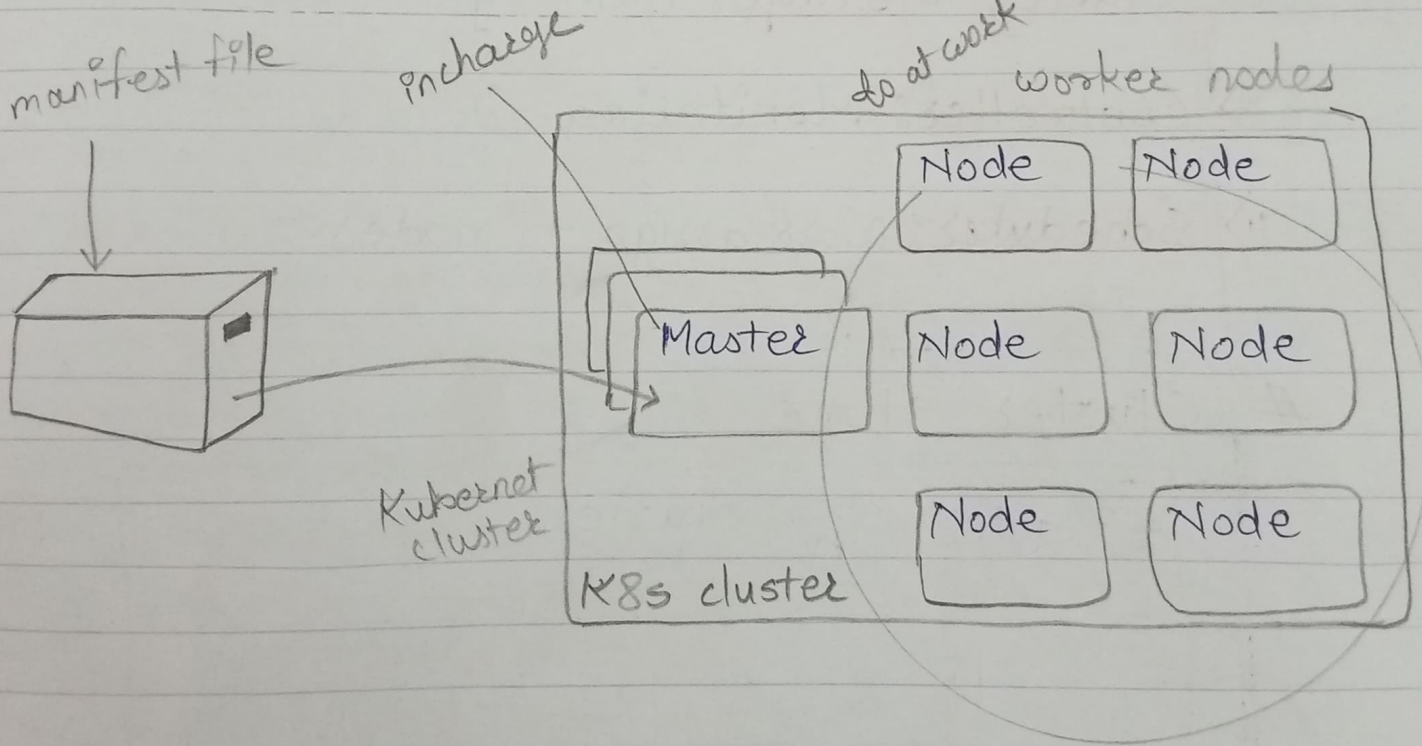
13/23

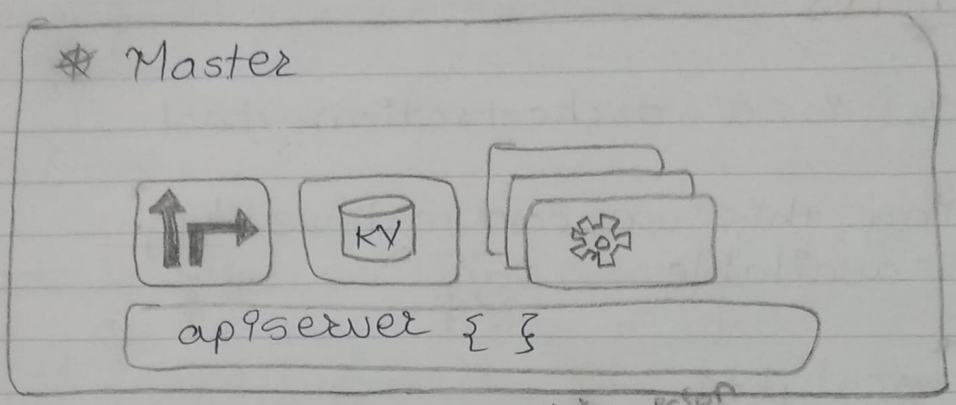
Kubernetes - docker next concept is kubernet

- Kubernetes is a orchestration tool

- By using this we can achieve high
  1. available - if appl° or container stops/down kubernet help to start again or up.
  2. scalable - as per requirements it can scalable
  3. Desired state - if we want how many pods it can be maintained without fails.



scheduler Memory KV controller



* Four Components

Based on controller storage take decision & assign work to scheduler

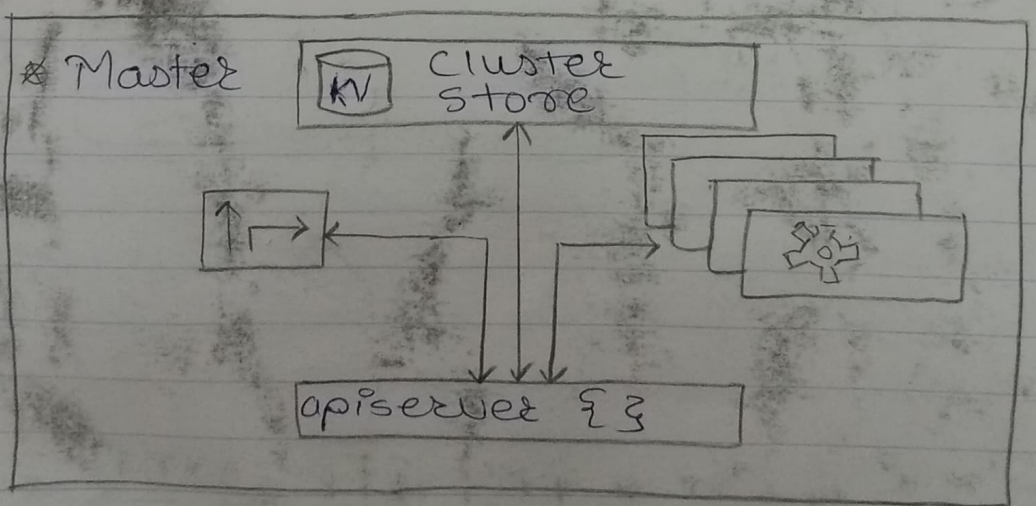
(3) 1) API Server: Authorticat° Person make a decision

(2) 2) Cluster Storage: memory→node data stored into here

(1) 3) Controller: Monitoring - fetched data from storage

(4) 4) Scheduler: Work assign to nodes - Perform the actions given by API server on nodes.

Cluster store

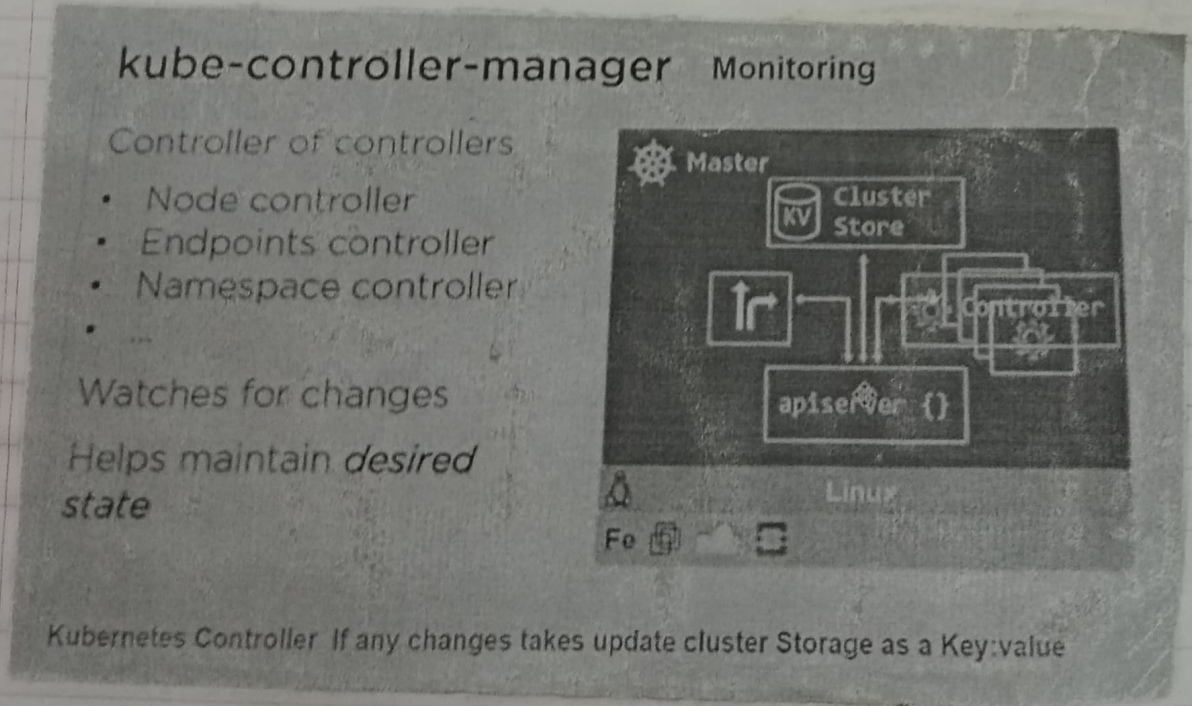


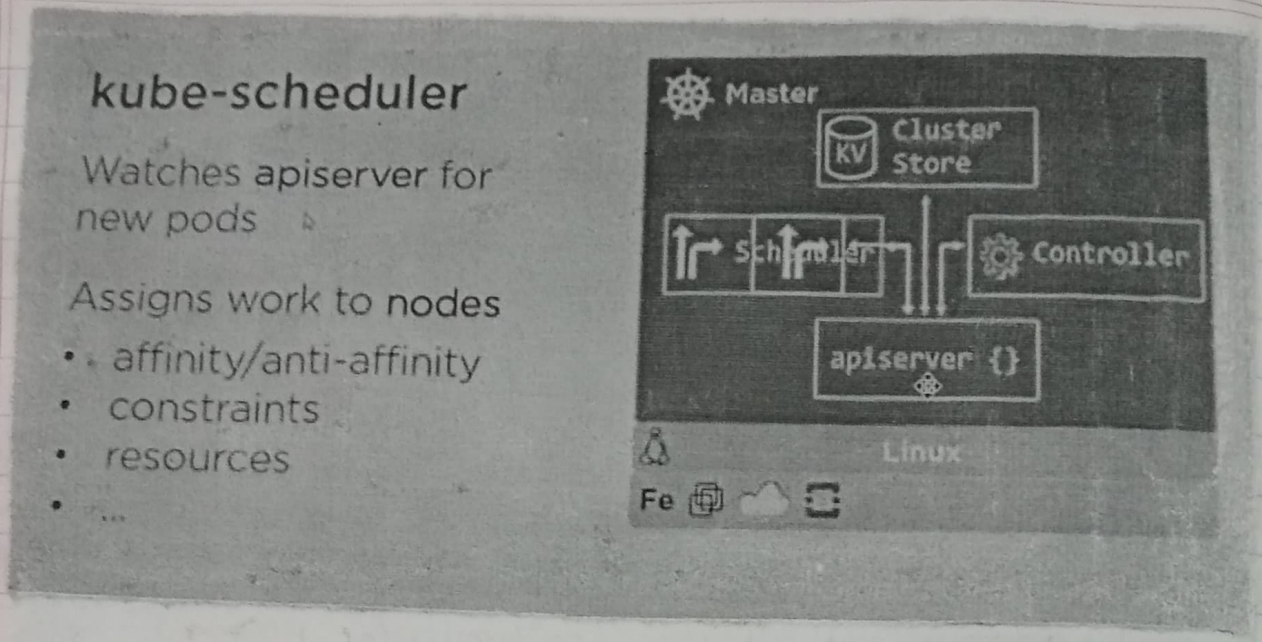
* Persistent storage
*

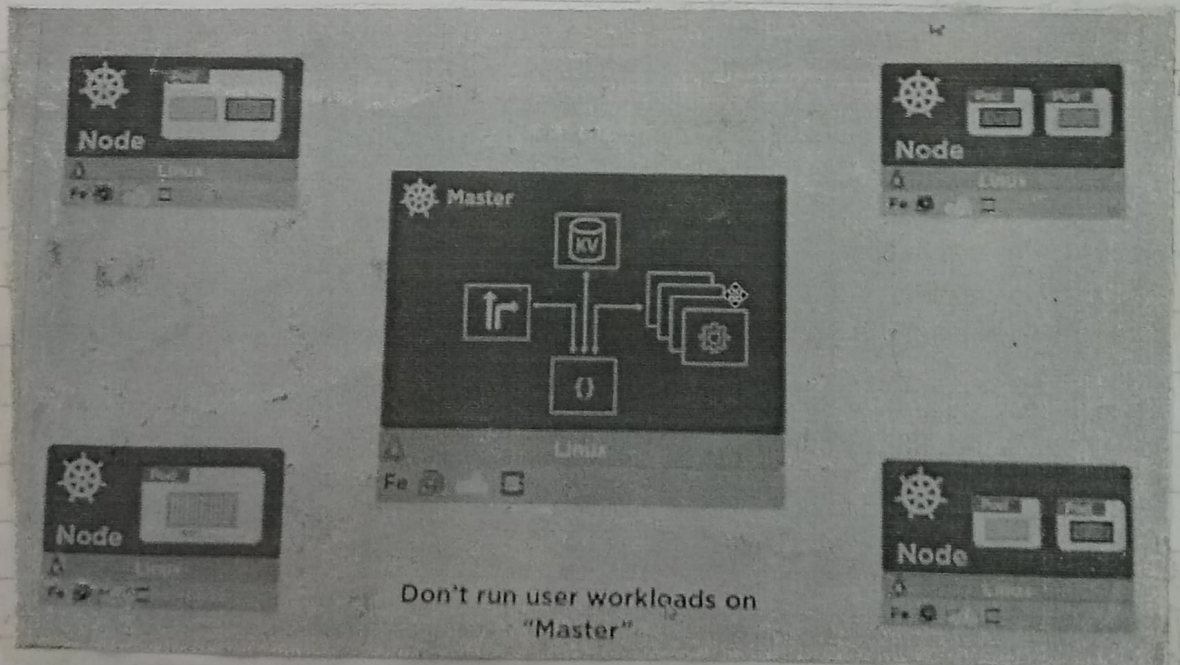
Cluster Storage

- Persistent storage
- Cluster state and config
- Used etcd
- Distributed, consistent, watchable....
- The "source of truth" for the cluster
- Have a backup plan for it !

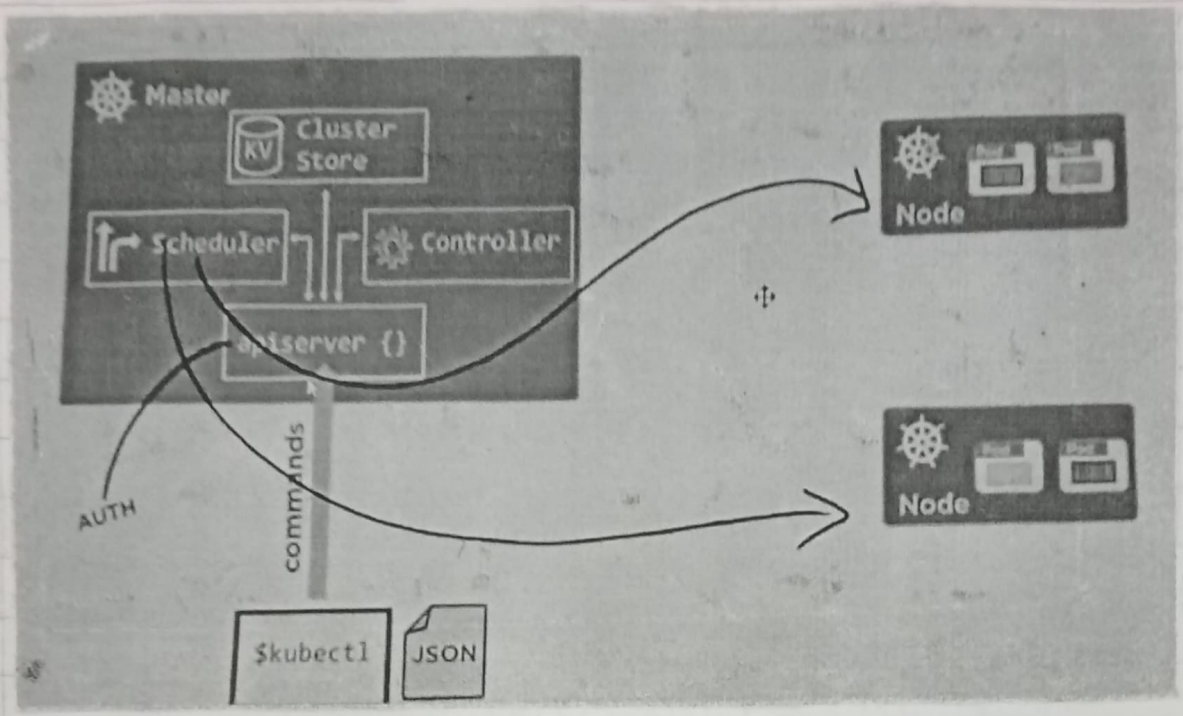
Cluster storage keep every info' in cluster as a Key:Value.
Ex. apiserver is brain which takes decision based on memory.



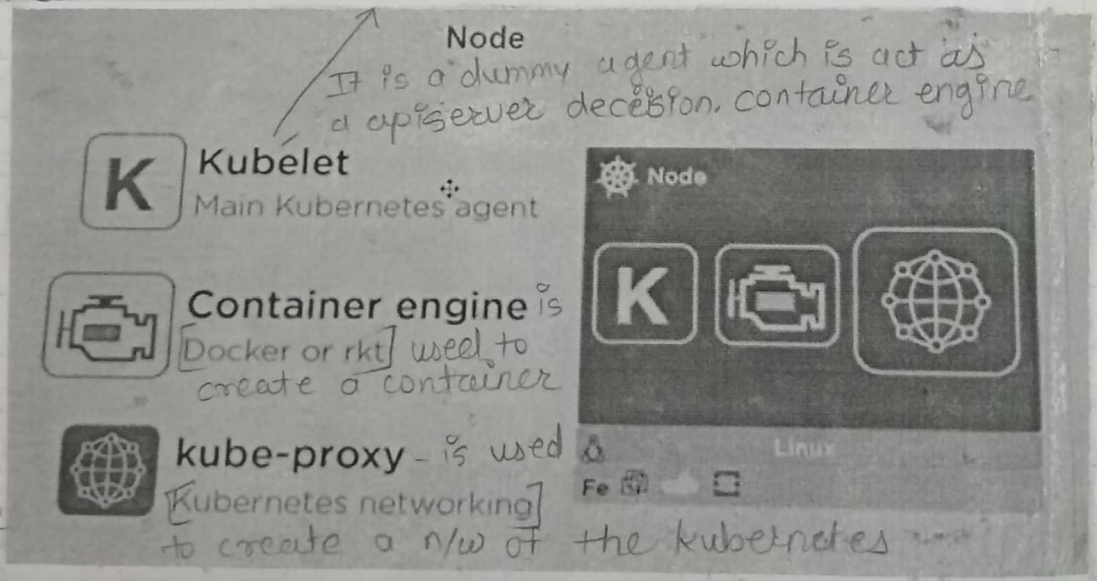




we are not execute any workloads of users on master. We have to runs user workloads on a nodes.

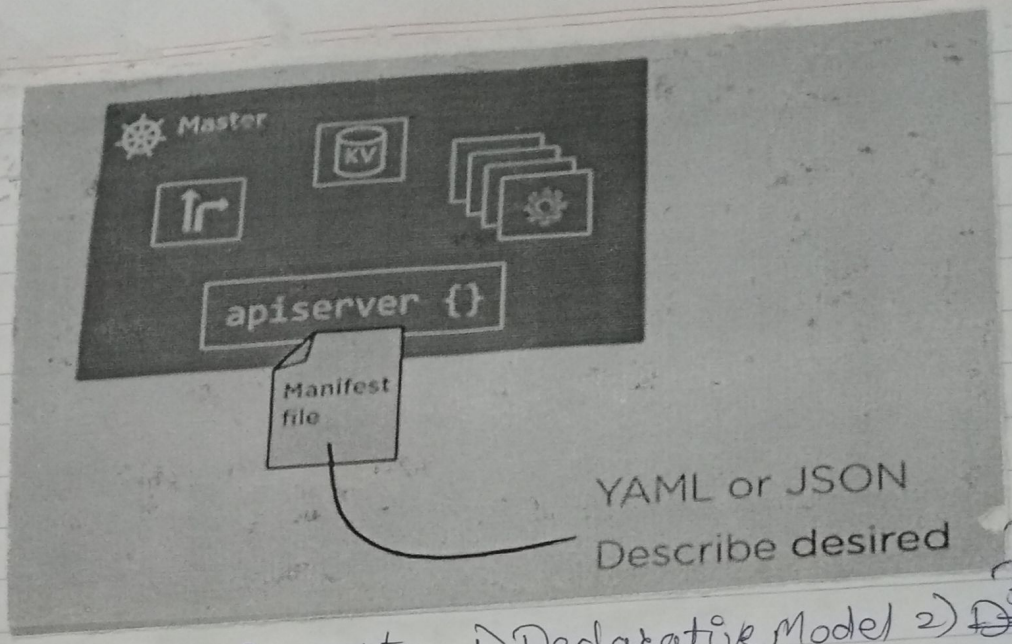


If any pod is deleted then it is responsibility to tell controller then, controlleler tell that to the schedular stored in cluster storage then take APIServer check pod is down the apiserver tell to the schedular tell that to the kubelet. Kubelet then tell that to the container engine. Then container engine create it.



Then we create these all things we have to n/w like n/w conn^n of master, node, pods & container etc In that container we hv to manage all things so that node has Kube-proxy. The main role of Kube-proxy to handle Kubernets n/w. request like one pod to anoth

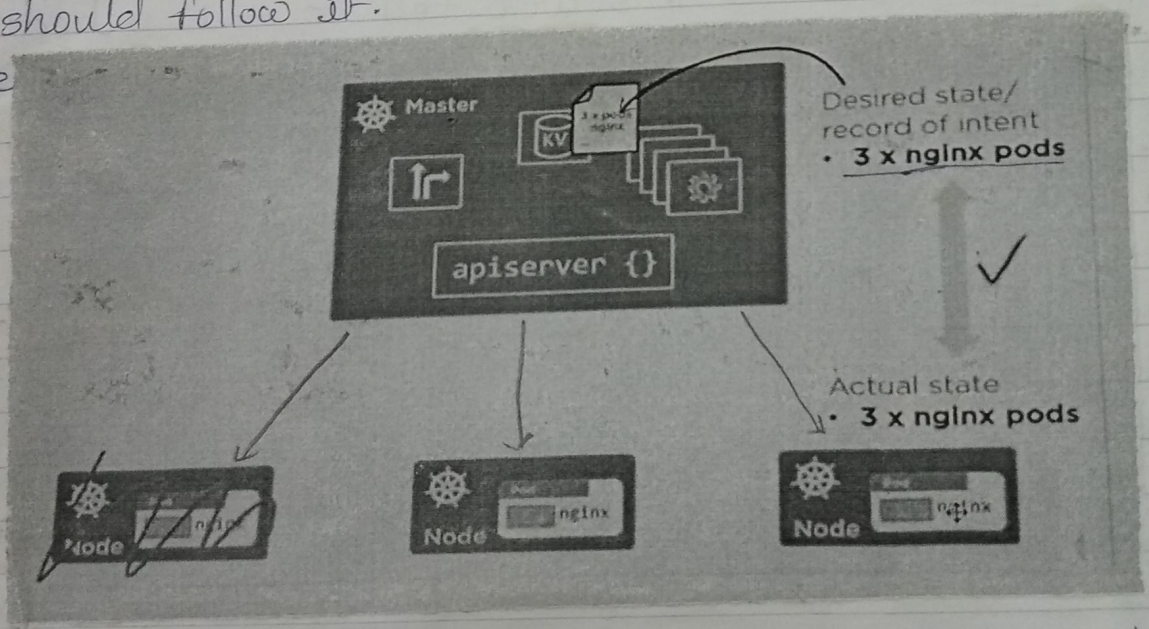
Kubernetes is a declarative model not procedural model

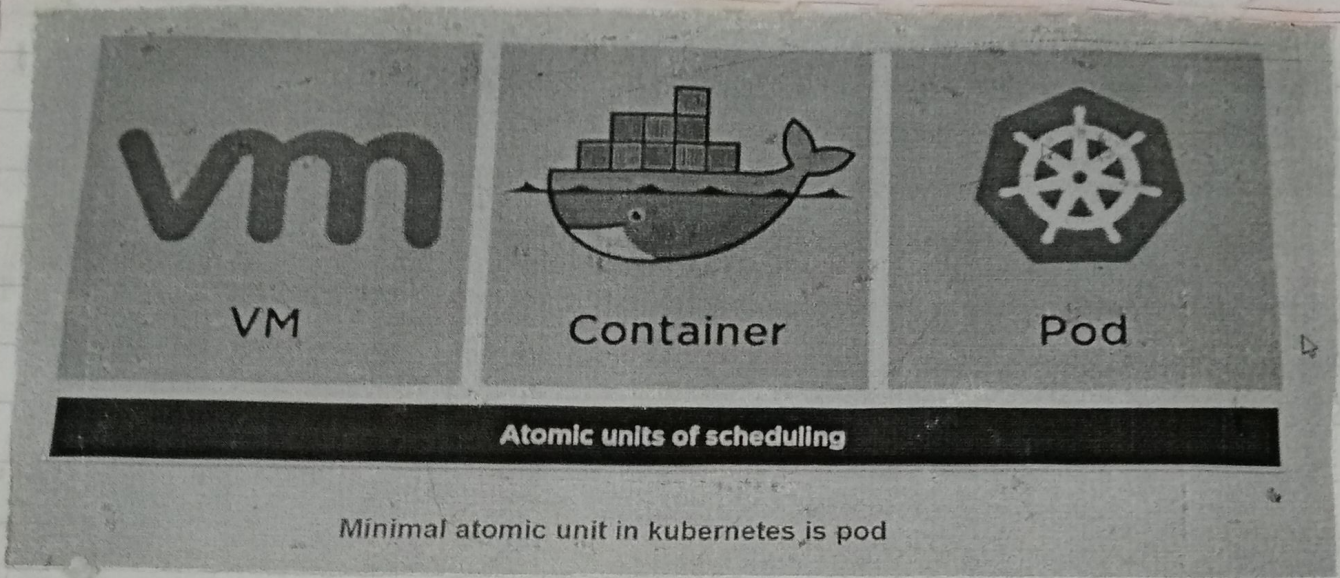


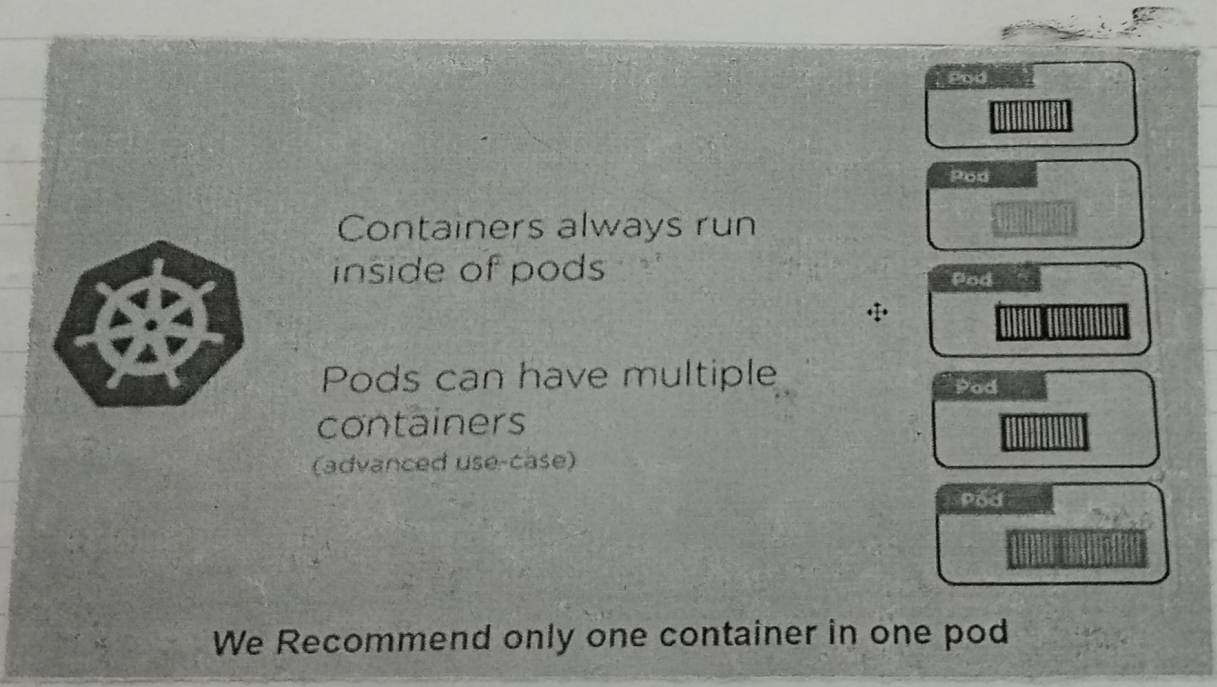
There are 2 parts - 1) Declarative Model 2) ~~Describe desired~~ Desired State

1) Declarative Model - If we have any requirements then we ho to tell Kubernetes then Kubernetes tells to master to all requirements. So, Kubernetes is a declarative model. Which kind of declares came we should follow it.

2) Desired State - If you have 10 pods then whatever situat° the pods or container should maintain properly. No any one pod should be down. So to maintain a count of that 10 pods i.e called desired state. Kubernetes is declarative model, the main work is to maintain desired state.




VM
Container
Pod
Atomic units of scheduling
Minimal atomic unit in kubernetes is pod

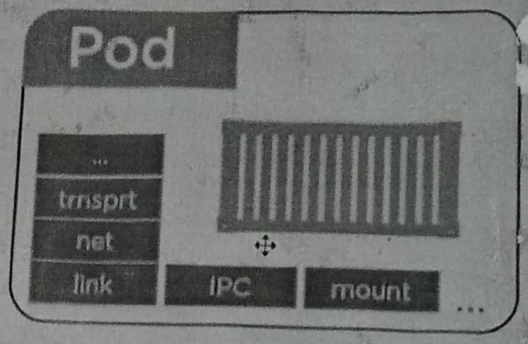

Containers always run inside of pods
Pods can have multiple containers
(advanced use-case)
Pod
Pod
Pod
Pod
Pod
We Recommend only one container in one pod

Ring-fenced environment

- Network stack
- Kernel namespaces
- ...

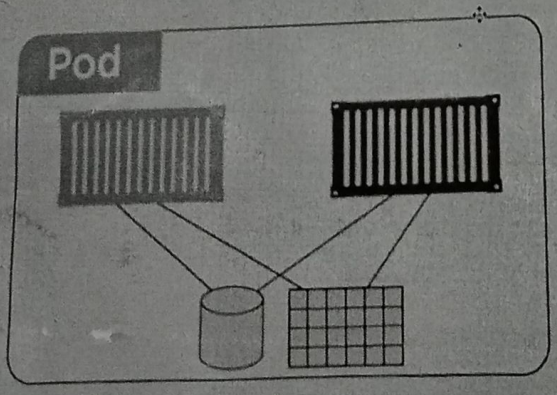
n containers

All containers in pod share the pod environment

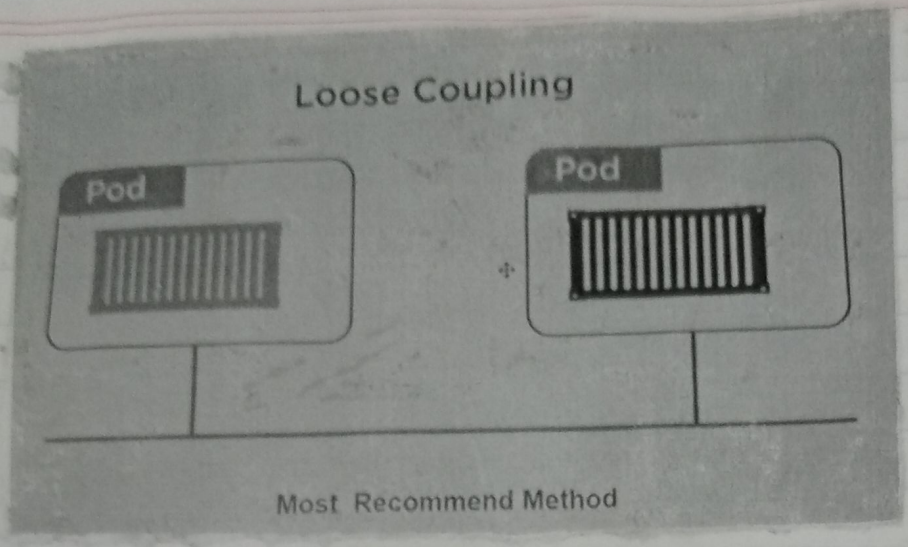


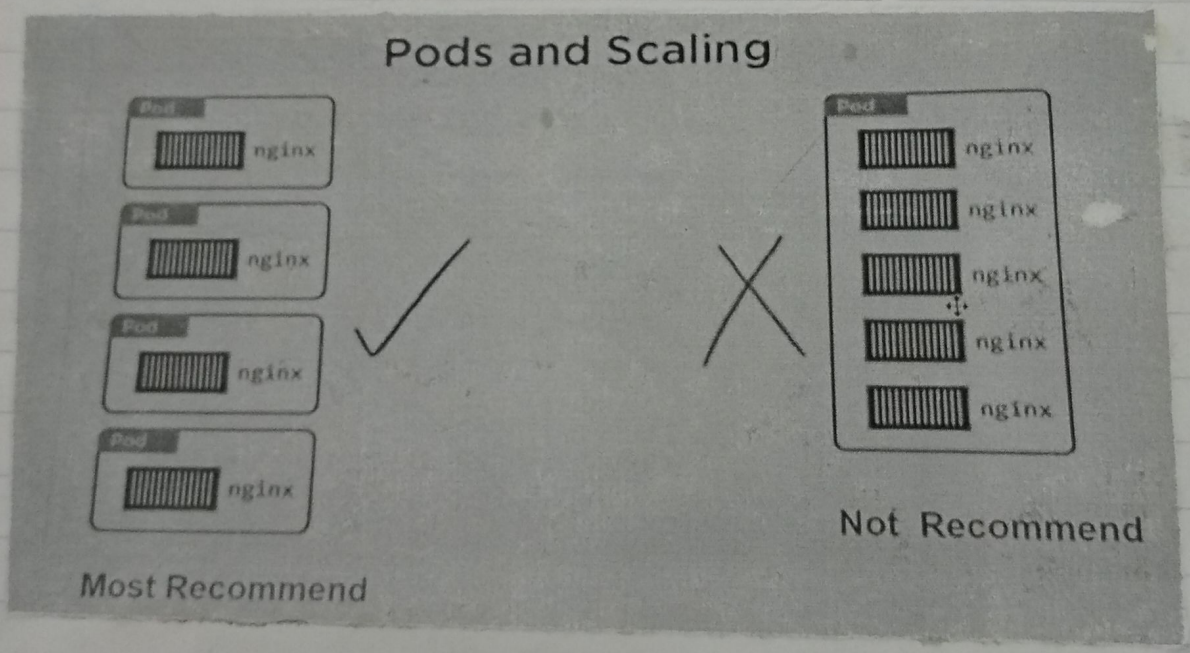
We Recommend only one container in one pod

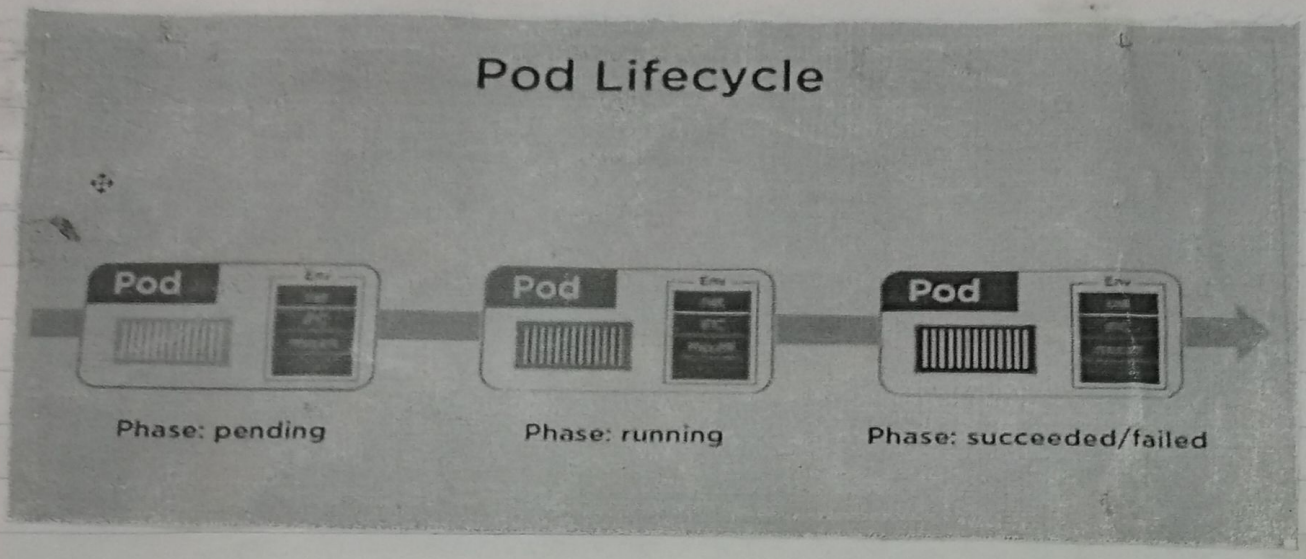
Tight Coupling



In Tight couple if any wrong in one container it affects second container


Loose Coupling
Pod
Pod
Most Recommend Method


Pods and Scaling
Pod
nginx
Pod
nginx
Pod
nginx
Pod
nginx
Most Recommend
Pod
nginx
nginx
nginx
nginx
nginx
Not Recommend



To setup cluster we need

1. To implement this we need three service : One master and two nods
2. On master and each node we have to install Docker
3. Install Kubernetes on master and each node
   1. Kubeadm
   2. Kubelet
   3. kubectl
4. After Installing Kubernetes one should be master and 2 Nodes by running init kubeadm unit on master we get a document by using this document we can configure this cluster

Install Docker

Don't use get.docker.com.

Search
install docker on ubuntu 18.04 ↵

Select official website
~~get.docker.com~~. docs.

~~Open~~

Create 3 instances & open individualy is bash file.

paste ssh ~~ttt~~ path ↵
↳ yes
↳ sudo -i
↳ apt-get update

Same for 3 git bash file.

Install docker from official website in 3 instance git bash.

1st uninstall docker - execute uninstall cmd in 3 git bash file

2nd install docker - Set up the repository

K8S Master Node Installation

Create 3 inst
1 instance of t2.medium
2 instance of t2.micro

(Note : Apply all cmds on all 3 instances untill you will know further)

Git bash file

```
sudo -i
apt-get update
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
```

(copy & paste all 5 lines at a time)

```
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Enter

```
y
```

Enter

```
sudo mkdir -m 0755 -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

Enter

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
y

cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
```

Enter

```
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker --version
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl

sudo curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

Enter

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

(Set on hold cmd shos up)

On Master

rm /etc/containerd/config.toml

systemctl restart containerd

kubeadm init (it takes time wait a bit)

(cont...)

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

On Both Nodes

(We've copied it form Master gitbash)

rm /etc/containerd/config.toml

systemctl restart containerd

(cont...)

kubeadm join 172.31.2.143:6443 --token h1bkxv.pslqnql60mwewe3m \
--discovery-token-ca-cert-hash
sha256:3f424276a29a9299e7536b9f5a156286a6c95ab212e7a2fec7ab926cc
ef09f6f

On Master

kubectl get nodes

(3 ips will show not ready)

(cont..)

kubectl apply -f
https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-
daemonset-k8s.yaml

(3 ips will show ready)

(****The End)

---

vi pod.yml

(Its open editor)

.....

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod
spec:
 containers:
   - name: first-container
     image: nginx
     ports:
     - containerPort: 80
```

.....

esc :wq! Enter

kubectl create -f pod.yml

(O/P---pod/hello-pod created)

kubectl get pods

(O/P---hello-pod 1/1 Running 0 27s)

K8S Cluster:

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/in stall-kubeadm/

https://kubernetes.io/docs/setup/production-environment/container-runtime%20s/#docker

https://docs.docker.com/engine/install/ubuntu/

STEPS TO CREATE PODS Execute the following commands

1. kubectl get nodes

Shows existing nodes in server

2. kubectl create -f pod.yml

create new pod.yml file in server

3. kubectl get pods

shows existing pods in server

4. kubectl describe pods

shows detail description of each pod

5. kubectl get pods -o wide

It shows pods info node running

6. kubectl get pods/hello-pod

7. kubectl get pods --all-namespaces

8. kubectl delete pods/hello-pod

It is used to delete specific pod i.e. hello-pod.

It is deleted a pod which name is hello-pod.

6/3/12

Replication Controller [RC]

Create

Create a yml file for multiple pod creation

myrc.yml

```
apiVersion: v1
kind: ReplicationController
metadata:
   name: myonlineapp-rc
spec:
   replicas: 3
   selector:
      app: myonlineapp
      version: 2.6.2
   template:
      metadata:
         labels:
            app: myonlineapp
            version: 2.6.2
      spec:
         containers:
         - name: myonlineapp-container
           image: [hub.docker.com-Repository name]
                  malladockerhub/myimage 28 feb23
           ports:
           - containerPort: 8080
```

name must be same

Check pods

~~MyKubeRC_Gmail# kubectl get pods ↵~~
~~shows pod list~~
MyKubeRC_GMail# kubectl get pods ↵

Shows list of pods
We mention here 3 so 3 pods are created.

Delete pod

kubectl delete pod/hello_pod ↵

delete selected pod in list

In Visual studio code

→ git add -A
→ git commit -m "NewRC"
→ git push
~~Sai~~ Before these cmd save myrc.yml file.

In master bash git pull

Create RC file

Kubectl create -f myRC.yml ↵

New ~~RC~~ RC file created

Look RC file

kubectl get rc ↵

shows rc list.

kubectl get pods ↵

shows list of pods created when rc file is created we will wrote in myrc.yml file 3 replicas so we have now 3 pods in list.

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
myonlineapp-rc-5wl7b    1/1     Running   0          39s
myonlineapp-rc-7zwqn    1/1     Running   0          39s
myonlineapp-rc-k8wl7    1/1     Running   0          39s
```

Using -o wide

kubectl get pods -o wide ↵

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP           NODE                  NOMINATED NODE   READINESS GATES
myonlineapp-rc-5wl7b    1/1     Running   0          7m51s   10.32.0.5    ip-172-31-26-109      <none>           <none>
myonlineapp-rc-7zwqn    1/1     Running   0          7m51s   10.32.0.4    ip-172-31-26-109      <none>           <none>
myonlineapp-rc-k8wl7    1/1     Running   0          7m51s   10.40.0.4    ip-172-31-43-211      <none>           <none>
```

Kubectl delete rc/[rc-name]

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get rc
NAME              DESIRED   CURRENT   READY   AGE
myonlineapp-rc    3         3         3       15m
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl delete rc/myonlineapp-rc
replicationcontroller "myonlineapp-rc" deleted
```

* How to create deleted rc

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# ls
README.md  cd  myRC.yml  mySvc.yml  mynwRC.yml
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get rc
No resources found in default namespace. - deleted existing RC then shows this msg.
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl create -f myRC.yml
replicationcontroller/myonlineapp-rc created
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl create -f mynwRC.yml
replicationcontroller/myonline-rc created
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get rc
NAME              DESIRED   CURRENT   READY   AGE
myonline-rc       2         2         0       5s
myonlineapp-rc    3         3         3       18s
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods
NAME                    READY   STATUS             RESTARTS   AGE
myonline-rc-8x4hn       0/1     ErrImagePull       0          13s
myonline-rc-zlkq5       0/1     ImagePullBackoff   0          13s
myonlineapp-rc-7l2vw    1/1     Running            0          26s
myonlineapp-rc-jm4tw    1/1     Running            0          26s
myonlineapp-rc-vz9dg    1/1     Running            0          26s
```

If refered image in myRC.yml file not proper then get error

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods
NAME                    READY   STATUS             RESTARTS   AGE
myonline-rc-8x4hn       0/1     ErrImagePull       0          13s
myonline-rc-zlkq5       0/1     ImagePullBackoff   0          13s
myonlineapp-rc-7l2vw    1/1     Running            0          26s
myonlineapp-rc-jm4tw    1/1     Running            0          26s
myonlineapp-rc-vz9dq    1/1     Running            0          26s
```

Correct Image file name in myRC.yml ↵

↳ git add -A
↳ git commit -m "NewRC"
↳ git push

Master bash ↓

When we delete the rc file using

kubectl delete rc/myonline-rc
the existing file was deleted.

Then we create some chenges in VScode means previous 3 replicas created but we changes file in 2 replicas then when we create new rc file using

kubectl create -f ~~myR~~ mynwRC.yml
then this cmd create new RC file & with 2 pods means 2 replicas.

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/BurlaManisha/MyKubeRC_6Mar
   3d5fbcc..a057f89  main       -> origin/main
Updating 3d5fbcc..a057f89
Fast-forward
 mynwRC.yml | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl delete rc/myonlinerc-demo
Error from server (NotFound): replicationcontrollers "myonlinerc-demo" not found
root@ip-172-31-20-70:~/MyKubeRC_6Mar# ^C
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl delete rc/myonline-rc
replicationcontroller "myonline-rc" deleted
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl create -f mynwRC.yml
replicationcontroller/myonline-rc created
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
myonline-rc-jw57t     1/1     Running   0          7s
myonline-rc-wppbd     1/1     Running   0          7s
myonlineapp-rc-7l2vw  1/1     Running   0          6m31s
myonlineapp-rc-jm4tw  1/1     Running   0          6m31s
myonlineapp-rc-vz9dg  1/1     Running   0          6m31s
```

The above output of
kubectl get pods ↵

Shows all pods which exists in that server.

The new rc file created 2 pods & previous one was created 3 pods.

Plz chcke Name of pods.

We wrote in yml file i.e. myonline-rc but chk in o/p it is automaticaly generate name by default.

The above end shows - Name - name of pod Ready - It is ready to use s/w & h/w. Status - If you are using any image it is correct then show you Running. Age - When you are create this Pod shows to time.

Delete a specific pod in RC

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl delete pod/myonline-rc-jw57t
pod "myonline-rc-jw57t" deleted
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
myonline-rc-qgbx4       1/1     Running   0          6s
myonline-rc-wppbd       1/1     Running   0          8m32s
myonlineapp-rc-7l2vw    1/1     Running   0          14m
myonlineapp-rc-jm4tw    1/1     Running   0          14m
myonlineapp-rc-vz9dg    1/1     Running   0          14m
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl delete pod/myonlineapp-rc-jm4tw
pod "myonlineapp-rc-jm4tw" deleted
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
myonline-rc-qgbx4       1/1     Running   0          45s
myonline-rc-wppbd       1/1     Running   0          9m11s
myonlineapp-rc-7l2vw    1/1     Running   0          15m
myonlineapp-rc-9954w    1/1     Running   0          6s
myonlineapp-rc-vz9dg    1/1     Running   0          15m
```

If we want delete specific pod then we are using pod name.

But in RC create automatically another one because in RC does not deleted any pod bcause the main feature is when any pod going down or dead RC create automatically new pod in existine RC-file.

2nd example also shows you same we delete one pod when we check

kubectl eget pods ↵

Again we have same no. of pods. check age of pod then we will understand.

But if you delete RC.yml file then pods also deleted.

kubectl apply -f myRC.yml ↵
do changes in myRC.yml file where replicas = 5 then — git -A
— git commit -m "myNewRC"
— git push

```
root@ip-172-31-20-70:~/MyKubeRC_6Mar# git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
unpacking objects: 100% (3/3), done.
From https://github.com/BurlaManisha/MyKubeRC_6Mar
   a057f89..140fd54  main       -> origin/main
Updating a057f89..140fd54
Fast-forward
 myRC.yml | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl apply -f myRC.yml
replicationcontroller/myonlineapp-rc configured
root@ip-172-31-20-70:~/MyKubeRC_6Mar# kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
myonline-rc-qgbx4       1/1     Running   0          5m38s
myonline-rc-wppbd       1/1     Running   0          14m
myonlineapp-rc-7l2vw    1/1     Running   0          20m
myonlineapp-rc-9954w    1/1     Running   0          4m59s
myonlineapp-rc-fnb5f    1/1     Running   0          14s
myonlineapp-rc-n4n7f    1/1     Running   0          14s
myonlineapp-rc-vz9dg    1/1     Running   0          20m
```

Previous one
New one

In this once we update file ~~for ex replicas~~ for ex - In previous myRC.yml has 3 replicas bt I was change their 5 then if I am using # kubectl create -f myRC.yml ↵ then shows error bcoz myRC.yml file exist. We chang RC file with 5 replicas.

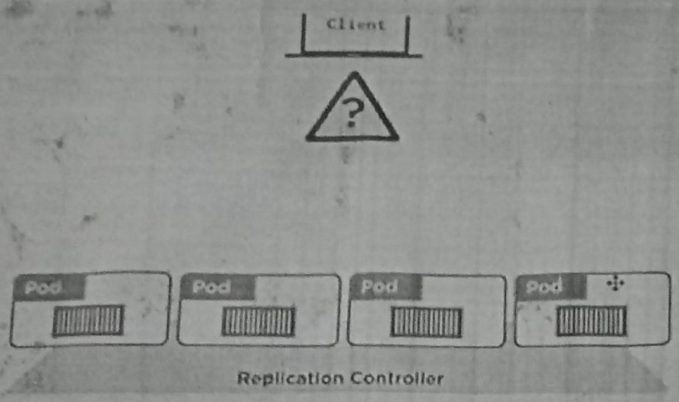
Then we use here apply
kubectl apply -f myRC.yml ↵

The is cmd create 5 replicas bt with existing replicas means already we have 3 now this cmd created 2 new one.

9/3/23

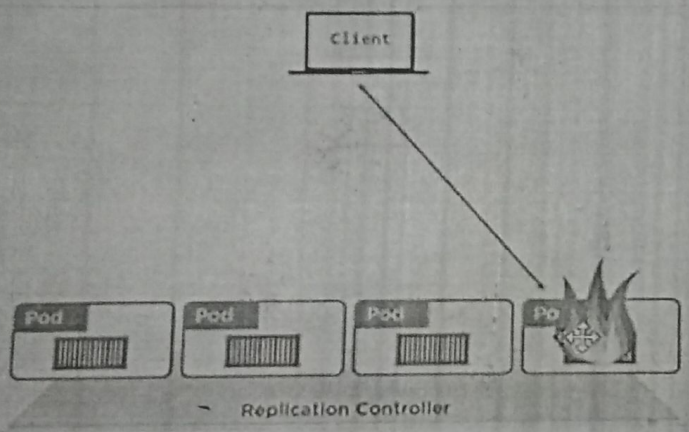
Services

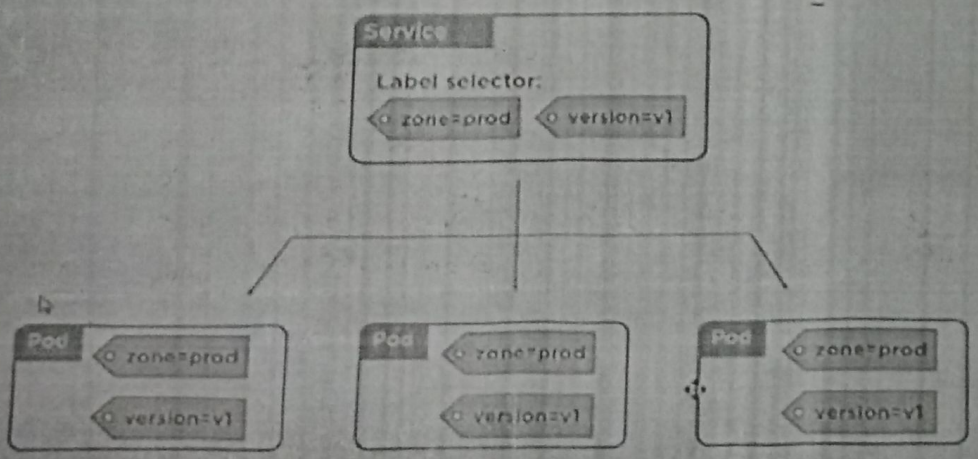
How we are going to asses pods from latop?



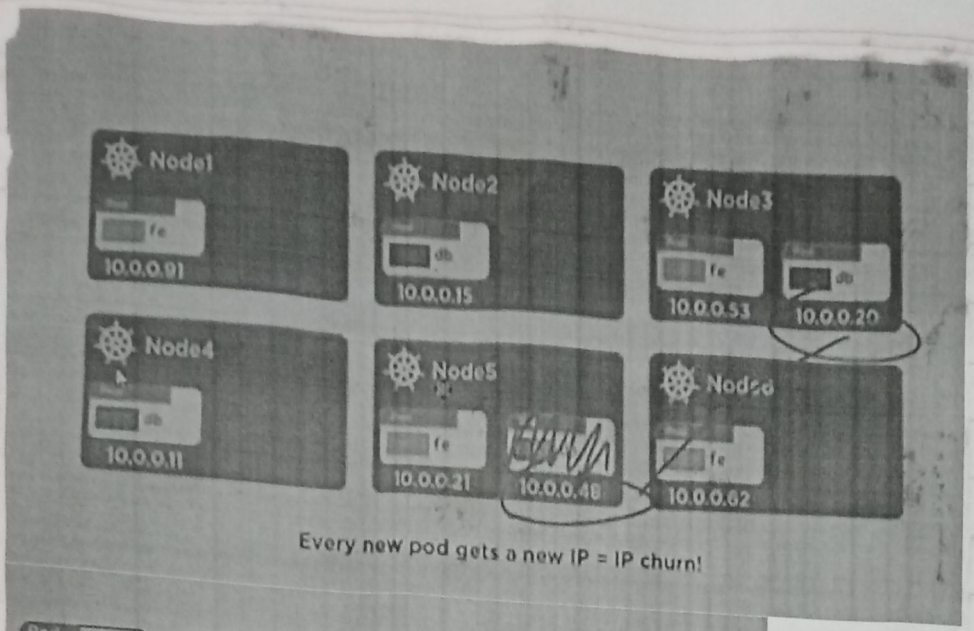
How we are going to asses pods from latop if any pod delete

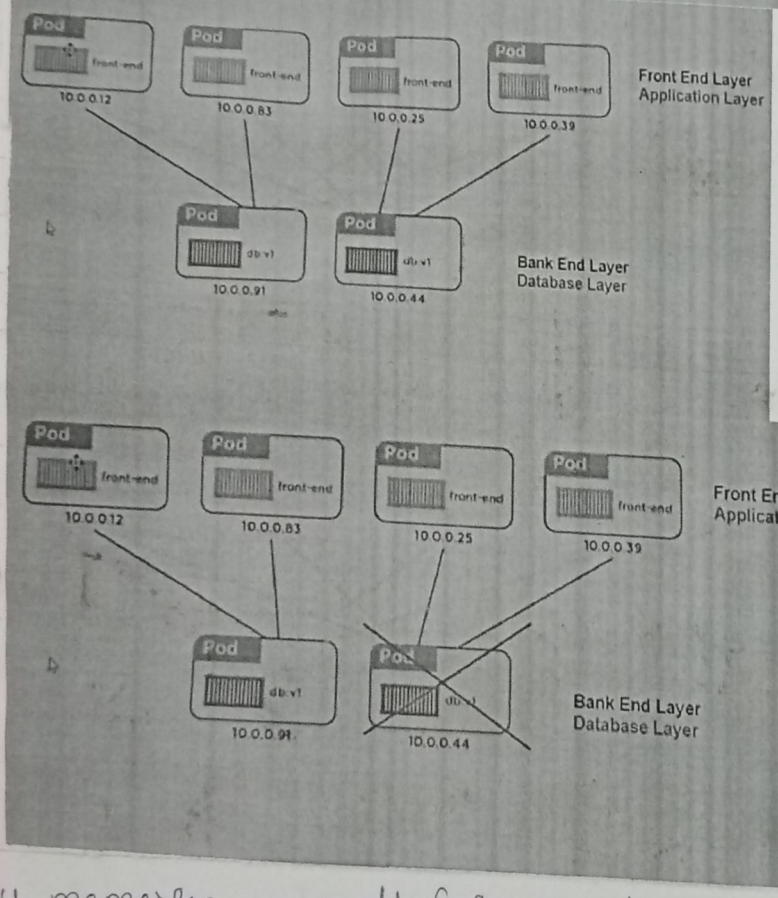
RC will create new pod with new IP





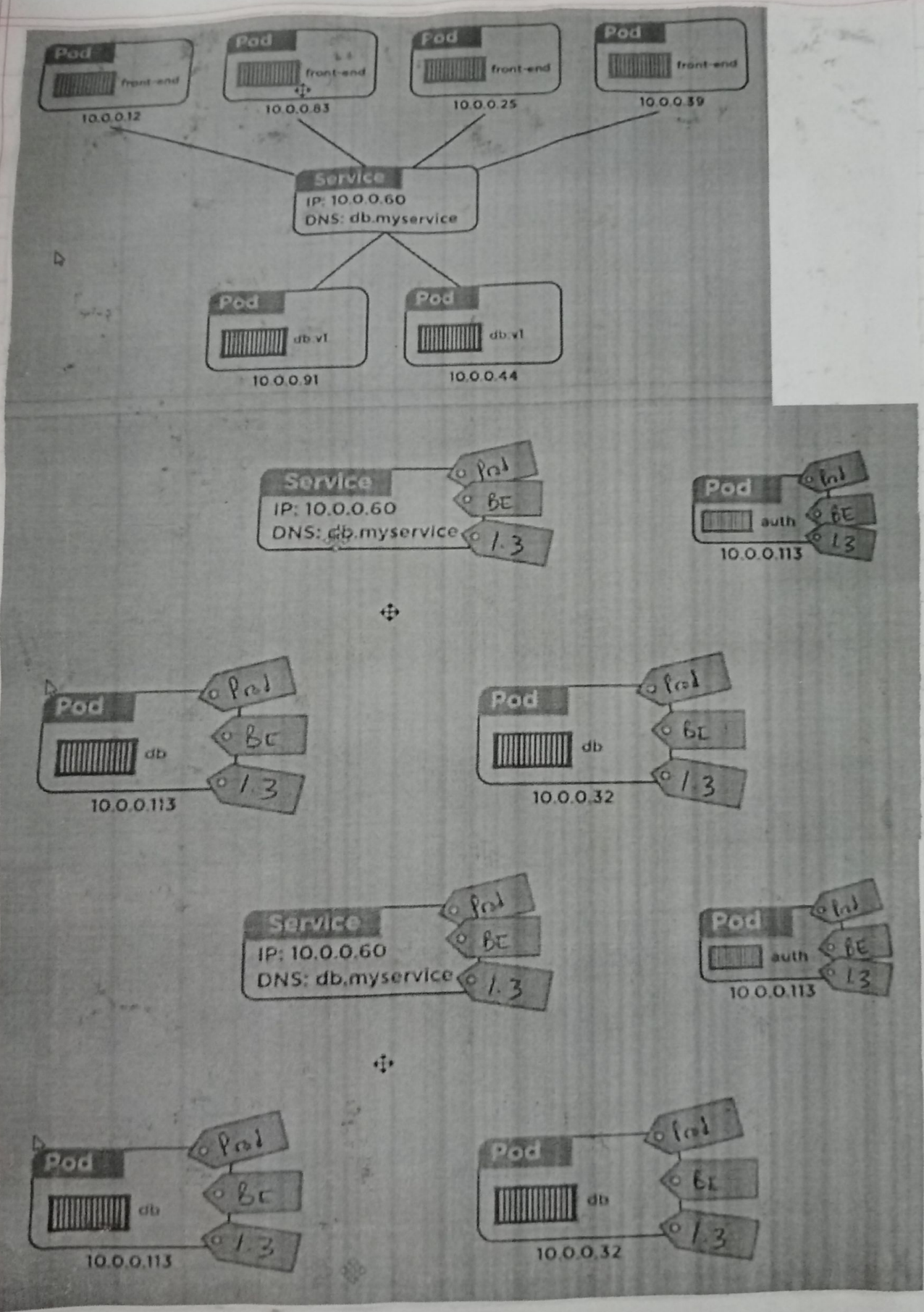
Service will solve this by giving fixed network,fixed IP by mapping of Labels

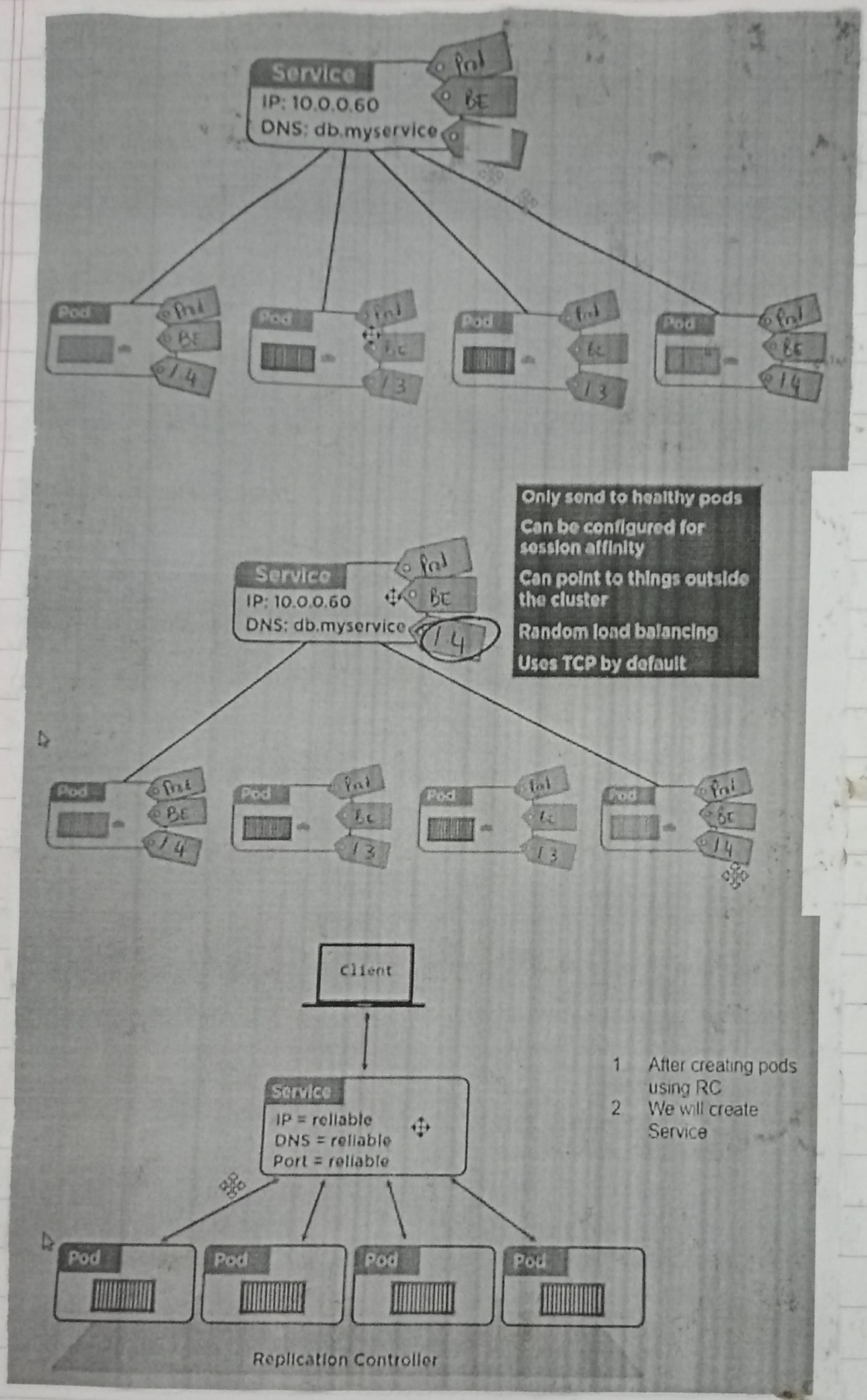




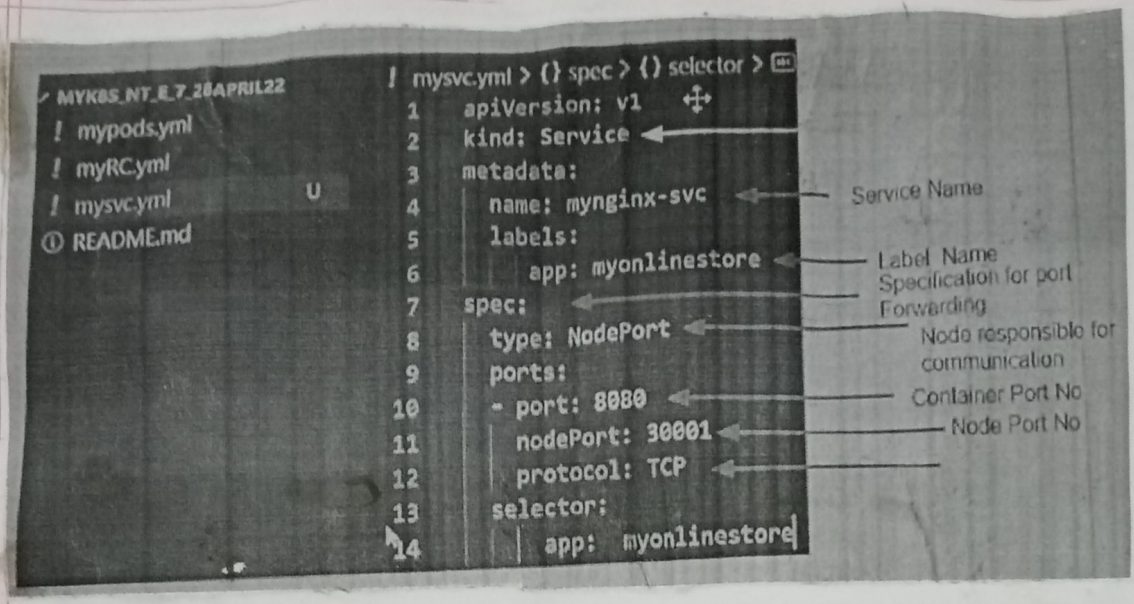
Kubernet Services

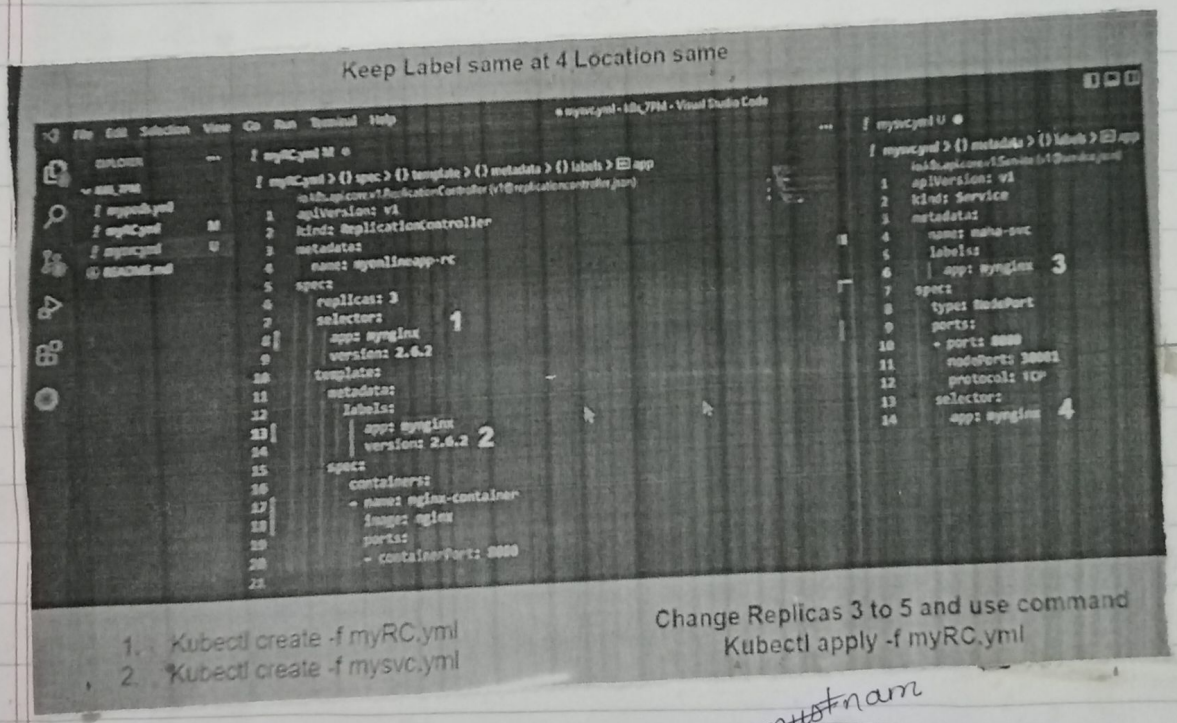
1) Can You memerise you all friends phone no. &
   → We can memrise names not phone no.
2) With one friends you want to talk. How will you talk by name or phone no. → can talk vai Phone no. only.
3) Name is permanent or phone no. → name
4) One of yr friend lost is phone no. & got new but you don't have his no. then how you will communicate to your friend.
   → By asking your matual friend by telling him your friend name.


Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db v1
10.0.0.91
Pod
db v1
10.0.0.44
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
auth
10.0.0.113
Pod
db
10.0.0.113
Pod
db
10.0.0.32
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
auth
10.0.0.113
Pod
db
10.0.0.113
Pod
db
10.0.0.32


Service
IP: 10.0.0.60
DNS: db.myservice
Pod
Pod
Pod
Pod
Only send to healthy pods
Can be configured for session affinity
Can point to things outside the cluster
Random load balancing
Uses TCP by default
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
Pod
Pod
Pod
Client
Service
IP = reliable
DNS = reliable
Port = reliable
1 After creating pods using RC
2 We will create Service
Pod
Pod
Pod
Pod
Replication Controller

Service





~~Here - 1, 2, 3, 4 these labels must be same.~~ must name

Here, 1, 2, 3 & 4 these labels name must be same.

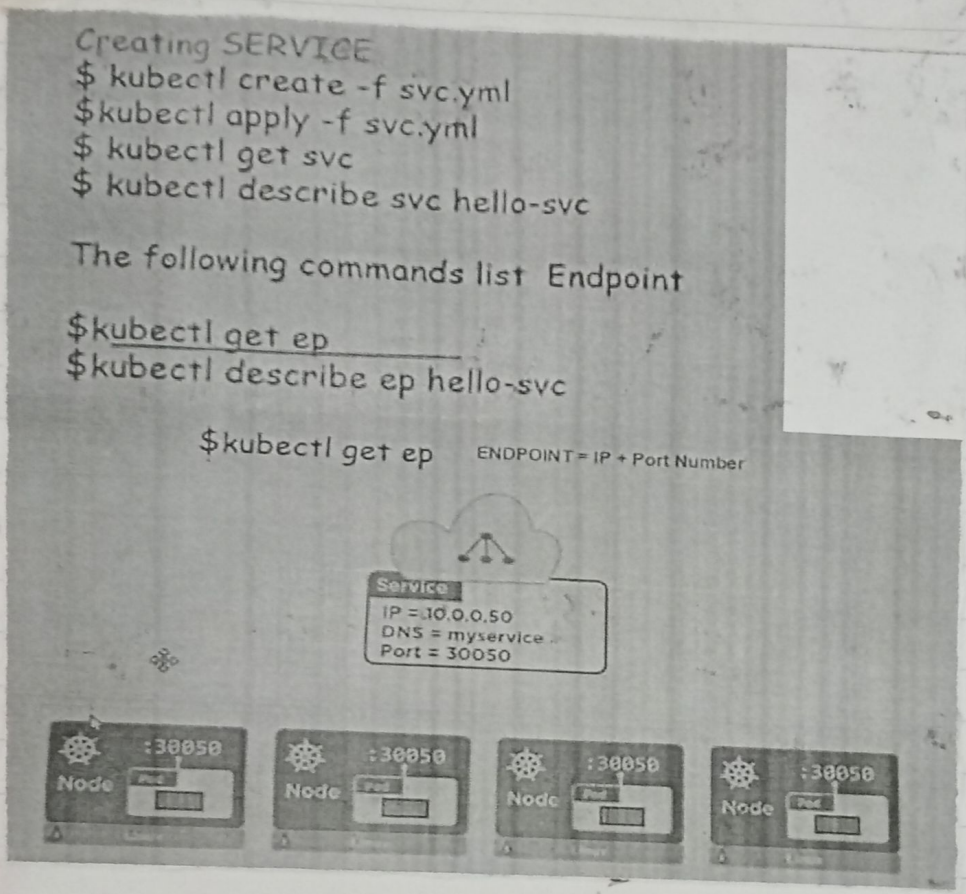
After execut° of svc file go to EC2 master inst copy public ip & paste it in browser url then put

Creating SERVICE
$ kubectl create -f svc.yml
$kubectl apply -f svc.yml
$ kubectl get svc
$ kubectl describe svc hello-svc

The following commands list Endpoint

$kubectl get ep
$kubectl describe ep hello-svc

$kubectl get ep ENDPOINT = IP + Port Number



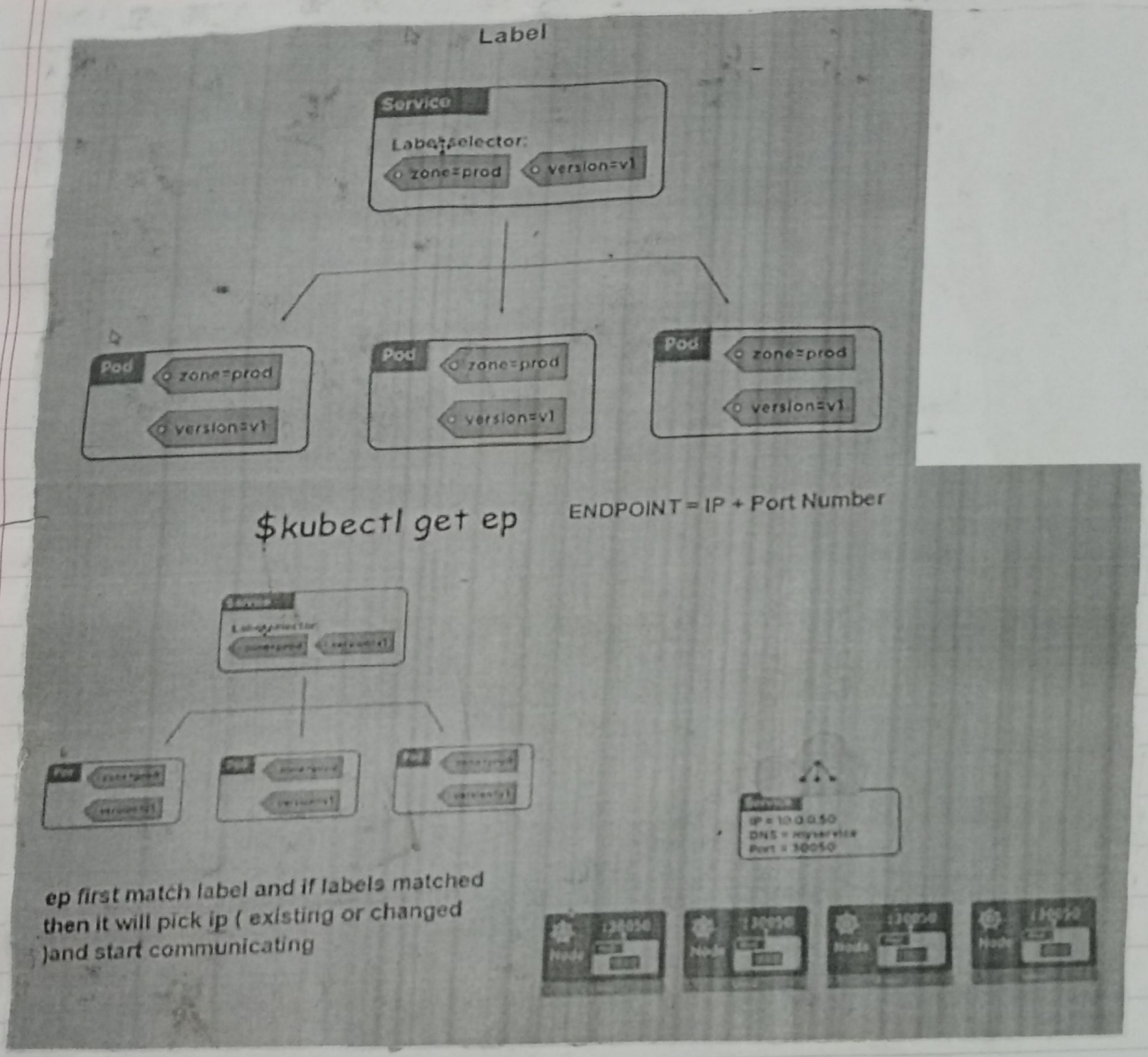
node port : 30001 then you will get Apache means we are given image name which is already has S3 bucket file with tomcat installation the add in url mahalogin-5.0/login

For. ex

192.168.1.2:30001/mahaLogin-5.0/login ↵

You will get mahalogin page.

~~Label~~ Endpoint



→ When we execute this cmd shows id add. & port no. of pods.

VERSION

Install Powershell

```
Name           Id                   Version  Source
-----------------------------------------------------
PowerShell     Microsoft.PowerShell 7.3.0.0  winget
PowerShell Preview Microsoft.PowerShell.Preview 7.3.101.0 winget
```

Open powershell as a run as administrator

Install chocolatey by googling install chocolatey —> install —> copy link and past in powershell

Google —> openjdk 8 using chocolatey in windows copy past below command on powershell

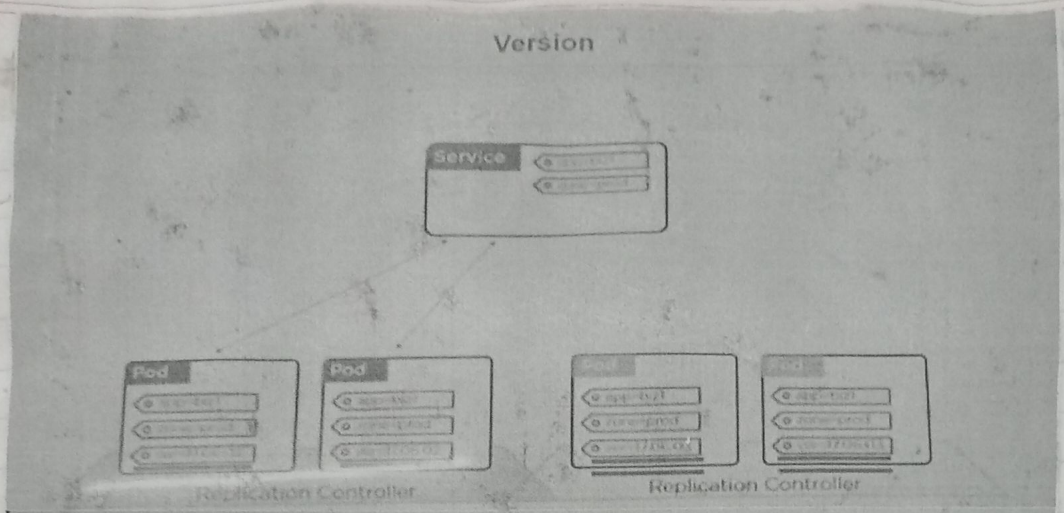
```
choco install openjdk8
```

Install maven —> just type install maven in powershell

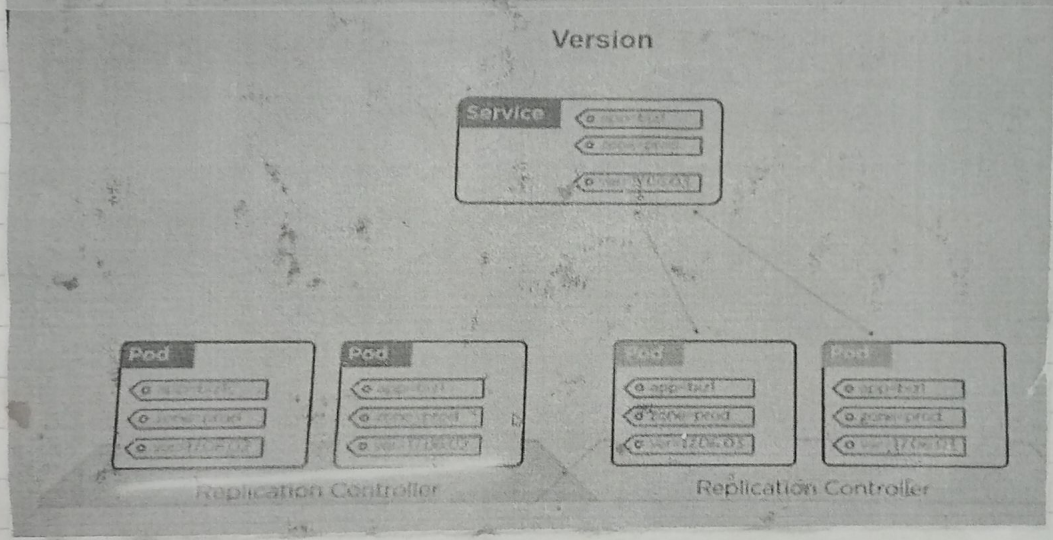
Vi Dockerfile

```
FROM tomcat:8.5.37-jre8

MAINTAINER nitin

RUN apt-get update

ADD  target/mahaLogin-5.0.war /usr/local/tomcat/webapps/

WORKDIR /usr/local/tomcat/webapps/

EXPOSE 8080

CMD ["catalina.sh", "run"]
```

RC create pod svc attach that

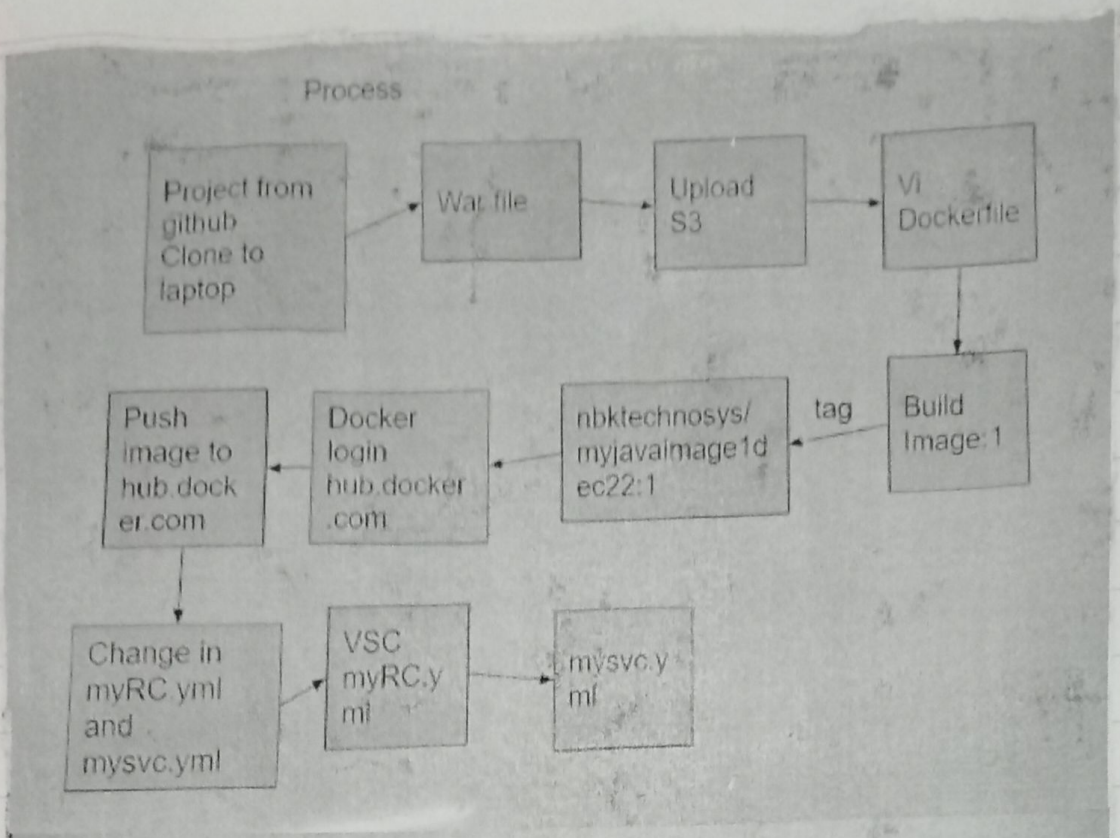


when I'm using VScode yml file then RC leave old & attach new.



Drawback Node :-

If one rc we ~~deleted~~ created or apply new pods then in that included RC leave old one & take control on new one



Version :1

mysvc.yml

```
apiVersion: v1
kind: Service
metadata:
  name: vertex-svc
  labels:
    app: myvertexapp
    version: 2.6.1
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 30001
    protocol: TCP
  selector:
    app: myvertexapp
    version: 2.6.1
```

myRC.yml

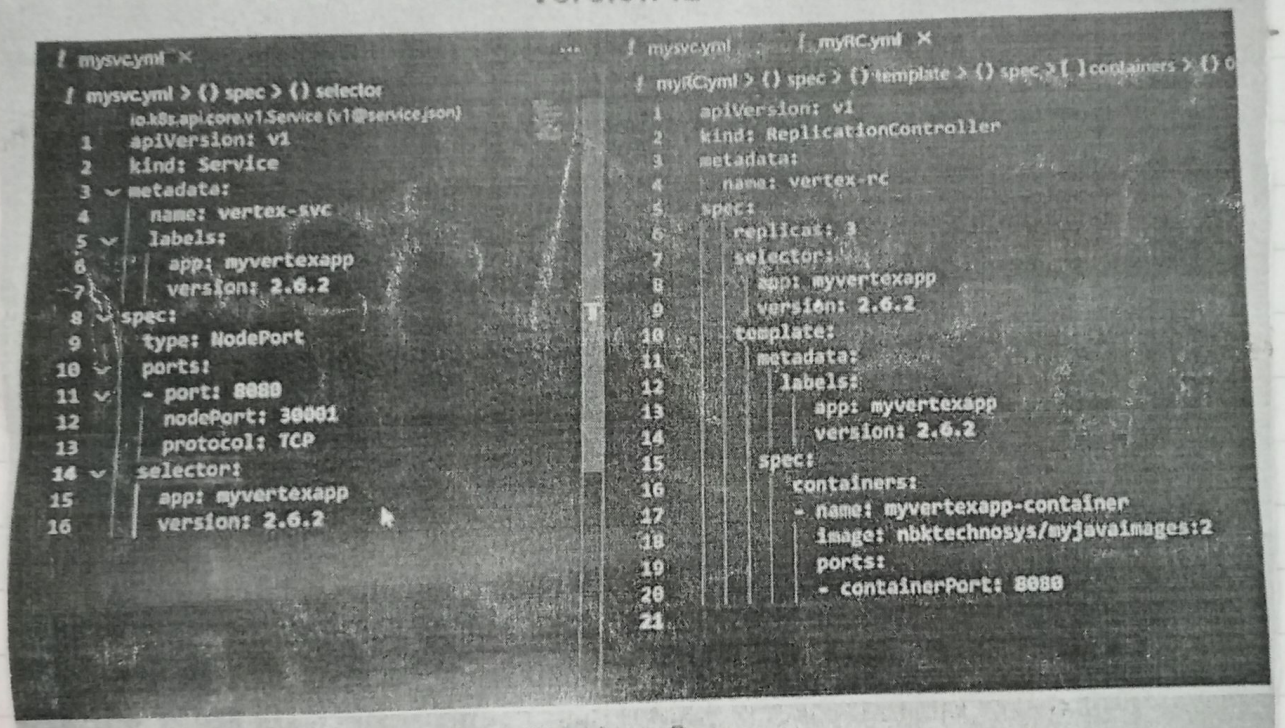
```
apiVersion: v1
kind: ReplicationController
metadata:
  name: vertex-rc
spec:
  replicas: 3
  selector:
    app: myvertexapp
    version: 2.6.1
  template:
    metadata:
      labels:
        app: myvertexapp
        version: 2.6.1
    spec:
      containers:
      - name: myvertexapp-container
        image: nbktechnosys/myjavaimages:1
        ports:
        - containerPort: 8080
```

Version :1

```
root@ip-172-31-3-28:~/K8s_7PM# kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
vertex-rc-8sv6t     1/1     Running   0          3m26s
vertex-rc-c862n     1/1     Running   0          3m26s
vertex-rc-tqw4d     1/1     Running   0          3m26s
```

```
root@ip-172-31-3-28:~/K8s_7PM# kubectl get ep
NAME         ENDPOINTS                                                  AGE
kubernetes   172.31.3.28:6443                                           7d1h
vertex-svc   10.38.0.1:8080,10.38.0.2:8080,10.40.0.3:8080               2m23s
```

Version :2



mysvc.yml

```
apiVersion: v1
kind: Service
metadata:
  name: vertex-svc
  labels:
    app: myvertexapp
    version: 2.6.2
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 30001
    protocol: TCP
  selector:
    app: myvertexapp
    version: 2.6.2
```

myRC.yml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: vertex-rc
spec:
  replicas: 3
  selector:
    app: myvertexapp
    version: 2.6.2
  template:
    metadata:
      labels:
        app: myvertexapp
        version: 2.6.2
    spec:
      containers:
      - name: myvertexapp-container
        image: nbktechnosys/myjavaimages:2
        ports:
        - containerPort: 8080
```

Version :2

```
root@ip-172-31-3-28:~/K8s_7PM# kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
vertex-rc-4wj78     1/1     Running   0          9s
vertex-rc-8sv6t     1/1     Running   0          15m
vertex-rc-c862n     1/1     Running   0          15m
vertex-rc-nmsfh     1/1     Running   0          9s
vertex-rc-tqw4d     1/1     Running   0          15m
vertex-rc-wmzz6     1/1     Running   0          9s
```

```
root@ip-172-31-3-28:~/K8s_7PM# kubectl get ep
NAME         ENDPOINTS                                                  AGE
kubernetes   172.31.3.28:6443                                           7d1h
vertex-svc   10.38.0.1:8080,10.38.0.2:8080,10.40.0.3:8080               2m23s
```

```
root@ip-172-31-3-28:~/K8s_7PM# kubectl get ep
NAME         ENDPOINTS                                                  AGE
kubernetes   172.31.3.28:6443                                           7d1h
vertex-svc   10.38.0.3:8080,10.38.0.4:8080,10.40.0.4:8080               17m
```

Version in a Kubernetes

1. Install Powershell (Search in Google)

#https://github.com/PowerShell/PowerShell
(in that scroll down)
#Select Windows(x64) download stable version (101MB) .msi
(it will start the download)
.....OR.....
#https://github.com/PowerShell/PowerShell/releases/download/v7.3.3/PowerShell-7.3.3-win-x64.msi
(direct link to download Powershell)

(Install the download, it will take time)
IMP#(In Start menu right click on Powershell 7, open with Run Administration)
#The powershell cmd prompt will open, don't close it

2. Install Chocolatey (Search in Google)

#paste the cmds in powershell cmd prompt from (*)

#https://docs.chocolatey.org/en-us/choco/setup
(in that scroll down) to (Install with PowerShell.exe)

#Get-ExecutionPolicy .....(*)
(O/P---RemoteSigned)
(We want Bypass as O/P)
#Set-ExecutionPolicy Bypass -Scope Process
#Get-ExecutionPolicy
(O/P---Bypass)

#Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex
((New-Object

System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
(O/P---downloading chocolatey on powershell)

#choco (O/P---Chocolatey v1.3.0)

3. Install openjdk 8 using chocolatey on powershell (Search in Google)

#paste the cmds in powershell cmd prompt from (*)

#https://community.chocolatey.org/packages/openjdk8
...OR...
#choco install openjdk8 .....(*)
(direct cmd to install openjdk8 on Powershell cmd prompt)

#a (it asks to run all the script)

(the process will take time)

4. Install Maven

#paste the cmds in powershell cmd prompt from (*)

#choco install maven .....(*)

#a (it asks to run all the script)

#close and open powershell again with run administration

5. Go to the war folder of Vertex (Desktop Folder)

src->main->webapps->WEB-INF->view->login

(click on login, it will open in VSC)

***#in line 40 <h2>(Change the name to specific)</h2> (Welcome to Hell)

***#delete the .war file in the target folder

#cd C:\Users\Vinit\Desktop\war → if any issue in path change drive.
(copy the path)

#mvn install
(it creats a new .war file in target folder)

6. Create a S3 Bucket

#upload the latest created .war file of vertex

#copy object url

#https://mynewvertex.s3.us-east-2.amazonaws.com/mahaLogin-5.0.war

7. Start the master node instances

#open master in gitbash

#sudo -i
#mkdir mydocker
#vi **Dockerfile**
(Editor opens)

FROM tomcat:8.5.37-jre8
MAINTAINER vinit
RUN apt-get update
ADD https://mynewvertex.s3.us-east-2.amazonaws.com/mahaLogin-5.0.war /usr/local/tomcat/webapps

```
EXPOSE 8080
CMD ["catalina.sh","run"]
-----------------------------
```

#docker build -t mywarimage:1 .
(creates a image/software)

#docker images
(checks if the mywarimage created)

#docker tag mywarimage:1 vinithttps/mywarimage11323:1
(creates a image/software via vinithttps .. hub.docker.com)

#docker images
(checks if the vinithttps/mywarimage11323 created)

#docker login

Username : vinithttps
Password : AWSAWS@9196

(Login Succeeded)

#docker push vinithttps/mywarimage11323:1
(pushes the files from gitbash to hub.docker.com)

8. Go to VSC open K8svc folder

(should contain MyRC.yml & MySVC.yml)

13/3/23

MyRC.yml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: myvertexapp-rc
spec:
   replicas: 2
   selector:
     app: myvertexapp
     version: 2.6.3
   template:
     metadata:
       labels:
          app: myvertexapp
          version: 2.6.3
     spec:
        containers:
        - name: myonlineapp-container
          image: vinithttps/mywarimage11323:1
          ports:
          - containerPort: 8080
```

MySVC.yml

```
apiVersion: v1
kind: Service
metadata:
  name: vertex-svc
  labels:
    app: myvertexapp
    version: 2.6.3
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 30001
    protocol: TCP
```

```
  selector:
    app: myvertexapp
    version: 2.6.3

#in vsc terminal
git add -A
git commit -m "war"
git push
-----------------------------------
```

9. Go gitbash

#cd K8SVSC
(go into the folder)

#git pull
(pulls files from vsc to gitbash)

#kubectl apply -f MyRC.yml
(Updates the file with new file)

#kubectl get pods
(O/P---myvertexapp-rc-vpf2z 1/1 Running 0 11s
myvertexapp-rc-zsjpk 1/1 Running 0 11s
to check if pods are created)

#kubectl apply -f MySVC.yml
(Updates the file with new file)

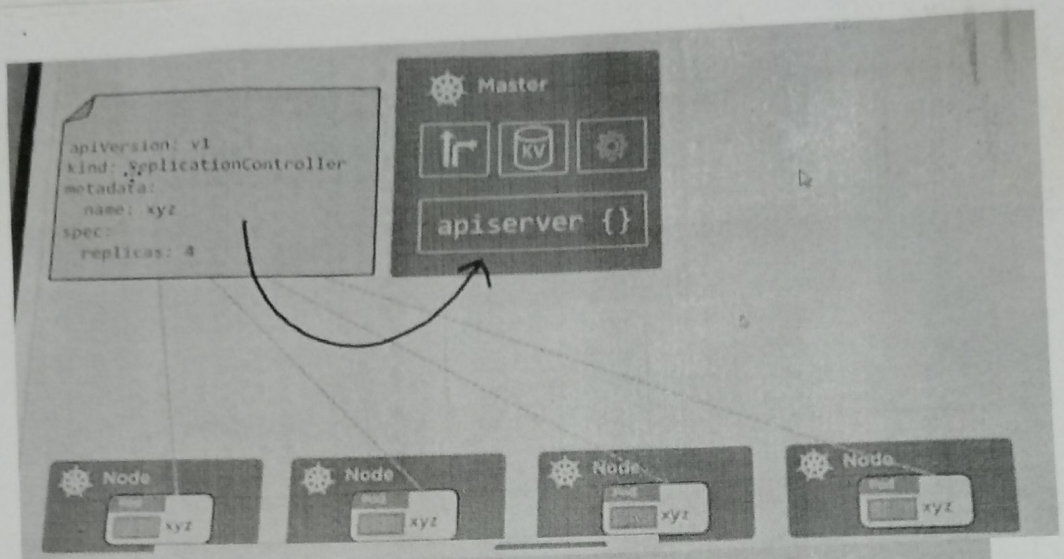
10. Copy nodes ip and paste in Google url

changes depend on instance

http://18.222.200.140:30001/mahaLogin-5.0/login
(The page will appear with the specific message you changed in Step 5.)

13/3/23

Deploy

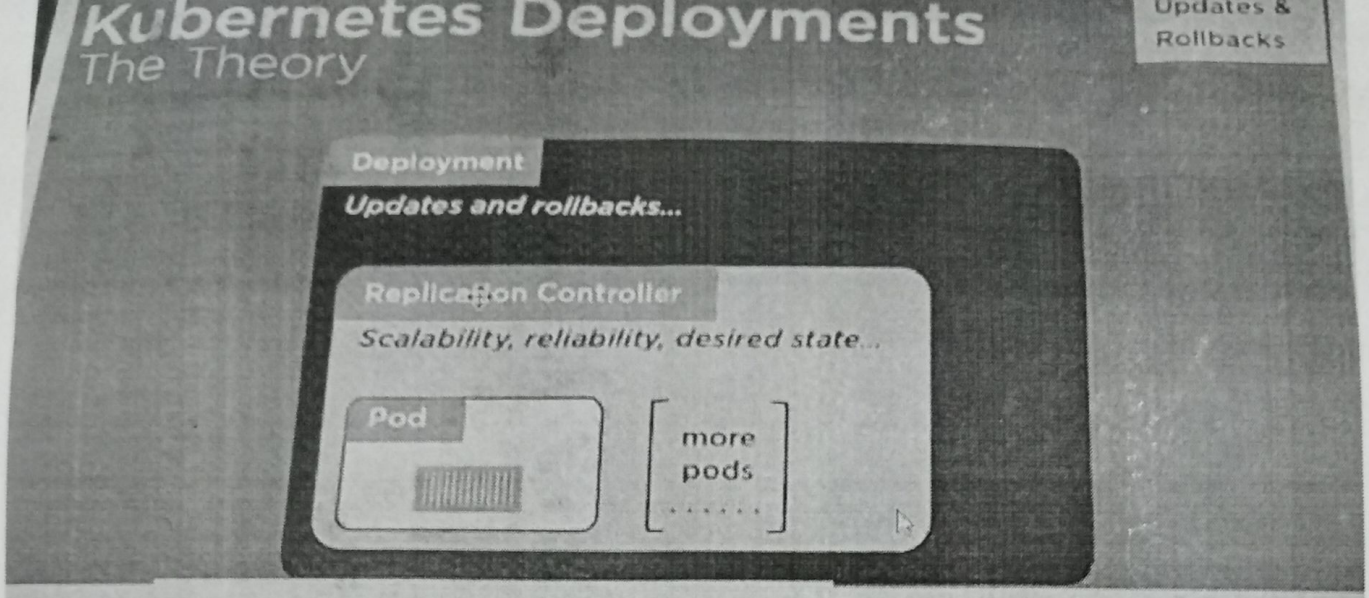


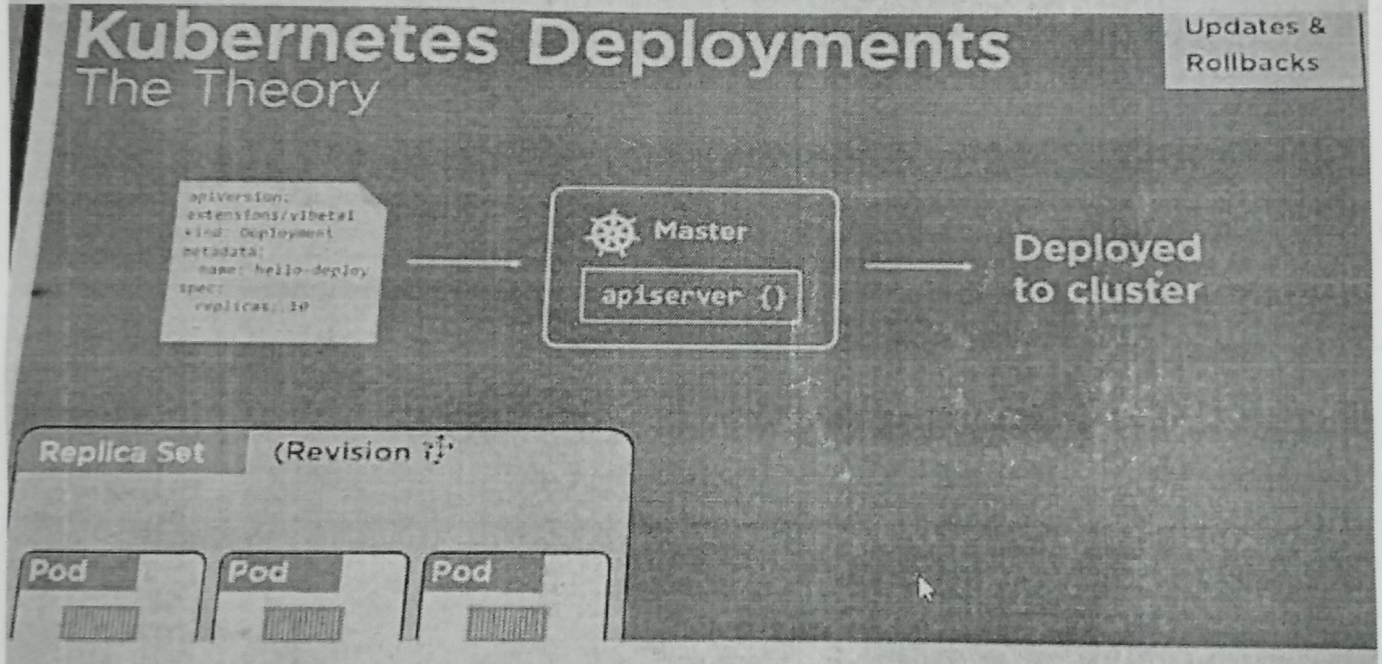
Deployments are all about declarations

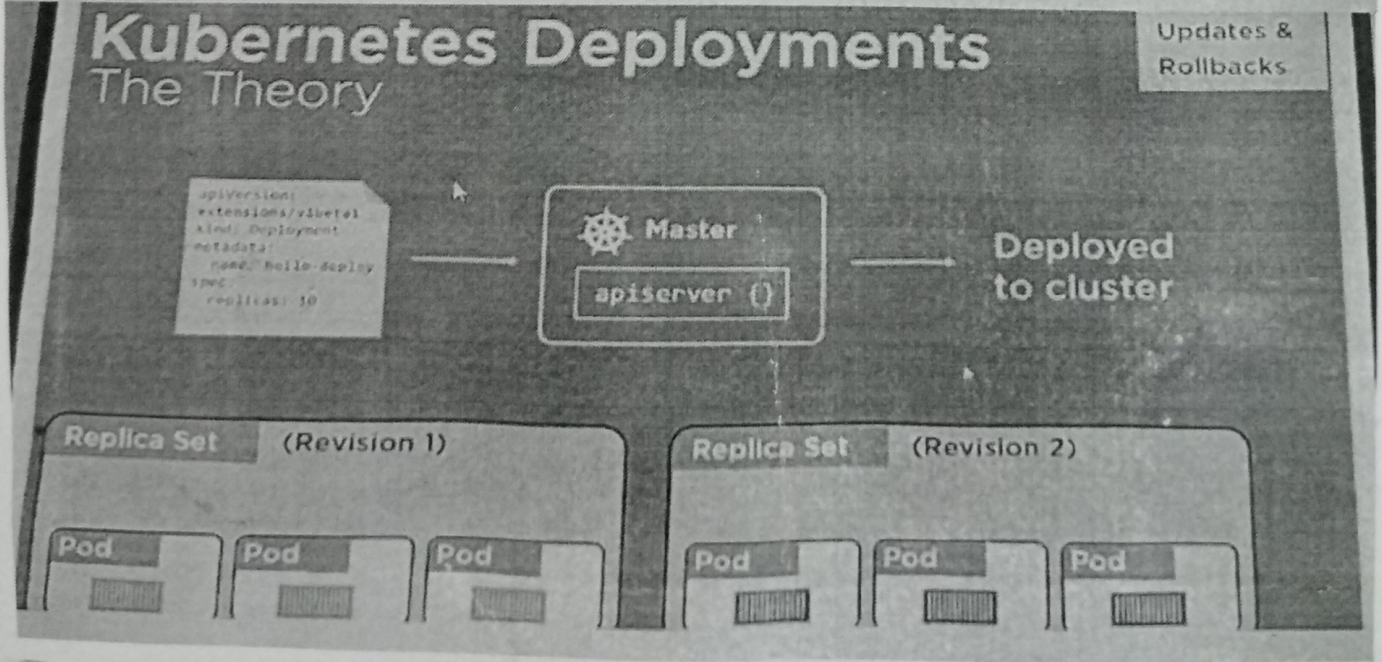
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: xyz
spec:
  replicas: 4
```

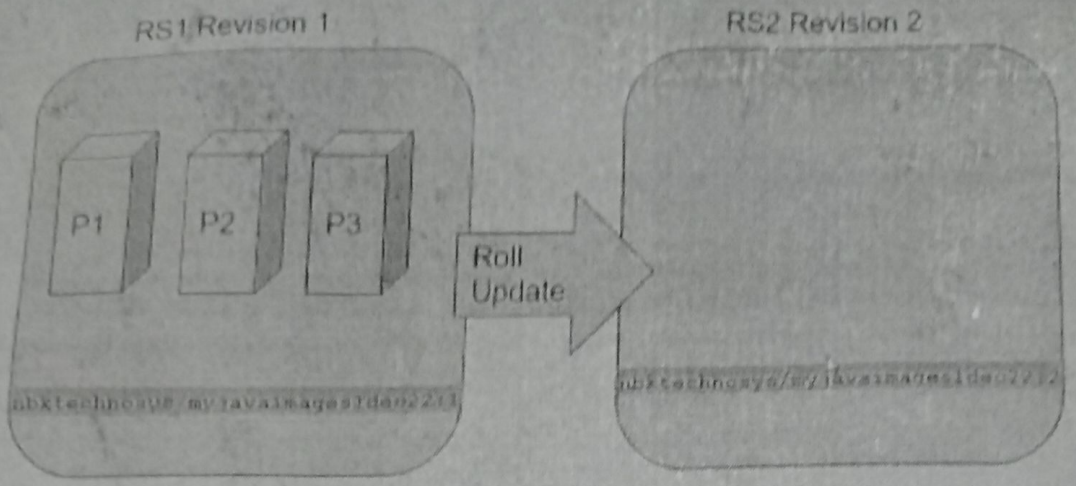
Objects in the K8s API

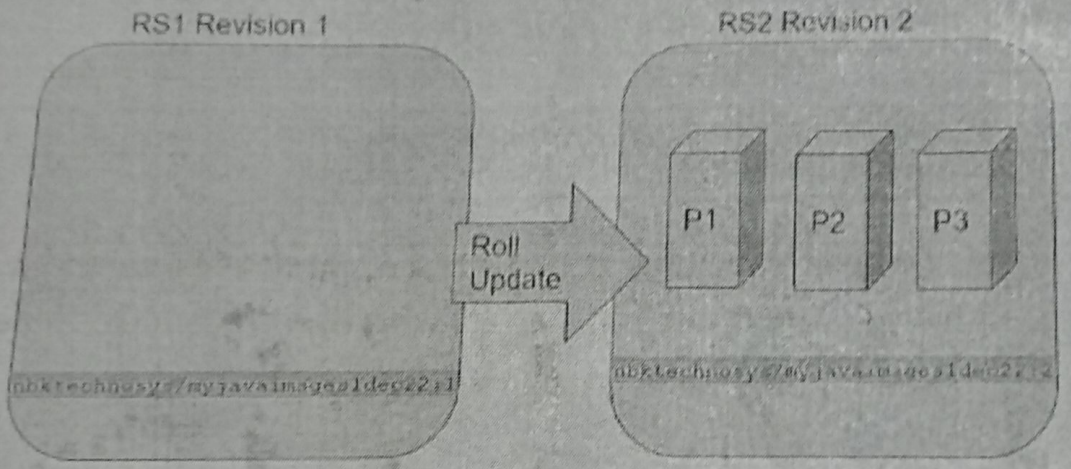
- **Pods** — Atomic unit of scheduling
- **Replication Controllers** — Scale pods, desired state etc.
- **Deployments** — RC + rolling updates, rollbacks
- **Services** — Stable networking

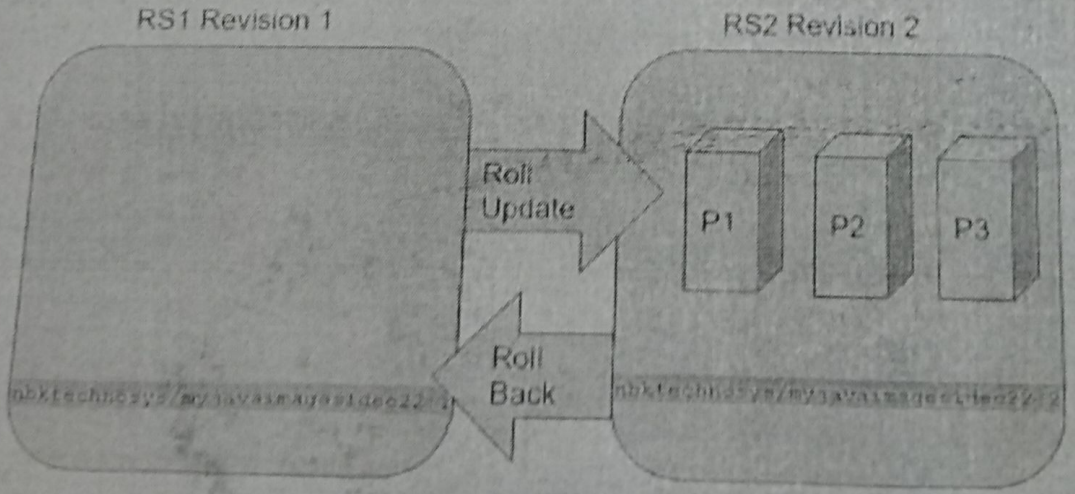

Kubernetes Deployments
The Theory
Updates & Rollbacks
Deployment
Updates and rollbacks...
Replication Controller
Scalability, reliability, desired state...
Pod
more pods

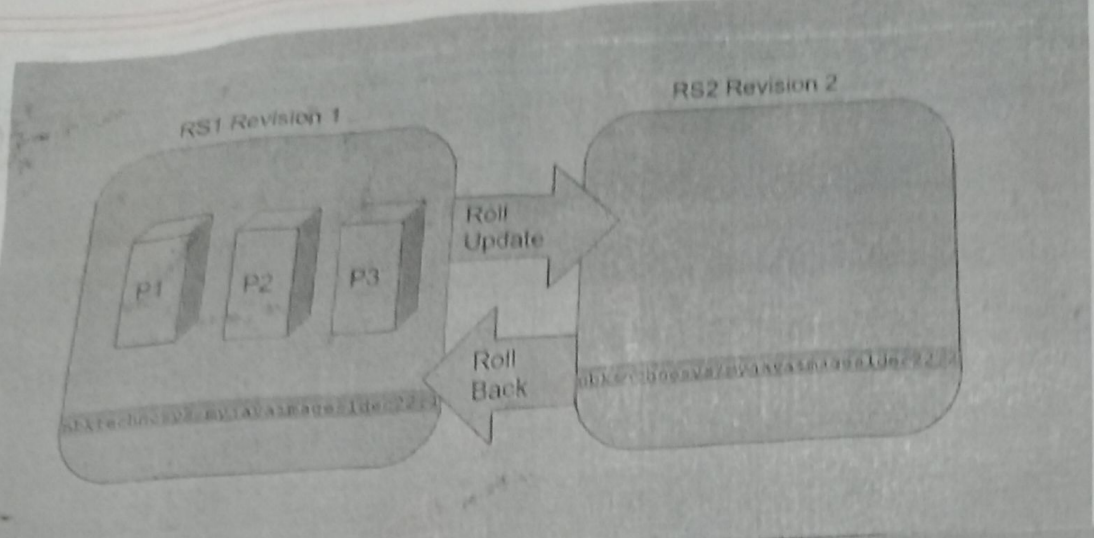

Kubernetes Deployments
The Theory
Updates & Rollbacks
apiVersion:
extensions/v1beta1
kind: Deployment
metadata:
name: hello-deploy
spec:
replicas: 10
Master
apiserver {}
Deployed to cluster
Replica Set
(Revision 1)
Pod
Pod
Pod


Kubernetes Deployments
The Theory
Updates & Rollbacks
apiVersion:
extensions/v1beta1
kind: Deployment
metadata:
name: hello-deploy
spec:
replicas: 10
Master
apiserver {}
Deployed to cluster
Replica Set
(Revision 1)
Pod
Pod
Pod
Replica Set
(Revision 2)
Pod
Pod
Pod


RS1 Revision 1
RS2 Revision 2
P1
P2
P3
Roll
Update


RS1 Revision 1
RS2 Revision 2
Roll
Update
P1
P2
P3


RS1 Revision 1
RS2 Revision 2
Roll
Update
Roll
Back
P1
P2
P3



Manifest file for deploy

Represents Deploy

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myjavaapp-deploy
  labels:
    app: myjavaapp
```

Represents RC

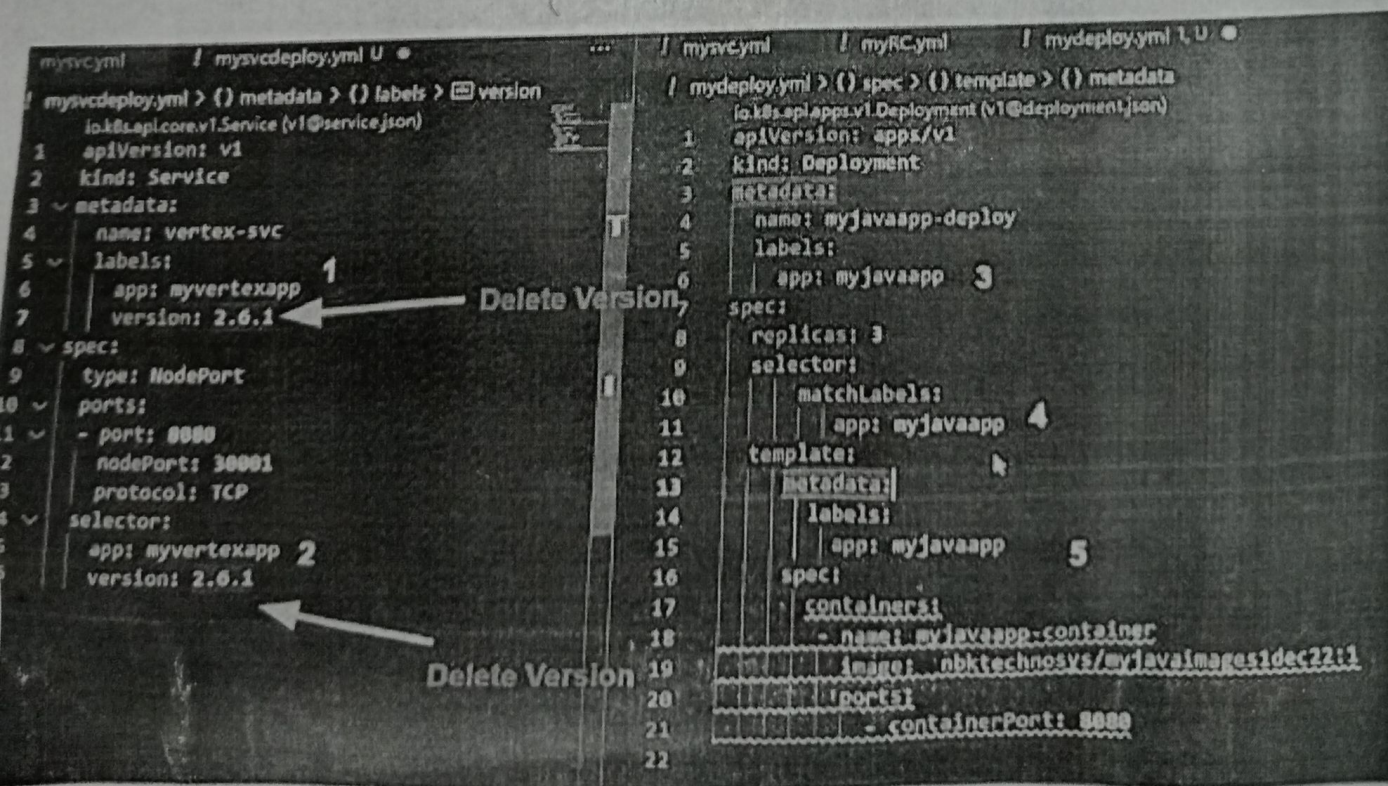
```
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myjavaapp
```

Represents Pods

```
  template:
    metadata:
      labels:
        app: myjavaapp
```

Represents container

```
    spec:
      containers:
        - name: myjavaapp-container
          image: nbktechnosys/myjavaimages1dec22:1
          ports:
            - containerPort: 8080
```



```
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl create -f mydeploy.yml --record
```

Without Record revision will not be mentioned

```
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl get deploy
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
myjavaapp-deploy    3/3     3            3           14s
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
myjavaapp-deploy-855bf985c7   3         3         3       27s
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
myjavaapp-deploy-855bf985c7-7cv46   1/1     Running   0          64s
myjavaapp-deploy-855bf985c7-bhd6n   1/1     Running   0          64s
myjavaapp-deploy-855bf985c7-gncdb   1/1     Running   0          64s
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22#
```

It will create rs

```
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl create -f mysvcdeploy.yml
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl get svc
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl describe svc
Endpoints:  10.40.0.1:8080,10.40.0.2:8080,10.46.0.3:8080
```

mysvc.yml

```
apiVersion: v1
kind: Service
metadata:
  name: vertex-svc
  labels:
    app: myvertexapp
    version: 2.6.1
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 30001
    protocol: TCP
  selector:
    app: myvertexapp
    version: 2.6.1
```

mydeploy.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myjavaapp-deploy
  labels:
    app: myjavaapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myjavaapp
  template:
    metadata:
      labels:
        app: myjavaapp
    spec:
      containers:
        - name: myjavaapp-container
          image:  nbktechnosys/myjavaimages1dec22:2
          ports:
            - containerPort: 8080
```

Only change image number

```
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl apply -f mydeploy.yml --record
```

```
root@ip-172-31-11-194:~/myK8s_NT_E_7_28Apri22# kubectl get rs
NAME                              DESIRED   CURRENT   READY   AGE
myjavaapp-deploy-7557f86fdd       3         3         3       16s
myjavaapp-deploy-855bf985c7       0         0         0       4m40s
```

CREATING DEPLOYMENT

kubectl create -f deploy.yml
kubectl describe deploy myjavaapp-deploy
kubectl get rs
kubectl describe rs
kubectl get deploy
kubectl delete deploy/myjavaapp-deploy
kubectl delete rs/myjavaapp-deploy-6858bf488c