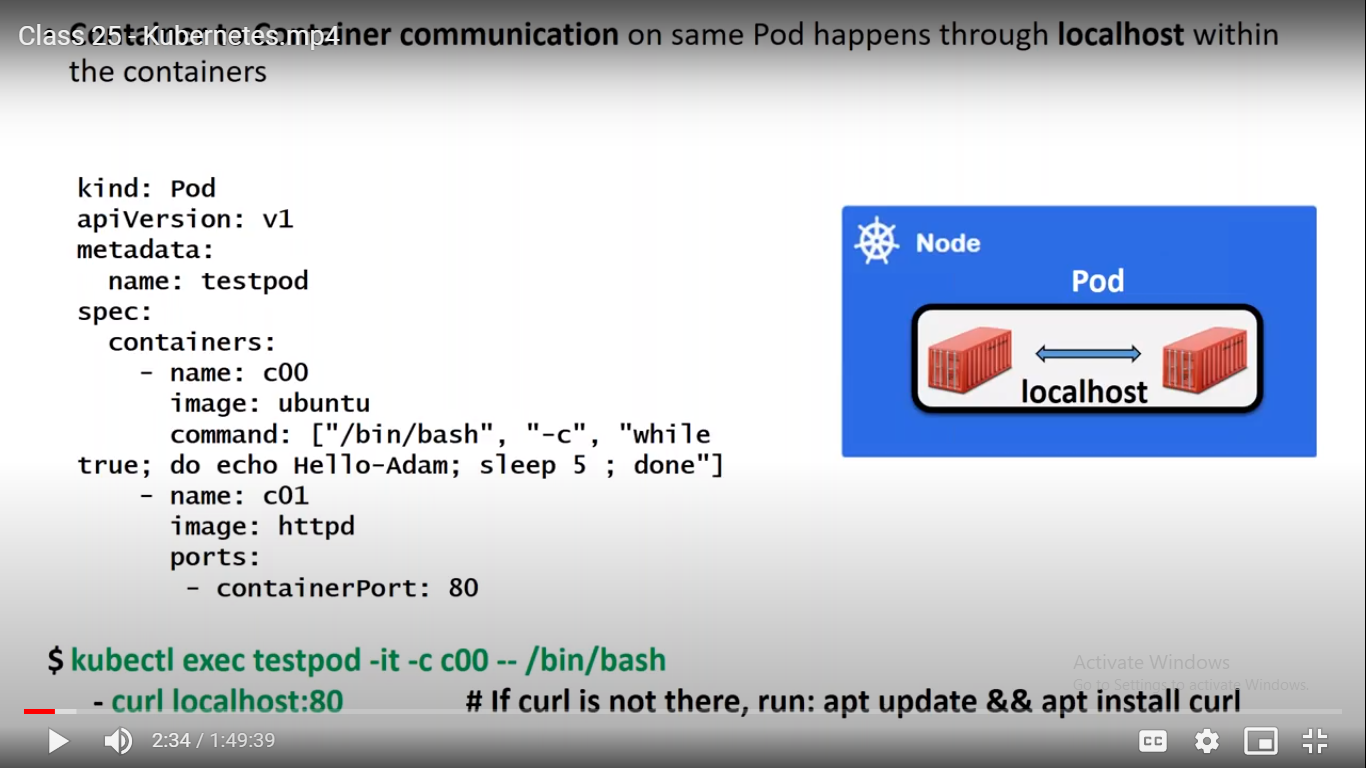
# Kubernetes Networking :-



In Pod we can have one container by default but if it requires we can have 2 container as well.

Here we can use localhost to connect other containers.

Created 2 containers using pod object.

ubuntu@ip-172-31-41-23:~$ kubectl exec testservice -it -c c00 -- /bin/bash

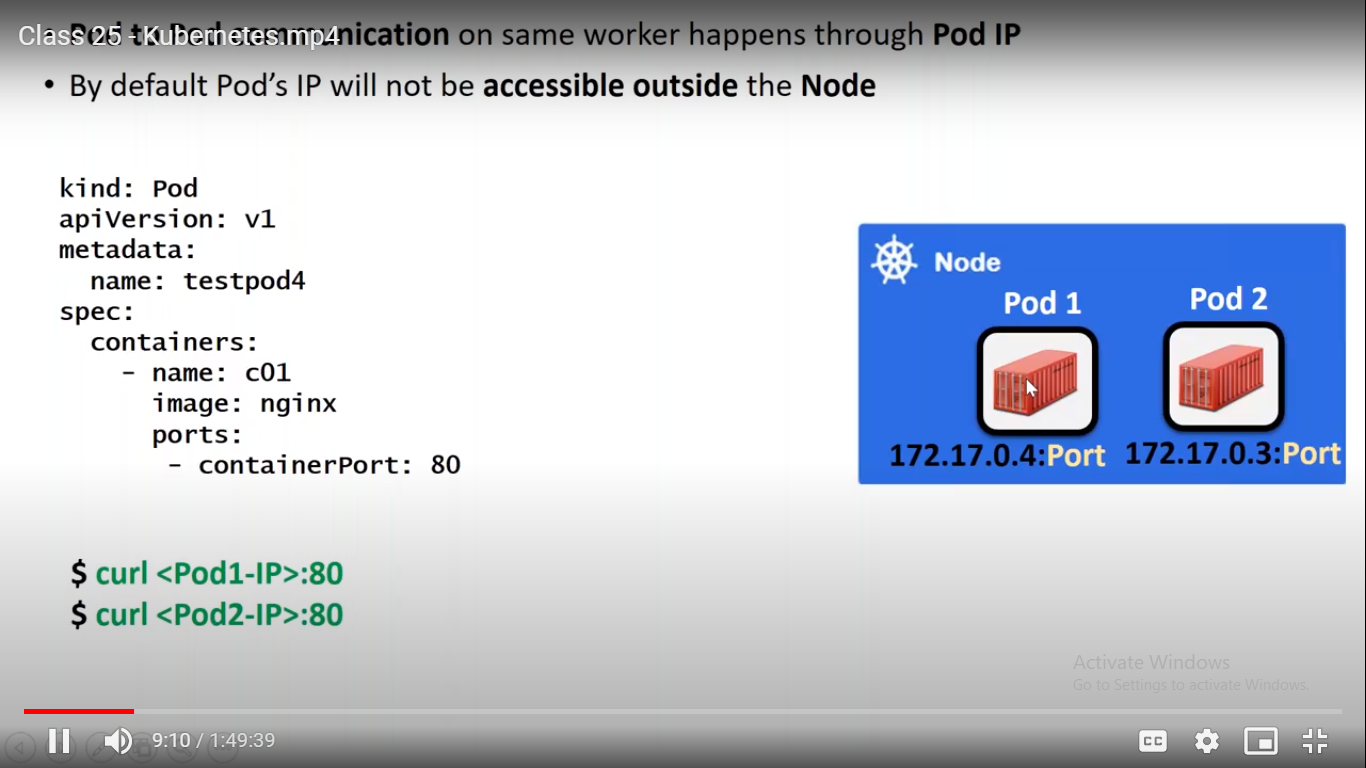
Going inside the container c00 and executing the first command.

Here we have to install curl as ubuntu image is the vanilla flavor so it won’t have curl in it.

apt update && apt install curl

So localhost will be used to connect inside pod and container to container.

## 2- Pod to Pod Communication.



Two scenario here.

1. If you have pod on the same machine and you want to communicate between it.

Use Pod IP

If you have pod on the same machine then will be using pod ip

ubuntu@ip-172-31-41-23:~$ kubectl apply -f pod3.yml

pod/testservice created

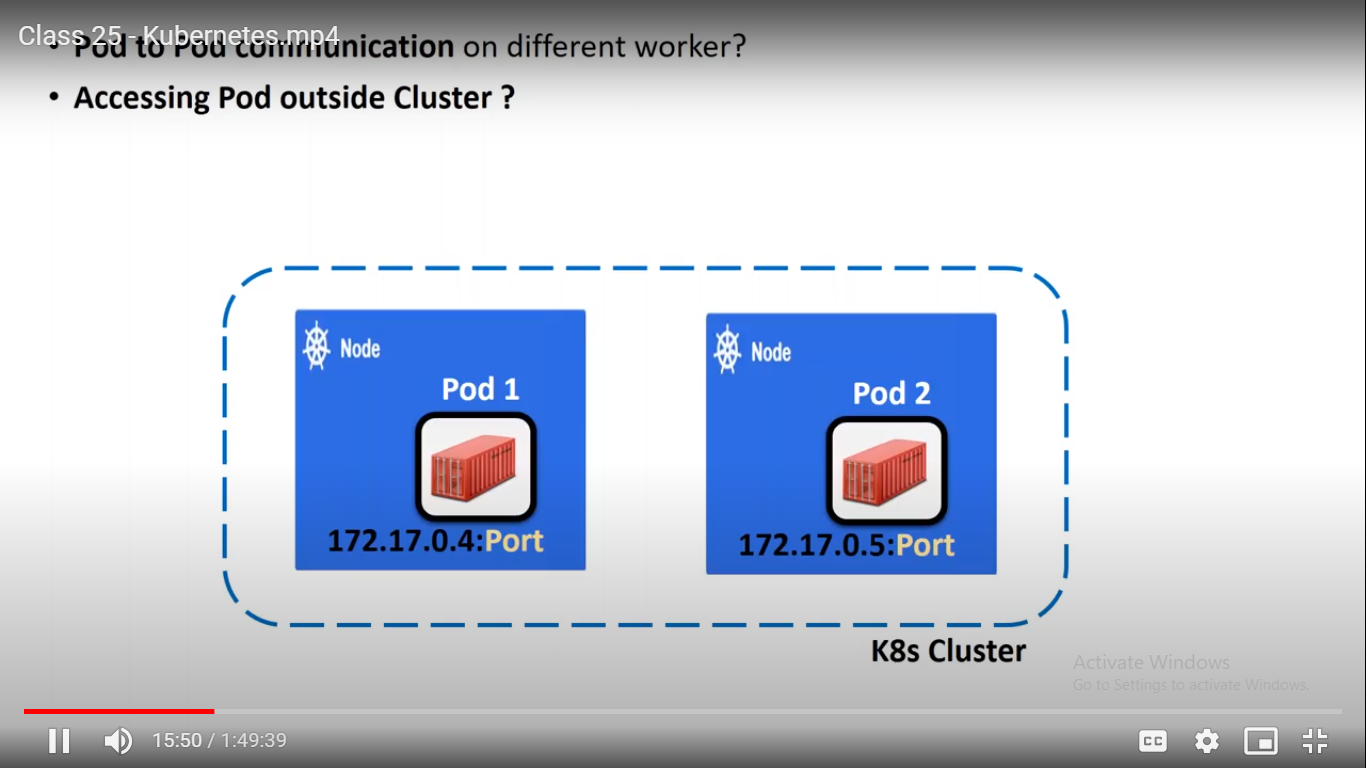
ubuntu@ip-172-31-41-23:~$ kubectl get pod -o wide

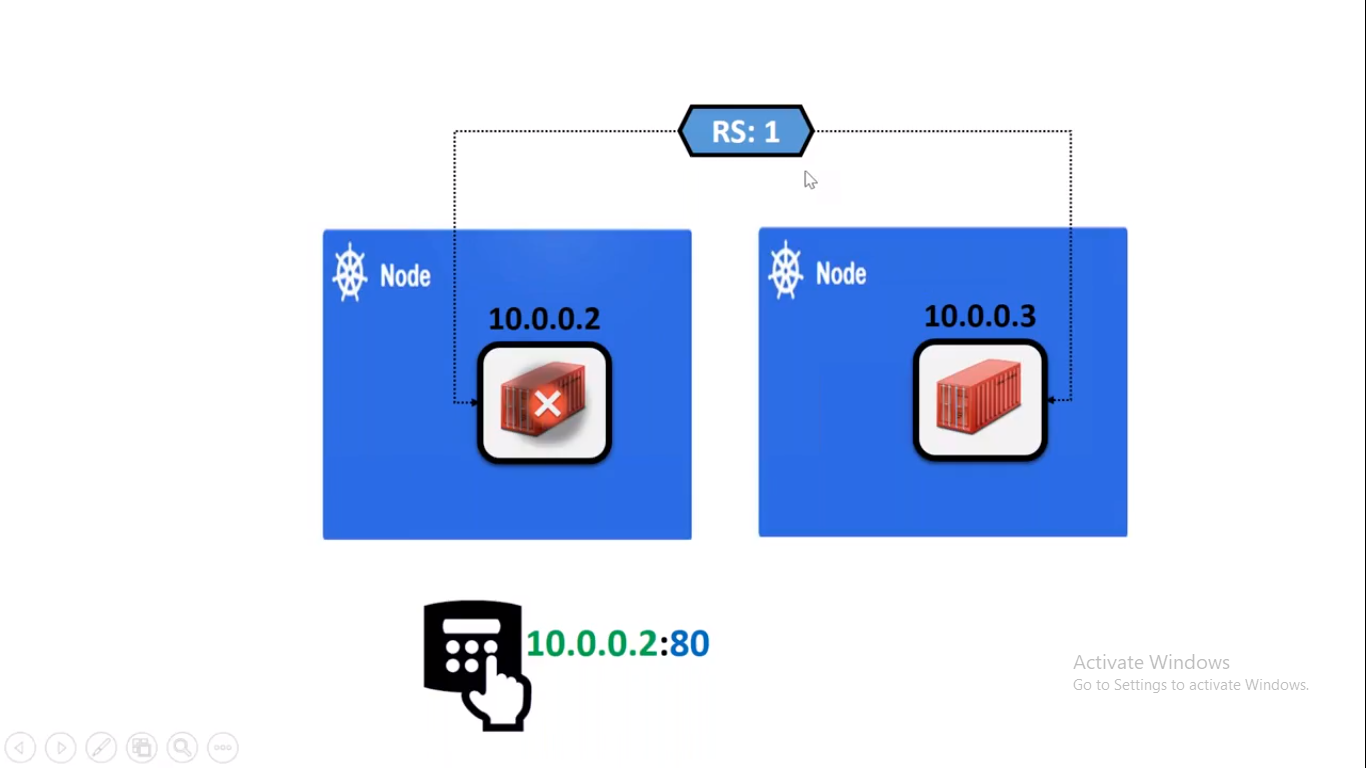
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

testservice 1/1 Running 0 23s 10.244.2.10 ip-172-31-45-172 <none> <none>

ubuntu@ip-172-31-41-23:~$ curl 10.244.2.10:80

How to do pod to pod communication if pod available in different machine.



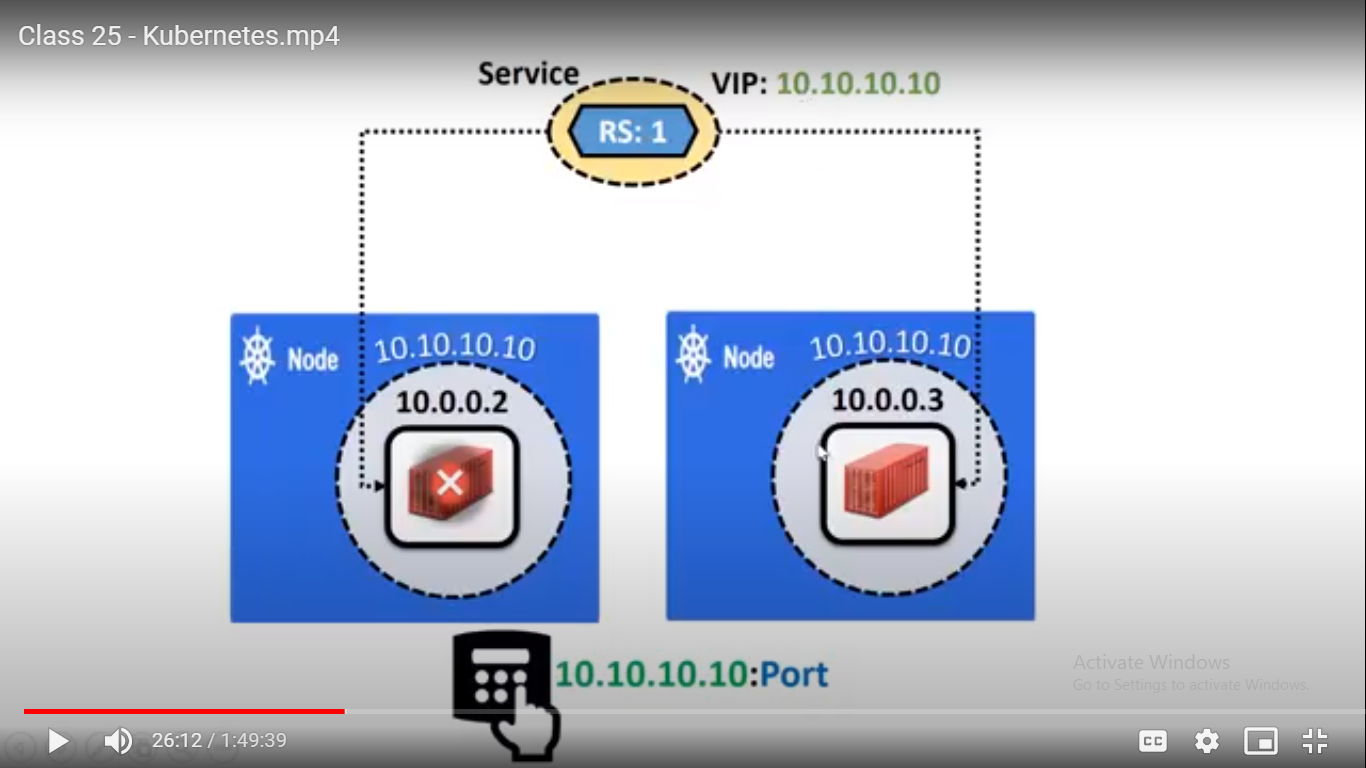


Hereone pod is created by RS object and to access it we can use the ip of that pod and able to access it.

But if this pod goes down than RS will create new pod and then IP will be also replaced so we cannot able to access it by using old IP.

This is the main problem here.

If you want to communicate between pod to pod in different machine or in same cluster or outside the cluster then new object is called services.



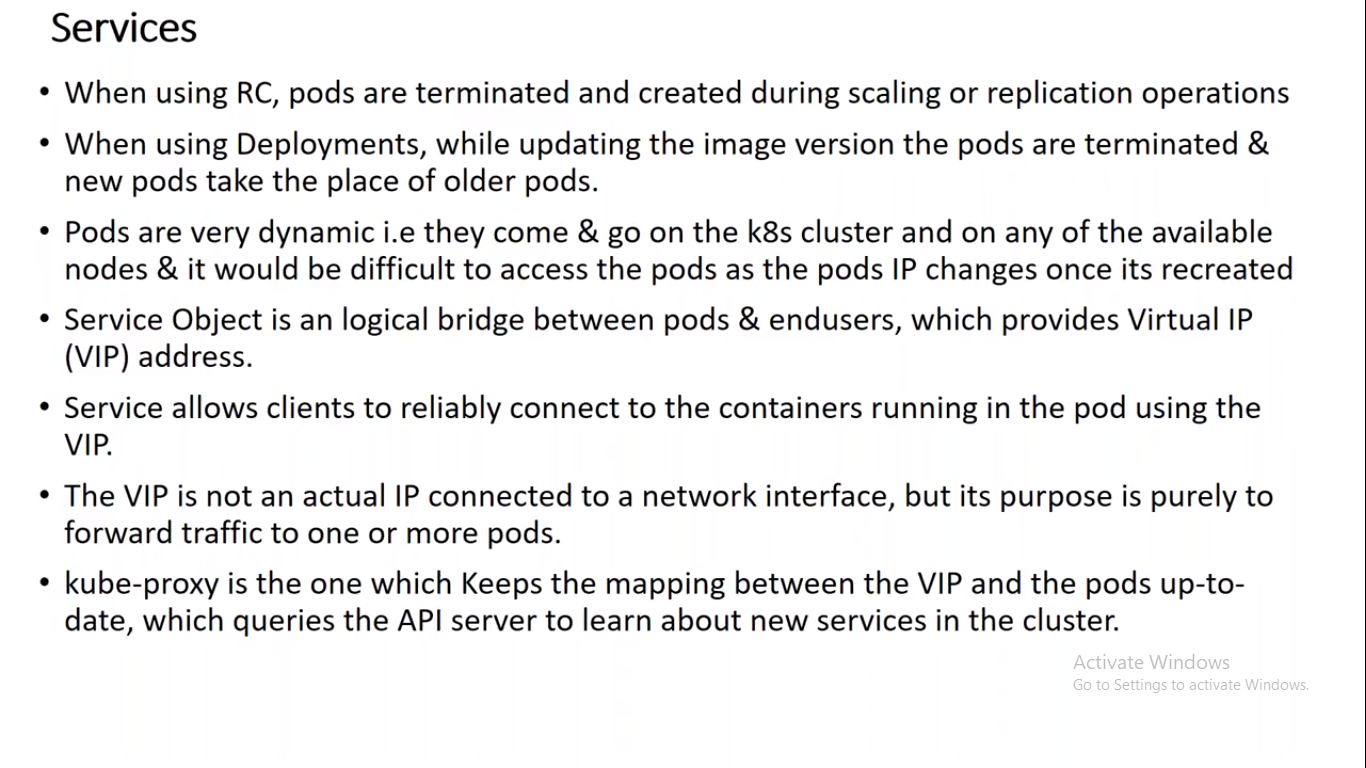
So to solve above problem we have new object called service.

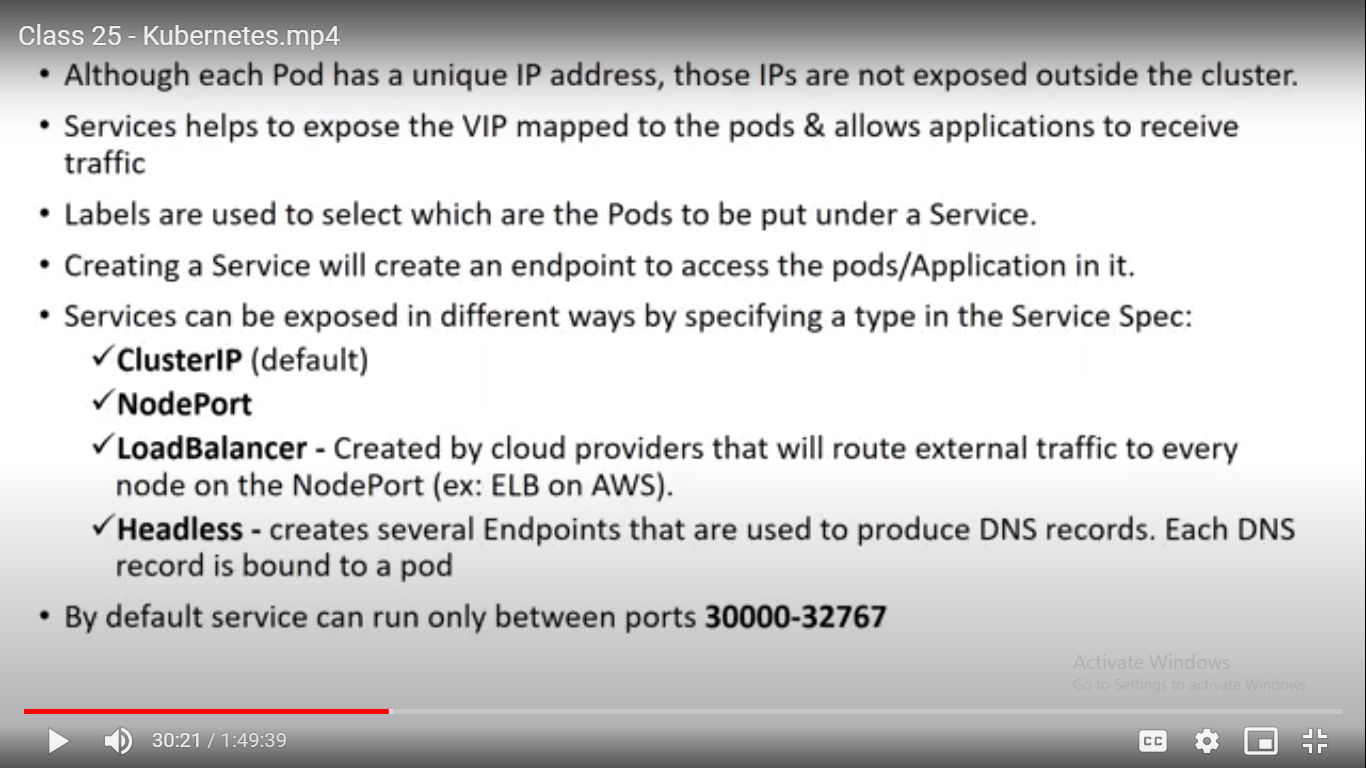
Here on top of RS object a new layer will be there and it the service object.

What service object will do here is that it will give one unique ID called **VIP**

Which will be constant and if we want access pod we will be using VIP:port and VIP will be communicating to the respective pod so even if pod goes down and it will create new pod with new IP then also we can access that pod using VIP:80.

**Defination:-**

****

****

Here VIP is working able to connect through pod is by using labels only.

It will tell the service that go to the pod which is having that label.

In Service object also we have different types.

Actually if we want to access the application outside the cluster i.e. machine not available in that cluster then service will also expose the **port** and to access the application we have to use the VIP:SEVICEPORT

The port ranges between **30000-32767**

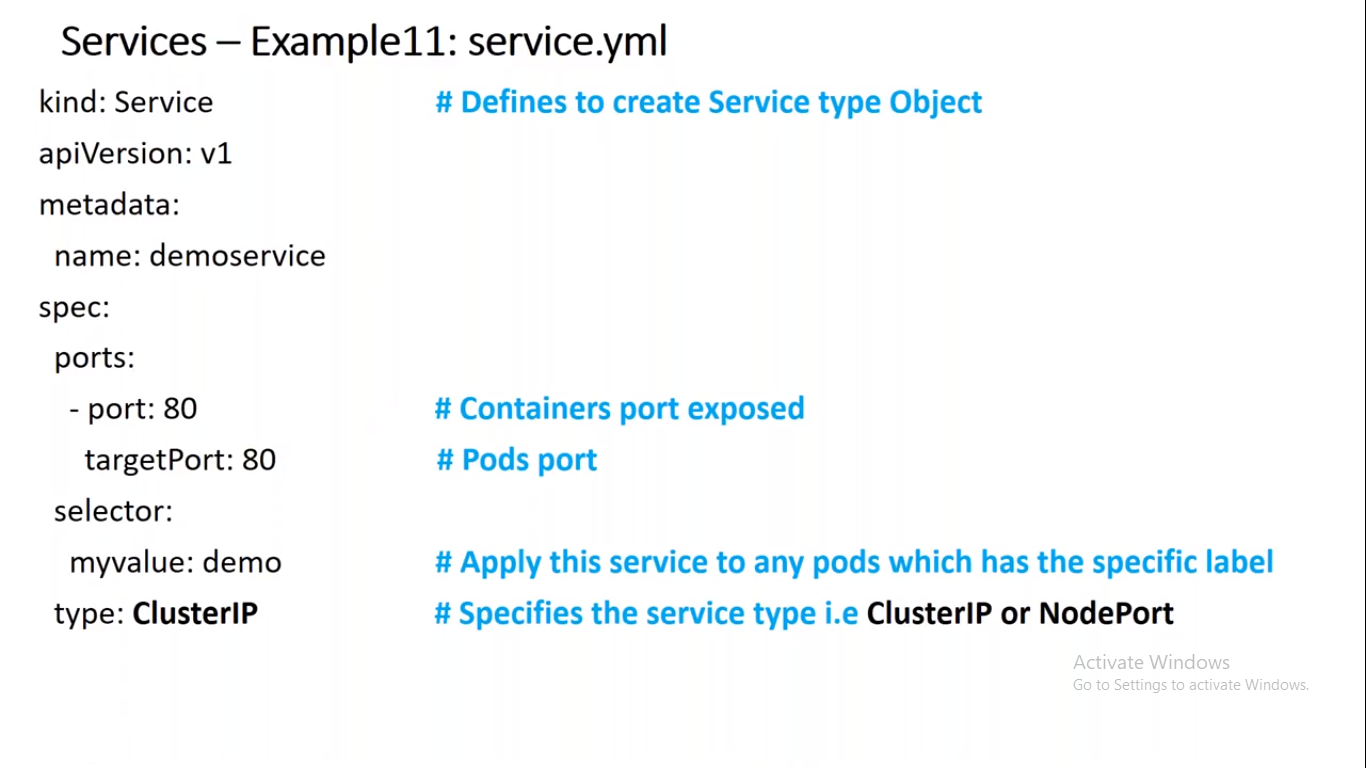
**Cluster IP**



This Cluster IP is only available within the cluster only.

It is mainly used to connect between 2 component of microservices.

For example:- If you are accessing facebook outside and internally facebook wants database so it can be achieved by using ClusterIP.



Untill now RS is going to call Pod.

Here service is layer which is not related to any object so it has separate definition.

Here main thing is spec.

Here we can see that in selector as demo.

So here it means apply this service to any pods which has specific label i.e. demo.

Here we can have any pod i.e. RS, Deploy.

type is CLUSTERIP and i.e. default IP.

Here First we created one pod which is having label as demo and similarly service object also with label as demo so here we are running the service object on top of pod object.

And we can see below we have one IP i.e. CLUSTER IP so we can access it from worker machine as well.

ubuntu@ip-172-31-41-23:~$ kubectl get service -o wide

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

demoservice ClusterIP 10.110.43.7 <none> 80/TCP 43m myvalue=demo

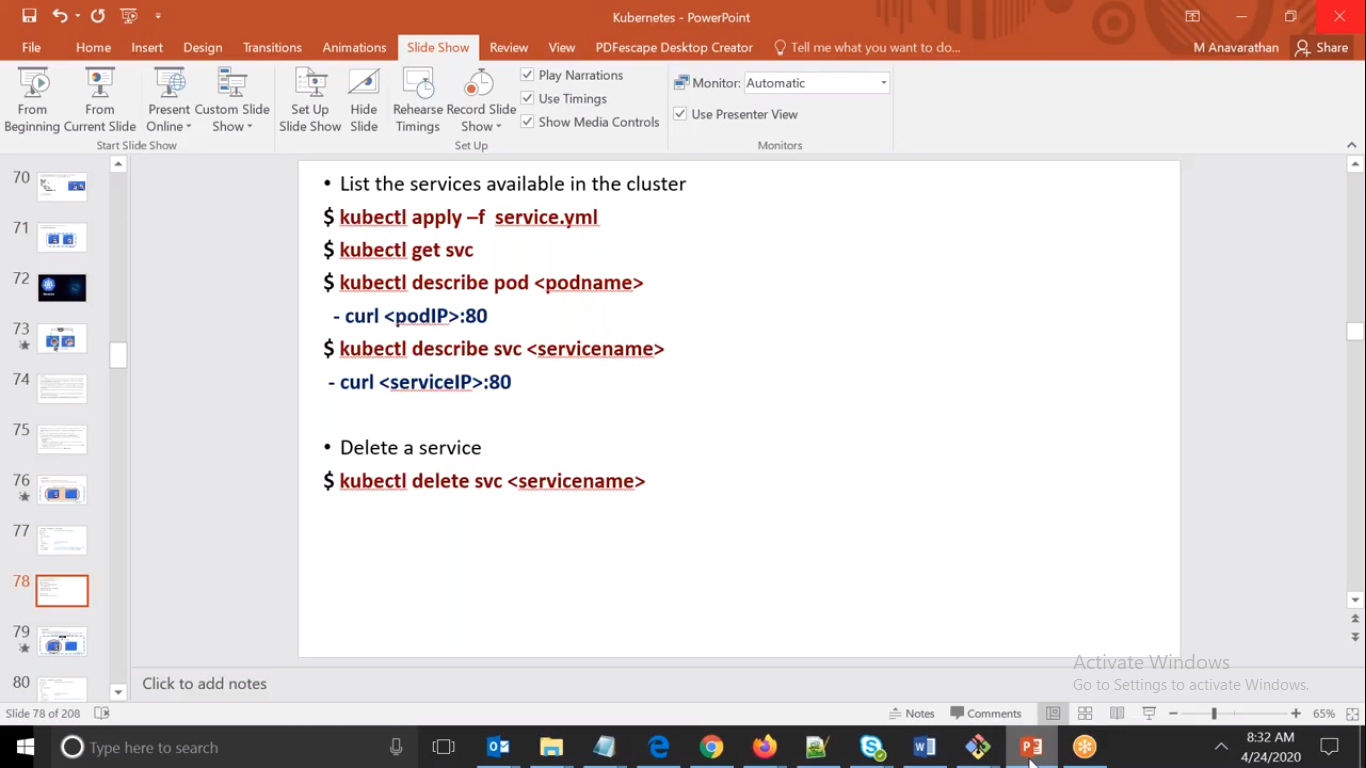
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 47h <none>

Worker Machine..

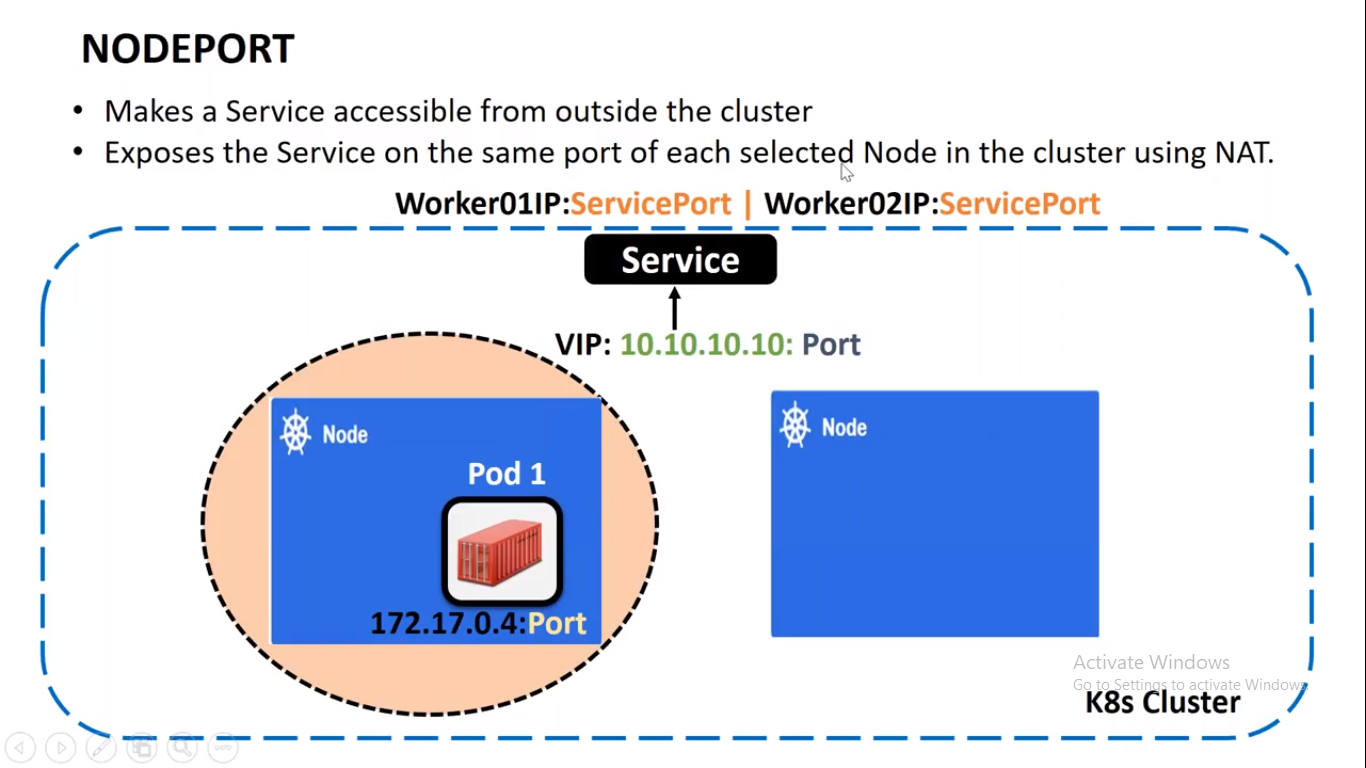
Last login: Sat Apr 25 10:14:13 2020 from 36.255.86.100

ubuntu@ip-172-31-45-172:~$ curl 10.110.43.7:80

<html><body><h1>It works!</h1></body></html>



**2.NODEPORT**



If you want to access outside the cluster then we have to use NODEPORT.

But here that Service will also expose one port i.e. Service Port that will be in range of 30000- 32767.

ubuntu@ip-172-31-41-23:~$ kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

demoservice NodePort 10.110.43.7 <none> 80:**30024**/TCP 63m

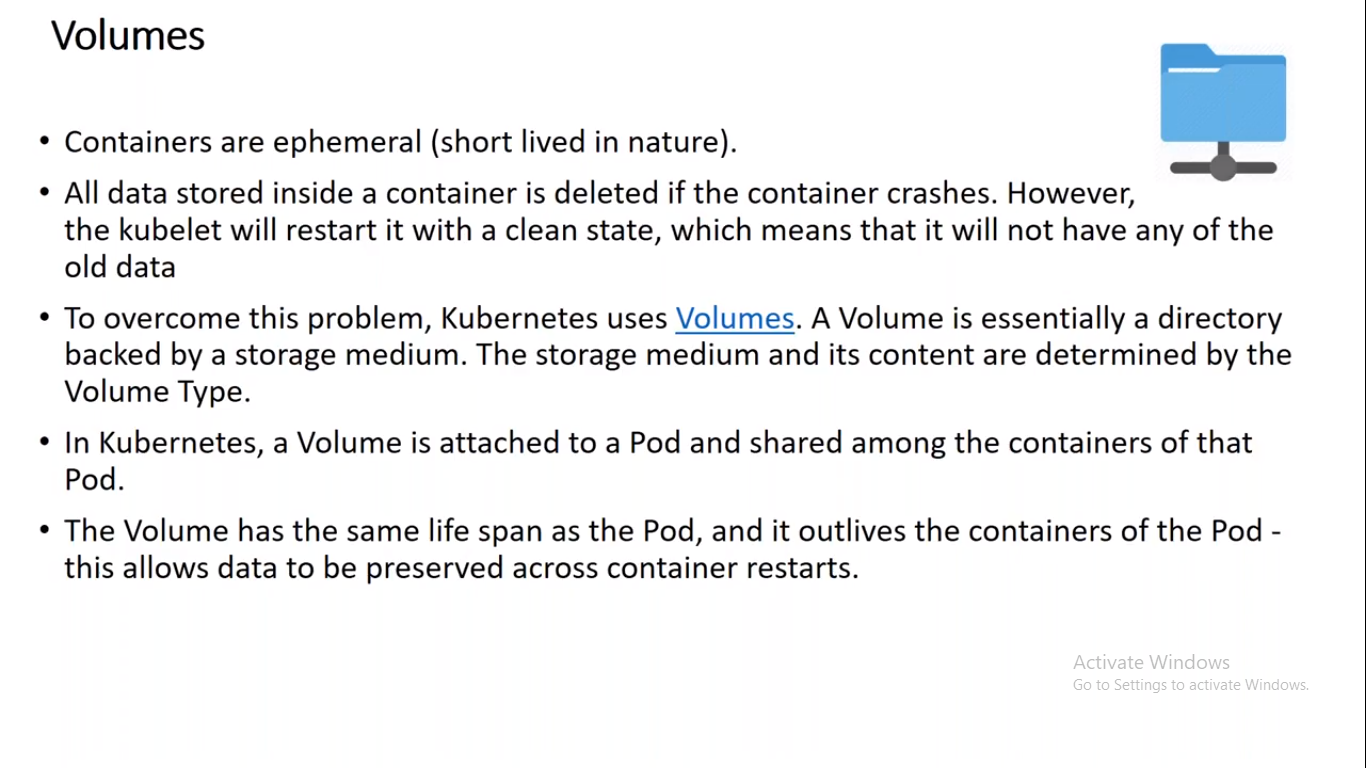
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 47h

Here we can see that it has given one additional port i.e. cold as service port which will be available to all the machine in the cluster.

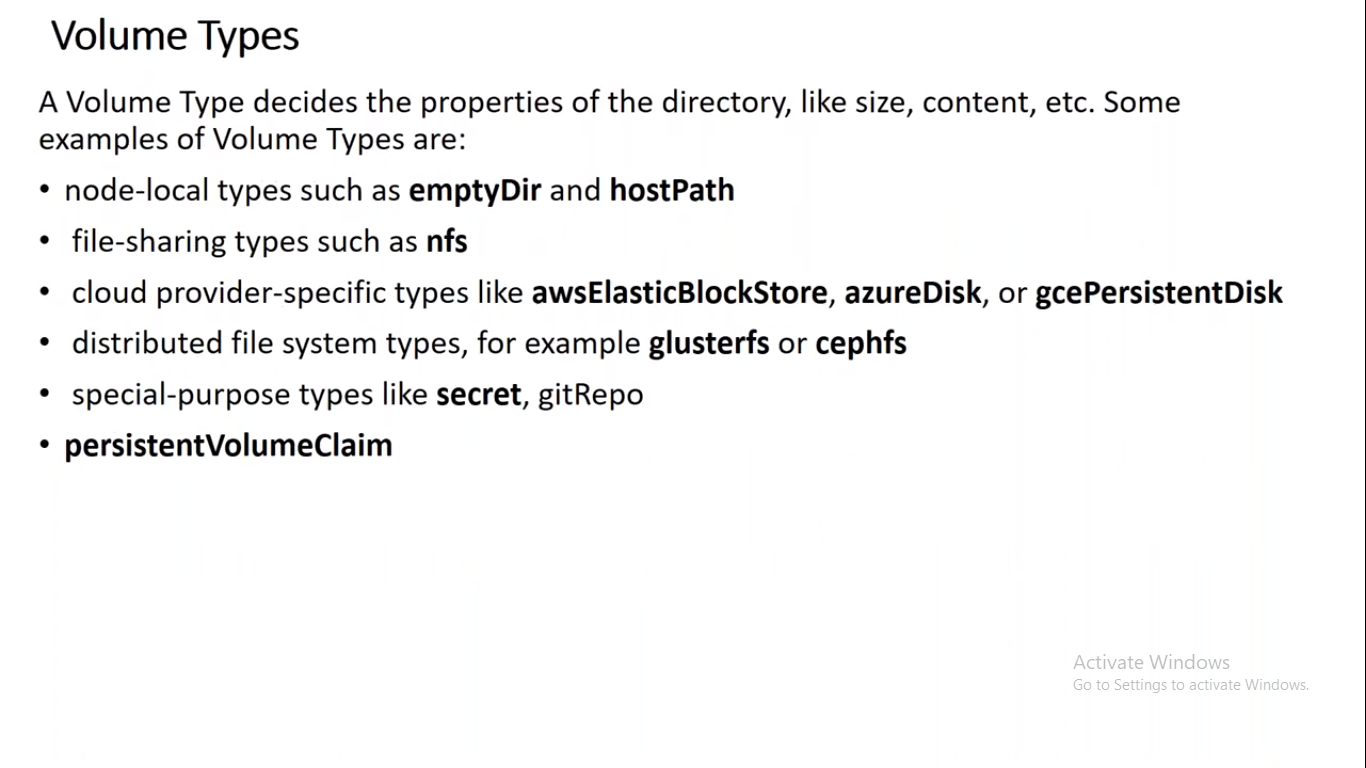
**LoadBalancer:- Actually** to access outside the cluster we requires IP of the worker or master machines but if we don’t know that ip we cannot access it.so to overcome this we have another object called as LoadBalancer, this is required when we have requirement in production.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

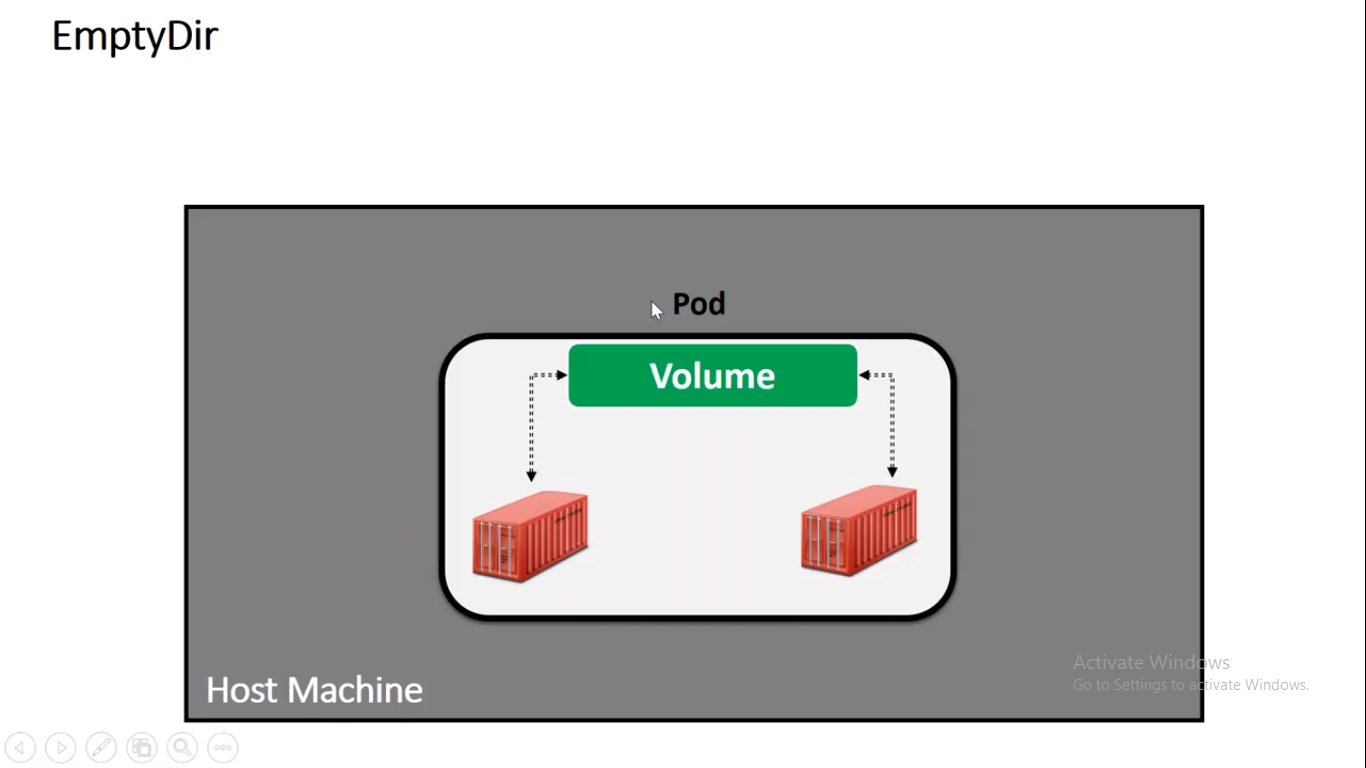
# Volumes



Here in k8s a volume is attached to Pod.



EmptyDir



It is used when one Pod is having multiple container.

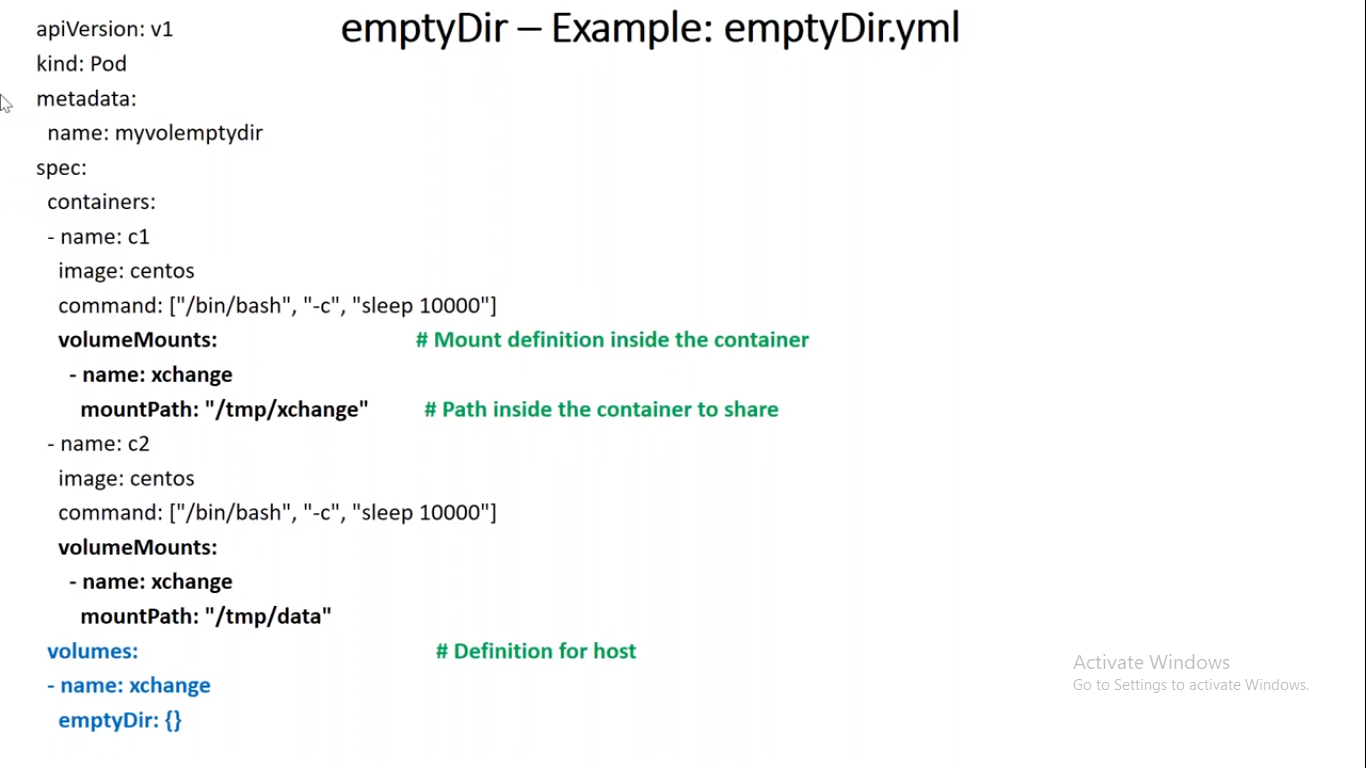
when you try to create a Pod and that time you will create a EmptyDir volume.

It will create Empty volume inside the pod.

Within each container you can access using different folders.  
When Pod is deleted then new Volume is created.

If any container goes down that time also Volume will be there and we can access it.

Example:-



Here we can see that we have 2 containers.

for both container we have Mount definition.

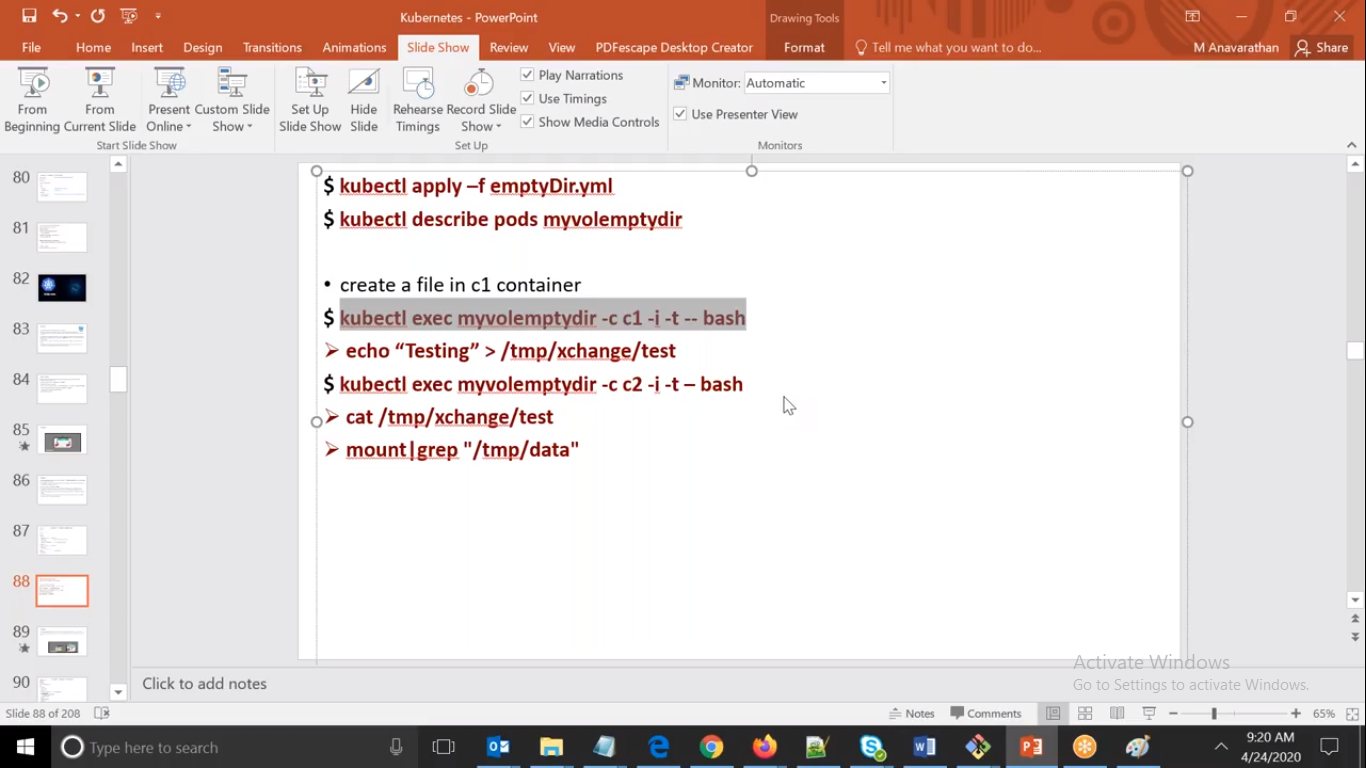
But Volumes is common for both the Containers.

So it is aligned to Containers.

And type is mentioned as emptyDir

Here in first container volume is mount in /tmp/xchange so whatever is available in second container in /tmp/data it will be accessed via volume to first container.

volume Mounts is for exchanging the volume.



**Connected to first container.**

ubuntu@ip-172-31-41-23:~$ kubectl exec myvolemptydir -c c1 -it -- bash

[root@myvolemptydir /]# cd /tmp/xchange/

**Connected to 2nd container**

ubuntu@ip-172-31-41-23:~$ kubectl exec myvolemptydir -c c2 -it – bash

[root@myvolemptydir /]# cd /tmp/data

[root@myvolemptydir data]# ls

**Created a file.**

[root@myvolemptydir data]# touch myemptyfile

**Again connected to ist container.**

ubuntu@ip-172-31-41-23:~$ kubectl exec myvolemptydir -c c1 -it – bash

[root@myvolemptydir /]# cd /tmp/xchange/

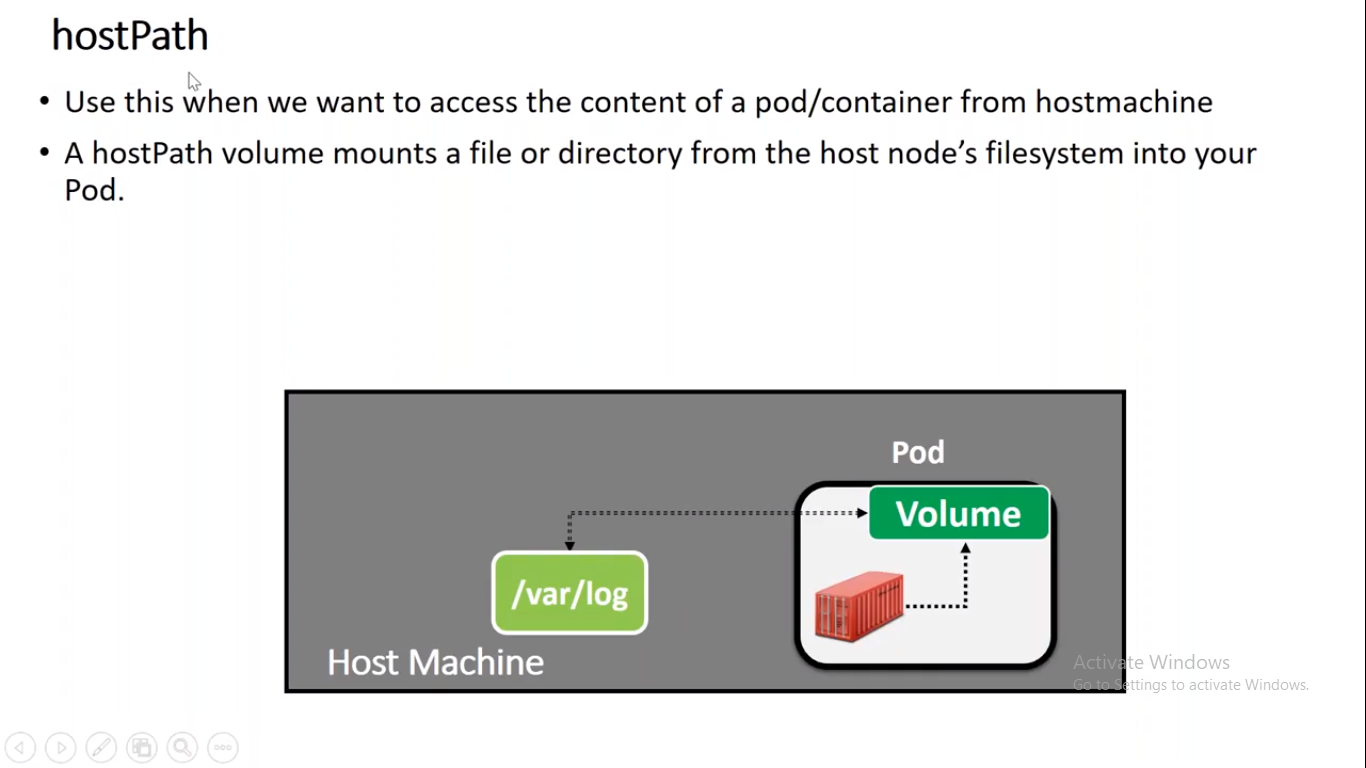
[root@myvolemptydir xchange]# ls

**We can see same file here also.**

myemptyfile

[root@myvolemptydir xchange]#

1. **HostPath**



Here we can access the volume from hostmachine.

