin. To use network Policies, You must be using networking solution which support NetworkPolicy as of now we are using

So there is ingress trafic & egress

Kubectl get netPol -A

Ingress — incoming from the Pod
egress — outgoing from that Pod

If we want that Pod does not communicate with outside to the internet so we egress trafic we use for databas generally.

Self Managed clusters / BareMetal K8s we use software like
Kubeadm
KubeProxy

Managed K8s clusters
AWS -> AKS
Integrate with other AWS services like IAM, ELB, EBS, ...etc
AZURE -> AKS
GCP -> GKE

In manage K8s lets take AKS so there is we can configure AWS service IAM, ELB EBS that the only difference

EKS Introduction

- Amazon Elastic Kubernetes Service (EKS) is fully managed Kubernetes service
- EKS is the best Place to run K8s applications becoz of it's security and scalibility.
- EKS can be integrated with other AWS service such as ELB, Amazon cloudWatch, Auto Scaling Group AWS Identity & Access Managment (AIM).
- EKS make it easy for you to run K8s on AWS without needing to install, operate, & maintain your own Kubernetes control plane.
- so without worrying about How to install & How to manage we only focoused on deploying application and manging data on my cluster.

Managed Control Plane

- Amazon EKS provides scalable and highly available control plane that run accross multiple AWS availablity zones. afor high avalablity.
- control plane is over masters node
- Amazon EKS run Kubernetes control plane accross three avalablity zones inorder to ensure high avalablity and it automatically detects & replace unhealthy masters.
- if something gone with your worker node so you get your another worker also.

→ K8s Cluster Autoscaler (we deploy as a pod)

cluster autoscaler is the tool that automatic -aly adjust the size of kubernetes cluster when one of the following condition is true.

- There is pods that fail to run in the clus -ter due to insufficient resorces
- There are node in the cluster that have been underutilized for some period of time & there pods can be placed on other existing node.

Ingress :-

Ingress is a kubernetes object in which we add rules for routing traffic from exter sources to the k8s service
ingress is one of the object in k8s

ingress controller : in k8s ingress controller is component responsible for handling external access to service within cluster based on rules defined in ingress resources

There is several ingress controller available & you can choose one based on your requirm

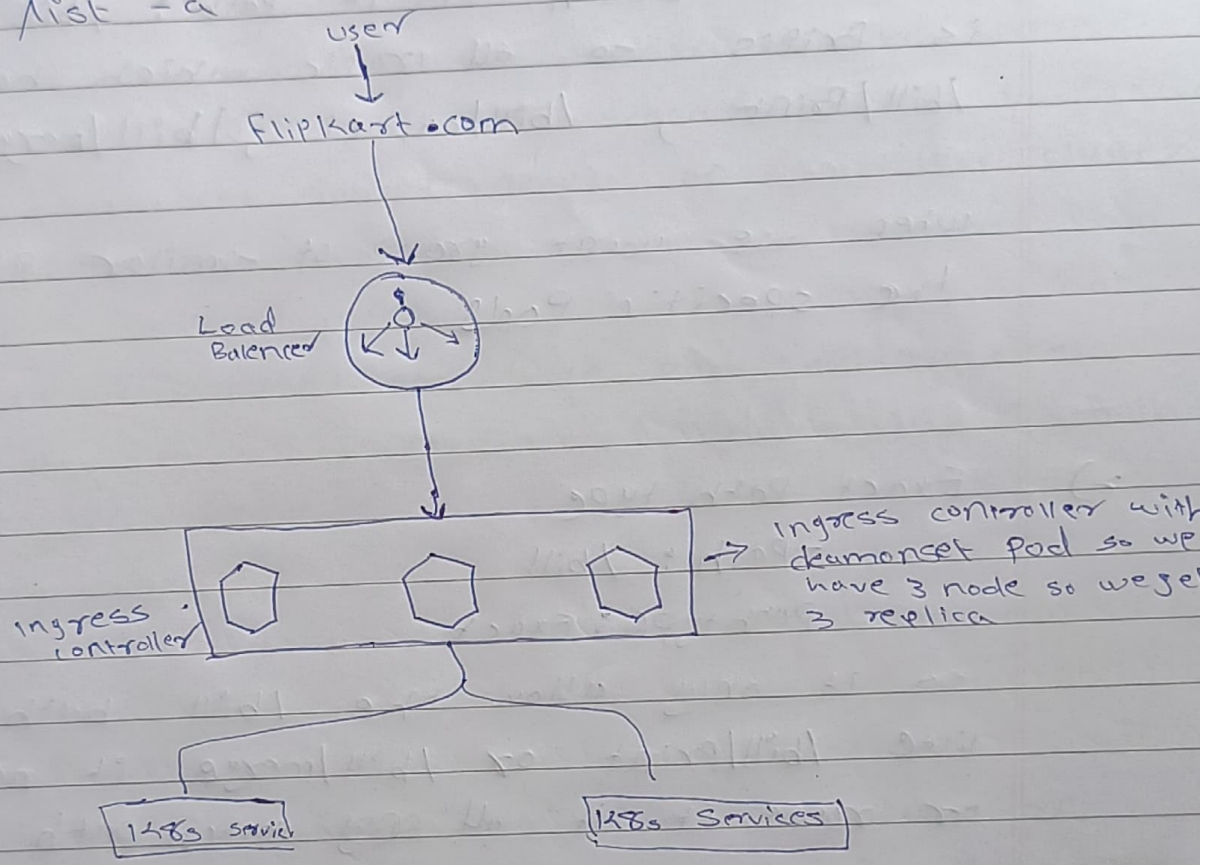
Nginx ingress controller
HAProxy ingress controller
Terrafik ingress controller
Istio ingress controller

we can deploy ingress controller as deployme or as a DaemonSet

- If we deploy as deployment we make sure we have more then one replica
- and after deploy ingress controller as a Po we need service (Loadbalancer) to access fron outside

Helm CLI commands

helm install myhelloworld
helm delete myhelloworld
helm list -a



we can defined the host base routing and pathbase routing in ingress rule

```
apiversion : networking.k8s.io/v1
kind : ingress
metadata :
  name : example-ingress
spec :
  rules :
  - host : flipkart.com
    http :
    - path : /bill
      pathtype : Prefix
      backend :
        name : flipkart-service
        port
          portnumber : 80
```

this is select based or every use case

there is diffrent path type is there

1) Prefix Path type

it use when we want the specified path the request is having that path match to specified path like if path: /bill or it is prifis so all traffic which have like /bill/Print , /bill/cancel , /bill/merge .

when we want group of similar path route the specific Pods.

2) Exact Path type

- Path: /bill
  Pathtype: Exact

so it only allow the /bill path request n like /bill/print or /bill/merge it not allow it or route it it all only match path like /bill.

3) Implementation specific

RBAC → Role Based Access control (Authorization)

who can do what within a K8s cluster.
RBAC is the way to define who can do what in K

if you are working on Kubernetes for some time you may have faced as scenario wher you ha to give some user limited access on you K8s cluste For Example you may want to user, who is from developmen, to have access only to some resorces that are in the development namespace and nothing else. so for that we use concept of Authentication & Auth -orization in Kubernetes.

Broadly, there is three kind of user that can access K8s cluster

① Developers / Admins
user which responsible for administrative or developmental tasks on the cluster, this include opera -tion like upgrading cluster & creating resources (PODs, Deployment, PV, PVC, ETC) onthecluster.

RBAC in Kubernetes is based on three key concepts:

1. verbs :- set of operation that can executed on resources (create, Read, Update or Dele

2) API Resources :- object available on ~~resources~~ cluster Pods, servic, nodes, PV & other

3) Subject :- (user, group, (service account) allow access on API based on verb & resorces.

→ Role & RoleBinding (work on ns level

- But the object like PV, node, storageclass and other which is cluster level resorces or objec so Role & RoleBinding not work

so for clusterlevel resorces

→ clusterRole & cluster-Rolebinding.

HELM :- Helm is the package manager of K8s it allow you to install / deploy / remove / update application on your kubernetes cluster in a similar manner to yum / apt for linux distribution

HELM chart → yml file of K8s resorces / objects. (collection of manifest)
like deployment.yaml , service.yaml
configmap.yaml , secrete.yaml , etc

HELM client (CLI) → helm install <AppName> <chart Name>
helm install javawebapp javawebchart
helm upgrade java

HELM Repository → helm repo add <repoName> <repoURL>
helm repo add sarimtechhelm http://helmrep.sarimt.com

why use Helm?

K8s can be difficult to manage with all objects we need to maintain, Helem manage all of this for us

Helem gretly simplifies the process of creating, managing and deploying application using Helm charts

Helem maintain version history of every chart (application) installation, if something goes wrong you can simply call "helem rollback"

Helm 3 Architecture :-

In helm 3 there is no tiller component Helm client directly intract with Kubernetes API for helm chart deployment

Helm 2 Architecture

Q What is difference between Helm 2 & Helm 3

Ans Here is some key difference between Helm 2 & 3

1) Tiller Removal :

Helm 2 required additional component called tiller. which responsible for maniging releas life cycle of K8s cluster

Helm 3 remove Tiller and all operation perform directly against K8s API server

any changes we want hardcode we do it in ~~values.yaml~~ deployment.yaml

and if we donot hardcord it we .values.yaml

include — it defined in helper.tpl