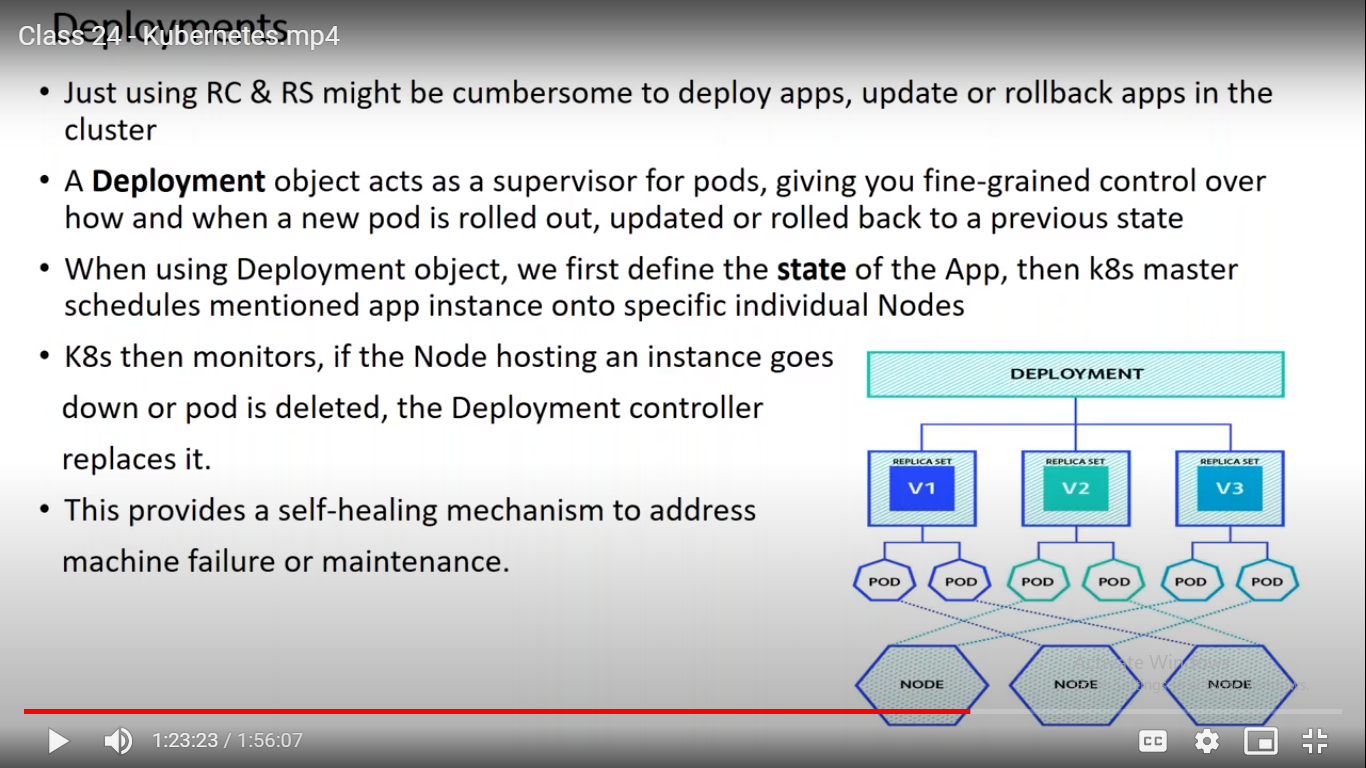
## Deployment:-



We saw 2 types of objects including pod.

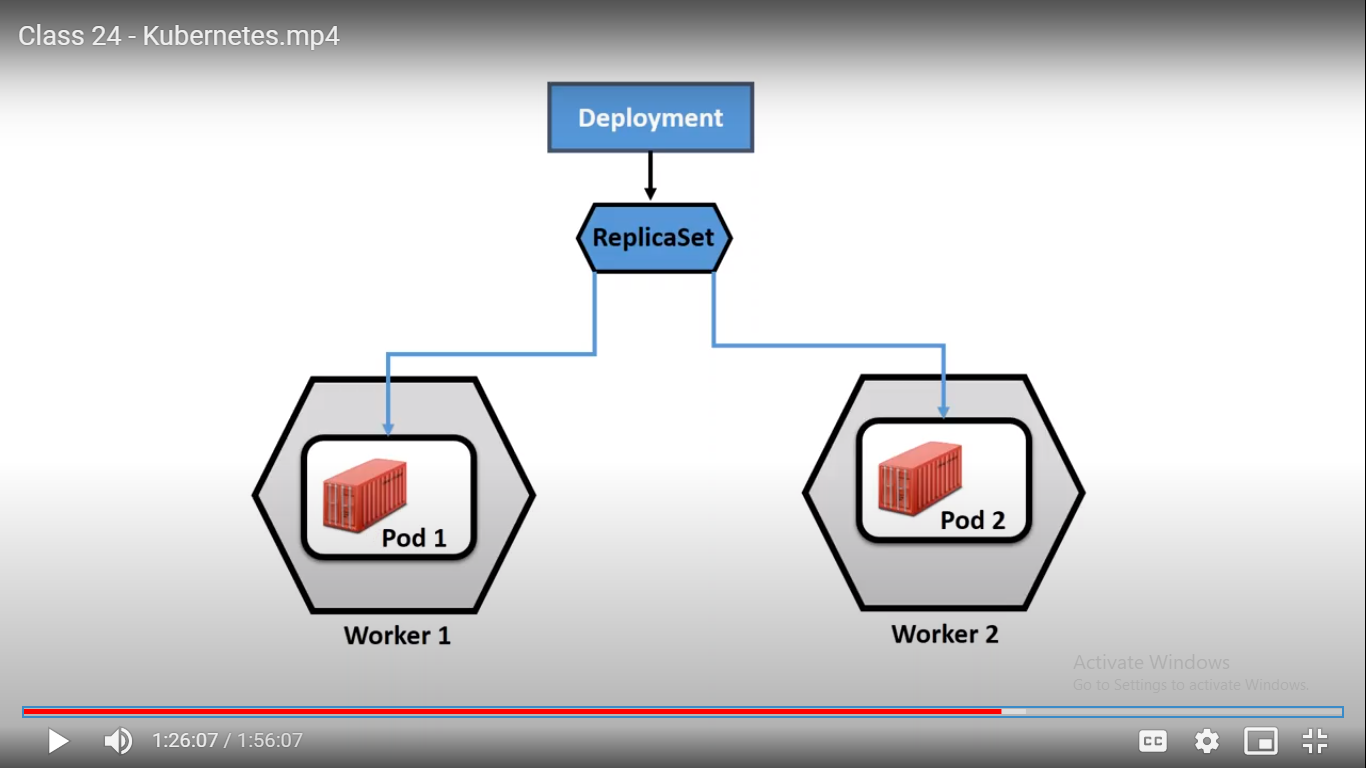
In that pod is default, but it does not ability to replicate itself.

when you are trying to deploy the application in production if due to certain reason new pod does not comes, so we should have ability or control to roll out from that.

So that in this case how do we go back and create the object from old set of image.

So we are having new object is called Deployment, which has options to have roll out back if any thing goes wrong.

Deployment will be in top of Replica set only.

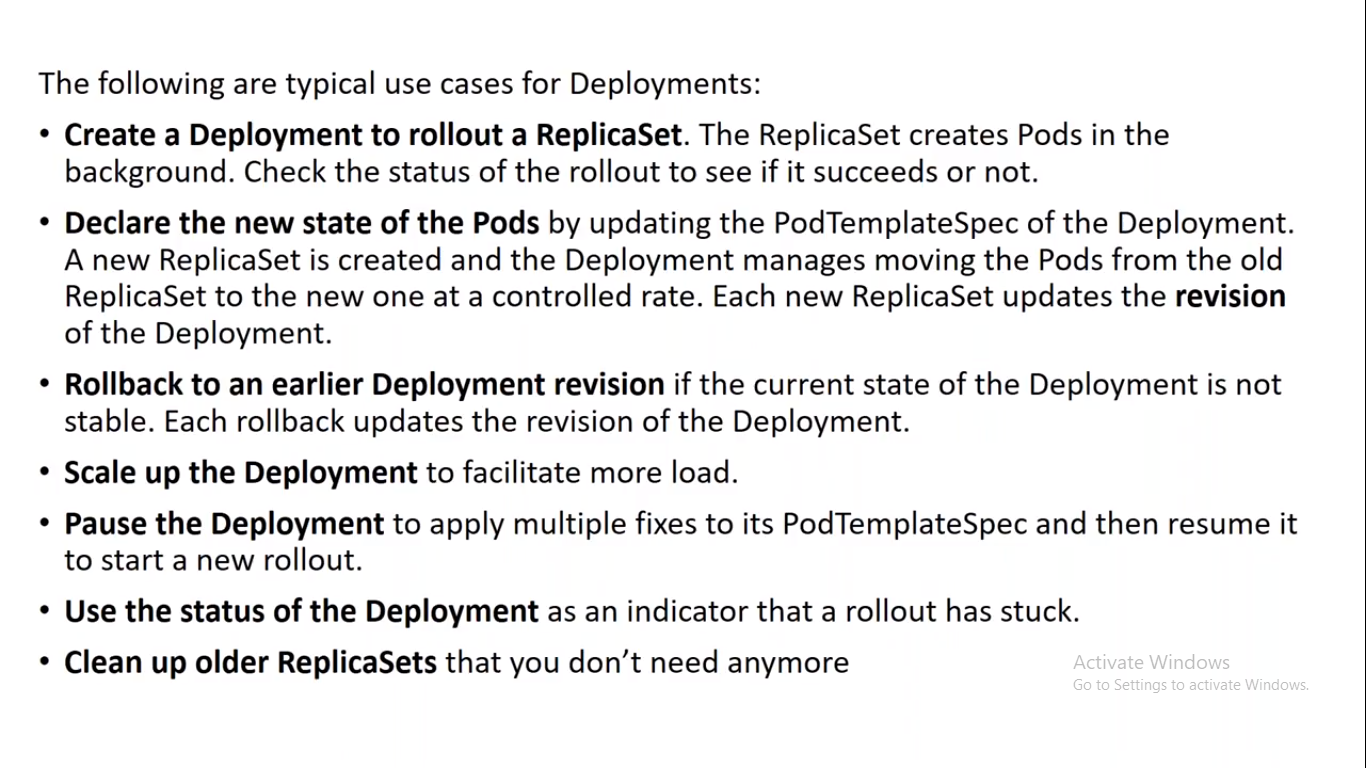


Deployment object will internally create Replica Set only, Replica set will be created the Pod.

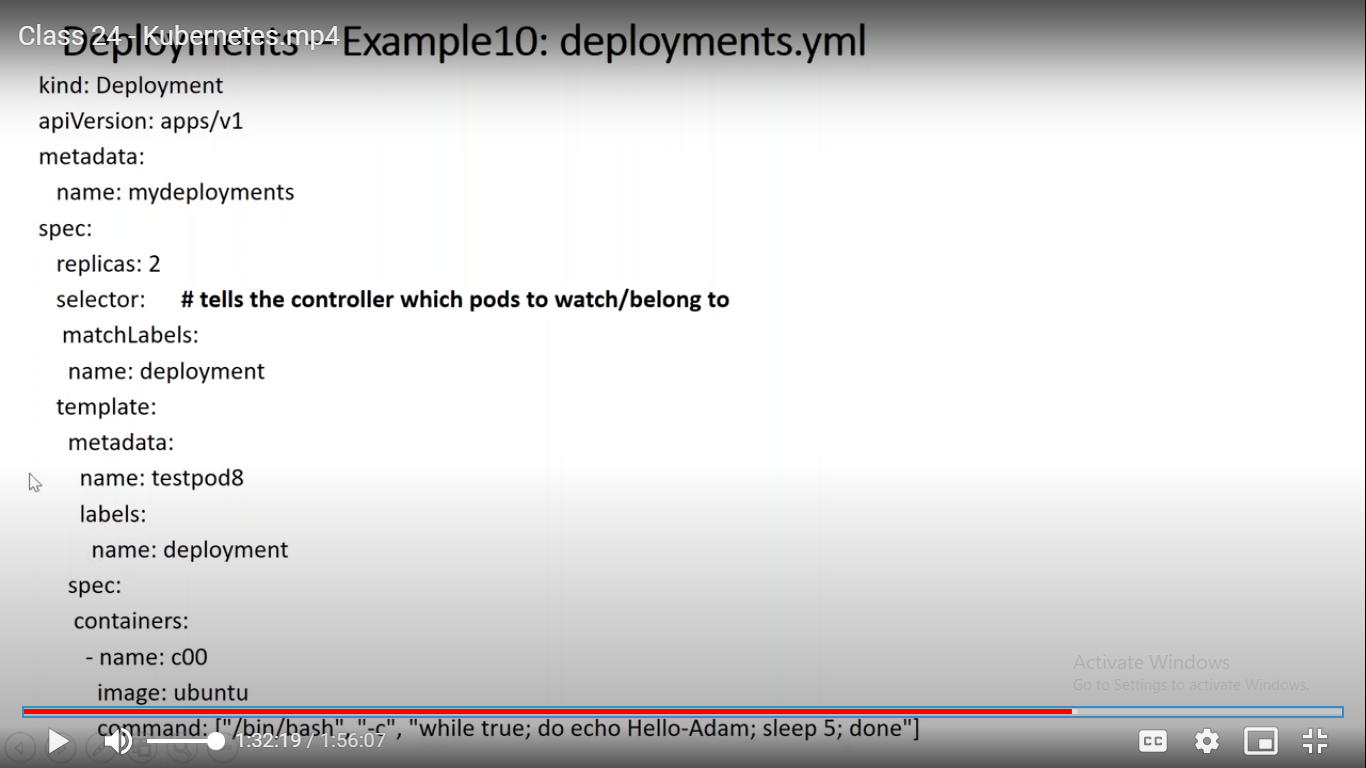
It will giving functionality of both Replication and Rolling Out and Rolling In.

If pod goes down then RS will manage it.

Uses Cases as below.



Example:-



Here we can see that Object is Deployment.

kind:Deployment

apiVersion :- apps/v1 i.e. similar to RS.

Here only one difference is that we are giving in selectors as matchLabels.

Example Analysis –

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

deployment.apps/mydeploy created

It created deployment objectd.

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

NAME READY STATUS RESTARTS AGE

mydeploy-9b6b9bd55-gsj54 1/1 Running 0 6s

mydeploy-9b6b9bd55-r2njm 1/1 Running 0 6s

Here we vcan see that pod is created from mydeploy only.

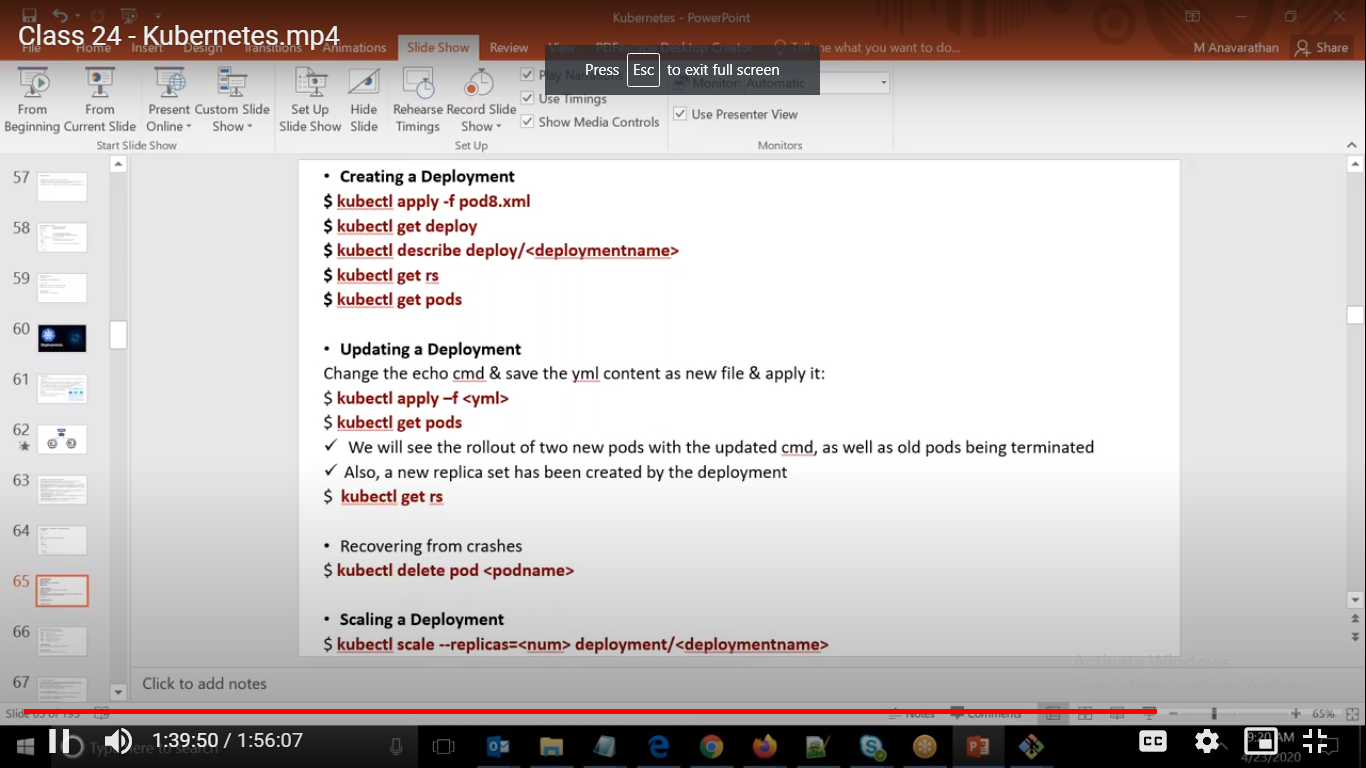
ubuntu@ip-172-31-41-23:~$ kubectl get rs

NAME DESIRED CURRENT READY AGE

mydeploy-9b6b9bd55 2 2 2 17s

Here we can see RS is created from my deploy only.

Commands:-



ubuntu@ip-172-31-41-23:~$ kubectl delete pod mydeploy-9b6b9bd55-gsj54

pod "mydeploy-9b6b9bd55-gsj54" deleted

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

NAME READY UP-TO-DATE AVAILABLE AGE

mydeploy 2/2 2 2 19m

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

NAME READY STATUS RESTARTS AGE

mydeploy-9b6b9bd55-gsj54 1/1 Terminating 0 19m

Here if we delete the pod then it will maintain the same state as it is maintained by deploy only.

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

NAME DESIRED CURRENT READY AGE

mydeploy-9b6b9bd55 2 2 2 31m

ubuntu@ip-172-31-41-23:~$ vi mydeploy.yml

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

deployment.apps/mydeploy configured

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

NAME DESIRED CURRENT READY AGE

mydeploy-7cbb7bdf57 1 1 0 4s

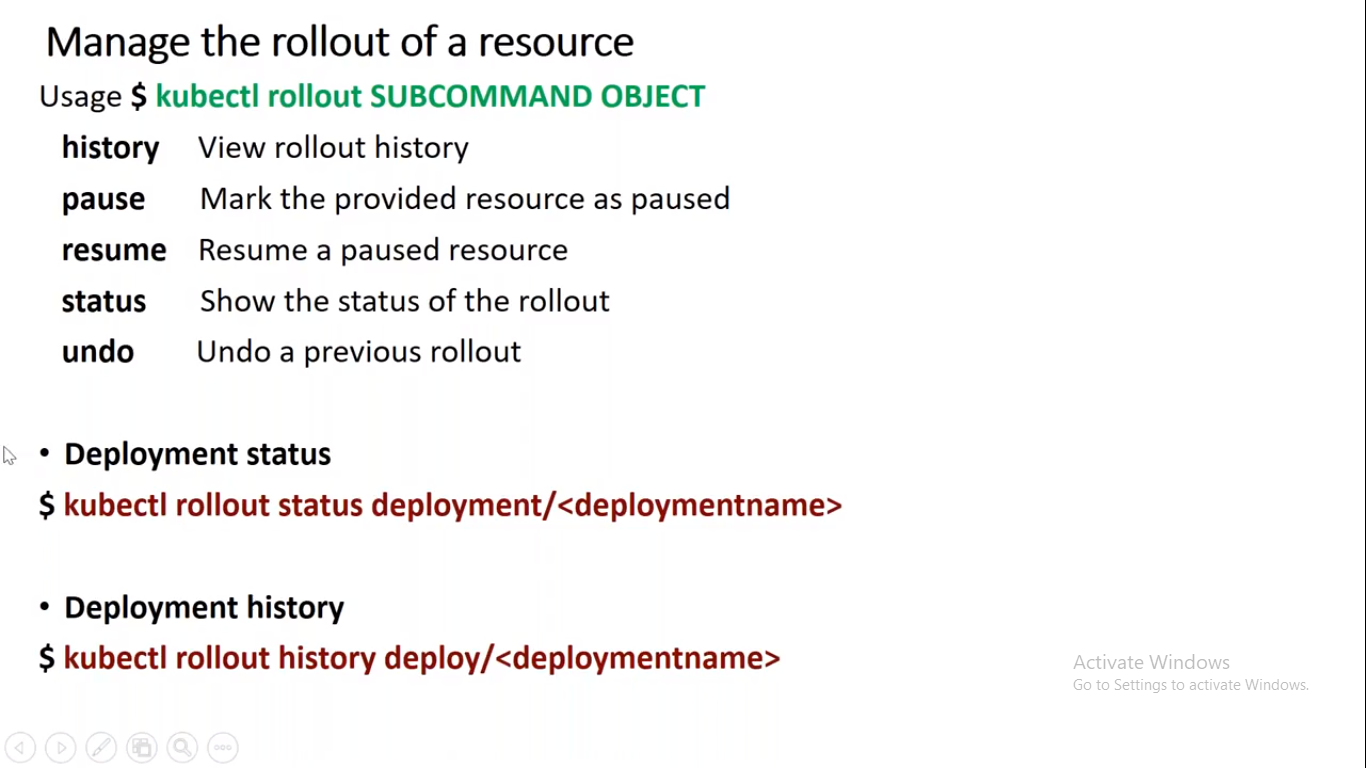
mydeploy-9b6b9bd55 1 1 1 32m

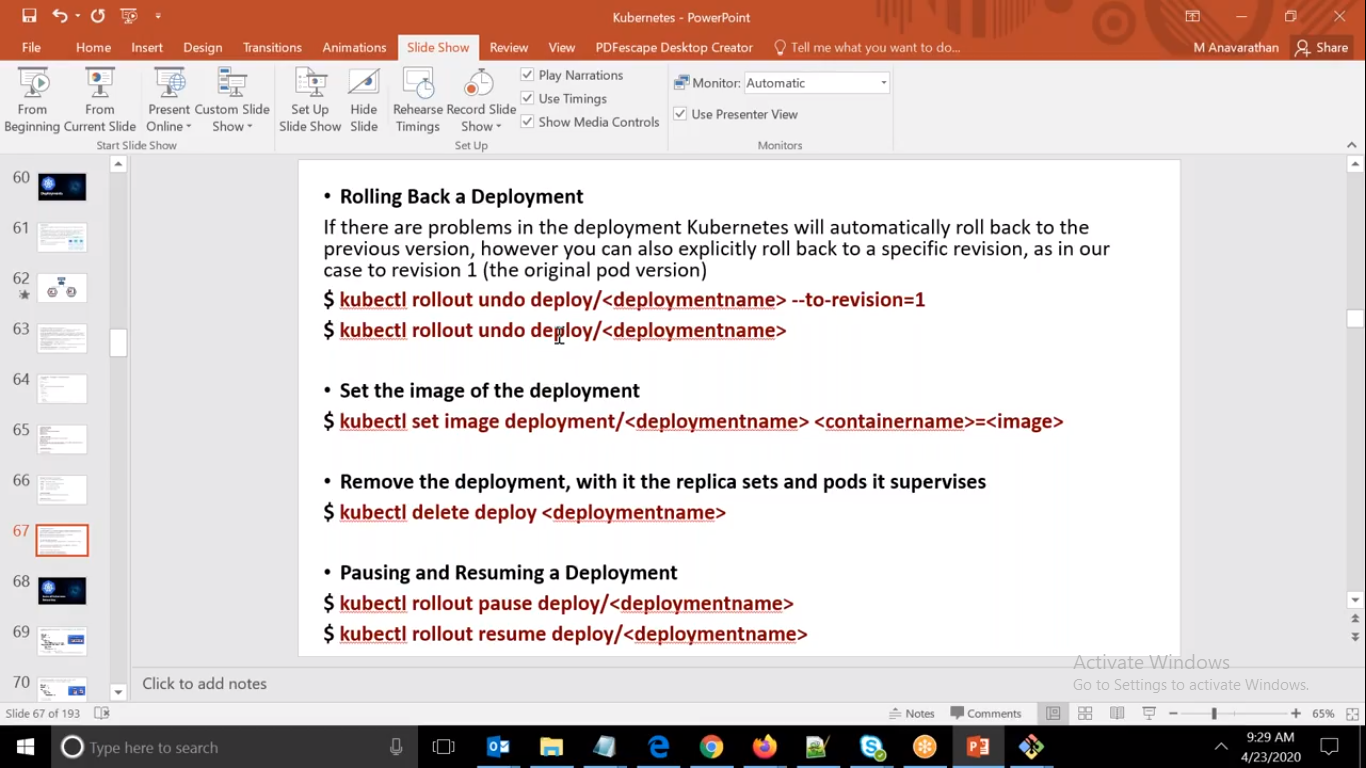
Here we can see that if we change the spec then only it will create new rs.

And if we make it to 1 then in rs

Here we have 2 RSet, everytime you change the configuration then it will create new rs so that it will be useful in tracking.

So by this we can roll back to previous one easily.





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

NAME DESIRED CURRENT READY AGE

mydeploy-7cbb7bdf57 0 0 0 14m

mydeploy-9b6b9bd55 1 1 1 47m

Here we can see that it roll back to last one..

We can go to back version as well.