Kubernetes

↳ It is an ~~ac~~ orchistration tool, which means how docker containers talk to each other, how they scale, heal it will be done by kubernetes.

Why kubernetes

↳ It is microservice

Monolithic

↳ Means all the services are under one roof / one repo, if one service fail all of them will crash, if one needs an update everything needs to be updated

Microservices

↳ One big application is divided into smaller ones, if one service is impacted other will keep on running

Architecture

If there is one node then it is called server.
If it is multinode (or more than one server) then it is called cluster.
All the ~~work~~ nodes are working in the worker nodes & the work is assigned in the master node, no work will be done in master node, what makes master node make workers node work?

↳ It contains API server (1): It keeps track of everything happening in the masternode & the working node, it is an communication gateway b/w all the components. If we want to talk to the K8's cluster we need to talk to the API server.

↳ (2) Scheduler: It will schedule the ~~work~~ containers in the working node, to also run the pods in the node

↳ (3) etcd: Key:Value pair data store which store all the data of the K8's cluster

↳ (4) Controller manager: It makes sure everything is running properly like nodes, clusters all the services.

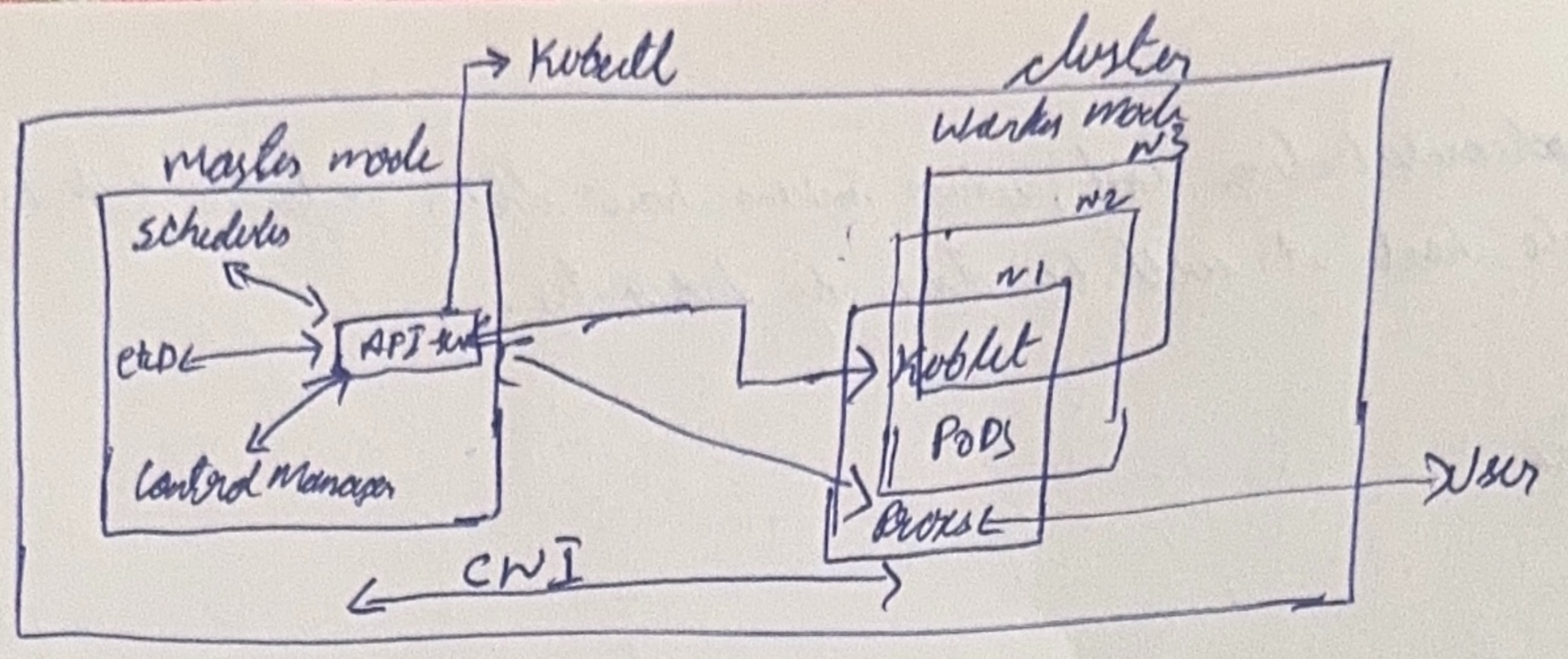
In worker node we have the following

↳ (1) Kubelet: makes sure all the things are working properly in the worker node if everything is fine it tells to the API servers & it tells it to the control manager

↳ (2) Service Proxy: To access the pods as an outside user we need to use this.

How do all the nodes communicate: using CNI (Container network interface).

Kubectl: To give instructions to the cluster using API servers.



How can you create a cluster?

↳ Kubeadm : Install kubeadm in 3 instance and join them in an cluster
(cost will increase as we are taking 3 instances)

↳ Minikube : we can create cluster in single instance (or) local system

↳ KIND : K8's in dockers, in dockers containers we will have K8's cluster.

↳ EKS/AKS/GKE : cloud based.

KIND cluster.

file to tell all the cluster information :- config.yml.

KIND : cluster → what kind of file to create a cluster

apiversion: kind.x-k8..... → API version found on KIND website

nodes: → we need multiple nodes so using a list

- role : control-plane → Control plane also known a master node

image : kindest/node → which image we are using

- role : worker → 1 worker nodes

image → which image

- role : worker
- role : worker

} → 2 more worker nodes & which image

extraPortMapping: → Mapping the port of contains & host (list again)
↳ As the cluster is running in containers, we need to map contains & host ports

- containerPort : 80 → which port in container

- hostPort : 80 → which port in local

protocol : TCP → which protocol

c→ Kind create cluster --name tuscluster --config = config.yml
↳ creating a KIND cluster called tuscluster using the config.yml file.

c→ kubectl get nodes
↳ How many nodes are in the cluster.

↳ Before this we need to give context for kubectl of which cluster

c→ kubectl cluster-info --context kind-tuscluster

Cluster using Minikube

↳ minikube start --driver=docker -vm=true

↳ if running on EC2

↳ where are we running the cluster in docker

↳ This will create an single node minikube cluster.

Now kubectl is in minikube to change it back to KIN we can run the following

↳ kubectl config use-context twocluster

In kubeadm

↳ We need to install kubeadm on the instance and if we do init then it is the master node, workers node will join on port 6443

↳ A token will be created in the master node and all the works node will use it to access the master node on port 6443

What will be running in the cluster?

↳ We can run one (or) more contains in the pod, for auto scaling and auto-healing we need deployment, to make it accessible to public we need service.

for eg: [Contains of nginx (image) → POD → Deployment → Service → User] In cluster 1

[Mysql → POD → Deployment → Service → User] In cluster 1

Both the ~~pods~~ things are running in the same cluster we can get confused which is which so we have namespace: whatever resources are present in the namespace that is independent to that namespace, it doesn't effect the other namespace, isolate to each other.

↳ kubectl get namespace (or) kubectl get ns → To list all the namespace.

→ If no namespace is defined it will be in default

↳ kubectl get pods -n default → Get all the pods running in default namespace.

↳ kubectl create ns nginx → To create a namespace nginx.

↳ kubectl run nginx --image=nginx -n nginx → In nginx namespace

↳ Create a pod nginx using the image nginx from docker hub

vim namespace.yml

-> Kind: Namespace → It is a namespace
api version: v1 → K8's version
metadata: → Object w multiple information
name: nginx → name of namespace.

-> kubectl create -f namespace.yml → This command will just crea
-f → flag for file; namespace.yml → file name

-> kubectl apply -f namespace.yml → Create & update

Creating a pod :- pod.yml

-> Kind: Pod → It is for creating pod
apiversion: v1 → which K8's version
metadata:
name: nginx-pod → name of the pod
namespace: nginx → Pod in the namespace nginx
spec:
containers:
- name: nginx → contains name
image: nginx:latest → creating container using the image nginx
ports:
- containerPort: 80 → Ports which the container will ex.

-> kubectl apply -f pod.yml
-> kubectl describe pod nginx-pod -n nginx → will tell us whatever happened with pod.
-> kubectl exec -it nginx-pod -n nginx --bash → To execute the pod.

Deployment kinds

-> What is replication controller :- It will manage the pods replicas
① Replica set :- if you say you want 4 replicas then 4 replicas of pods will be created
② Stateful set :- The pods can be created and deleted anytime but here numbering will be given to each of the pods
③ Deployment :- Same as replica set, but rolling update is given to it
what is rolling update :- If we change the image pod all the pods will get updated at once there will be downtime but in deployment one pod gets updated while other is running.
④ Daemon set :- It make sure how much ever pods we have there is always one pod is running.

apiversion: apps/v1 } → deployment kind using version1

metadata:

name: nginx-deployment } → name of the deployment set & in what namespace

namespace: nginx

spec:

replicas: 2 → How many replicas

selector:

matchlabels:

app: nginx } → Applying this deployment to any app with the name of nginx

template:

metadata:

name: nginx-pod.

labels:

app: nginx } → Created a pod with the label nginx & name of the pod: nginx-pod

spec:

containers:

- name: nginx

image: nginx: latest

ports:

- containers port: 80 } → To create a pod we need an image we are using image nginx: latest & port 80

c> kubectl get deployment -n nginx

c> kubectl scale deployment / nginx-deployment -n nginx --replicas = 5

↳ we are scaling the deployment with the name nginx deployment to 5

→ Instead of deployment if we do replica set we get replica sets.

↳ if we do daemon set we get daemon sets, but we do not need replicas.

as there will 1 pod in each nodes.

<u>Job - Cron Jobs</u>

↳ Job is something when an container has a single task, we do not want it running as a server like :- taking a backup, patching, update.

↳ we can have single processing (or) batch processing.

↳ The image which we will use here is called busybox

status:- of the pod will be completed after the completion of running the pod

<u>pod lifecycle</u> :- Container Creating, Pending, Running, Terminating, Completed, imagepull backoff error.

<u>when</u> we want a job to run at a particular time then it is cron-job.

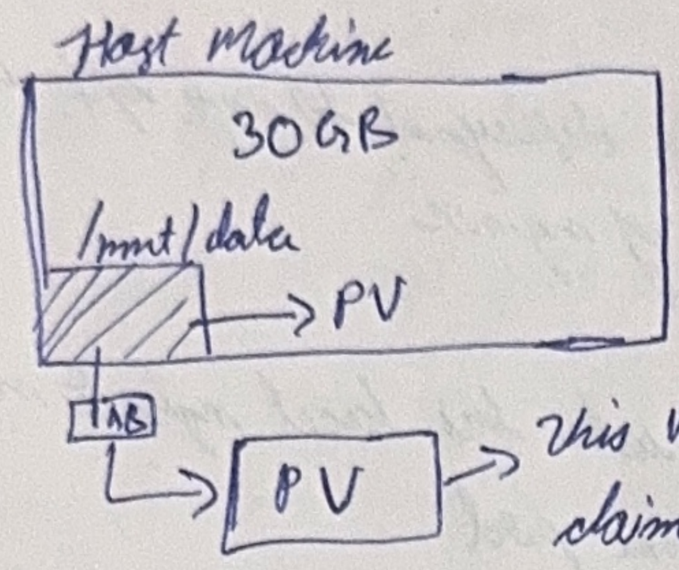
↳ ★★★★★ :→ we can get in cron-guru website

Storage

→ the pods running on the server can be deleted at any time but what about the data?

→ If there is 2 containers running in the deployment step but one gets deleted what about the data?

→ We need to bind the data to the host, How? by using Persistent Volume (PV) & PV claims (PVC)



→ This volume is PV which is available but to use it we have to claim it that is PVC, who reclaims it, it is theirs.

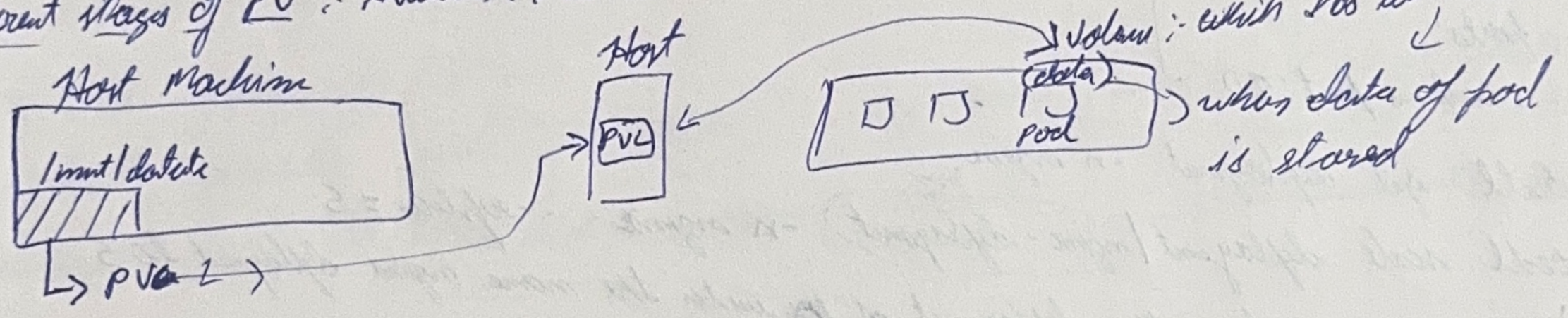
→ PV cannot be created in a namespace, Any namespace can claim the PV if the requirement match, its dedicated to that until released (depends on reclaim policy)

Different Reclaim Policy:

↳ Retain :- underlying storage is not deleted (when data is important & must be preserved)

↳ Delete :- Everything is deleted

Different stages of PV :- Available, bound



Services

↳ It will also use label and selectors like deployment.

↳ we have port: which port from the device outside do you want to connect

target port :- the port inside the pod.

↳ Different types :- Cluster IP :- the IP address of the cluster in that a particular port will be assigned (defaut)

node Port :- In all of our node we will take a node and assign to it

load balancer :- Used for cloud services.

↳ kubectl get all nginx
↳ namespace.

As our cluster is in a docker container we need to port forward, IP address need to exposed.

↳ kubectl port-forward service service/nginx-service -n nginx
which service are we forwarding
↳ In the namespace nginx

→ Creating a docker image & uploading it to docker registery.
→ Create a namespace
→ Create a deployment (Creation of pod is here)
→ Create a service
→ Expose the port.

Ingress

→ Traffic routing in the cluster.
→ Managing routes & services (traffic) in a cluster we use ingress
→ All the services should be in the same namespace.
→ We need ingress controller.

```
Kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: nginx-nato-ingress
  namespace: nginx
spec:
  rules:
  - http:
    paths:
    - pathtype: Prefix
      path: /nginx
      backend:
        service:
          name: nginx-service
          port: 80
    - pathtype: prefix
      path: /app
      backend:
        service:
          name: nato-service
          port: 8000
```

→ whenever we get /nginx path prefix type we want to go to the service which is backend which service in backend is nginx-service on port no 80.

→ whenever we get /app path prefix type we want to go to the service which is nato service on port 8000

→ After this we need to expose the ingress controller.