

# Kubernetes

## About Kubernetes:

Kubernetes automates operational tasks of container management and includes built-in commands for deploying applications, rolling out changes to your applications, scaling your applications up and down to fit changing needs, monitoring your applications, and more—making it easier to manage applications.

## Azure Kubernetes:

Azure Kubernetes Service (AKS) **offers the quickest way to start developing and deploying cloud-native apps, with built-in code-to-cloud pipelines and guardrails.** Get unified management and governance for on-premises, edge, and multi cloud Kubernetes clusters.

## What is Kubernetes container orchestration?

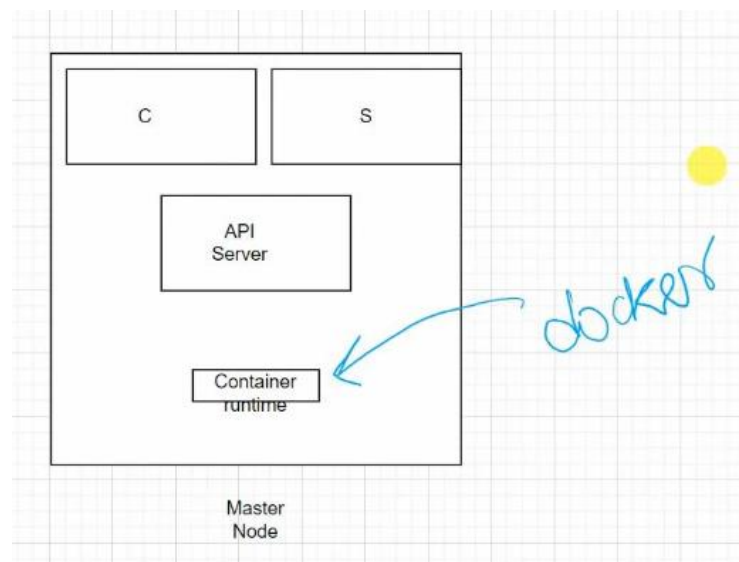
Kubernetes is **a popular open-source platform for container orchestration.** It enables developers to easily build containerized applications and services, as well as scale, schedule and monitor those containers.

## what is container orchestration?

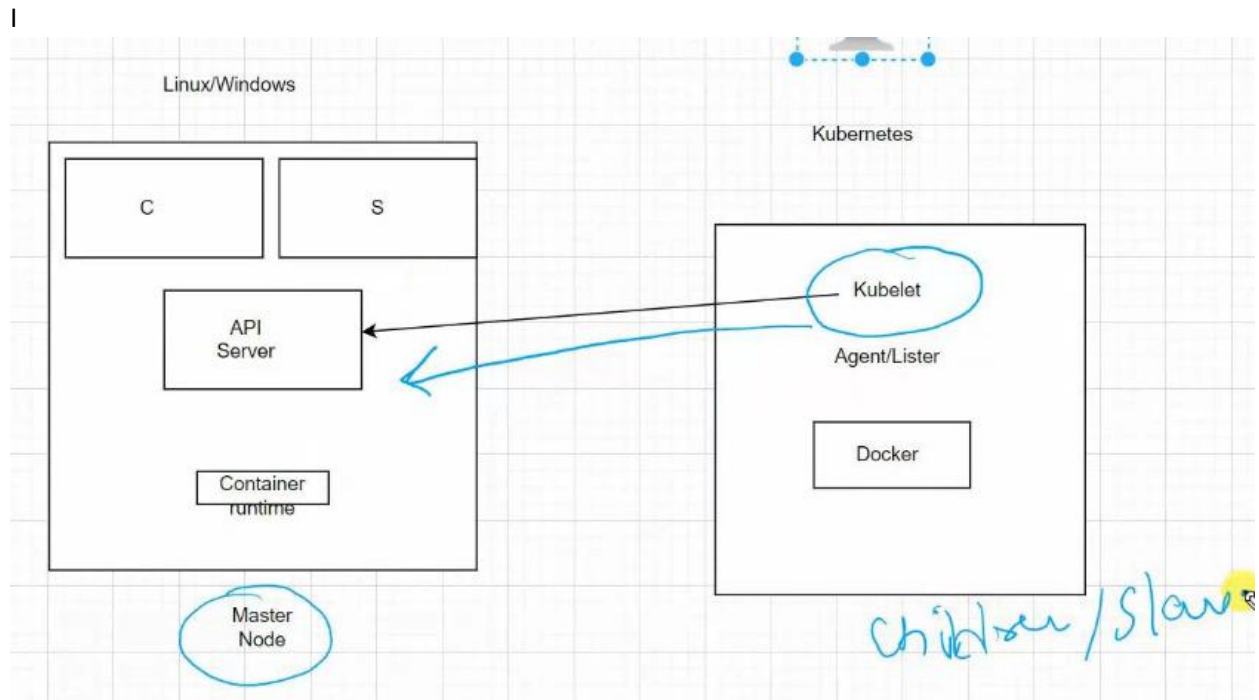
Container orchestration is **the automation of much of the operational effort required to run containerized workloads and services.** This includes a wide range of things software teams need to manage a container's lifecycle, including provisioning, deployment, scaling (up and down), networking, load balancing and more.

## Kubernetes:

- Node is similar to VM in Kubernetes terminology.



- API server is the heart of the Kubernetes.
- We can interact with API server using “Kube CTL” tool
- In above diagram C is nothing but controller
- In above diagram S is nothing but Scheduler
- We have to create Child Nodes on top of Master Nodes.



- As we seen in above diagram Master node API server will Interact with Kubelet on Child Node.
- Child Node is also called as Slave node
- We need to deploy POD on top of Child Nodes.
- Container will run in POD
- Containers are called “POD’s” in Kubernetes.
- We can run multiple PODs on same Node.

### Controller and Scheduler:

- Controller and scheduler are running on Master Node
- Controller will understand the problems in Nodes and PODs it will send the instruction to scheduler that why Node or POD is down.
- Scheduler will replicate the POD if POD is down, if Node is down it will create replica of the Node.
- Scheduler will ensure to up the PODs within in a fraction of minutes.
- This will also behave like load balancer, and it will also do auto scaling.
- Master Node completely handled by Microsoft, because it's a PaaS service.

### Creating Kubernetes Services:

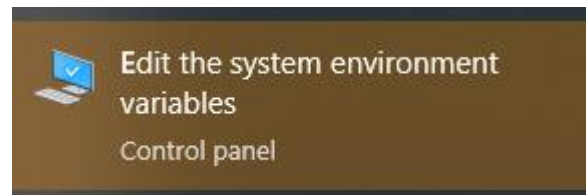
- Search as Kubernetes in market place.
- Create
- Config : dev/test
- Cluster name: democluster322
- Node size: B2ms
- Scale method : manual
- Node count: 1
- Click next to Node pool
- If we want we can create New Node Pool, but here we can go with default.
- Note: Node pool OS by default it will be Linux, if we want we can go to Windows also. But better to use linux, because Linux is light weight OS. It will be too faster comparatively.
- In Integration tab enable container monitoring.
- Review and create.
- Note: Before we start using the Kubernetes we need to install below on our local machine

1. Command line tool (cmd/powershell)
2. AZ CLI software
3. Kubernetes command line tool (kubectl)

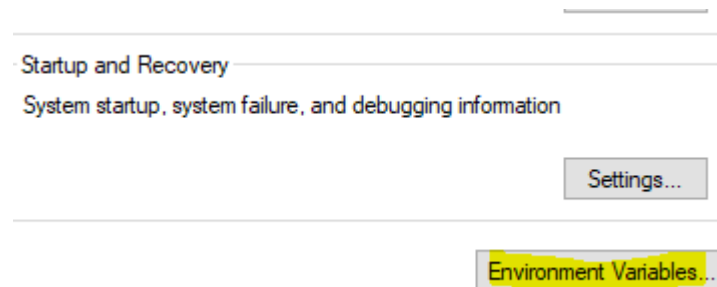
- CMD and powershell pre-installed on our Local machine
- To install AZ CLI open power shell and run “AZ”
- To install Kubernetes we need to run “az aks install-cli”, the command will install.
- Now copy below specified path and paste in file manager.

```
PS C:\Users\home> az aks install-cli
Downloading client to "C:\Users\home\azure-kubectl\kubectl.exe" from "https://storage.googleapis.com/kubernetes-release/release/v1.25.2/bin/windows/amd64/kubectl.exe"
Please add "C:\Users\home\azure-kubectl" to your search PATH so the `kubectl.exe` can be found. 2 options:
  1. Run "set PATH=%PATH%;C:\Users\home\azure-kubectl" or "$env:path += 'C:\Users\home\azure-kubectl'" for PowerShell. This is good for the current command session.
  2. Update system PATH environment variable by following "Control Panel->System->Advanced->Environment Variables", and re-open the command window. You only need to do it once
Downloading client to "C:\Users\home\AppData\Local\Temp\tmp52e0_xer\kubelogin.zip" from "https://github.com/Azure/kubelogin/releases/download/v0.0.20/kubelogin.zip"
Please add "C:\Users\home\azure-kubelogin" to your search PATH so the `kubelogin.exe` can be found. 2 options:
  1. Run "set PATH=%PATH%;C:\Users\home\azure-kubelogin" or "$env:path += 'C:\Users\home\azure-kubelogin'" for PowerShell. This is good for the current command session.
  2. Update system PATH environment variable by following "Control Panel->System->Advanced->Environment Variables", and re-open the command window. You only need to do it once
PS C:\Users\home>
```

- Open environment variables as below



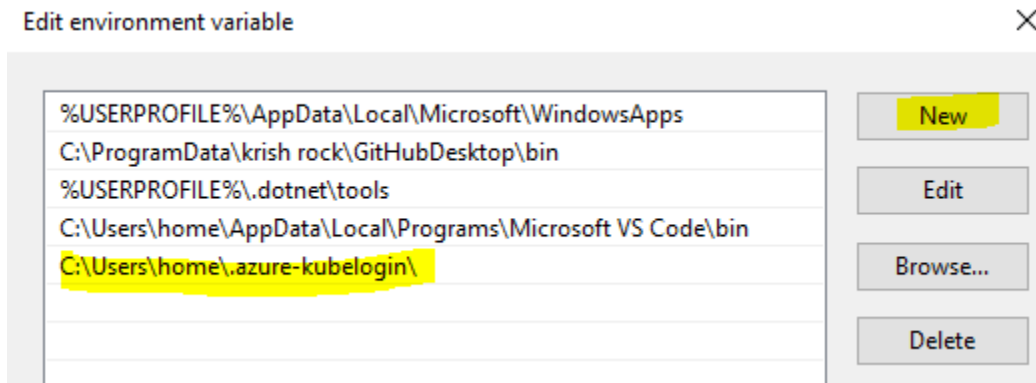
- Click on environment variables



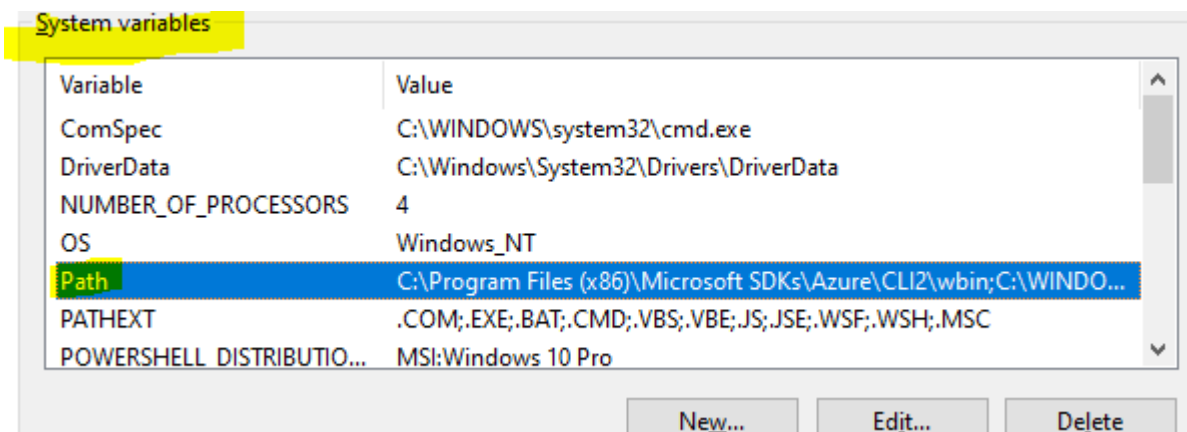
- Now double click on “Path”

User variables for krish rock	
Variable	Value
OneDrive	C:\Users\home\OneDrive
Path	C:\Users\home\AppData\Local\Microsoft\WindowsApps;C:\Progra
TEMP	C:\Users\home\AppData\Local\Temp
TMP	C:\Users\home\AppData\Local\Temp

- Now click on new and Paste the path which we copied above  
“C:\Users\home\azure-kubelogin\”



- Click on OK
- Now double click on Path under system variables
- Add the new path like above.



- Now click on and close this environment variables and powershell
- Now re open the Powershell again and run below command
- To check whether the kubelet cli installed properly or not by running below command
  - CMD: kubectl
- The above command will show kubectl commands
- Now to connect our cluster open cluster in portal
- Click on connect and copy the credentials from below snap shot.

Connect to your cluster using command line tooling to interact directly with cluster using kubectl, the command line tool for Kubernetes. Kubectl is available within the Azure Cloud Shell by default and can also be installed locally. [Learn more](#)

1. Open [Cloud Shell](#) or the Azure CLI
2. Run the following commands

```
az account set --subscription b36294ff-b022-4ff3-9fcd-39716852e936
```

```
az aks get-credentials --resource-group kuber-rg --name democluster322
```

Sample commands

- Now paste the above command in powershell and run.
  - CMD: `az aks get-credentials --resource-group kuber-rg --name democluster322`
- The above command will login to kluster and we can use it
- Now to check PODs in kluster run below command
  - CMD: `kubectl get pods`
- To check nods run below commands
  - CMD: `kubectl get nodes`

## Creating POD:

- Now run below command to create POD
  - CMD: `kubectl run myfirstpod --image nginx`

```
PS C:\Users\home> kubectl run myfirstpod --image nginx
pod/myfirstpod created
PS C:\Users\home>
```

- As shown in the above picture we have created POD.
- If we want to run latest image run below command
  - CMD: `kubectl run myfirstpod --image nginx:latest`

```
PS C:\Users\home> kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
myfirstpod    1/1     Running   0           3m48s
PS C:\Users\home>
```

- To chek our pod details run below command
  - CMD: `kubectl describe pods myfirstpod`
- Like below it will show so many details, I have taken snapshot for few details.

```

PS C:\Users\home> kubectl describe pods myfirstpod
Name:          myfirstpod
Namespace:     default
Priority:       0
Service Account: default
Node:          aks-agentpool-41022184-vmss000000/10.244.0.4
Start Time:    Fri, 30 Sep 2022 18:08:51 +0530
Labels:        run=myfirstpod
Annotations:    <none>
Status:        Running
IP:            10.244.0.13
IPs:
  IP: 10.244.0.13
Containers:

```

- Now we can install one more POD for “httpd” image

- CMD: kubectl run httpdpod --image httpd

```

PS C:\Users\home> kubectl run httpdpod --image httpd
pod/httpdpod created
PS C:\Users\home>

```

- We can check the pods below commands

- CMD: kubectl get pods

```

PS C:\Users\home> kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
httpdpod      1/1     Running   0           114s
myfirstpod    1/1     Running   0           19m
PS C:\Users\home>

```

- We can also check our pods in portal

- Go to kluster
  - Click on Workloads on blade
  - Filter by default, the we can see our pods

Filter by pod name

Filter by label selector ⓘ

Status

Filter by namespace

Enter the full pod name

foo=bar,key!=value

All statuses ▾

default

<input type="checkbox"/>	Name	Namespace	Ready	Status	Restart count	Age ↓
<input type="checkbox"/>	myfirstpod	default	✔ 1/1	Running	0	22 minutes
<input type="checkbox"/>	httpdpod	default	✔ 1/1	Running	0	4 minutes

- Now run below commands to get few major details of the PODs

- CMD: kubectl get pod -o wide

```

myfirstpod 1/1 Running 0 19m
PS C:\Users\home> kubectl get pod -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP            NODE                                     NOMINATED NODE   READINESS GATES
httpdpod      1/1     Running   0           6m46s  10.244.0.14   aks-agentpool-41022184-vmss000000    <none>           <none>
myfirstpod    1/1     Running   0           24m    10.244.0.13   aks-agentpool-41022184-vmss000000    <none>           <none>
PS C:\Users\home>

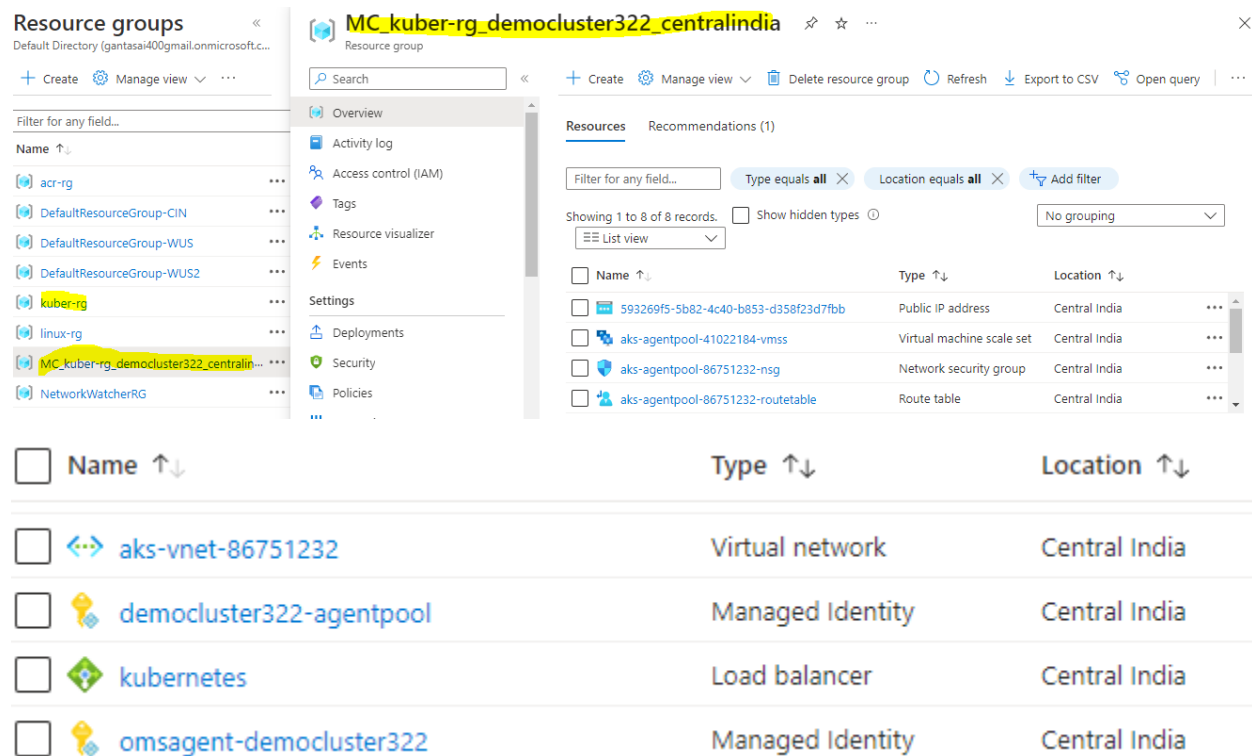
```

- To remove any pod run below command
  - CMD: `kubectl delete pod httpdpod`
- Now check pods
  - CMD: `kubectl get pod`

```
PS C:\Users\home> kubectl get pod -o wide
NAME          READY   STATUS    RESTARTS   AGE
httpdpod      1/1     Running   0           6m46s
myfirstpod    1/1     Running   0           24m
PS C:\Users\home> kubectl delete pod httpdpod
pod "httpdpod" deleted
PS C:\Users\home> kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
myfirstpod    1/1     Running   0           28m
PS C:\Users\home>
```

- We can see in the above picture httpdpod is deleted and only one pod is available

Note: Whenever we are creating kubernetes cluster it will also create other resource group in that it will create few resources like VNet, IP, Virtual machine scale set, kubernetes loadbalancer and more.



- As shown in above screenshot we have created only kuber-rg and in that kubernetes cluster, but microsoft will deploy other resource group as shown as MC\_kuber-rg. In that it will deploy above shown resources.



Now we need to expose our POD to internet.

- Now check pods
  - CMD: kubectl get pod
- Below command will expose the POD
  - CMD: kubectl expose pod myfirstpod --type=LoadBalancer --port=80 --name=nginxlb

```
PS C:\Users\home> kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
myfirstpod    1/1     Running   0           93m
PS C:\Users\home> kubectl expose pod myfirstpod --type=LoadBalancer --port=80 --name=nginxlb
service/nginxlb exposed
PS C:\Users\home>
```

- The above command has exposed the POD
- Now to check our external IP run below command
  - CMD: kubectl get svc

```
service/nginxlb exposed
PS C:\Users\home> kubectl get svc
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes    ClusterIP     10.0.0.1     <none>        443/TCP          5h1m
nginxlb       LoadBalancer  10.0.145.199 20.219.235.26 80:32169/TCP     9m43s
PS C:\Users\home>
```

- Now go to browser and search with external IP we can reach out to nginx page.
- Now we can run “httpd” image using below command.
  - CMD: kubectl run httpdpod --image httpd
- Now we can expose the POD using below command.
  - CMD: kubectl expose pod httpdpod --type=LoadBalancer --port=80 --name=httpdlb
- Now run below command to see external IP
  - CMD: kubectl get svc
- Then it will show the the external IP
- Search that IP in browser we can see httpd page.

Now we create Kubernetes service using below Yaml file.

```
! basic-skeleton.yml  ! service.yml x  ! p
! service.yml > {} spec > {} selector > app
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-loadbalancerservice
5  spec:
6    type: LoadBalancer
7    ports:
8      - port: 80
9    selector:
10   app: mynginxapp
```

- Now go to power shell and run below command
  - CMD: `kubectl apply -f .\service.yml`

```
PS C:\Users\home> cd "C:\Krish\Devops\AKS\Crating PODS using YAML"
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\service.yml
service/my-loadbalancerservice created
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- Now run below command to check the IP of our just installed loadbalancer
  - CMD: `kubectl get svc`

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get svc
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes                         ClusterIP           10.0.0.1        <none>           443/TCP          4d
my-loadbalancerservice             LoadBalancer        10.0.15.14      20.219.237.168   80:30182/TCP     6m37s
nginxlb                           LoadBalancer        10.0.202.251    20.244.72.35     80:32460/TCP     2d22h
```

- Now we can see without IP address without running “expose” command.
- We can see the same in portal also.
- Go to cluster
- Services and ingresses
- Then we can see the loadbalancer service

democluster322 | Services and ingresses ...

Kubernetes service

Search

« + Create ▾ Delete Refresh Show labels Give feedback

Kubernetes resources

- Namespaces
- Workloads
- Services and ingresses**
- Storage
- Configuration

Settings

- Node pools
- Cluster configuration
- Networking
- Open Service Mesh

Services Ingresses

Filter by service name Enter the full service name

Filter by namespace All namespaces ▾

<input type="checkbox"/>	Name	Namespace	Status	Type	Cluster IP	External IP
<input type="checkbox"/>	kubernetes	default	✓ Ok	ClusterIP	10.0.0.1	
<input type="checkbox"/>	kube-dns	kube-system	✓ Ok	ClusterIP	10.0.0.10	
<input type="checkbox"/>	metrics-server	kube-system	✓ Ok	ClusterIP	10.0.170.67	
<input type="checkbox"/>	nginxlb	default	✓ Ok	LoadBalancer	10.0.202.251	20.244.72.35
<input type="checkbox"/>	my-loadbalancerservice	default	✓ Ok	LoadBalancer	10.0.15.14	20.219.237.168

- Now if you want to reach to nginx we need to go through the LoadBalancer. If POD goes down we can't be able to reach the webpage. So we need to install multiple pods on same Node. So if one POD goes down then LoadBalancer will send to other POD.
- Now we need to delete the existing POD and LoadBalancer.
  - Go to Cluster
  - Click on services and Ingresses
  - Now remove POD and Loadbalancer

democluster322 | Services and ingresses ...

Kubernetes service

Search

« + Create ▾ Delete Refresh Show labels Give feedback

Kubernetes resources

- Namespaces
- Workloads
- Services and ingresses**
- Storage
- Configuration

Settings

- Node pools
- Cluster configuration
- Networking
- Open Service Mesh

Services Ingresses

Filter by service name Enter the full service name

Filter by namespace All namespaces ▾

<input type="checkbox"/>	Name	Namespace	Status	Type	Cluster IP	External IP
<input type="checkbox"/>	kubernetes	default	✓ Ok	ClusterIP	10.0.0.1	
<input type="checkbox"/>	kube-dns	kube-system	✓ Ok	ClusterIP	10.0.0.10	
<input type="checkbox"/>	metrics-server	kube-system	✓ Ok	ClusterIP	10.0.170.67	
<input checked="" type="checkbox"/>	nginxlb	default	✓ Ok	LoadBalancer	10.0.202.251	20.244.72.35
<input checked="" type="checkbox"/>	my-loadbalancerservice	default	✓ Ok	LoadBalancer	10.0.15.14	20.219.237.168

- Now check PODs using below commands, there won't be any POD.
  - CMD: kubectl get pods
- If any pod not deleting run below command to remove
  - CMD: kubectl delete pod myfirst-pod

- Now we can create multiple PODS using YAML script
- Now we can use earlier written POD.YML, service.yml and we need to create new script pod2.yml as below

```

• apiVersion: v1
• kind: Pod
• metadata:
•   name: second-pod
•   labels:
•     app: mynginxapp
•     type: front-end
• spec:
•   containers:
•     - name: nginxcontainer
•       image: nginx:latest

```

- Now save the script as pod2.yml

```

! pod.yml > apiVersion
io.k8s.api.core.v1.Pod (v1@pod.json)
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: myfirst-pod
5    labels:
6      app: mynginxapp
7      type: front-end
8  spec:
9    containers:
10     - name: nginxcontainer
11     image: nginx:latest

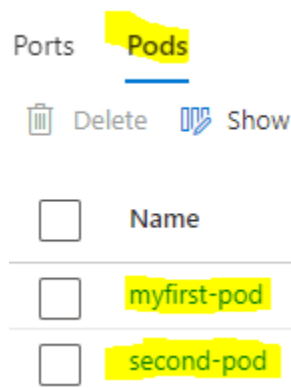
```

- Now run below commands one by one.
- The below command will install the my-firstpod
  - CMD: `kubectl apply -f .\pod.yml`
- The below command will install the second-pod
  - CMD: `kubectl apply -f .\pod2.yml`
- The below command will install LoadBalancer and will attach to both pods as we specified in service file "mynginx"
  - CMD: `kubectl apply -f .\service.yml`

- Now we can see in below snap shot that how they are deployed.

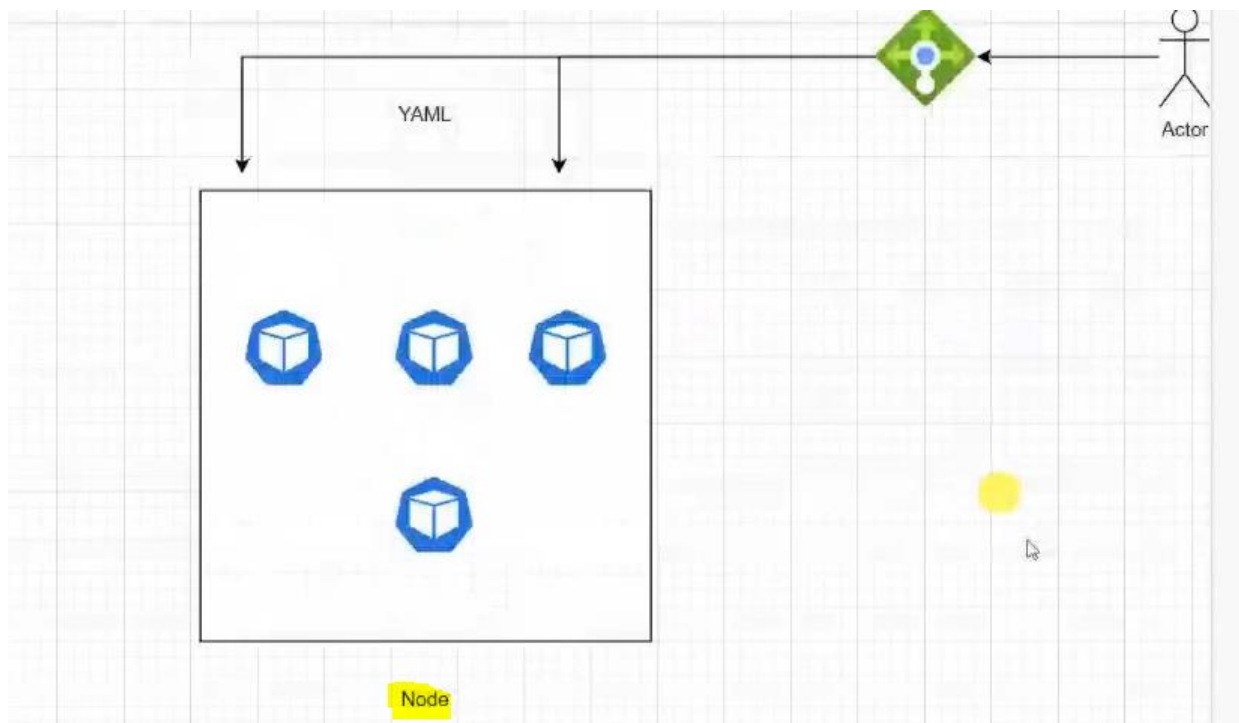
```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get pods
No resources found in default namespace.
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\pod.yml
pod/myfirst-pod created
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\pod2.yml
pod/second-pod created
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\service.yml
service/my-loadbalancerservice created
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get svc
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes                          ClusterIP           10.0.0.1        <none>           443/TCP          4d3h
my-loadbalancerservice              LoadBalancer       10.0.135.92     20.219.239.198   80:31276/TCP     22s
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
myfirst-pod       1/1     Running   0           72s
second-pod        1/1     Running   0           60s
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- Now we can see that loadbalancer is handling 2 pods in portatl
- Go to cluser
- Click on services and ingresses
- Select loadbalancer
- Now click on pods
- Then it will show the both pods which it is handling



# Replication Set

- Earlier we have deployed the PODs on Node directly.
- For high availability we have created multiple PODs and attached to loadbalancer. But while deploying PODs it will be so manual using earlier method. If we need 1 or 2 pods then we can create the yaml script for each pod. But if we need many the manual approach is not good. So we need to use the Replication set approach
- We will deploy the replication set on the Node.
- Under replication set we can deploy many pods.



- Before we deploying the replication set, we ensure to delete POD and loadbalancer which we have earlier deployed.
- Now create the new file as “replicaset.yml” in Visual studio code.
- Write below script on replicaset file

```
• apiVersion: apps/v1
• kind: ReplicaSet
• metadata:
•   name: first-rs
•   labels:
•     type: front-end-rs
• spec:
```

- `template:`
- `metadata:`
- `name: myfirst-pod`
- `labels:`
- `app: mynginxapp`
- `type: front-end`
- `spec:`
- `containers:`
- `- name: nginxcontainer`
- `image: nginx:latest`
- `selector:`
- `matchLabels:`
- `app: mynginxapp`
- `replicas: 3`

- Now go to PowerShell and run below command
  - CMD: `kubectl apply -f .\replicaset.yml`

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\replicaset.yml
replicaset.apps/first-rs created
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- As per above we can see our replica set has created. To check our replica set run below command
  - **CMD: `kubectl get rs`**
- As per YAML script we have given 3 replicas, and it has deployed the same as below

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get rs
NAME          DESIRED  CURRENT  READY  AGE
first-rs      3        3        3      2m28s
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get pods
NAME                READY  STATUS   RESTARTS  AGE
first-rs-91s17      1/1    Running  0          5m25s
first-rs-9rkd2      1/1    Running  0          5m25s
first-rs-tmsk2      1/1    Running  0          5m25s
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- We can see the same in portal, this pods under replicaset.
- Now go to cluster in portal
- Click on workloads on blade
- Click on replicaset on top
- Filter by default

- Now click on our replicaset

**democluster322** | Workloads ...  
Kubernetes service

Search << + Create ▾ Delete Refresh Show labels Give feedback

Overview  
Activity log  
Access control (IAM)  
Tags  
Diagnose and solve problems  
Microsoft Defender for Cloud

Kubernetes resources

Namespaces  
Workloads

Deployments Pods **Replica sets** Stateful sets Daemon sets Jobs Cron jobs

Filter by replica set name  
Enter the full replica set name

Filter by label selector ⓘ  
foo=bar,key!=value

Filter by namespace  
default

<input checked="" type="checkbox"/>	Name	Namespace	Ready	Current
<input checked="" type="checkbox"/>	first-rs	default	✓ 3/3	3

- Now it will show the pods.

Namespace

default

Labels

type : front-end-rs ⓘ

Selector

app=mynginxapp

[See more](#)

## Pods



Delete Show labels

<input type="checkbox"/>	Name	Ready	Status
<input type="checkbox"/>	first-rs-9ls17	✓ 1/1	Running
<input type="checkbox"/>	first-rs-9rkd2	✓ 1/1	Running
<input type="checkbox"/>	first-rs-tmsk2	✓ 1/1	Running



- If our pod goes down or deleted it automatically create one more pod. In below we tried to delete the one pod and it is terminating one and creating one as below.

Pods

 Delete  Show labels

<input type="checkbox"/>	Name	Ready	Status
<input checked="" type="checkbox"/>	first-rs-9lsl7	✓ 1/1	Terminating
<input type="checkbox"/>	first-rs-9rkd2	✓ 1/1	Running
<input type="checkbox"/>	first-rs-tmsk2	✓ 1/1	Running
<input type="checkbox"/>	first-rs-zm5m7	⚠ 0/1	ContainerCreating

- If you delete 2 or 3 then also it automatically creates same. If 3 goes down it will create other 3.
- For example if you want to deploy 6 containers on same node using existing template we need to change the replicas 3 to 6. And follow below.

```

• apiVersion: apps/v1
• kind: ReplicaSet
• metadata:
•   name: first-rs
•   labels:
•     type: front-end-rs
• spec:
•   template:
•     metadata:
•       name: myfirst-pod
•       labels:
•         app: mynginxapp
•         type: front-end
•     spec:
•       containers:
•         - name: nginxcontainer
•           image: nginx:latest
• selector:
•   matchLabels:
•     app: mynginxapp
• replicas: 6

```

- Now we need to run below command to execute the above script.
  - CMD: `kubectl replace -f .\replicaset.yml`
- Note: earlier we have used apply but now we have used replace because same script we have used and those PODs are running so we need to replace them.

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl replace -f .\replicaset.yml
replicaset.apps/first-rs replaced
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- To check the pods run below command

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
first-rs-9rkd2      1/1     Running   0           6h15m
first-rs-cfvzh      1/1     Running   0           3m34s
first-rs-hnsw9      1/1     Running   0           3m34s
first-rs-tmsk2      1/1     Running   0           6h15m
first-rs-x62ft      1/1     Running   0           3m34s
first-rs-zm5m7      1/1     Running   0           6h2m
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- Above are the 6 pods which we have deployed on the above replace command.

## Orchestration:

If you delete pods or if pods will go down the kubernetes controller will deploy the other PODs within seconds. This is called as orchestration and replication controller.

- Now we can attach Loadbalancer using our "service.yml" file
- The script as below.

```
• apiVersion: v1
• kind: Service
• metadata:
•   name: my-loadbalancerservice
• spec:
•   type: LoadBalancer
•   ports:
•     - port: 80
•   selector:
•     app: mynginxapp
```

- The LoadBalancer will attach to the PODs because we are giving selector as "mynginxapp"

```

PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\service.yml
service/my-loadbalancerservice created
PS C:\Krish\Devops\AKS\Crating PODS using YAML>

```

- Below are steps in high level how we did the process.

Node : Virtual machine Scale Set: Master Node and a Slave  
 Pod: Smallest unit: Container  
 ReplicSet: Controller : 3 POD's - YAML  
 Service: Loadbalancer

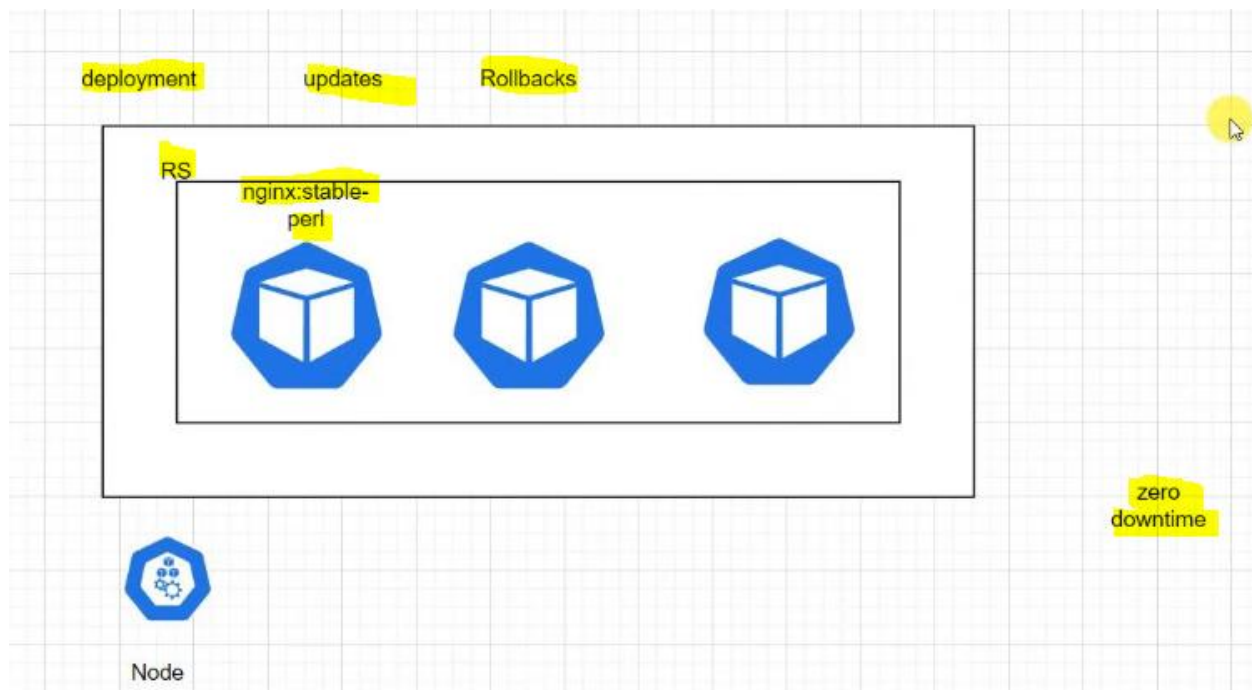
Commands:

```
kubectl get pods
      svc
```

```
kubectl run podname --image=
```

```
kubectl apply -f name.yaml
```

## Deployment



- Deployment having advantages over replicaset. In above diagram "RS" is nothing but Replicaset.

- When we do deployment it will also install Replicaset as shown in above picture.
- If we are using deployment if we want to update the container image, kubernetes will do in incremental manner. It will update the 1 POD then second POD like that it will be update.
- If you don't want to use the update and you want to go back to earlier version we can roll back to previous version same in incremental manner.
- Kubernetes always zero down time

Note: Before going to install deployment we need to ensure to delete the PODS and services and Replicaset.

- CMD: `kubectl get pods`

```
PS C:\Users\home> kubectl get pods
No resources found in default namespace.
PS C:\Users\home>
```

- Now write below script and save as "deployment.yml"
- We can use replicaset and need to do few changes, major change in kind has to change Replicaset to deployment.
- The code as below

```
• apiVersion: apps/v1
• kind: Deployment
• metadata:
•   name: first-deployment
•   labels:
•     type: front-end
• spec:
•   template:
•     metadata:
•       name: nginxpod
•       labels:
•         app: mynginxapp
•         type: front-end
•     spec:
•       containers:
•         - name: nginxcontainer
•           image: nginx:stable-alpine-perl
• selector:
•   matchLabels:
•     type: front-end
• replicas: 3
```

- Now run below command
  - CMD: `kubectl apply -f .\deployment.yml`

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\deployment.yml
deployment.apps/first-deployment created
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- It has created deployment in that it has created Replicaset and pods as shown in below.

```
PS C:\Krish\Devops> kubectl get rs
NAME                                DESIRED    CURRENT    READY    AGE
first-deployment-5d57c4f49f         3          3          3        2m58s
PS C:\Krish\Devops> kubectl get pods
NAME                                READY      STATUS    RESTARTS   AGE
first-deployment-5d57c4f49f-b7dgl  1/1       Running   0           3m8s
first-deployment-5d57c4f49f-qwncj  1/1       Running   0           3m8s
first-deployment-5d57c4f49f-z6tpc  1/1       Running   0           3m8s
PS C:\Krish\Devops>
```

- As shown above deployment script has created the one Replicaset and 3 pods.
- Earlier we have created nginx stable perl version now we want to update to latest. So we need to do small change in yaml script at image

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: first-deployment
  labels:
    type: front-end
spec:
  template:
    metadata:
      name: nginxpod
      labels:
        app: mynginxapp
        type: front-end
    spec:
      containers:
      - name: nginxcontainer
        image: nginx:latest
      selector:
        matchLabels:
          type: front-end
      replicas: 3
```

- Now run below command.

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\deployment.yml --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/first-deployment configured
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- It has created new replicaset as shown in below.

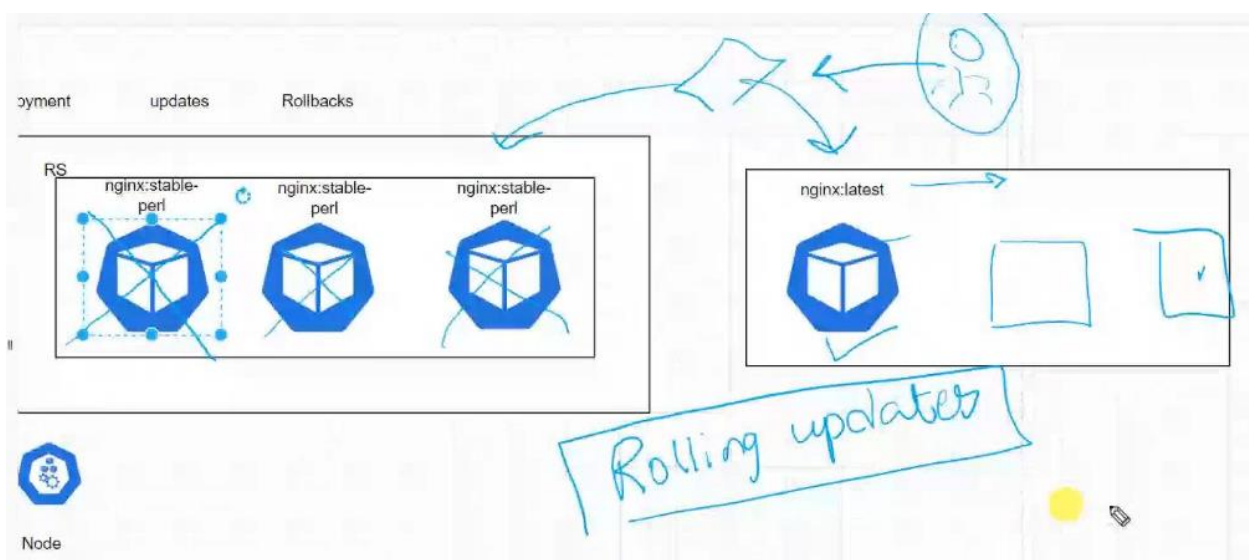
```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get rs
NAME                                DESIRED    CURRENT    READY    AGE
first-deployment-5d57c4f49f         0          0          0        32m
first-deployment-7cc4855cdf         3          3          3        2m40s
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- As shown in above picture PODs removed from earlier Replicaset and created new replicaset inside that new 3 pods created.

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get pod
NAME                                READY    STATUS    RESTARTS    AGE
first-deployment-7cc4855cdf-68xfg  1/1     Running   0           4m34s
first-deployment-7cc4855cdf-g7tmh  1/1     Running   0           4m34s
first-deployment-7cc4855cdf-mbt9h  1/1     Running   0           4m32s
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- Above are the 3 new pods.
- Note: the new replicaset created in same node as where our 1<sup>st</sup> replicaset created.

## Rolling out strategy



- As shown in above diagram rolling out will happen.
- If one POD goes down it will create one more POD one by one. It will not allow down time
- If you are updating the container image it will rollout the image one by one. 1<sup>st</sup> it will create new container image in new Replicaset and then it will destroy the old container.
- Then it will update the second POD as like above. It will do same for all.
- As per below picture we have to replica set, old replicaset had perl version of nginx image and current replicaset has latest version of nginx image.

Pods Replica sets

Delete Show labels

Current replica set

<input type="checkbox"/>	Name	Ready	Current	Age ↓	Images
<input type="checkbox"/>	first-deployment-7cc4855cdf	✓ 3/3	3	15 minutes	nginx:latest

Old replica sets

<input type="checkbox"/>	Name	Ready	Current	Age ↓	Images
<input type="checkbox"/>	first-deployment-5d57c4f49f	✓ 0/0	0	45 minutes	nginx:stable-alpine-perl

- If we want to go back to old version of “alpine-perl” we need to run below command
  - CMD: `kubectl rollout undo deployment/first-deployment`

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl rollout undo deployment/first-deployment
deployment.apps/first-deployment rolled back
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- To check revisions history we can run below command
  - CMD: `kubectl rollout history deployment/first-deployment`

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl rollout history deployment/first-deployment
deployment.apps/first-deployment
REVISION  CHANGE-CAUSE
2          kubectl.exe apply --filename=.\deployment.yml --record=true
3          <none>
```

```
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get rs
NAME                                DESIRED  CURRENT  READY  AGE
first-deployment-5d57c4f49f         3        3        3      3h16m
first-deployment-7cc4855cdf         0        0        0      166m
PS C:\Krish\Devops\AKS\Crating PODS using YAML>
```

- Now we can go to portal and see that our perl version shows in current version and latest version shows in older version as shown in below picture.

Pods Replica sets

 Delete  Show labels

Current replica set

<input type="checkbox"/>	Name	Ready	Current	Age ↓	Images
<input type="checkbox"/>	<a href="#">first-deployment-5d57c4f49f</a>	✓ 3/3	3	3 hours	nginx:stable-alpine-perl

Old replica sets

<input type="checkbox"/>	Name	Ready	Current	Age ↓	Images
<input type="checkbox"/>	<a href="#">first-deployment-7cc4855cdf</a>	✓ 0/0	0	3 hours	nginx:latest

- If we do roll out undo again latest will go to current and perl will come in old version.
- Below are the commands what we have used so far

```
## Get pods
kubectl get pods
## Get replica set
kubectl get rs
## Get deployments
Kubectl get deployment
## get service
Kubectl get svc
## Get everything
kubectl get all
#Creating a pod
kubectl run podname --image=nginx:latest
#Create a rs
kubectl create/apply -f filename.yml
#create a deployment
kubectl create/apply -f filename.yml
#Roll out
kubectl rollout history deployment/nameofdeployment
##Roll back
kubectl rollout undo deployment/nameofdeployment
```

- Note: While writing yaml, if write wrong in image it will show error.
- Now below we are using existing deployment.yml script and giving wrong image name.



```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: first-deployment
  labels:
    type: front-end
spec:
  template:
    metadata:
      name: nginxpod
      labels:
        app: mynginxapp
        type: front-end
    spec:
      containers:
        - name: nginxcontainer
          image: krish
      selector:
        matchLabels:
          type: front-end
      replicas: 3

```

- Image name we have given as krish that doesn't exist and it will fail in pulling image.
- Now run below command to execute the above script.
  - CMD: `kubectl rollout history deployment/first-deployment`

```

PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl apply -f .\deployment.yml
deployment.apps/first-deployment configured
PS C:\Krish\Devops\AKS\Crating PODS using YAML>

```

- As per below picture the above command failed to deploy 3 pods it just failed in pulling the image.

```

deployment.apps/first-deployment configured
PS C:\Krish\Devops\AKS\Crating PODS using YAML> kubectl get rs
NAME                                DESIRED   CURRENT   READY   AGE
first-deployment-5d57c4f49f         3         3         3       3h53m
first-deployment-7cc4855cdf         0         0         0       3h23m
first-deployment-98f896585         1         1         0       66s
PS C:\Krish\Devops\AKS\Crating PODS using YAML>

```

- We can also see the same in portal, as below.

Pods Replica sets

Delete Show labels

Current replica set

<input type="checkbox"/>	Name	Ready	Current	Age ↓	Images
<input type="checkbox"/>	first-deployment-98f896585	0/1	1	3 minutes	krish

Old replica sets

<input type="checkbox"/>	Name	Ready	Current	Age ↓	Images
<input type="checkbox"/>	first-deployment-5d57c4f49f	3/3	3	4 hours	nginx:stable-alpine-perl
<input type="checkbox"/>	first-deployment-7cc4855cdf	0/0	0	3 hours	nginx:latest

- As per above picture the script trying to pull the image krish but failed because krish image doesn't exist. So it's failed and the kubernetes are diverting traffic to existing Replicaset as shown above.
- We can pull any image, once it successfully pulled the image it will create pods one by one and it will delete the pods from old Replicaset.