What is kubernetes

Docker is a pre-requisite before you proceed to understand kubernetes

Kubernetes (popularly known as k8s) is a container orchestration engine, which as the name suggests lets you create, delete, and update containers

This is useful when

1. You have your docker images in the docker registry and want to deploy it in a cloud native fashion
2. You want to not worry about patching, crashes. You want the system to auto heal
3. You want to autoscale with some simple constructs
4. You want to observe your complete system in a simple dashboard(LEAST IMP METRIC)

its very similar to ECS and EC2

IN THIS ALL THE MACHINES CAN DISCOVER EACH OTHER AND CAN TALK TO EACH OTHER WHILE WORKING

THIS IS CALLED AS A K8S CLUSTER

VERY SIMILAR TO ECS (ELASTIC CONTAINER SERVICE)
It also starts a lot of containeres, lets you visit the containers, let you start the containers, scale them etc without you login to the vm
orchestrator is the kubernetes which is starting or stopping the containers etc
it helps us to easily move from aws to gcp, to vulter,to digitalocean , to fifth cloud provider
all clouds support the kubernets

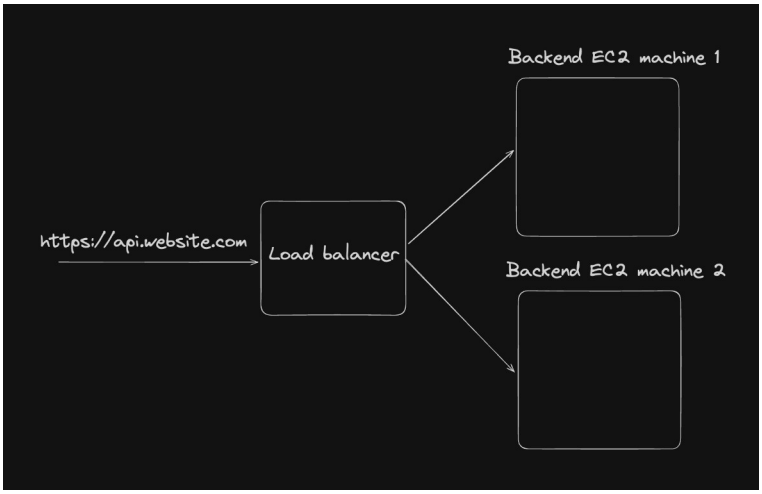
IT AUTOHEALS THE SERVER IF THE LIKE DESIRED CAPACITY MACHINES ARE NOT WORKING IT WILL MAINTAIN THE LIMIT AS THE ECS DOES THAT ALSO IT ALSO MAINTAIN THAT IF ANY SYSTEM CRASHES

IN ECS WE USED TO USE FARGATE
WE JUST GIVE KUBERNETES THE DOCKER FILE IT JUST TAKES IT AND DEPLOY THEM ON THE MACHIENS WE JUST PROVIDE THEM WITH THE DOCKER FE, DOCKER BE FILES TO USE
IT HAS ITS OWN CODE AND MAINTAIN THE MACHINES ON WHICH THE CODE IT HAS TO RUN

Before kubernetes Backend(HOW WE USED TO DO EARLIER)

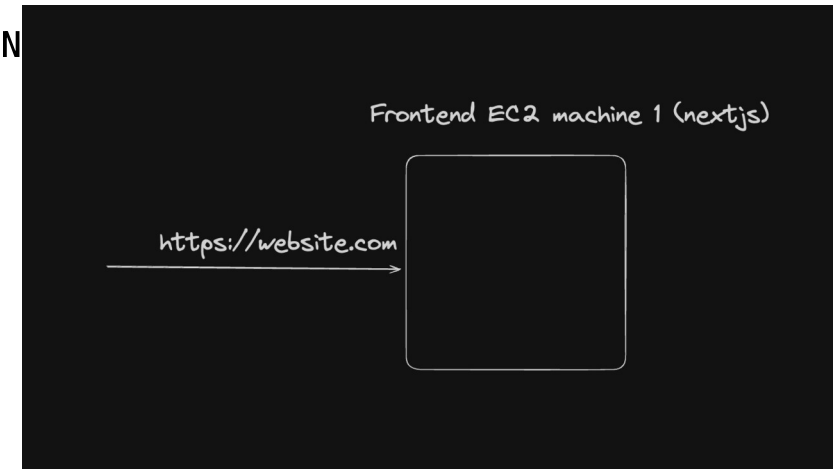
WE KEPT LIKE A LB IN FRONT OF THE MACHINE WHICH WAS RUNNING NGINX, CERTBOT, NODEJS APPLICATION AND WAS EXPOSED ON A PORT

TO REACH THE WEBSITE WE FIRST GO TO THE LB AND THEN THE MACHINES IF NOT THE LB THEN DIRECTLY THE MACHINES



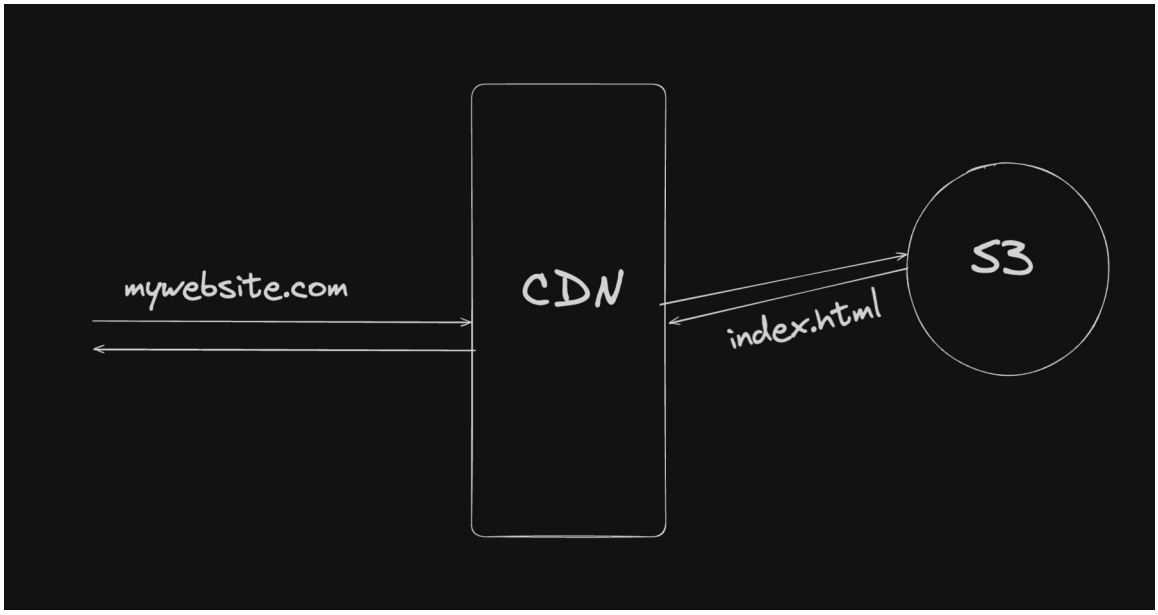
Frontend (Nextjs)

THIS IS JUST FOR NEXT JS WE CAN USE THIS OR THE UPPER METHOD ALSO

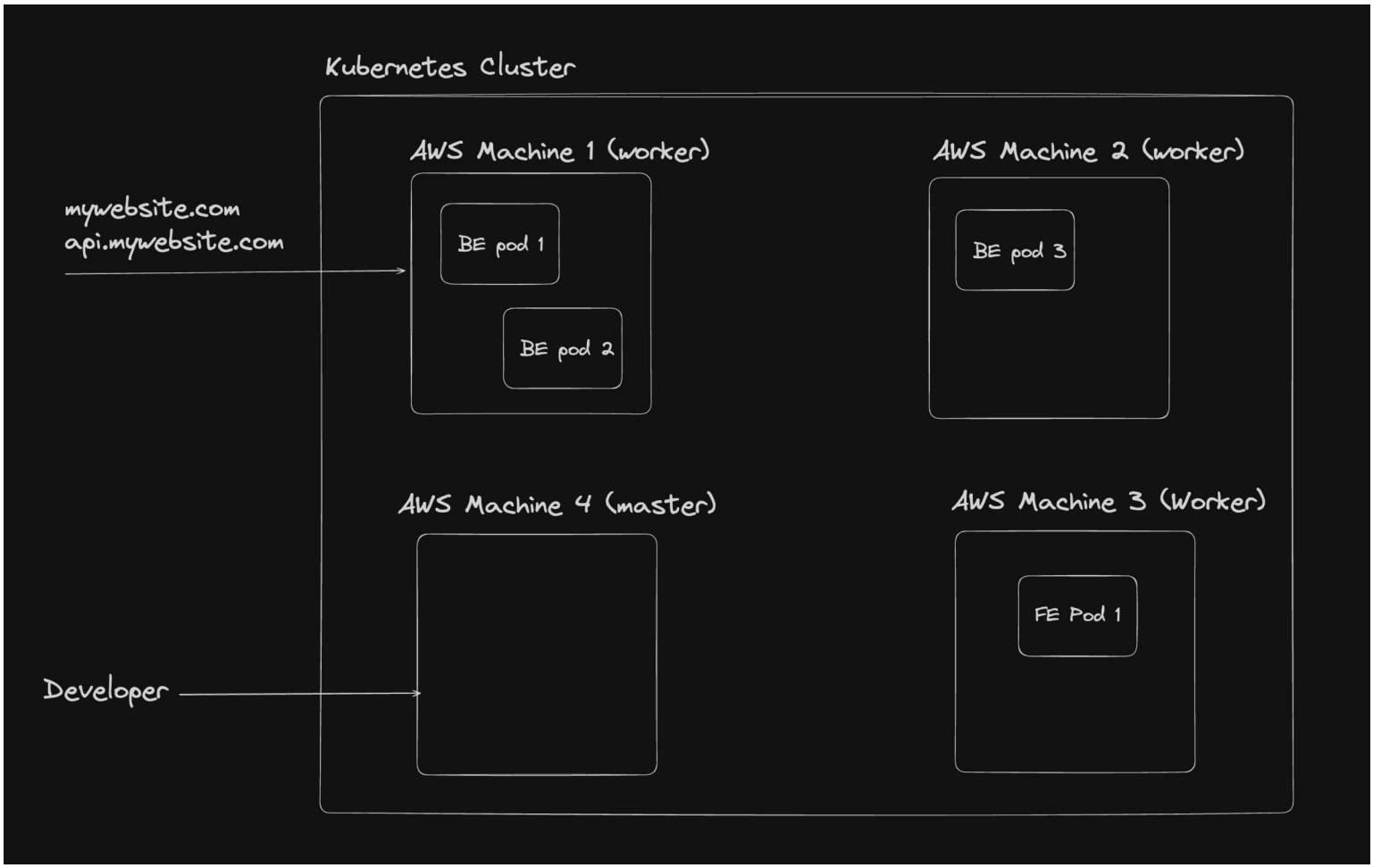


Frontend (React)

FOR STATIC WEBSITE, REACT WEBSITE WE USE A S3 MACHINE AND A CONTENT DELIVERY NETWORK(CDN) IN BETWEEN



After kubernetes
Your frontend, backend are all pods in your kubernetes cluster



THERE ARE 2 JARGONS FOR NOW
CLUSTER
MASTER NODE VS WORKER NODES

WHAT IS A MASTER NODE?
A node which is controlled by the developer in which we specifiy things like we need 3 machines in which I want to run this dockerhub/pallab/fe image and dockerhub/pallab/be image and just run this machines master node itself wont run on the machine it will start multiple machines , it also has the

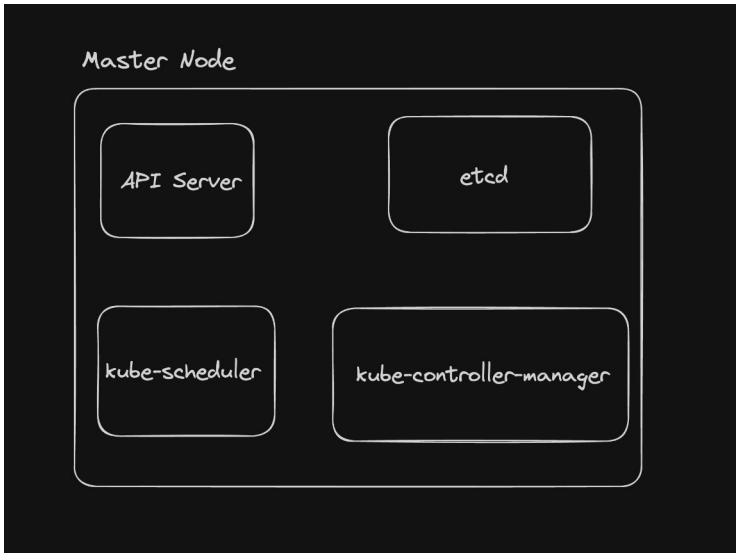
intelligence if some machine goes down it will automatically moves the fe code to machine 2 is assume like the machine 3 was down

BIGGEST RESP OF MASTER NODE MACHINE IS
TAKE COMMANDS FROM THE DEVELOPER ,
TAKE DECLERATIVE COMMANDS/DESIRED STATE COMMANDS FROM THE DEVELOPER, AND SCHEDULE THE COMMANDS ON THE VARIOUS MACHINES

ONE MACHINE IS CALLED AS NODE
LIKE IN THERE THERE WERE 4 NODES AND COMBINATION OF THESE IS CALLED AS CLUSTER AND IN THE NODES 3 WERE WORKER NODES AND ONE WAS MASTER NODE

Nodes
In kubernetes, you can create and connect various machines together, all of which are running kubernetes. Every machine here is known as a node
There are two types of nodes

Master Node (Control pane)
- The node that takes care of deploying the containers/healing them/listening to the developer to understand what to deploy



BUT WHAT IS ACTUALLY
RUNNIGN INSIDE THE MASTER NODE/ THE ARCHITECTURE?

IN SHORT EXPLANATION
first you interact with the api server in which you send a request on such as http://masternode.100xdevs.com/deployment/container/start
we are just telling it please start 3 machines
when we say that it get stored in the etcd(it stores things like key value pair , creds etc) its like a postgress, redis its a db that kubernetes choose
its like a enough db to store info in kubernetes

now what does the **kube-scheduler** does?
Now it know from etcd that harkirat want to start some machines so I will start some machines and I know where to start those machines
it has metric of the three machines to run and it knows on which machine harkirat fe should start on, on which harkirat be should start on etc

then there is something called **kube-controller-manager**
what is controller?
This is like controlled loop(a inf loop which is checking something) eg a thermostat like it constantly checks the temp of the house like if we setupped the temp should be 25 it will turn on the ac if temp is 30 and will turn it of if its 22 it will monitor the temp everytime

NOW THE KUBE-CONTROLLER-MANAGER MEANS
it has many kube controller running whose job is to check konsi process abhi taak schedule nhi hui hai and konsi abhi nhi chal rhi hai , or which pod is needed to be schdeuled that is one conroller which is constantly checking what is running or not working like node js controller, deployment controller etc
ITS MAIN TASK IS TO START BUNCH OF CONTROLLERS
AND ONE OF THOSE CONTROLLER IS THE NODE CONTROLLER
WE CAN WRITE OUR OWN CONTROLLER ALSO
WHAT ARE THE CONTROLLER DOING?
They are checking the process and all running or not

THIS IS THE ARCHITECTURE OF THE KUBERNETES AND THERE ARE MORE COMPONENTS ALSO

API Server

1. **Handling RESTful API Requests**: The API server processes and responds to RESTful API requests from various clients, including the kubectl command-line tool, other Kubernetes components, and external applications. These requests involve creating, reading, updating, and deleting Kubernetes resources such as pods, services, and deployments
2. **Authentication and Authorization**: The API server authenticates and authorizes all API requests. It ensures that only authenticated and authorized users or components can perform actions on the

cluster. This involves validating user credentials and checking access control policies.

3. **Metrics and Health Checks**: The API server exposes metrics and health check endpoints that can be used for monitoring and diagnosing the health and performance of the control plane.
4. **Communication Hub**: The API server acts as the central communication hub for the Kubernetes control plane. Other components, such as the scheduler, controller manager, and kubelet, interact with the API server to retrieve or update the state of the cluster.

etcd

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.
Ref - https://etcd.io/docs/v3.5/quickstart/

kube-scehduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on. Its responsible for pod placement and deciding which pod goes on which node.

kube-controller-manager

Ref - https://kubernetes.io/docs/concepts/architecture/controller/

The kube-controller-manager is a component of the Kubernetes control plane that runs a set of

controllers. Each controller is responsible for managing a specific aspect of the cluster's state.

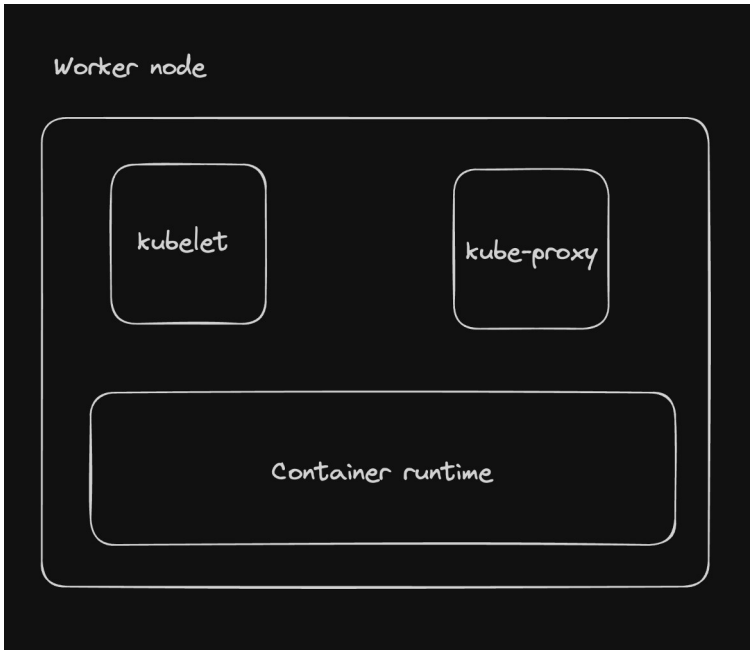
There are many different types of controllers. Some examples of them are:

Node controller: Responsible for noticing and responding when nodes go down.

Deployment controller: Watches for newly created or updated deployments and manages the creation and updating of ReplicaSets based on the deployment specifications. It ensures that the desired state of the deployment is maintained by creating or scaling ReplicaSets as needed.

ReplicaSet Controller: Watches for newly created or updated ReplicaSets and ensures that the desired number of pod replicas are running at any given time. It creates or deletes pods as necessary to maintain the specified number of replicas in the ReplicaSet's configuration.

Worker Nodes - The nodes that actually run your Backend/frontend



Worker machine do the polling part means it always checks for work from the master node abhi kuch aaya kya? Abhi kuch karna hai kya like that

the process which is checking is called as the

kubelet

is taking to the master api node its understanding the request and find if any work is there
its scheduling those jobs on the pod and running those pods also telling like its starting right now , its running right now etc means all the updates also posting all the statues back to the api server master node
kube proxy helps in generating the ip addresss,making you communicate between the pods or the containers that you are starting or running
Conatiner runtime == this is where the container actually running

Three things present in the architecture of Worker node:
kubelet - An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

kube-proxy - The kube-proxy is a network proxy that runs on each node in a Kubernetes cluster. It is responsible for you being able to talk to a pod

Container runtime - In a Kubernetes worker node, the container runtime is the software responsible for running containers.
(similar to docker container)

Common Container Runtimes for Kubernetes

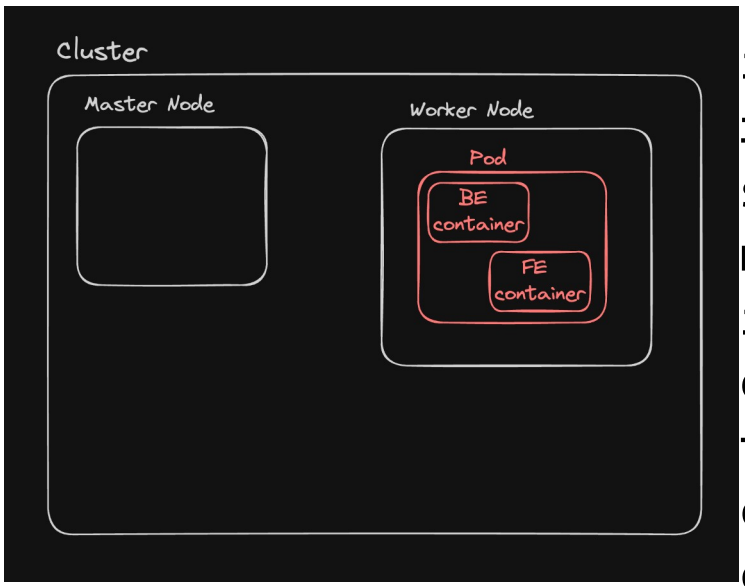
1. containerd
2. CRI-O
3. Docker

Cluster
A bunch of worker nodes + master nodes make up your kubernetes cluster . You can always add more / remove nodes from a cluster.
Images
A **Docker image** is a lightweight, standalone, and executable software package that includes everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and

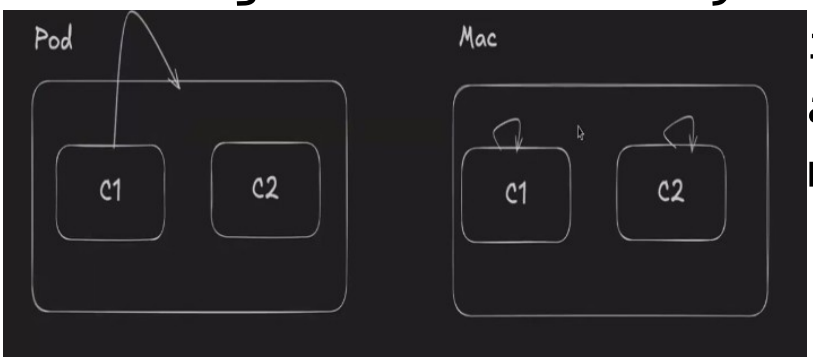
configuration files. Images are built from a set of instructions defined in a file called a Dockerfile.

Containers
A container is an image in execution. For example if you run

Pods
A pod is the smallest and simplest unit in the Kubernetes object model that you can create or deploy



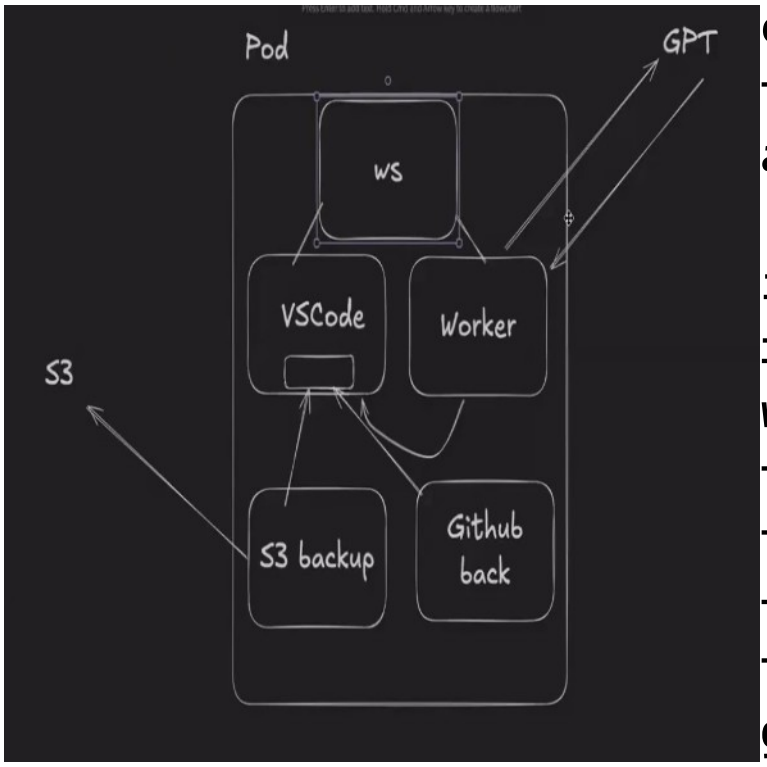
inside the pod a conatiner runs
single pod can also run multiple containers
in this the containers can easily talk to each other
the reason of using two containers in a pod is to connect them in a network to share data or like connect them which means one be can hit another be from there container
pod cannot directly run the node process it can only run a containers
POD is just in kuberenets its kubernetes term
a pod can run multiple containers but most of the time single container only



in a pod they are sharing a same machine, and in mac they are not sharing

Pod helps in running multiple containers together and it helps to scale up together and also scale down together and scale up together atomicaly

there are very specific usecase in which you will run multiple containers in a single pod



example for the usage of the pod in a bolt like application

in this whenever give a request It goes to the worker and the worker asks the gpt and the gpt returns the answer and creates the files while during that time the s3 backup is made, github backup is made etc and these all are in a network so that it can work properly, for every user there is a pod and each pod has a limit(1cpu, 1gb etc)
install the kind, kubectl(use choco to install)
we rarely run k8 cluster on our machine do it on the cloud only most of the time

USING KIND
kind create cluster --name local

kind delete cluster -n local(used to delete a cluster)

control-plane == master node

the create command create a single node which creates a node which ascts like both master and worker node

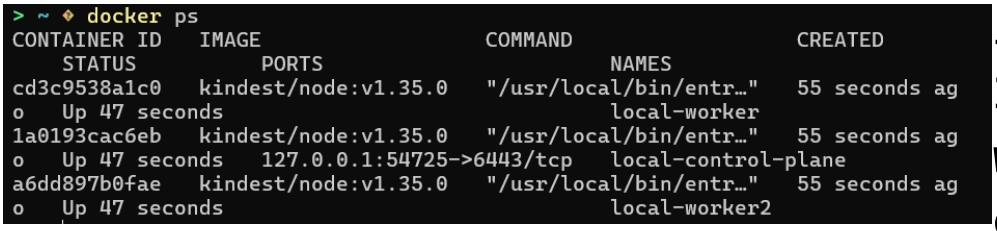
NOW IF WE WANT TO CREATE A MULTI NODE SETUP
CREATE A CLUSTERS.YML FILE

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
- role: worker
```

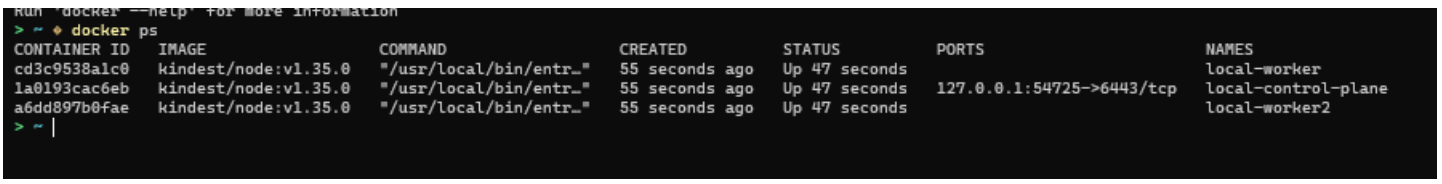
kind create cluster --config clusters.yml --name local

docker ps

this command will show that there are 3 containers running



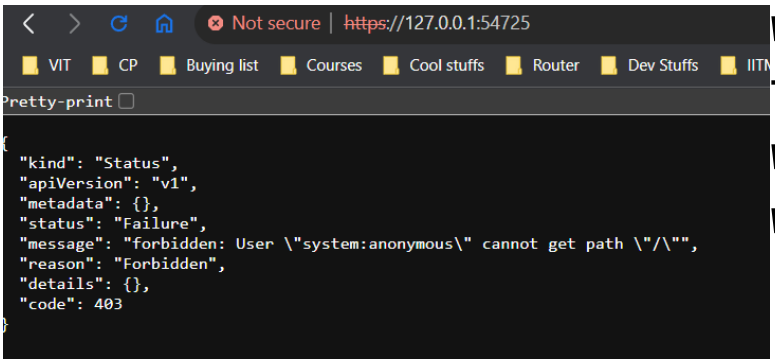
it has 2 workers 1 master node == which is called control plane



zoom out to see the url also, that url can be used to talk to the api server of the master node which we studied in the architecture

https://127.0.0.1:54725

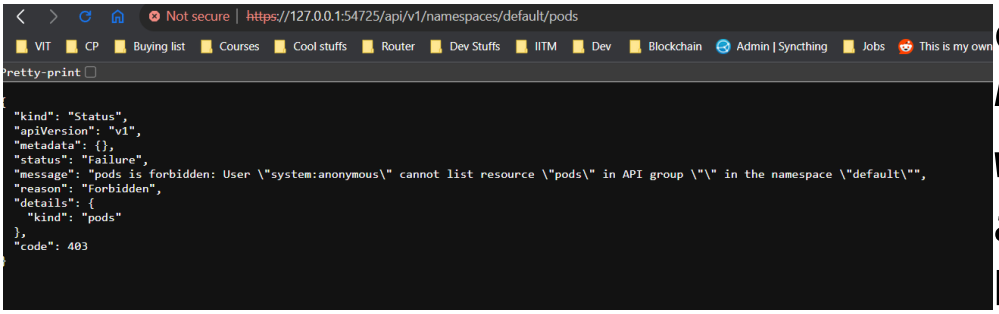
click on proceed bcoz there is no certification for now



whenver we start a cluster there are no pods running we have to schedule them to work with

https://127.0.0.1:54725/api/v1/namespaces/default/pods

this tell the number of pods running

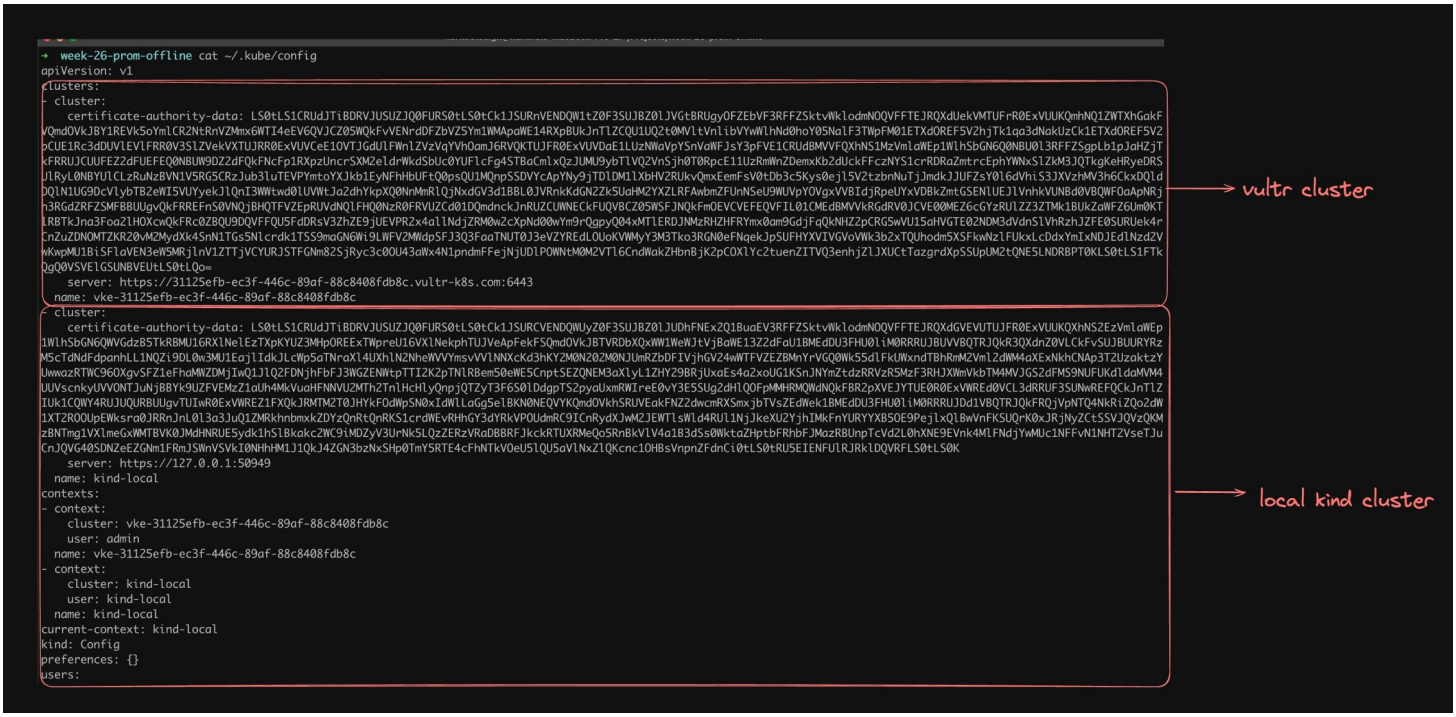


currently it gives 403 error because we have not sent any creds of kubernetes

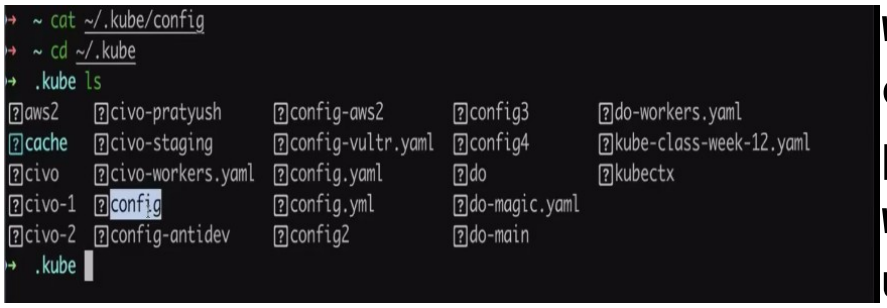
when ever we create a kube is shoves it creds for the cluster in this file

Kubernetes API server does authentication checks and prevents you from getting in.

All of your authorization credentials are stored by kind in **~/.kube/config**



do cat **~/.kube/config**



we keep like all the config file in one place and whenever which one we want to use we just copy that in the config file and then we can use that

now we have the config file but how we can talk to the kube cluster with this config file?

We can use the kubectl

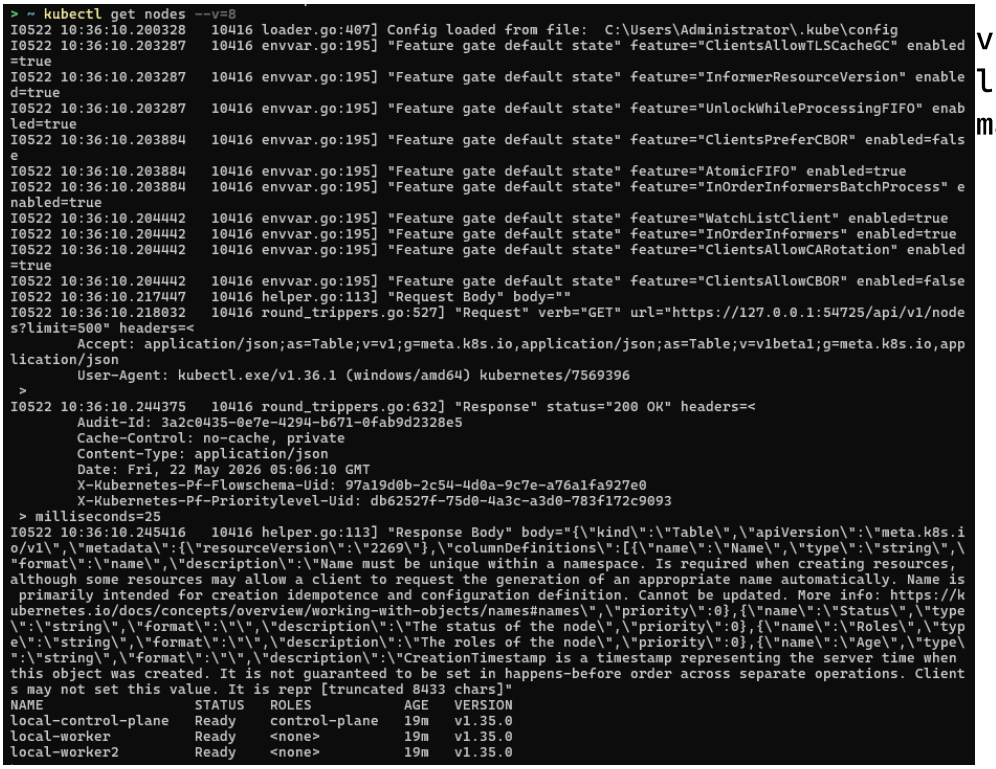
kubectl get pods (get the url of the pods)

this kubectl puts the auth config while talking to the pod so that it will not give the 403 error

kubectl get pods

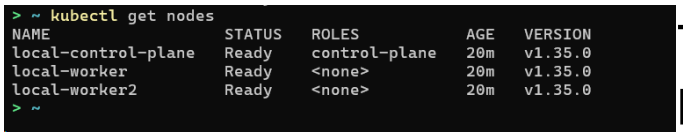
If you want to see the exact HTTP request that goes out to the API server, you can add --v=8 flag

kubectl get nodes --v=8



v == verbose (this means the logs must be printed and as many as possible print them)

kubectl get nodes



this helps us to see all the pods running

(WHENEVER YOU SEE YOUR KUBERNETES CLUSTER IS DOWN YOU USE THIS COMMAND TO SEE IS THE NODE STOPPED?)

We have created a cluster of 3 nodes

How can we deploy a single container from an image inside a pod ?

Starting using docker
docker run -p 3005:80 nginx

Starting a pod using k8s

- Start a pod

  kubectl run nginx --image=nginx --port=80

  in this the first nginx is the name then the image name and the port which is written does not export the pod on that url that is just for the documentation purpose

- Check the status of the pod

  kubectl get pods

- Check the logs

  kubectl logs nginx

kubectl delete pod nginx

```
> ~ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          20s
```

now the pod is running but we don't know on which node the pod is running
kubectl get pods -w(to constantly watch the status of the pod)
kubectl logs nginx
kubectl logs nginx -f(to tail all the logs which are coming)

```
> ~ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             local-worker2/172.18.0.3
Start Time:       Fri, 22 May 2026 10:47:12 +0530
Labels:           run=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.3.3
IPs:
  IP:  10.244.3.3
```

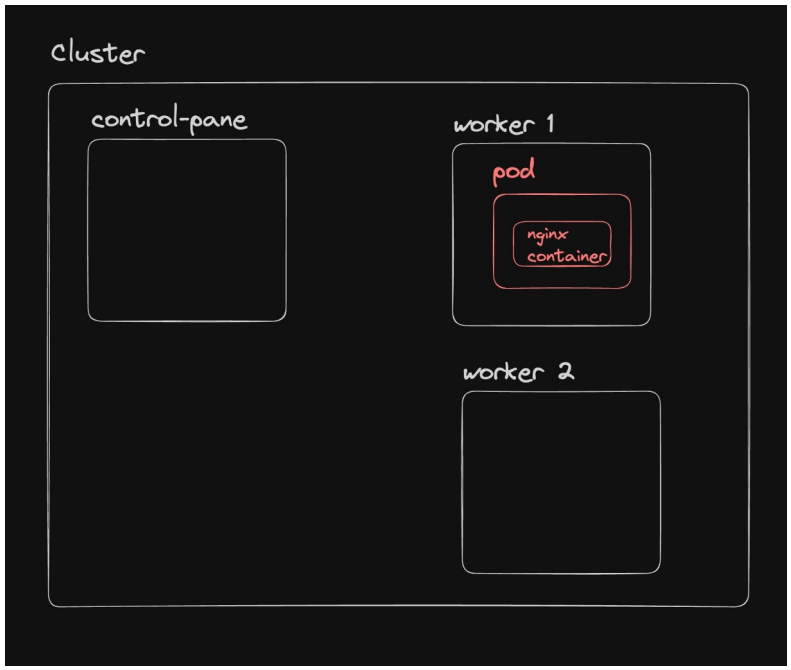
kubectl describe pod nginx (gives bunch of info about the file)

in this we can see on which worker node the pod is running

now to delete a pod kubectl delete pod mongo-pod

What our system looks like right now

Good questions to ask
How can I stop a pod?
How can I visit the pod? Which port is it available on? (this require another kubenetes service)
How many pods can I start?



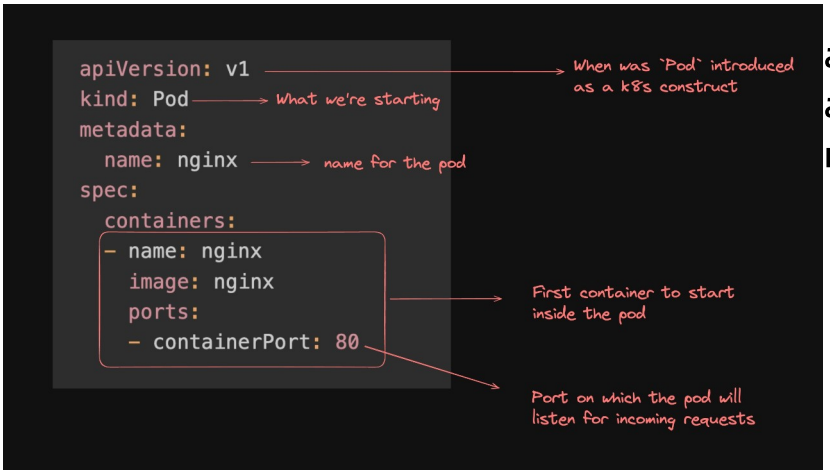
THE WAY WE STARTED THE FILE WAS NOT THE ACTUALL WAY AND THE CORRECT WAY TO USE IT PROPERLY WE HAVE TO DEFINE A MANIFEST.YML FILE AND THEN WE SHOULD USE IT

JUST THE COMMAND DEFINED IN A YML FILE

before we used this
kubectl run nginx --image=nginx --port=80
now we create this manifest.yml file
apiVersion: v1
kind: Pod

```
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

kubectl apply -f manifest.yml(it is completely same to the thing to the command we ran above)
kubectl delete pod nginx



always create the file and apply the file never use the naked command

https://github.com/code100x/staging-ops
explore some of the kubectl files for reference

MAIN USE OF KUBERNETES IS THAT YOU CAN CHANGE THE PLATFORM VERY EASILY LIKE IF YOU ARE USING ASG IN AWS IT WILL BE DIFFICULT TO MOVE TO MIGS IN GCP
NOT TRUE FOR SERVERLESS WORKERS CLOUDFARE TO LAMBDA AWS

THATS WHY ONE OF THE BEST APPROACH TO USE KUBERNETES

kubernetes ko yahi samaj aata hai ki aek pod start kardo uske andar woh container chalu kardega
agar 1 container bhi chalana hua toh bhi pod hii lagega

we should add resource limit and resource minimum for the machine which we have scheduled

KIND = KUBERNETES INSIDE DOCKER

ls ~/.kube/config (at this location the config file is present) this holds the creds of the cluster
and whenever you want to
cat ~/.kube/config (to print the config file)

set alias k=kubectl
its similar to docker if we close the pod the data get lost and like whenever we used to store the data we used volumes which helps us to persist the data

for fetching data use the naked commands of kubectl but not for pushing some data

you should always create a manifest.yml file

vim manifest.yml (create the file)
kubectl apply -f manifest.yml

SO IMAGINE YOU WANT TO RUN TWO NGINX PROCESS FIRST PRICIPLE SAYS
JUST CREATE ONE MORE MANIFEST2.YML AND THEN JUST CHANGE THE NAME TO NGINX2 AND THEN START IT
WHEN YOU START THAT THEN YOU CAN USE IT
BUT THE ISSUE IS IF SOMEHOW ONE CRASHES IT CANNOT START ON ITS OWN THERE IS ONLY A THING COMES CALLED AS

REPLICASET

A ReplicaSet in Kubernetes is a controller that ensures a specified number of pod replicas are running at any given time. It is used to maintain a stable set of replica Pods running in the cluster, even if some Pods fail or are deleted.

SO IF YOU GIVE IT A TASK TO RUN LIKE 5 MACHINES IT WILL ENSURE THAT ITS SAME LIKE THE ASGS AND IF YOU START A 6TH MACHINE THEN ALSO IT WILL CLOSE THAT AND WILL RUN ONLY 5

Create a replicaset

Let’s not worry about deployments, lets just create a replicaset that starts 3 pods

IN THIS ALSO WE HAVE TO CRATE A YML FILE

Create rs.yml

```
apiVersion: apps/v1
kind: ReplicaSet (NOW THIS IS THE NAME OF THE REPLICASET)
metadata:
  name: nginx-replicaset
spec:
  replicas: 3 (NO OF PODS)
  selector:
    matchLabels:
      app: nginx
  template: (HAME KYA START KARNA HAI WOH YAHA DEFINE KARO)
    metadata:
      labels:
        app: nginx(NAME OF THE POD)
    spec:
      containers:
      - name: nginx (NAME OF THE CONTAINER THIS IS THE PREFIX OF THE CONTAINER)
        image: nginx:latest
        ports:
        - containerPort: 80
```

THE SELECTOR THING HELPS THE REPLICASET TO UNDERSTAND WHICH PODS WAS STARTED BY HIM AND HELPS TO IDENTIFY THAT HE STARTED THESE WITH THEES NAMES, AND IF WE START A CONTAINER WITH THAT SAME NAME ONLY UNFORTUNATELY IT WILL NOT ABLE TO IDENTIY THE DIFFERENCE BUT IT WILL KILL THE EXTRA THEN BCOZ IT WILL THINK THAT ALSO BELONG TO ME AND THEN WILL MANTAIN THE LIMIT

vim rs.yml
cat rs.yml
kubectl apply -f rs.yml
replicaset.apps/nginx-replicaset created
kubectl get replicaset

NAME	DESIRED	CURRENT	READY	AGE
nginx-replicaset	3	3	3	26s

kubectl get pods (you will see three pods are running)

SUPPOSE WE INCREASED THE NUMBER OF MACHINES IN THE YML FILE THEN JUST DO

kubectl apply -f rs.yml
replicaset.apps/nginx-replicaset configured

kubectl delete rs rs_name (helps delete the rs)
kubectl get pods -w (watch mode)
kubectl describe pod pod_name (to actually see whats happ)

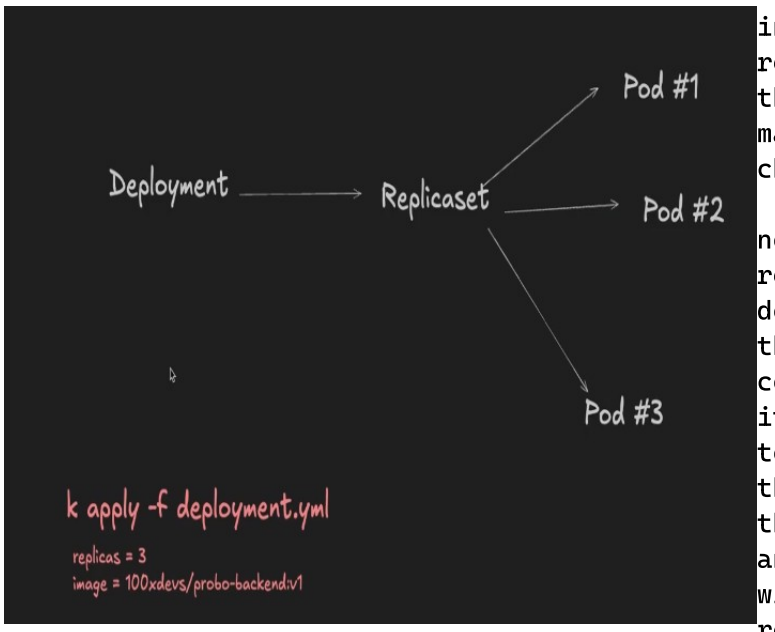
ENJOY

PRODUCTION MAI MAINLY USE HOTA HAI DEPLOYMENT NOT REPLICASET

Deployment
A **Deployment** in Kubernetes is a higher-level abstraction that manages a set of Pods and provides declarative updates to them. It offers features like **scaling**, **rolling updates**,

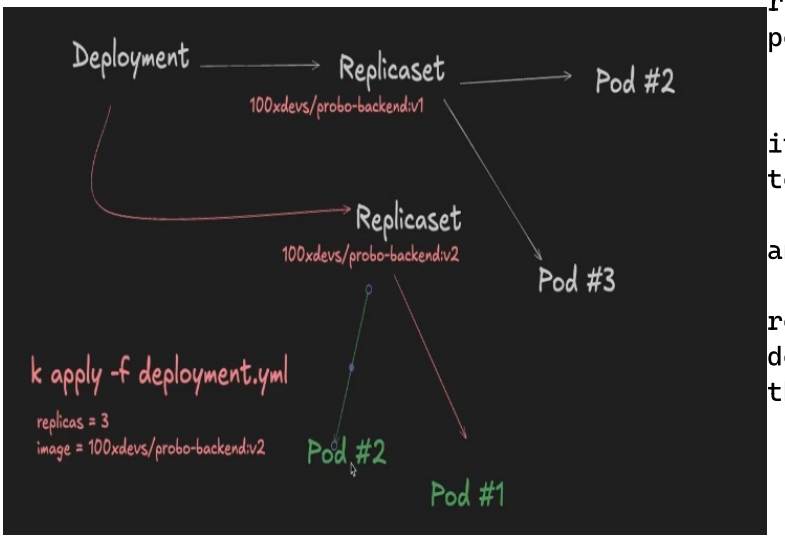
and **rollback capabilities**(can get back to older version to), making it easier to manage the **lifecycle of applications**(JUST THE PUSHING, DEPLOYING CODE BY THE DEVS).

WHAT IS ROLLING UPDATE?
Its like imagine you had 10 machine and new code is pushed so to run the new code it will not stop the whole machine directly it will start one machine with the new code and will stop one machine with the old code this is the way it will do



initially you create a deployment which creates a replicaset and the replicaset creates 3 pods
these 3 pods has the image code mentioned now the main thing comes how it works when we make some changes to the code

now whenever a new file or code is pushed a new replicaset is made which has the code for the new docker file
the procedure of making the machine run the new code is very simple it uses rolling updates
it starts one pod in the new replicaset and try to run the code if it works it marks green and then closes one pod of the older replicaset
this happens until all the pods are shifted
and if any false code is done in docker then it will never able to make a pod green and will return error but then also the old code will persist and the server will not go down



its like when the new pod become healthy then it tells the old replicaset to close the pod

and then the traffic is sent to new pod

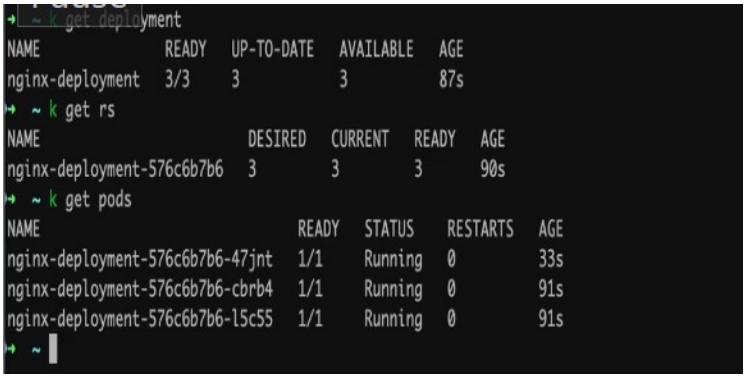
replicaset manages the pod
deployment manages the replicaset and also does the rolling updates

Create a deployment
Lets create a deployment that starts 3 pods
before that deleting the old rs
kubectl get rs

kubectl delete rs nginx-replicaset

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

create a deployment.yml file
kubectl get deployment
kubectl get rs
kubectl get pods



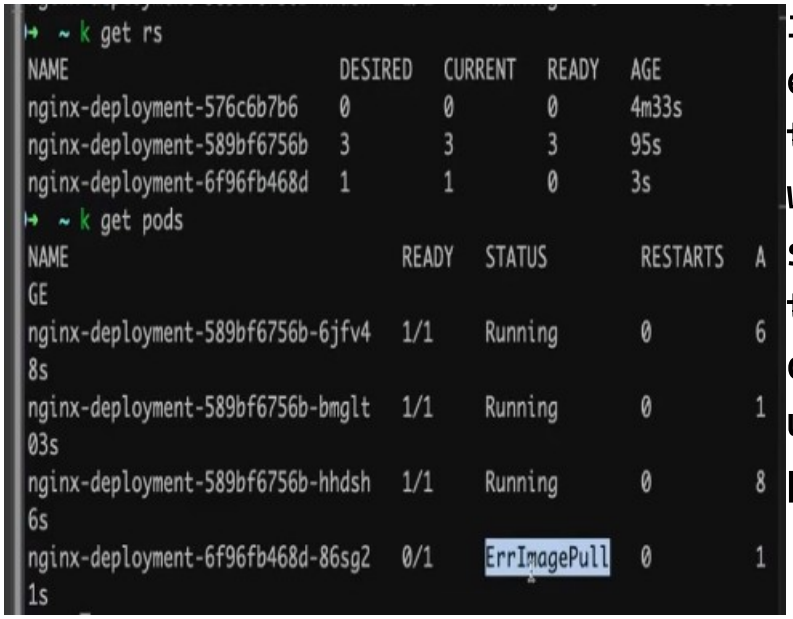
kubectl apply -f deployment.yml

THE MAIN THING OF THIS IS LIKE IF I DELETE NOW ONE OF THE POD FILE IT WILL AUTOMATICALLY CREATE A NEW ONE AGAIN IN SECONDS BECAUSE IN THE REPLICASET THE DESIRED COUNT WAS THREE THATS WHY

UPDATING THE DEPLOYMENT FILE
changing the nginx:latest to mongo:latest
then do
kubectl apply -f deployment.yml

now with this you can watch the changes
kubectl get pods -w
kubectl get rs (to see the changes currently happening like rolling update thingy)

```
> ~ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-59f86b59ff   3         3         3       20m
nginx-deployment-68c69f7975   1         1         0       28s
> ~ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-59f86b59ff   3         3         3       20m
nginx-deployment-68c69f7975   1         1         0       34s
```



if we add a name of a non existing image it will show the replicaset is running but when we see the pods it will show the ErrImagePull
the good thing about this the old code is still running until the proper code is not pushed

YOU CAN ALSO DO DEPLOYMENT ROLLBACK CHECK GOOGLE WHEN REQUIRED
kubectl rollout undo deployment/nginx deployment --to revision=1
TO SCALE THE APPLICATION
kubectl scale deployment nginx-deployment --replica 1
kubectl get pods

MOST OF THE TIME WE WILL CREATE DEPLOYMENT ONLY NOT THE REPLICASET NOR THE PODS INDIVIDUALLY

```
➜  ~ k get pods
NAME                    READY   STATUS    RESTARTS   AGE
nginx                   1/1     Running   0          2m43s
nginx-replicaset-2fdb8  1/1     Running   0          82s
➜  ~ cat rs.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
➜  ~
```

the matchlabel name helps to create the difference between the pods created but the issue is whenever we create one pod with the same name deployment get confused and see I have to start 2 but one is already working so I will create only one you can see that in the name of the file

SERVICES IN KUBERNETES

now we created a application but it is not exposed to the world to do that we use services
SERVICES ARE USED TO HOST THE POD INSIDE , OUTSIDE THE CONTAINER AND MACHINE AND TO THE OTHER PODS TOO LIKE IF ONE POD HAS A CODE OF THE NODE AND OTHER HAS THE CODE FOR MONGO THEY NEED TO CONTACT THEY CAN USE KUBERNETES
THERE ARE THREE TYPES OF SERVICES:

1. NodePort
2. ClusterIP
3. LoadBalancer

we can setup this on digitalocean,vulter,gcp and aws(tricky) but you can also setup the offline but the loadbalancer will not work in the local machine

go to digital ocean and crate a kubernetes cluster from the create button

for **offline setup** we were using kind and we can use that again
create a kind.yml file

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  extraPortMappings:
  - containerPort: 30007
    hostPort: 30007
- role: worker
- role: worker
```

the above one was complicated the simple one is this

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
- role: worker
```

we saved this one earlier in the clusters.yml
now to create use this
kind create cluster --config clusters.yml -n local
n is for the name
this is setuped locally and in the digital ocean its take time to start

if the local one is working then you can do docker ps to check if running or not

```
> ~ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS                       NAMES
514c1849fd61   kindest/node:v1.35.0   "/usr/local/bin/entr…"   About a minute ago   Up About a minute                               local-worker2
60e97a9ad837   kindest/node:v1.35.0   "/usr/local/bin/entr…"   About a minute ago   Up About a minute                               local-worker
b3cceb794953   kindest/node:v1.35.0   "/usr/local/bin/entr…"   About a minute ago   Up About a minute   127.0.0.1:60856->6443/tcp   local-control-plane
> ~ kubectl get pods
No resources found in default namespace.
> ~ kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   2m38s
> ~ kubectl get deployment
No resources found in default namespace.
> ~ kubectl get rs
No resources found in default namespace.
> ~
```

in this you can get the info

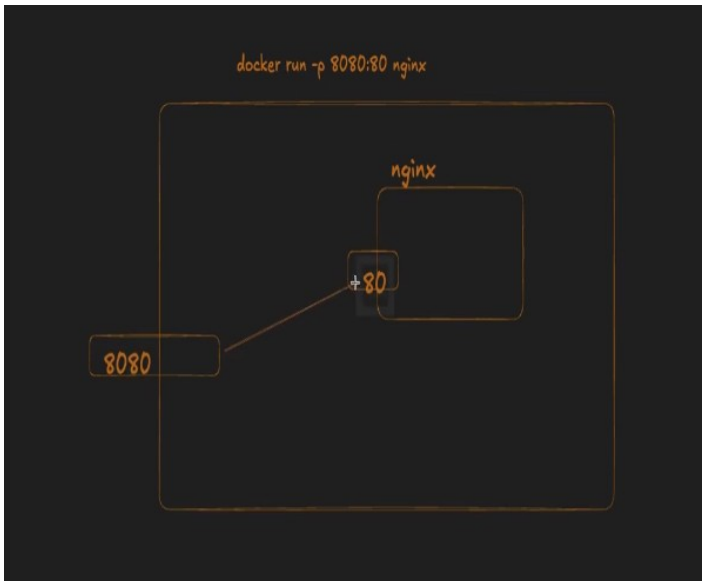
now we are creating a pod
we need to first create a yml file

```
> ~ cat manifest.yml
apiVersion: v1
kind: Pod
metadata:
    name: nginx
    labels:
      app: nginx
spec:
    containers:
    - name: nginx
      image: nginx
      ports:
      - containerPort: 80
      - containerPort: 443
```

now you can see in this the pods ports are exposed but then also you cannot use the ip to access the pod
creating the pods
kubectl apply -f manifest.yml
with
kubectl get pods
you can see the pods running

kubectl logs -f nginx

other than this we know
docker run -p 8080:80 nginx (this will start the nginx on the containter port 80 and on the system port 8080 it will point to container 80 port to be used)




```
> ~ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          5m34s
> ~ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                       NAMES
514c1849fd61   kindest/node:v1.35.0   "/usr/local/bin/entr…"   12 minutes ago   Up 12 minutes                               local-worker2
60e97a9ad837   kindest/node:v1.35.0   "/usr/local/bin/entr…"   12 minutes ago   Up 12 minutes                               local-worker
b3cceb794953   kindest/node:v1.35.0   "/usr/local/bin/entr…"   12 minutes ago   Up 12 minutes   127.0.0.1:60856->6443/tcp   local-control-plane
> ~
```

for now as you can see the pod is runnning the nginx code on some url but we cannot visit it for that we need to use services of kbnetes

to get the specific ip of a pod we use
kubectl get pods -owide(gives more details about the pod)

NOW TO EXPOSE IT WE NEED TO USE SERVICES

Services

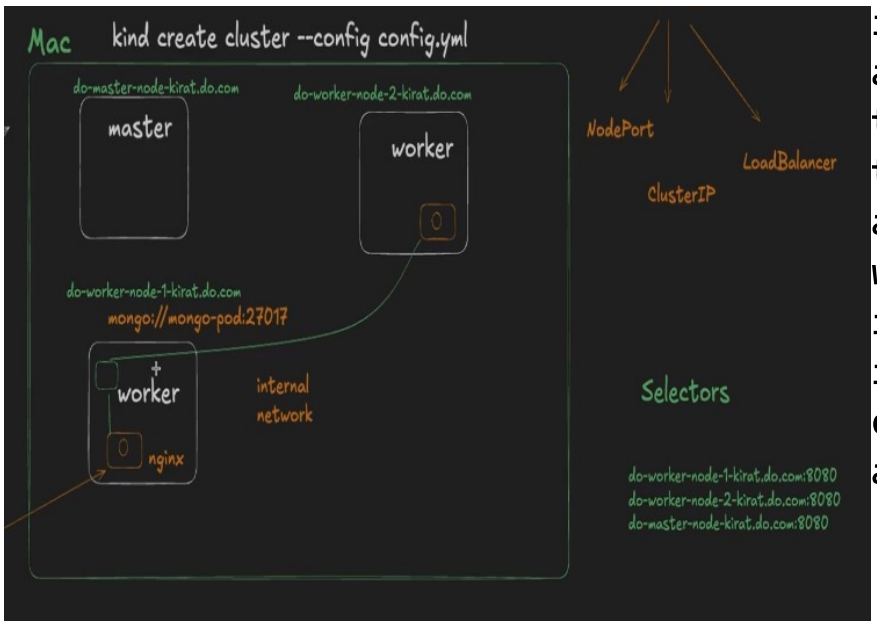
In Kubernetes, a "Service" is an abstraction that defines a logical set of Pods and a policy by which to access them. Kubernetes Services provide a way to expose applications running on a set of Pods as network services. Here are the key points about Services in Kubernetes:

Key concepts

1. **Pod Selector**: Services use labels to select the Pods they target. A label selector identifies a set of Pods based on their labels.(THIS IS USED FOR MENTIONING WHICH ALL PODS SHOULD BE EXPOSED IN THE NETWORK AND TO BE USED,AND ALSO IF ONE POD DIES IT AUTOMATICALLY REMOVES THAT ALSO)
2. **Service Types**:
    - **ClusterIP**: Exposes the Service on an internal IP in the cluster. This is the default ServiceType. The Service is only accessible within the cluster.
    - **NodePort**: Exposes the Service on each Node’s IP at a static port (the NodePort). A ClusterIP Service, to which the NodePort Service routes, is automatically created. You can contact the NodePort Service, from outside the cluster, by requesting <NodeIP>:<NodePort>.
    - **LoadBalancer**: Exposes the Service externally using a cloud provider’s load balancer. NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created.
3. **Endpoints**: These are automatically created and updated by Kubernetes when the Pods selected by a Service's selector change.

EASIESET WAY TO EXPOSE A POD USE

NODEPORT:This is the most easiest one but the issue is anyone can access and can do a ddos attacks and all



imagine the urls of the pods are the green one then with the url:8080 port in which they were exposed can be used and also one ip can show any worker pods it is not necssary if the url is for the 1st pod it will show the first one only it can show the 3rd one also

the labels which is present in the yml file are very imp and this also have the SELECTOR thing which means it assigns the url name based on the app name only like if its nginx all the request will be transferred to that pod(that's why while accessing the url it can be any pod)
idhar joh service.yml file mai selector hoga who udhar labels that replicaset mai

```
> ~ cat manifest.yml
apiVersion: v1
kind: Pod
metadata:
    name: nginx
    labels:
      app: nginx
spec:
    containers:
    - name: nginx
      image: nginx
      ports:
      - containerPort: 80
      - containerPort: 443
```

what is the difference between this two files codes?
First one does not have label and second one does
name is different
and whenvere nameport will run it will forward request to second one if the lable is nginx and not to the first one

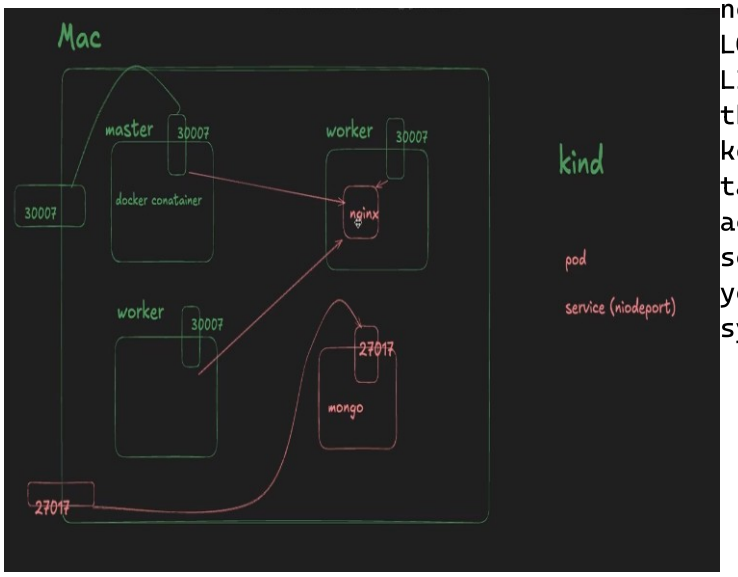
```
→ ~ cat manifest2.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx2
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    - containerPort: 443
→ ~ k man
→ ~ cat manifest.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    - containerPort: 443
---
```

this same label thins in service.yml called as a selector which gives labels to the pods(this is how it will find who to forward the req to)
ports: konse konse ports expose karne hai
service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80(yeh woh port hai jispar hamara pod ka code chal rha hai jaise nginx mai default port 80 hota hai isliye idhar 80 hai usko change be kar skte hai)
      nodePort: 30007  (IS PORT PAR CODE EXPOSE HOGA SYSTEM PAR HII BUT IN A RANGE 30K-50K)
  type: NodePort
```

default ports
postgres = 5432
mongo = 27017
THIS MEATHOD IS OK BUT WE DONT USE IT
now create a service.yml file
kubectl apply -f service.yml
kubectl get svc

now for the digitalocean machine we can just get the ip and then ip:30007 to access the code but for the local its running on our machine only



now the issue in this is THIS HASSLE IS JUST FOR THE LOCAL SETUP
LIKE now we have did everything correct but for now the port is directing to the pod but pod ki khudki koi accessible ip nhi hai jaise digitalocean ki hai tabhi woh chij kae liye hame khud phele usko accessiable banana padega woh karne kae liye hame service.yml mai kuch change karna padega
yeh dikkat online walae mai nhi hogi kuyki uska system exposed hai

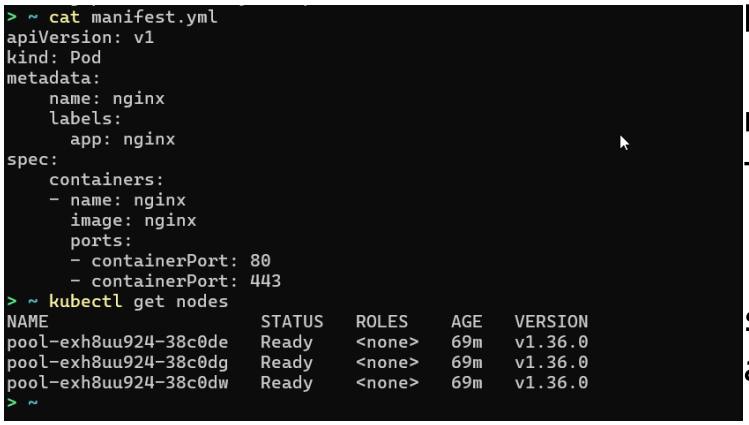
toh hamm bss service kae port ko kind.yml file mai add kardenge takki who request forward ho ske and access ho ske
Restart the cluster with a few extra ports exposed (create kind.yml)

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  extraPortMappings:
  - containerPort: 30007(yeh hogya yaha add)
    hostPort: 30007
- role: worker
- role: worker
```

LEAVE THIS AND NOW USING THE DO WEBSITE MACHINE
go to get started and get the config file from the manual page
this is like a pass to the DO(DIGITAL OCEAN) cluster
download that file
open this
cd ~/.kube
then do start .(windows), linux (open.) and paste the file here
and just copy the new file login to the config file

cp .\k8s-1-36-0-do-0-blr1-1779560482617-kubeconfig.yaml .\config
just do this to copy the file data
now just do kubectl get pods (to get those clusters)

kubectl get nodes(to see the nodes running)
these same name are in the resources section of the website
lets apply the manifest.yml which we created earlier



kubectl apply -f manifest.yml

now at kubectl get pods we can see them running

see this was so easy to deploy the applications

agar yahi vm par karte toh ssh karna padhtha fhir npm install karo, fhir docker fhir, docker file run karo etc etc

kubectl logs -f nginx (to see the logs)
kubectl describe pod(to see on which pod the code is running)
now lets install the service

```
> ~ cat service.yml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30007
  type: NodePort
> ~ kubectl apply -f service.yml
service/nginx-service created
> ~ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.109.0.1      <none>        443/TCP        76m
nginx-service   NodePort    10.109.17.232   <none>        80:30007/TCP   12s
> ~
```

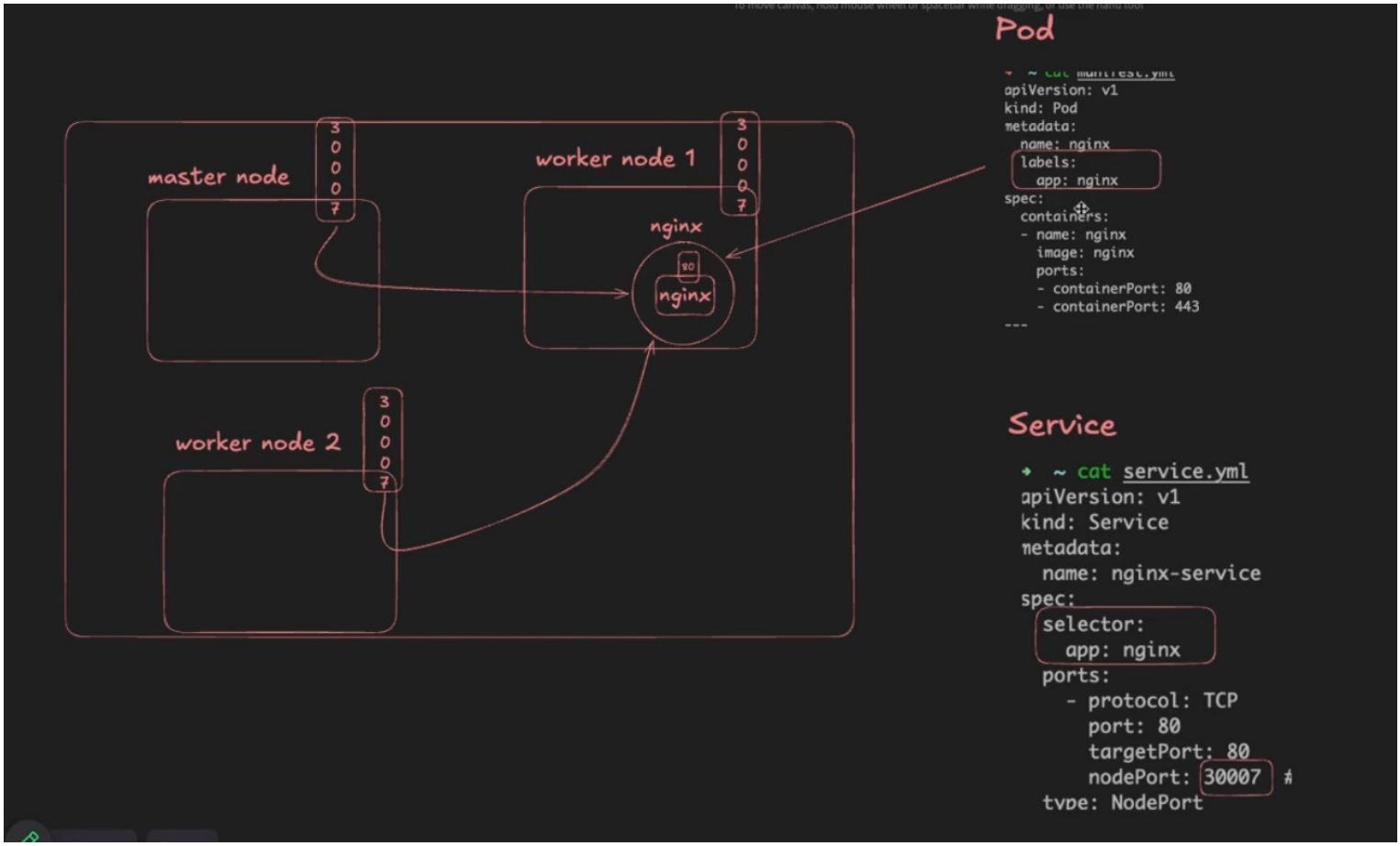
now we ran the service using kubectl apply -f service.yml

we have one pod running and one service running
this service is of type nodeport
kubectl get nodes
kubectl describe nodes(FROM HERE GET THE EXTERNAL IP)
IP:30007 (ENJOY THE WEBSITE)

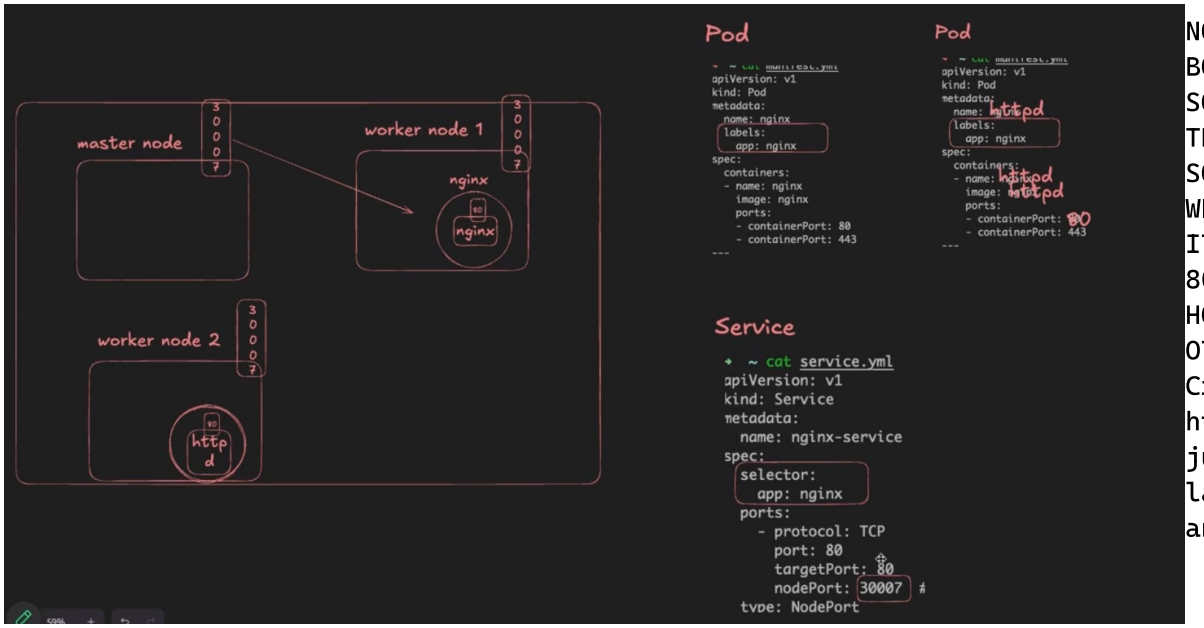
ITS LIKE WITH JUST 2 FILES MANIFEST.YML, SERVICE.YML SAE HII HAMARI FILE KUBERNETES CLUSTER PAR JISSE HAMARI WEBSITE EXPOSE HOGYI INTERNET PAR (BUT NOT HTTPS LOL)
WE CAN USE ANY POD IP TO ACCESS THE CODE
WHICH WE CAN GET FROM
kubectl describe nodes

kind (IT IS JUST FOR LOCAL DEVELOPMENT)
ON DO(THEY ARE RUNNING A BINARY FILES)
NODEPOOL ≡ ITS JUST A MACHINE
ITS JUST LIKE ONE MACHINE IS RUNNING MASTER NODE CODE, OTHER 2 ARE RUNNING WORKER NODE CODE(SAME AS THE ARCHITECURE WE LEARNT)

THIS ALL WAS DONE IN THE NODEPORT



NOW IMAGINE A SCENARIO IN WHICH YOU START A POD WITH SAME LABEL BUT DIFFERENT IMAGE WHAT WILL HAPPEN THEN?



NOW JUST AT THE URL THE BOTH POD WILL RUN BUT SOMETIME IT WILL SHOW THE NGINX CODE, SOMETIME HTTPD CODE(WHY WE CHOOSE HTTPD? BCOZ IT ALSO RUNS DEFAULT ON 80 PORT)
HOW WE APPLIED THE OTHER MANIFEST FILE?
Create a new mainfest-httpd.yml file
just don't change the label change the name and image

now just do
kubectl apply -f mainfest-httpd.yml
using kubectl get pods (you can see that running with the httpd name)
do kubectl describe pod nginx
kubectl describe pod httpd

you can see both are running
now if you check the if sometime nginx code and sometime httpd code

BOOM GHODE KHUL GYE LETS GO ……

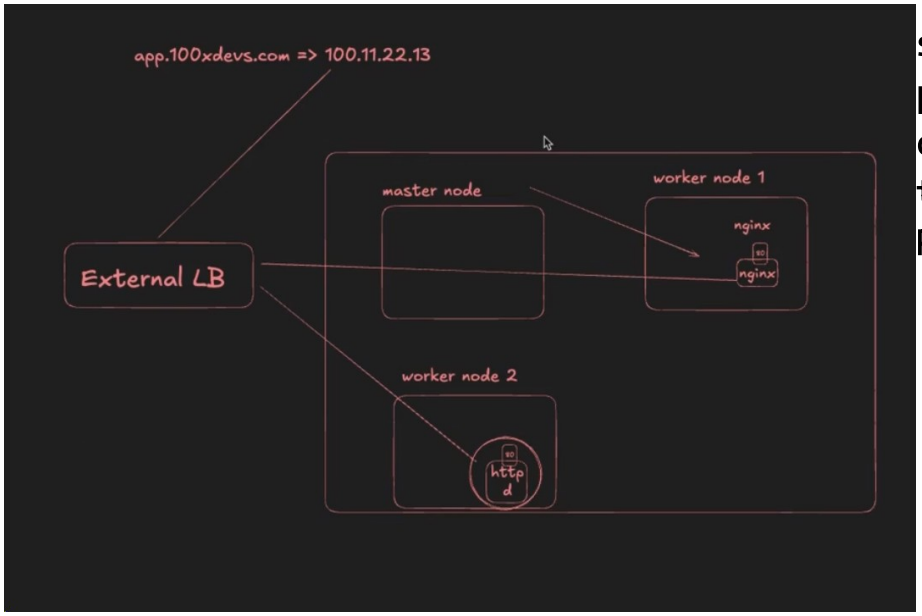
THERE IS AUTOSCALE OPTION IN THE RESOURCS SECTION IN THE DO WE CAN CHECK THAT ALSO

THREE DOWNSIDE OF NODEPORTS

1. the ip is exposed to the world(master node ip is exposed)
2. the port range is always between 30k - 50k
3. it can never get the https security the certificate and all

LOADBALANCER SERVICE

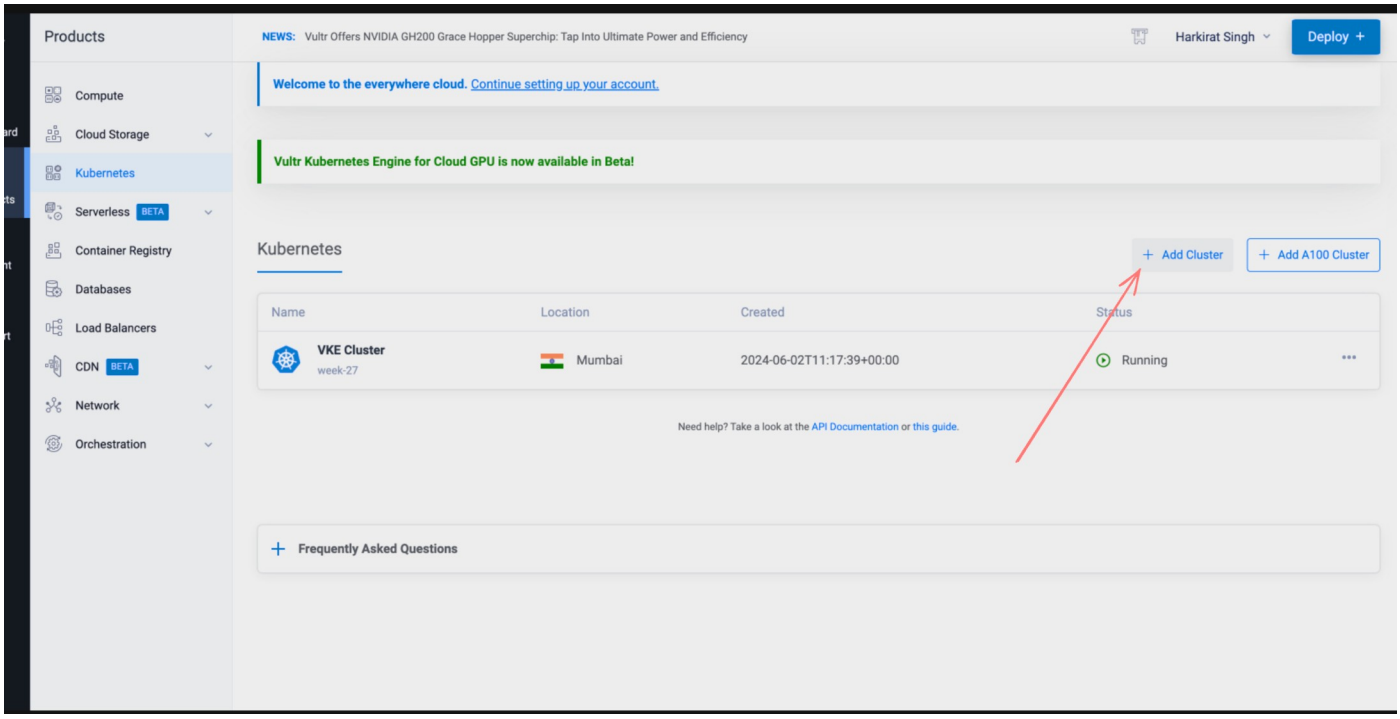
like we exposed the loadbalancer for the ec2 machines, here we can do that for the kuberenetes also
in do in the network section
In this like earlier we were exposing node of the master node and the worker node now we are just exposing the lb ip which will connect to the worker and the master node whenever required



see in the external lb is present and whenver we close the cluster we forget to close the external lb so REMEMBER THAT

In Kubernetes, a LoadBalancer service type is a way to expose a service to external clients. When you create a Service of type LoadBalancer, Kubernetes will automatically provision an external load balancer from your cloud provider (e.g., AWS, Google Cloud, Azure) to route traffic to your Kubernetes service

Creating a kubernetes cluster in vultr



creating service-lb.yml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

```
> ~ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.109.0.1      <none>        443/TCP        15h
nginx-service   NodePort    10.109.17.232   <none>        80:30007/TCP   14h
> ~ kubectl delete svc nginx-service
service "nginx-service" deleted from default namespace
> ~
```

delting the old service because the name of the new service is also the same

btw when you will create the lb it will cost money again differently

kubectl apply -f service-lb.yml
use this to apply

kubectl get svc(this will show the lb is running)

the code is very much similar to the nodeport just the change is not node port and also the external lb Ip is just exposed and we can use multiple ports not just the 30k-50k and also we can do ssh, https also JUST THE ISSUE IS IT WILL CHARNGE EXTRA MONEY FOR THE LB

NOW TO ADD THE IP JUST GO TO THE NAMECHEAP WEBSITE ADD A A RECORD AND ADD THE IP IT WILL POINT TO THAT IP
NOW TO GET HTTPS
GO TO THE LB settings and then the SSL
kubectl delete svc nginx-service

CLUSTERIP

this one is just exposed to the internal node proces like the db and not exposed to the outside world and no one outside person can use it