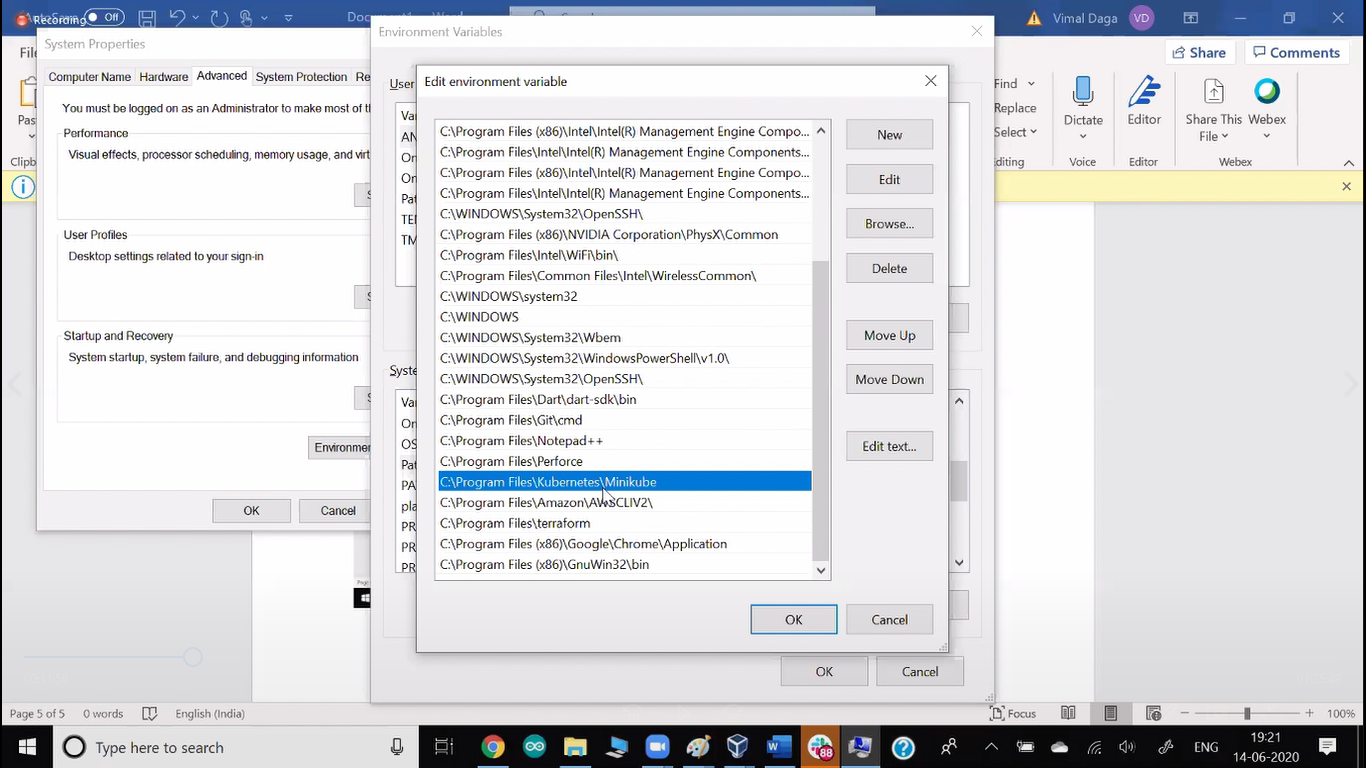
**Kubernetes Setup**

* Minikube Installation

🡪Go to <https://kubernetes.io/docs/tasks/tools/install-minikube/>

🡪Click on <https://github.com/kubernetes/minikube/releases/latest/download/minikube-installer.exe>

🡪It creates a program inside program files.   
 🡪Set environmental variable.



🡪Now run for tell your minikube you are using vbox.

-> Minikube start --vm-driver=virtualbox

🡪Now setup is ready.

🡪When minikube command it automatically downloads all the iso file for it and install vm and setup k8s for us (Both master and slave).

* Kubectl installation

🡪 Go to <https://kubernetes.io/docs/tasks/tools/install-minikube/>

🡪 Click on <https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-windows>

🡪Go to minikube folder in cmd and run this command

-> curl -LO <https://storage.googleapis.com/kubernetes-release/release/v1.18.0/bin/windows/amd64/kubectl.exe>

WHY to run in same folder?

🡪Because if you copy anywhere else you have to set environment variable path one more time.

* Kubectl

🡪Kubectl is a client command.

🡪For connect to k8s we use this command.

🡪Kubectl contact to master node.  
🡪 Master Node connect to docker engine and ask docker to create a container.

🡪If master node and slave node both are in same os it is known as single cluster node. If both working in different os it is known as multi cluster Node.

🡪kubectl must know IP address of master and port number of API server.

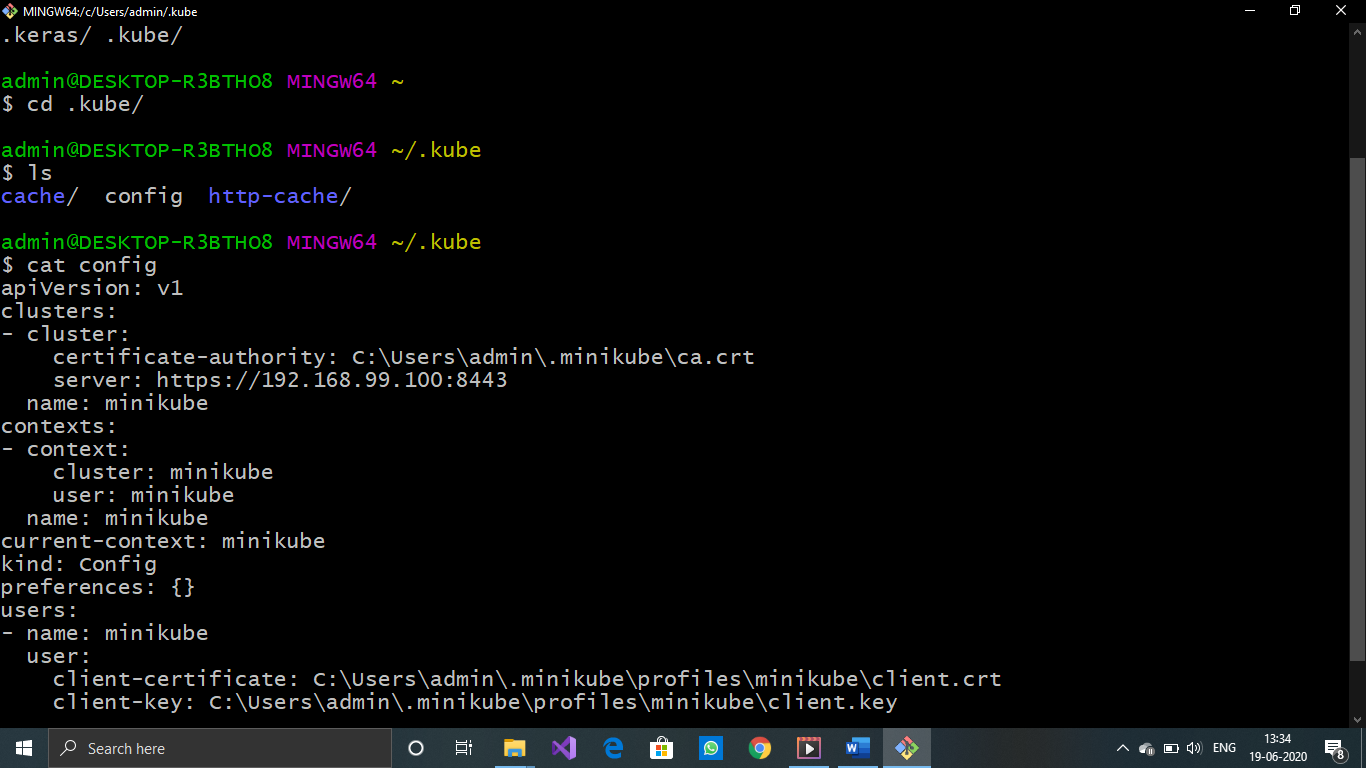
🡪port number of API server is 8443

🡪One program is running in Master Node known as API server.  
🡪Main responsibility of this program is to listen the client (Kubectl).

🡪In Kubernetes master node has one more program known as Kube schedular.  
🡪They will contact to different different slave node and launch pod there.

* Kubectl Configure

🡪In kubectl config file it is mentioned they have to contact to IP 192.168.99.100 port 8443. This file is created by minikube.

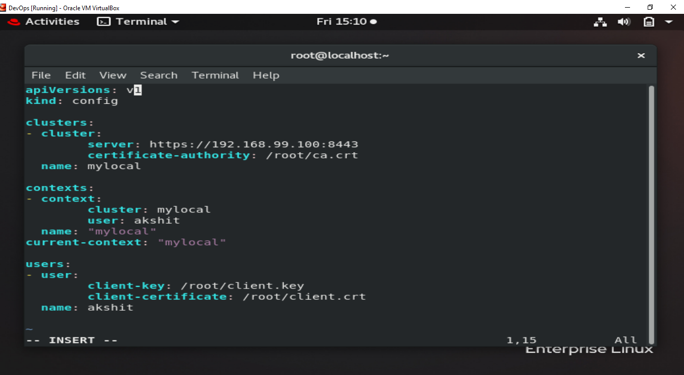


🡪This file automatically created so we can run kubectl without configuration.

🡪If you are using kubectl and k8s setup in different OS then do following for configuration.

🡪kubectl get pods --server <https://192.168.99.100:8443> --client-key /root/client.key   
--client-certificate /root/client.crt   
--certificat-authority /root/ca.crt (Below steps given)

🡪Instead of using this big command every time, create one yml file



🡪use kubectl get pods --kubeconfig my.yml

🡪We can also create a config file.  
 🡪go to /root

🡪create mkdir /.kube

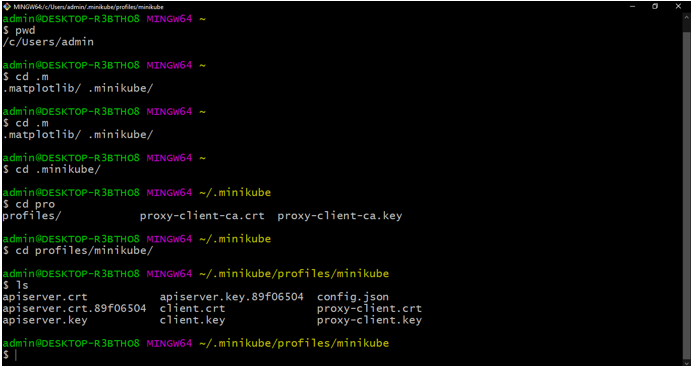
🡪Copy here this yml file (name should be config)

🡪provide Authentication:

🡪login-password -- For human

🡪key -- For program

🡪We have minikube, so in our case minikube provides key.



🡪provide this key to kubectl (client).

🡪ca certificate can be found in two folders back.

🡪Whenever http protocol used (File transferred) we require three things.  
 🡪key 🡪ca-certificate 🡪certificate

**Kubernetes**

🡪Kubernetes is made for management purpose known as COE (Container orchestration Engine).

🡪By default, user-id password for minikube is.

->user=docker ->password=tcuser

🡪Kubernetes is managing pods.

🡪Here launching a pod is known as deployment.

🡪Kubernetes keep an eye on this pod.

🡪As soon as pod goes down it automatically launch a new pod.

🡪If it deleted somehow its automatic starting to create a new pod.

🡪 If you only launch a pod using Kubernetes run pod, this time k8s don’t work as fault tolerance.

🡪This thing provided by deployment.

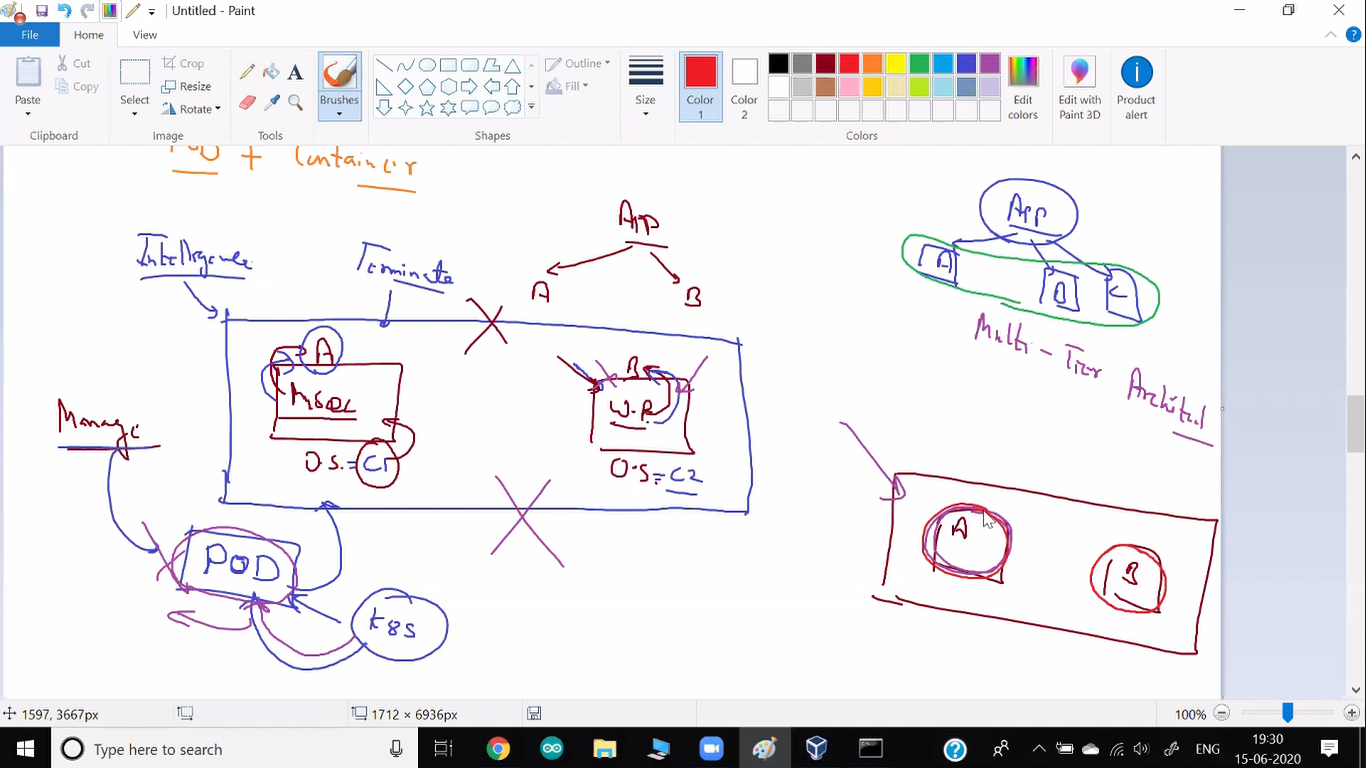
🡪Kubectl create deployment myweb --image=akshit/php-webserver

* Pod

🡪kubernetes manage the pods.

🡪pod contact to Docker engine and launch container.

🡪Pod is one who go and download the image for you.  
🡪Pod keep on monitoring containers.  
🡪If container failed, pod will also fail and k8s create one more pod for you.



🡪This box is known as pod.  
🡪There are two containers are running, both are highly dependent on each other.  
🡪If anyone of the container fail, terminate the complete Pod/box and launch one more container

* Tag/labels

🡪Pod name and IP of the container is dynamic.  
🡪Whenever something fails k8s automatically launch one more pod for you but this time the name of a pod and IP of a container are changed.

🡪For solving this we can give tag name that never changes.  
🡪One more use case is we can give tag like, env=test, so easily manage them and retrieve only required pods.

🡪 Kubectl label pods myweb1 env=test

* Selectors

🡪Selector are used to search the pod.  
🡪It uses labels for searching the pod.

🡪There are two types of selectors.

1. Equality Based Selector

🡪 Kubectl get pods --selector dc!=US

1. Set Based Selector  
   🡪 Kubectl get pods -l “dc in (US, IN), env=prod”

🡪It also provides to search using AND OR (give multiple searching criteria)

**Controller**

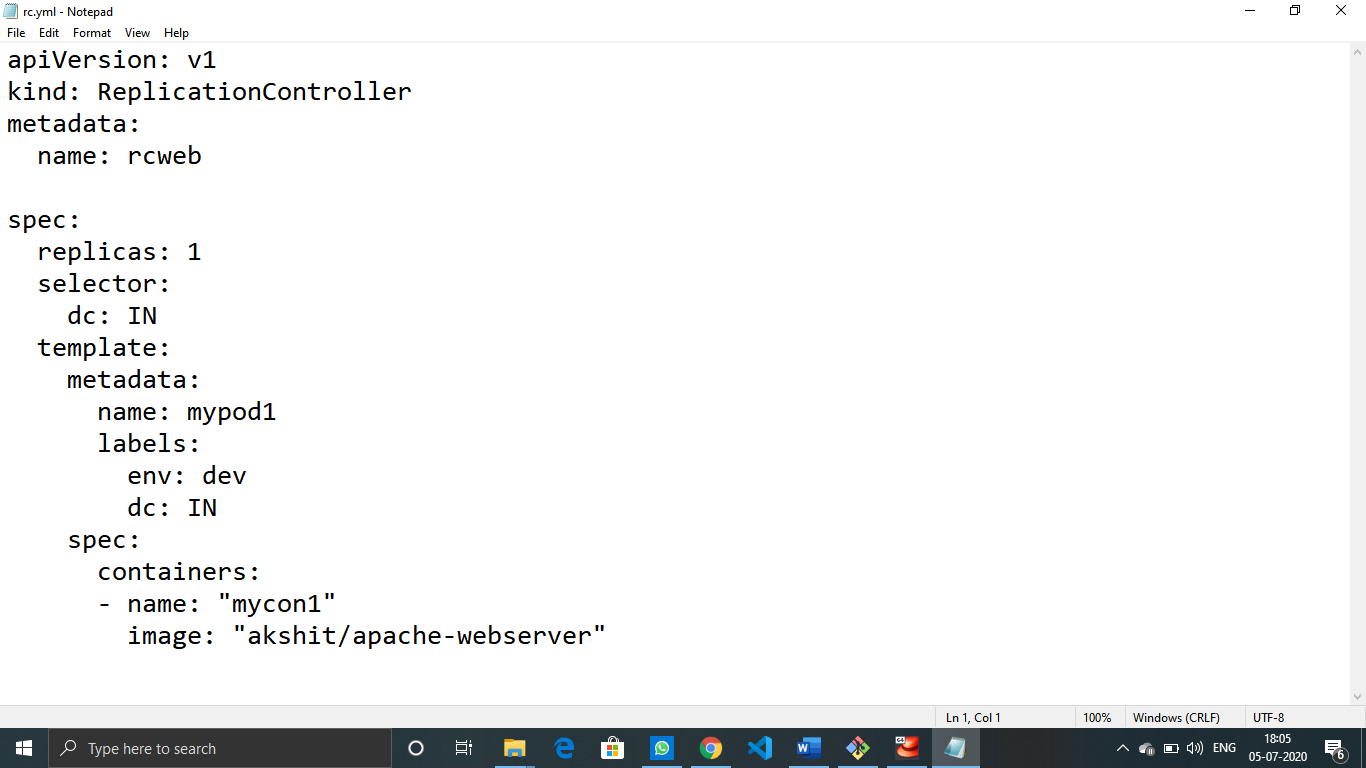
🡪There are two different controllers

1. Replication Controller

🡪It is older and has some limitation.

🡪It uses Equality Based selector.

🡪Kubectl get pods --selector dc!=US



🡪Kubectl create -f rc.yml

🡪 Kubectl replace -f rc.yml

🡪while replica already created and you edit something.

🡪Here kind is a keyword, different different tool has different keyword.  
🡪So this is also known as DSL (Domain Specific Language).

🡪Only difference between RC and RS is set based selector.

2. ReplicaSet

🡪It overcome with limitation of RC.

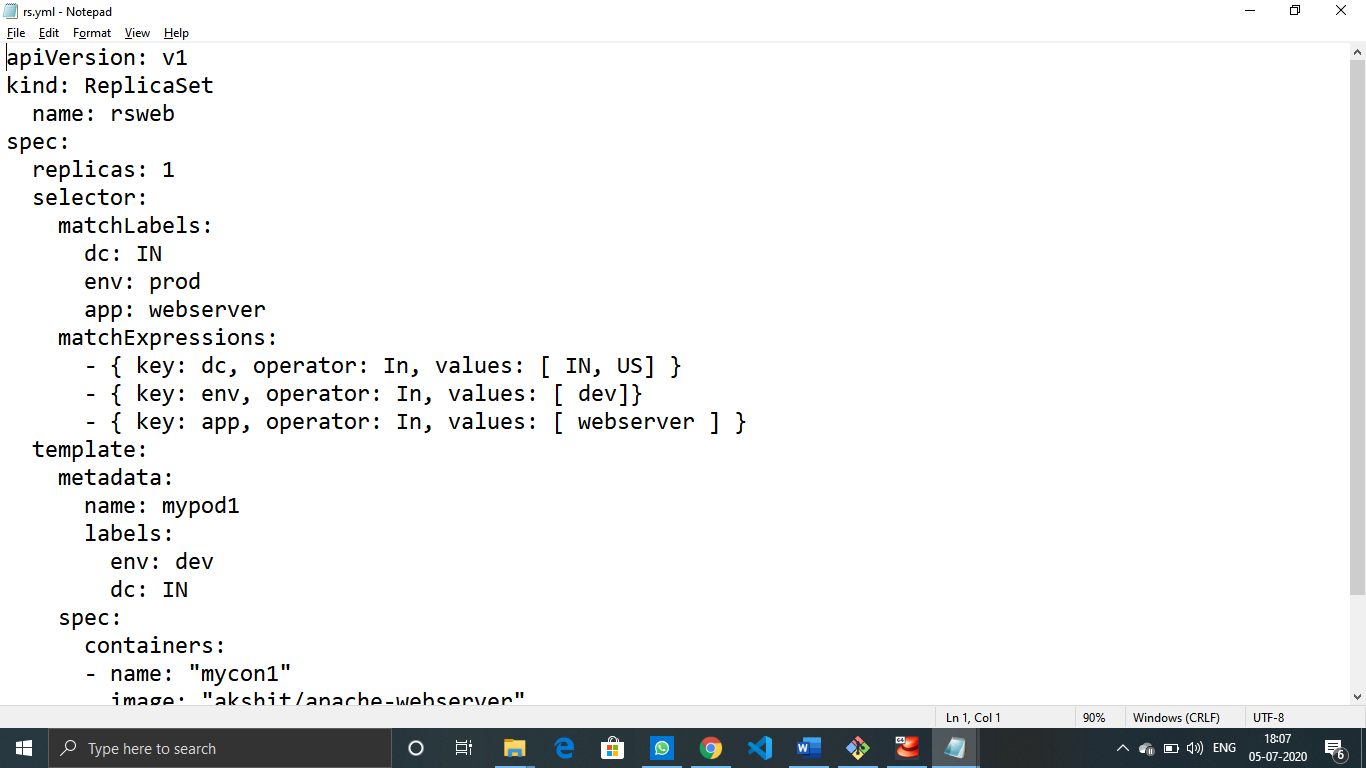
🡪It uses both Set Based selector and Equality based selector.

🡪set based --matchexpressions

🡪equality based --matchlabels

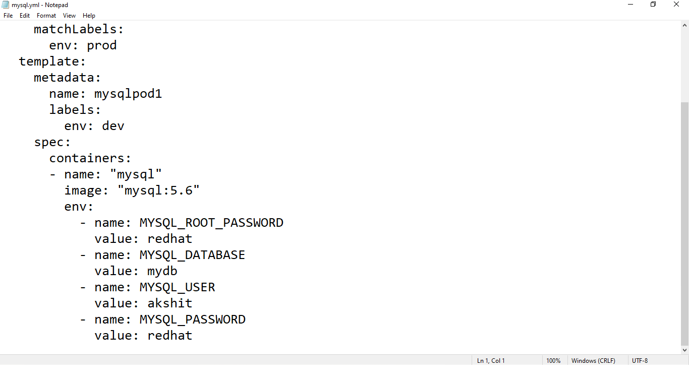
🡪 Kubectl get pods -l “dc in (US, IN), env=prod”

🡪It always watches labels.



Environment variables

🡪many times, we use environment variables for different purpose.  
🡪If we edit bash.rc file then we can set environment variables, but here whenever old pod deleted, new pod created then these variables does not exist there.  
🡪For this we can write env variables inside yml file.



**PVC**

🡪PVC (Persistent Volume Claim)

🡪PVC get the storage from PV.

🡪PVC is per folder.

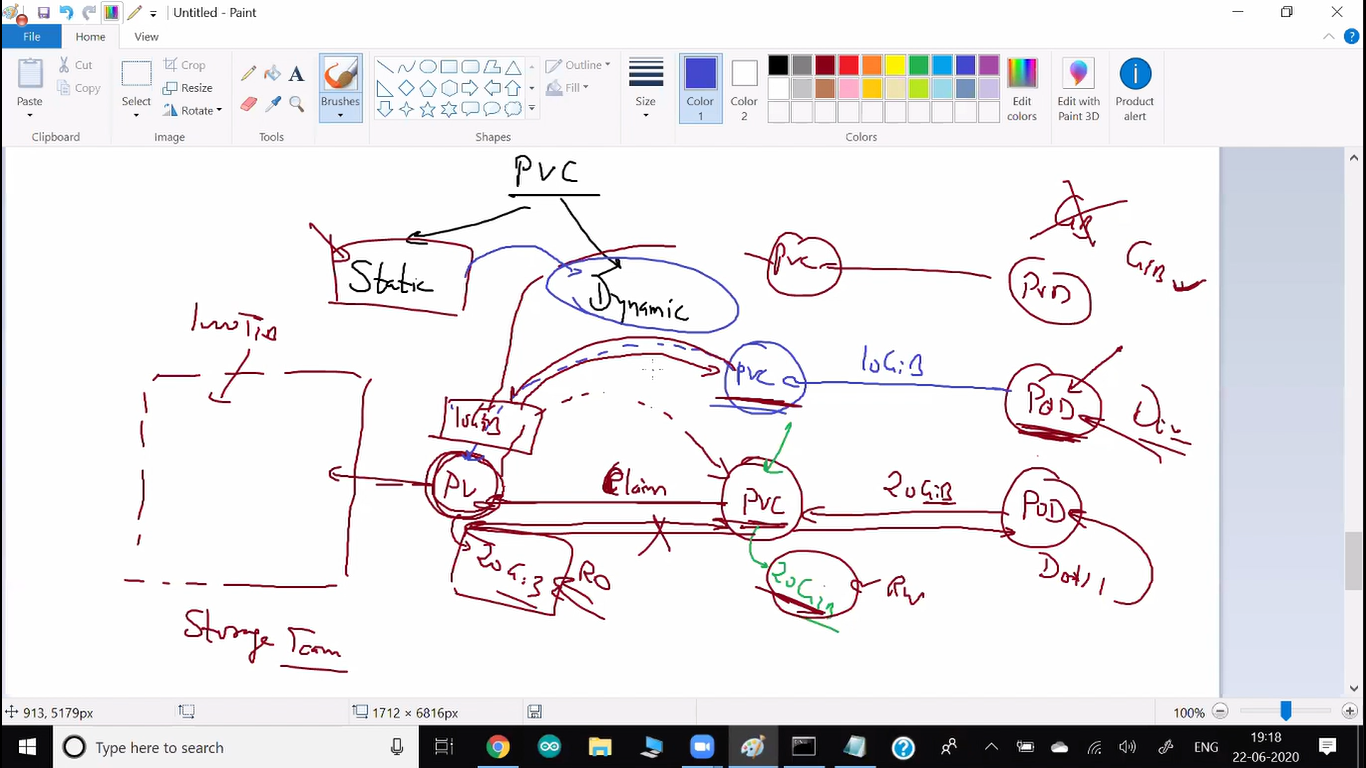
🡪There is two different type of PVC available.

1. Static

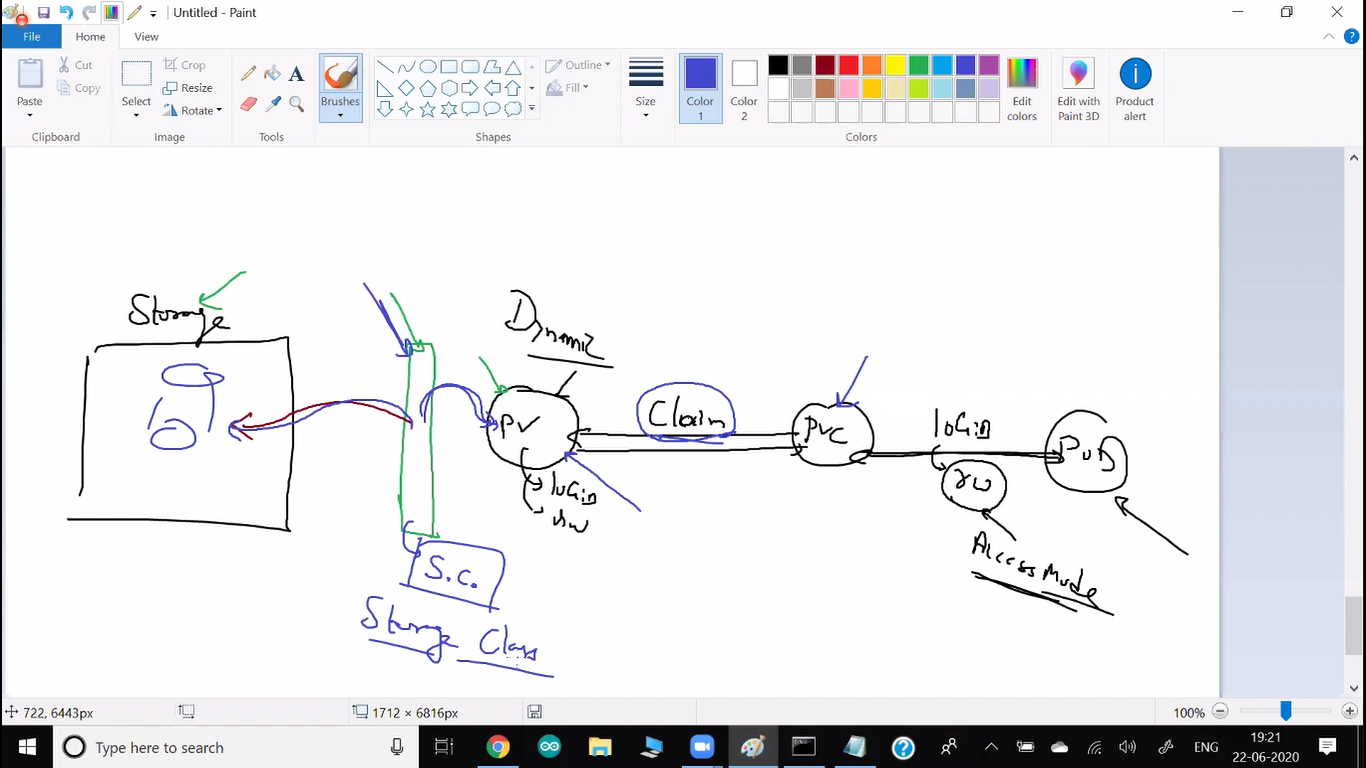
🡪Older, we don’t use this much.

🡪Here our team have to first create, then they only get the storage.

🡪Our team have to know from where to get the storage.



1. Dynamic



🡪Here PVC don’t require any PV to be there.

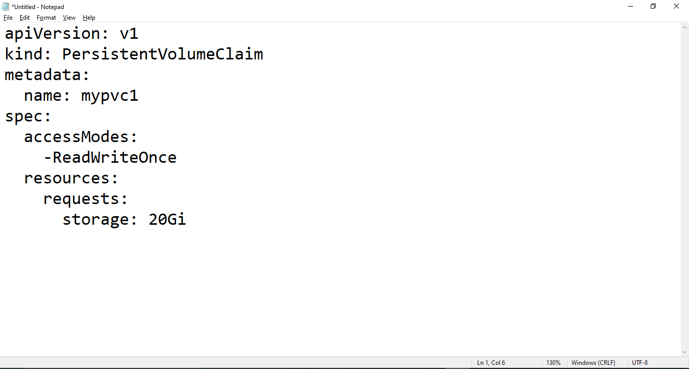
🡪Without a team dynamically-automatically when the request or claim goes on the fly PV creates.

🡪Here SC (Storage class) get the data from storage (like aws or somewhere).

🡪First step to create PVC.

🡪Second step is mounting this folder to /var/www/html

🡪pv will be dynamically created by sc.



🡪kubectl create -f pvc.yml



🡪Rs with PVC.

**SECRET**

🡪kubectl create secret generic mysecret --from-literal=user=vimal --from-literal=mypass=redhat

🡪kubectl get secret mysecret -o yaml

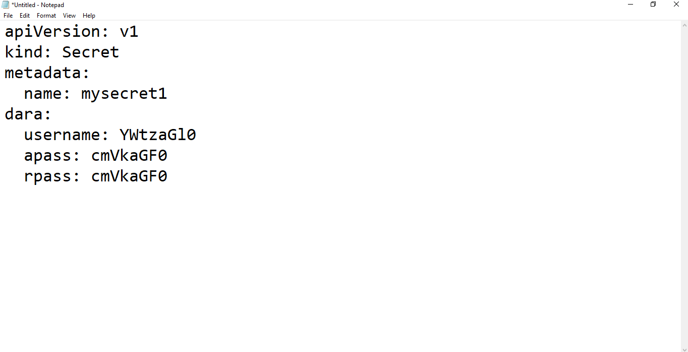
🡪It converts output to the yaml format. It gives you some extra information.

🡪most off resource gives this option to convert into yaml.

🡪Here value is not encrypted.

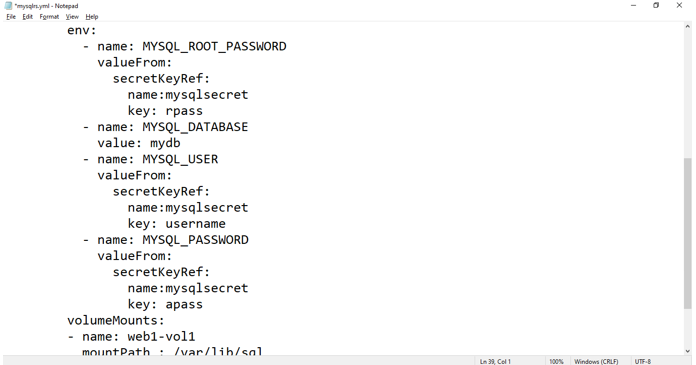
🡪value is encoded in base64.

🡪For using this with yaml file you have to convert the value into base64 at a time of given.



🡪here username, apass, rpass all are variable.

🡪Add this variable to the mysql-pvc file.



🡪Now if you describe command you cannot able to see the login details.

🡪This would very useful when your developer is uploading a complete yaml file.

🡪So, no one can able to get login details from a GitHub repo.

**Service**

🡪Service have their own IP and port number.

🡪This IP never changes.

🡪One static IP one static port we have provide to client.

🡪This IP-port also known as endpoint.

🡪This endpoint is to manage the things as a load balancer.

🡪Common mechanism we use for LB is round robing.

🡪In the Kubernetes world you want load balancer, component for this is known as service.

🡪kubectl get services 🡪kubectl get svc

* ClusterIP

🡪It works as a load balancer

🡪It works inside your cluster

* NodePort

🡪If you want outside laptop, cluster they have connectivity for this pod.

🡪For this you have to use one more service-load balancer.

🡪But here you have to use NodePort

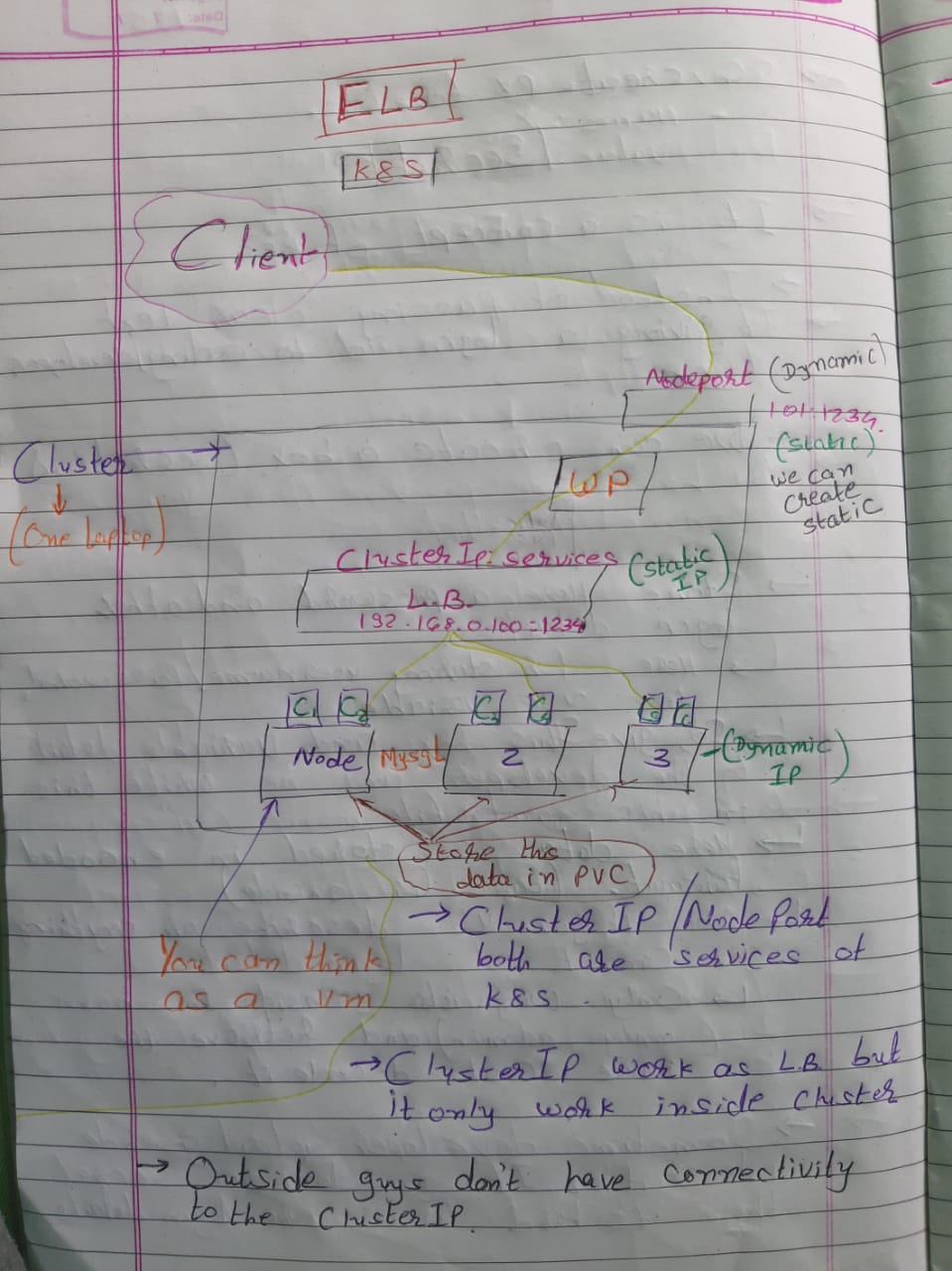
* Load Balancer

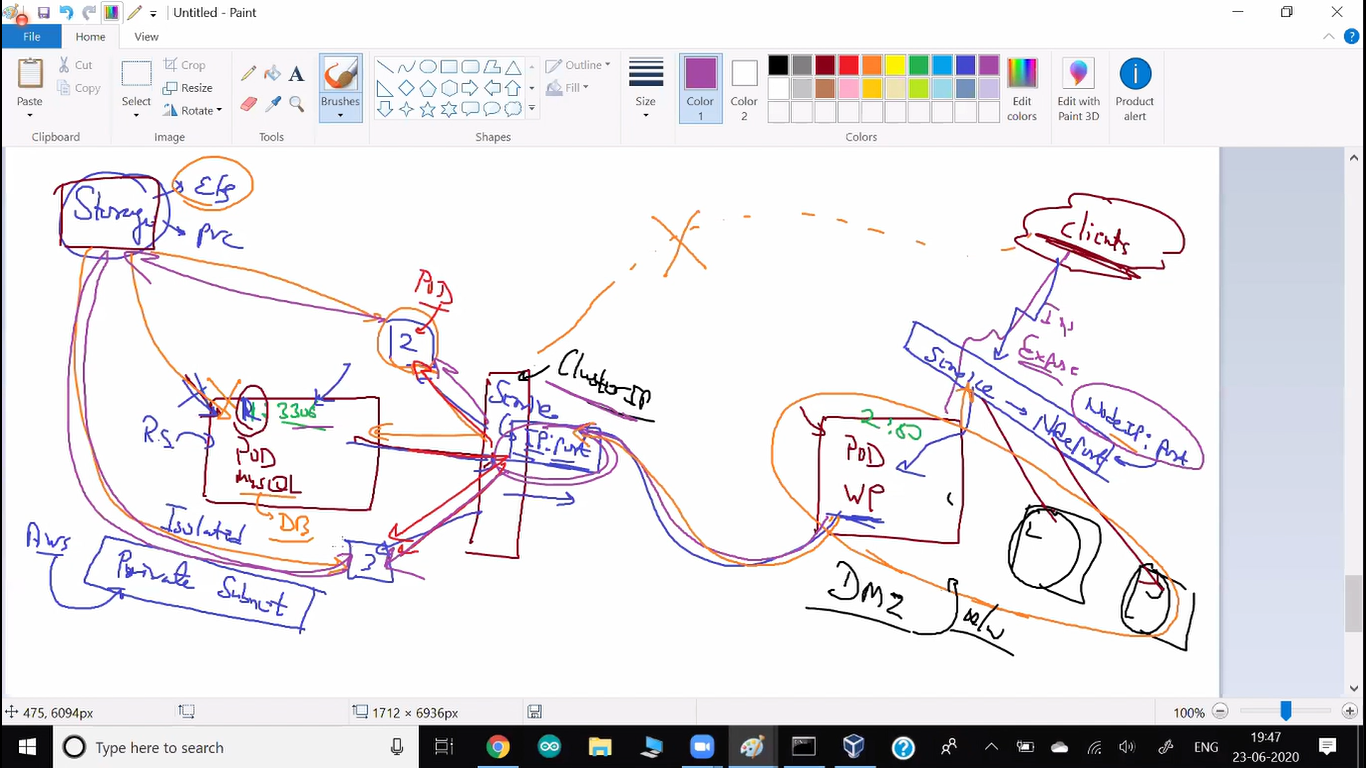
🡪If you create service using type load balancer.

🡪We use this service only when you have multi node cluster.

🡪It also has outside connectivity.

🡪Here when client come, first they connect to first node, then it sends to the second node. (If client hit different container but same node so ram-cpu load increase).





🡪Here client connect to the NodePort.

🡪It works as a load balancer.

🡪 NodePort has outside connectivity.

🡪Pod get data from mysql between this there is clusterIP.

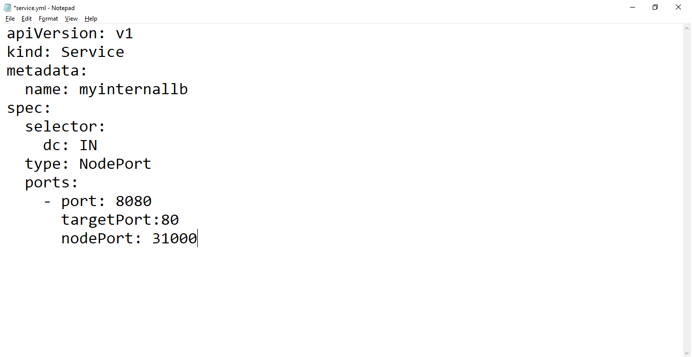
🡪So, no one can connect from outside world.

🡪This clusterIP also work as a load balancer.

🡪You can store data in PVC.

🡪Here, OS working in an isolated world and you provide outside connectivity, so this is known as DMZ (Demilitarized zone).





**Kustomization**

🡪To launch multiple yaml file in a sequence you have to use this facility of Kubernetes

🡪kubectl create -k .

🡪. represent current directory.

**Deployment**

🡪RS can do many things for you.  
🡪But they cannot do rollout for you. They are not made for this.

🡪when the new version come up deployment automatically delete older pods and launch new pods.

🡪main thing is you don’t get downtime.

🡪deployment will use multiple strategy for rollout.

1. Ramped

🡪It will slowly update all the pods.

🡪suppose 3 pods running, 2 have clients and one is free.

🡪now new version come. So, deployment terminate the 3rd pod and create a new pod. Now when new client come, they are connected to newer version.

🡪Now whenever 1st or 2nd pod free they do same, launch new pod.

🡪ramped is also known as rolling update.

🡪It is by defalut strategy in deployment and it is main strategy.

1. Recreate, canary, blue/green strategy.

🡪kubectl apply -f deployment.yml

🡪kubectl get deploy

🡪kubectl get deployment

🡪kubectl get deployments



🡪Now whenever new code come, edit the file and add the version.

🡪Run kubectl apply -f deployment.yml

🡪kubectl rollout status deploy/<id>

🡪kubectl rollout undo deploy/<id>

**Master-Slave Architecture**

🡪Slave is also known as worker node.

* **Kubectl – req to API server**

🡪kubectl (client) must know IP address of master and port number of API server.

🡪port number of API server is 8443.

🡪One program is running in Master Node known as API server.  
🡪Main responsibility of this program is to listen the client (Kubectl).

* **Kube Schedular**

🡪In Kubernetes master node has one more program known as Kube schedular.

🡪They will decide on where to launch pod.

🡪When you ask kube schedular to launch two pods. They always try to distribute the pod in different different slave.

* **kubelet**

🡪master communicate to a program kubelet (In slave node).

🡪Kubelet is a program contact to docker engine launch a container and wrap it in the pod.   
🡪Without kubelet you can’t launch the pod.

🡪In master also have to launch pod.

🡪All the master services are going to launch inside one container-pod.

🡪So, we require docker engine and kubelet in the master.

* **Kube Controller – monitoring**

🡪They keep on doing monitoring.

🡪They keep on checking all the node and pod.

🡪Whenever something failed, controller send this information to schedular and ask to launch one more pod.

🡪If node failed, they also tell this to schedular (so they launch pod in other nodes only).

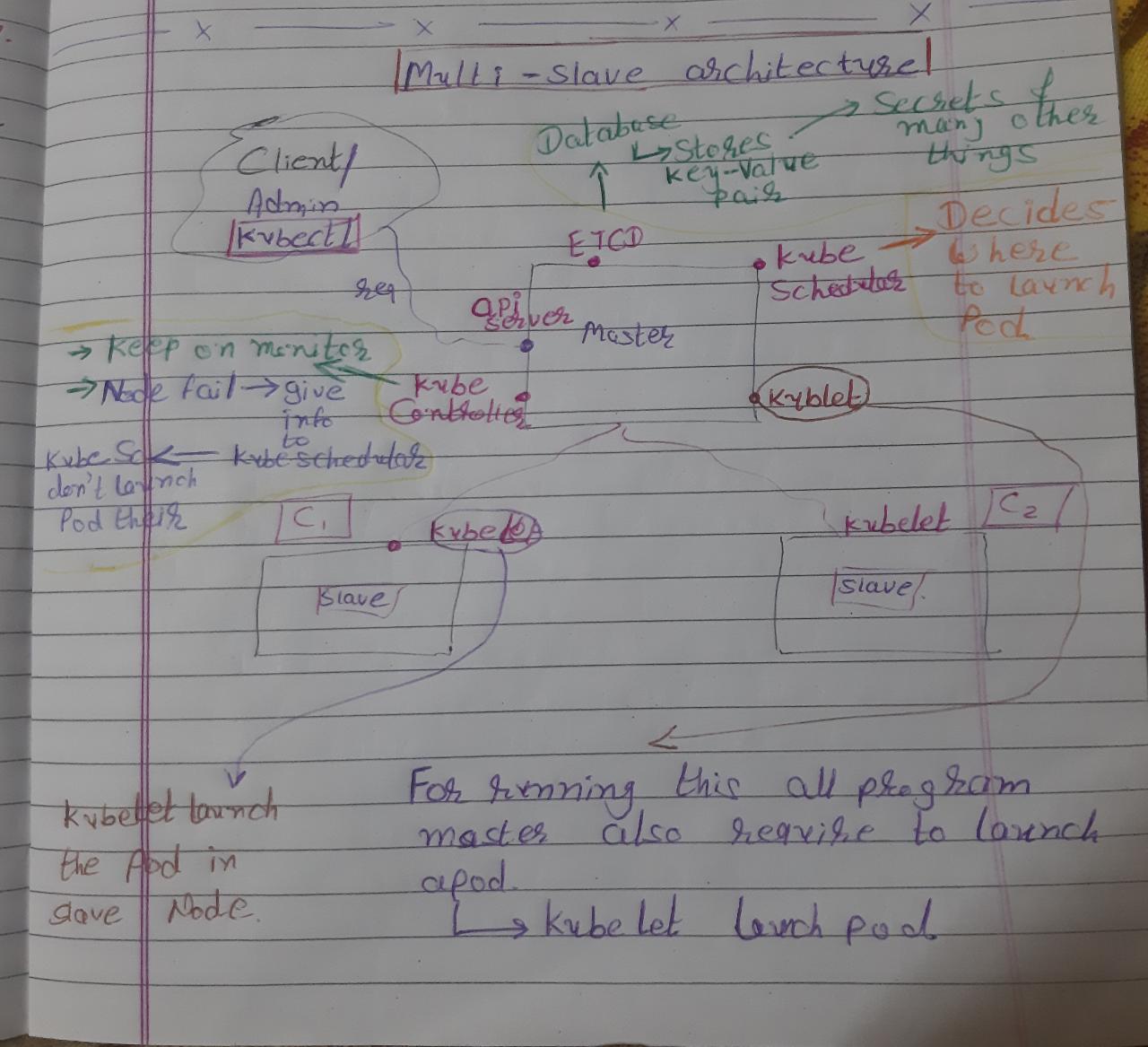
* **ETCD - database**

🡪Master has lot of things it wants to store permanent.

🡪For this there is also one database service running inside master.

🡪Name of this database is etcd.

🡪key-value pairs are stored in etcd database (Secret).



**🡪Requirement to create this architecture**

🡪mkdir dvd 🡪mount /dev/cdrom /dvd

🡪for permanent /etc/rc.d/rc.local

🡪configure yum at /etc/yum.repos.d/dvd.repo

🡪yum install net-tools vim

🡪configure yum for docker

<https://download.docker.com/linux/centos/7/x86_64/stable/>

🡪yum install docker ---nobest

🡪disable firewall

🡪Or you have to manually add the port of API server kubelet and all and enable.

🡪Disable selinux 🡪vim /etc/selinux/condif make permissive

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

exclude=kubelet kubeadm kubectl

EOF

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

🡪For install kubeadm, kubelet, kubectl

🡪kubeadm is only for setup your cluster.

🡪vim /etc/docker/daemon.json

🡪{

"exec-opts": ["native.cgroupdriver=systemd"]

}

🡪write this

🡪docker don’t support cgroup driver you have to change driver to systemd

🡪systemctl restart docker

🡪disable swap

🡪vim /etc/fstab

* Go to this comment last line.

🡪yum install iproute-tc

🡪cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sudo sysctl –system

🡪systemctl start kubelet

🡪systemctl enable kubelet

🡪Now system configured. Clone it or do this configuration manually in all the slave vm.

🡪Cloning time click on generate new mac address

**After Configuration steps**

🡪Set hostname to all master and slave

🡪hostnamectl set-hostname master

🡪hostnamectl set-hostname slave1

🡪hostnamectl set-hostname slave2

🡪exec bash -- so it updates name in system

🡪set dns or set local dns in every system

🡪vim /etc/hosts

192.168.43.253 master

192.168.43.152 slave1

192.168.43.61 slave2

🡪kubeadm init --pod-network-cidr=10.10.1.0/16

🡪init kubeadm in the master

🡪By using --pod-network-cidr command you can give IP range.

🡪mkdir -p $HOME/.kube

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

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

🡪kubectl get nodes

🡪kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

🡪Here applies command first time work as a create, and then whenever you change something it is work as a replace.

🡪Now join slave to master

🡪For this use token.

kubeadm join 192.168.43.253:6443 --token ve2lxi.fjyxg36rryiovegl --discovery-token-ca-cert-hash sha256:92f8381ca357bed0dc5ff50429310c793abe45331653c17d2b1759ab8260155c

🡪Run this in both the slave

🡪Now set windows as a client.  
 🡪There is one config file inside /.kube in master.

🡪copy this file in the windows

🡪kubectl get nodes --kuberconfig config

🡪Run this from desktop (config file is in desktop)

**Usecases**

🡪reverse proxy

🡪Load balancer

🡪Fault Tolerance  
🡪Service discovery   
🡪Scale in scale out automatically

🡪Increase/Decrease Ram = Vertical scalling

🡪Increase/Decrease OS = Horizontal scalling

🡪Use PVC so you can get a permanent storage.

🡪Use secret box so you can hide your login details directly.

**Commands**

* Kubectl create deployment myweb --image=akshit/php-webserver
* Kubectl get pods
* Kubectl delete pods myweb-a23ebf-aq55
* Kubectl describe pods myweb-a23ebf-aq55

🡪They show you IP and many other information.

🡪Because they launch using container, they can’t able to connect with outside world.

🡪Containers are isolated.

🡪You have to expose it, patting concept.

* Kubectl expose deployment myweb --type=NodePort --port=80
* Kubectl get all

🡪They auto assign a port to expose you can see it using services.

* Kubectl scale deployment myweb --replicas=3

🡪Here behind the seen they provide load balancer for you.

🡪

* Kubectl delete pods --all
* minikube IP.

🡪It gives you IP of a system.

* Kubectl run mywebpod --image=akshit/apache-webserver

🡪If you only launch a pod using this command, this time k8s don’t work as fault tolerance.

* Kubectl get pods --show-labels
* Kubectl label pods myweb1 env=test
* Kubectl get pods --selector dc=US
* Kubectl get pods --selector dc!=US
* Kubectl get pods -l dc=US,env=prod

🡪You can use -l instead of –selector

🡪AND

* Kubectl get pods -l “dc in (US, IN)”

🡪OR

🡪It shows all dc which are present in US or IN.

* Kubectl get pods -l “dc in (US, IN),env=prod”

🡪 It shows all dc which are present in US or IN where env=prod

* Kubectl delete pods -l env=dev
* kubectl create secret generic mysecret --from-literal=user=vimal --from-literal=mypass=redhat
* kubectl get secret mysecret -o yaml

🡪It converts output to the yaml format. It gives you some extra information.

🡪most off resource gives this option to convert into yaml.

* kubectl get services 🡪kubectl get svc
* kubectl get deploy
* kubectl get deployment
* kubectl get deployments
* kubectl rollout undo deploy/<id>

--------x--------x---------x----------x----To be continued -----x---------x----------x----------x-------

Why worker node connected to master

🡪Because we have kubelet program running in the slave

🡪worker node has to launch kubelet program and ask master to connect.

This connection is known as join, we have to join only once.

Where master have to launch the pod.

🡪kube schedular is a program is plan – schedule

🡪If you want to create multi node cluster by your own.

🡪For this we have to use kubeadm.

Kubectl get ns

🡪it gives all namespace.

🡪You can think namespace as a room.

🡪There is a different different namespace in a Kubernetes so one namespace data does not mess with other namespace.

🡪Our all pods, rs and all are running in default namespace.

🡪Kubernetes internal database-etcd, kube schedular and all things are there inside kube-system namespace

🡪kubectl get pods -n kube-system

Configmap

🡪Configure file inside pod.

🡪If pod terminate our configuration file also loose.

🡪So, you can put all the detail of configure file in configmap.

🡪kubectl create configmap mycm1 –from-literal=file1=hello

🡪kubectl get cm

🡪kubectl describe cm mycm1

🡪you can make a permanent Prometheus file (prom.yml)