**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?**

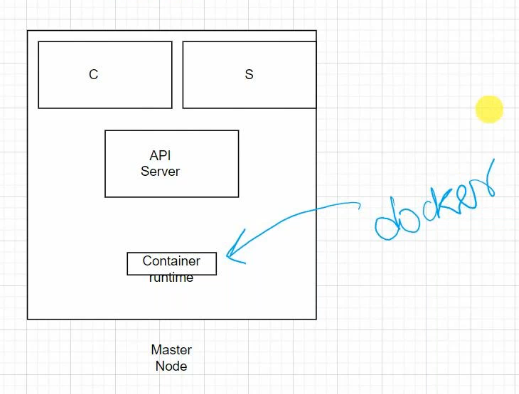
Kubernetes is **a popular open-source platform for container orchestration**. It enables developers to easily build containerized applications and services, as well as scale, schedule and monitor those containers.

**what is container orchestration?**

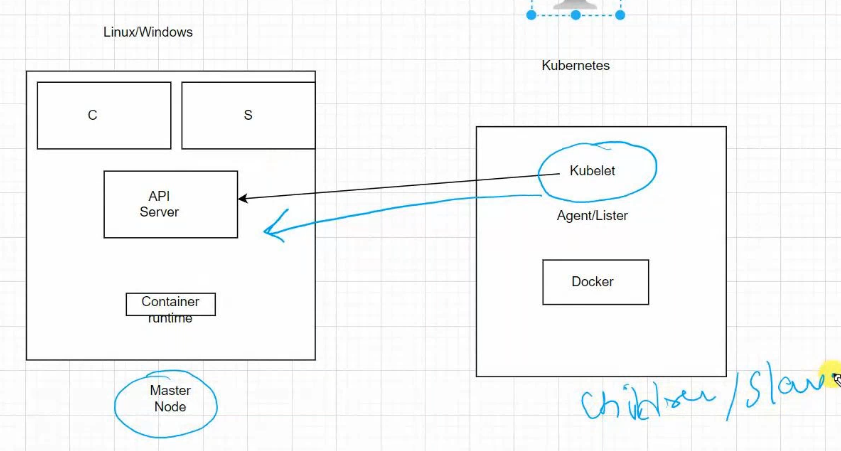
Container orchestration is **the automation of much of the operational effort required to run containerized workloads and services**. This includes a wide range of things software teams need to manage a container's lifecycle, including provisioning, deployment, scaling (up and down), networking, load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

**Kubernetes:**

* Node is similar to VM in Kubernetes terminology.



* API server is the heart of the Kubernetes.
* We can ineract 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.

I 

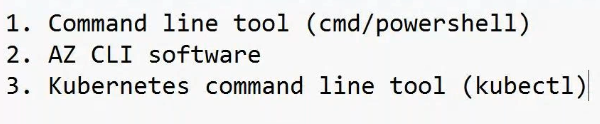
* As we seen in above diagram Master node API server will Interact with Kubelet on Child Node.
* Child Node is also called as Slave node
* We need to deploy POD on top of Child Nodes.
* Container will run in POD
* Containers are called “POD’s” in Kubernetes.
* We can run multiple PODs on same Node.

**Controller and Scheduler:**

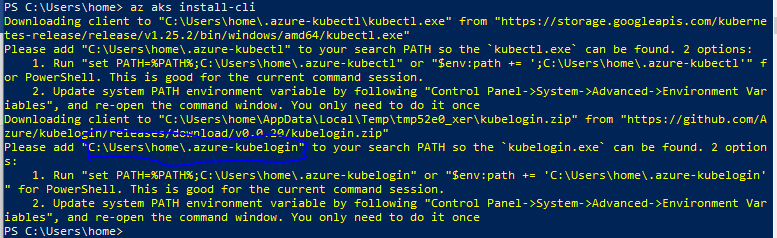
* Controller and scheduler are running on Master Node
* Controller will understand the problems in Nodes and PODs it will send the instruction to scheduler that why Node or POD is down.
* Scheduler will replicate the POD if POD is down, if Node is down it will create replica of the Node.
* Scheduler 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:**

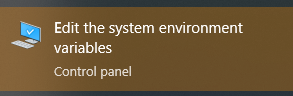
* Search as Kubernetes in market place.
* Create
* Config : dev/test
* Cluster name: democluster322
* Node size: B2ms
* Scale method : manual
* Node count: 1
* Click next to Node pool
* If we want we can create New Node Pool, but here we can go with default.
* Note: Node pool OS by default it will be Linux, if we want we can go to Windows also. Bute 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



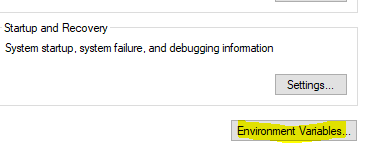
* CMD and powershell pre-installed on our Local machine
* To install AZ CLI open power shell and run “AZ”
* To install Kuberenetes we need to run “az aks install-cli” , the command will install.
* Now copy below specified path and paste in file manager.



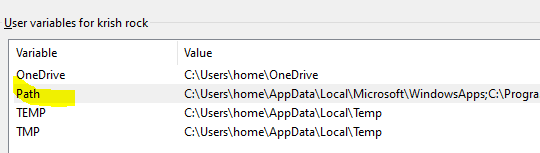
* Open envoironment variables as below



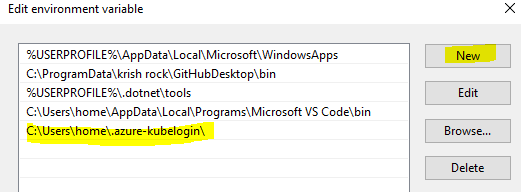
* Click on environment variables



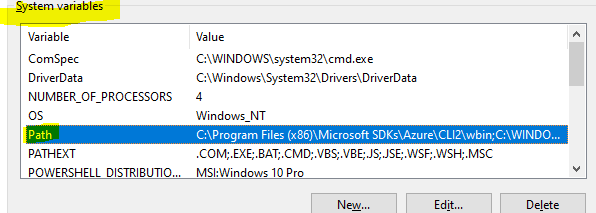
* Now double click on “Path”



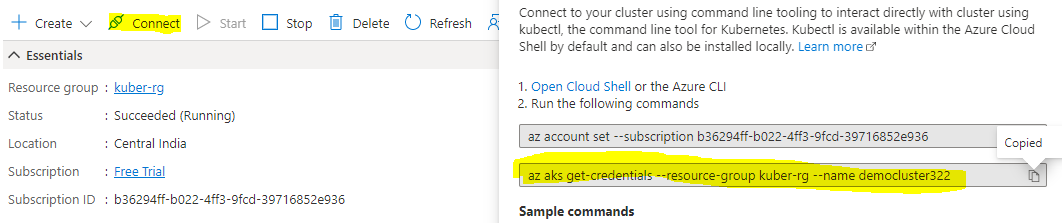
* Now click on new and Paste the path which we copied above “C:\Users\home\.azure-kubelogin\”



* 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.



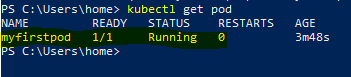
* Now paste the above command in powershell and run.
  + CMD: az aks get-credentials --resource-group kuber-rg --name democluster322
* The above command will login to kluster and we can use it
* Now to check PODs in kluster run below command
  + CMD: kubectl get pods
* To check nods run below commands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

**Creating POD:**

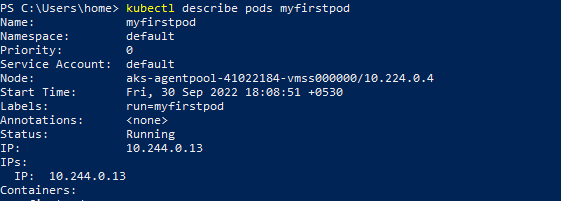
* Now run below command to create POD
  + CMD: kubectl run myfirstpod --image nginx



* 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



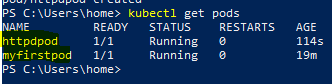
* To chek our pod details run below command
  + CMD: kubectl describe pods myfirstpod
* Like below it will show so many details, I have taken snapshot for few details.



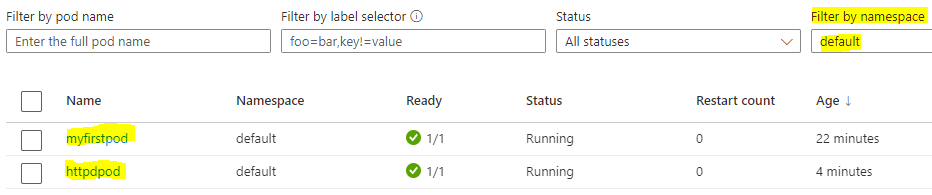
* Now we can install one more POD for “httpd” image
  + CMD: kubectl run httpdpod --image httpd



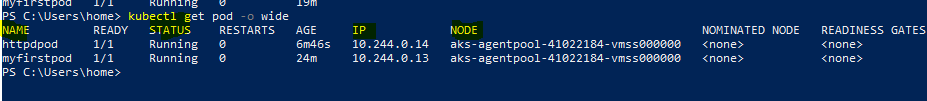
* We can check the pods below commands
  + CMD: kubectl get pods



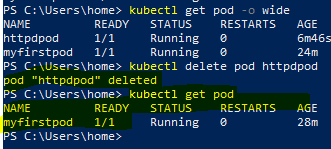
* We can also check our pods in portal
  + Go to kluster
  + Click on Workloads on blade
  + Filter by default, the we can see our pods



* Now run below commands to get few major details of the PODs
  + CMD: kubectl get pod -o wide

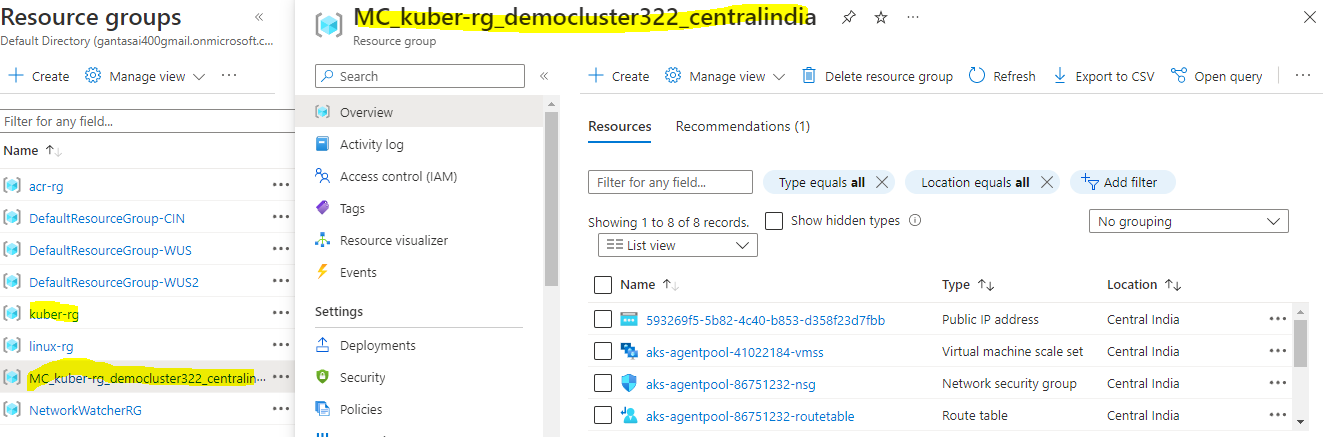


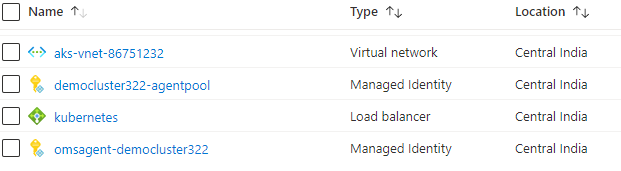
* To remove any pod run below command
  + CMD: kubectl delete pod httpdpod
* Now check pods
  + CMD: kubectl get pod



* We can see in the above picture hpptd pod is deleted and only one pod is available

Note: Whenever we are creating kuberenetes kluster it will also create other resouce group in that it will create few resources lik VNet, IP, Virtual machine scale set kubernetes loadbalance and more.

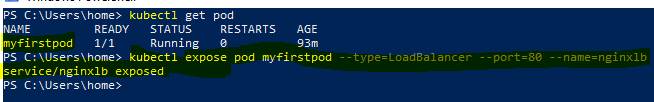




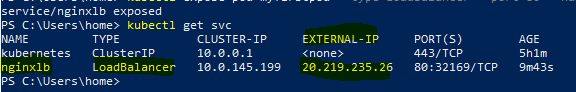
* As shown in above screenshot we have created only kuber-rg and in that kubernetes kluster, but microsoft will deploy other resource group as shown as MC\_kuber-rg. In that it will deploy above shown resources.

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

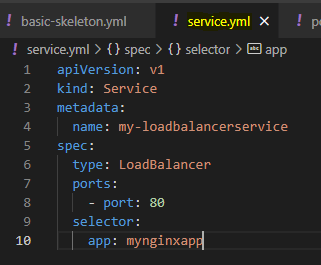


* The above command has exposed the POD
* Now to check our external IP run below command
  + CMD: kubectl get svc

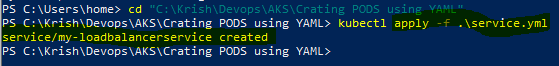


* Now go to browser and search with external IP we can reach out to nginx page.
* Now we can run “httpd” image using below command.
  + CMD: kubectl run httpdpod --image httpd
* Now we can expose the POD using below command.
  + CMD: kubectl expose pod httpdpod --type=LoadBalancer --port=80 --name=httpdlb
* 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.

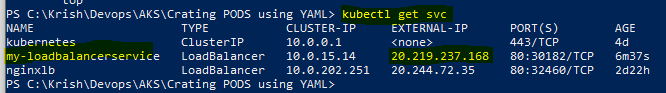
Now we create Kubernate service using belo Yaml file.



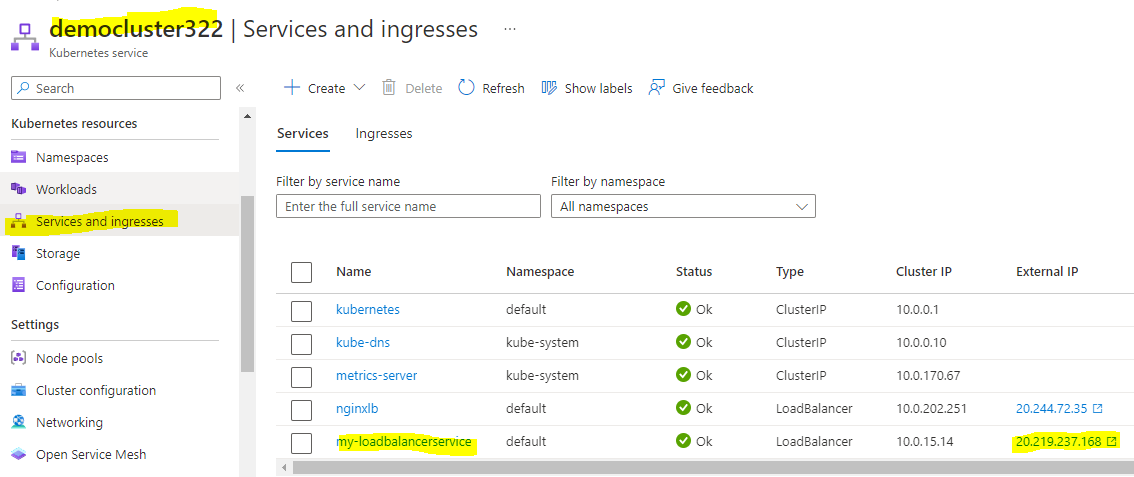
* Now go to power shell and run below command
  + CMD: kubectl apply -f .\service.yml



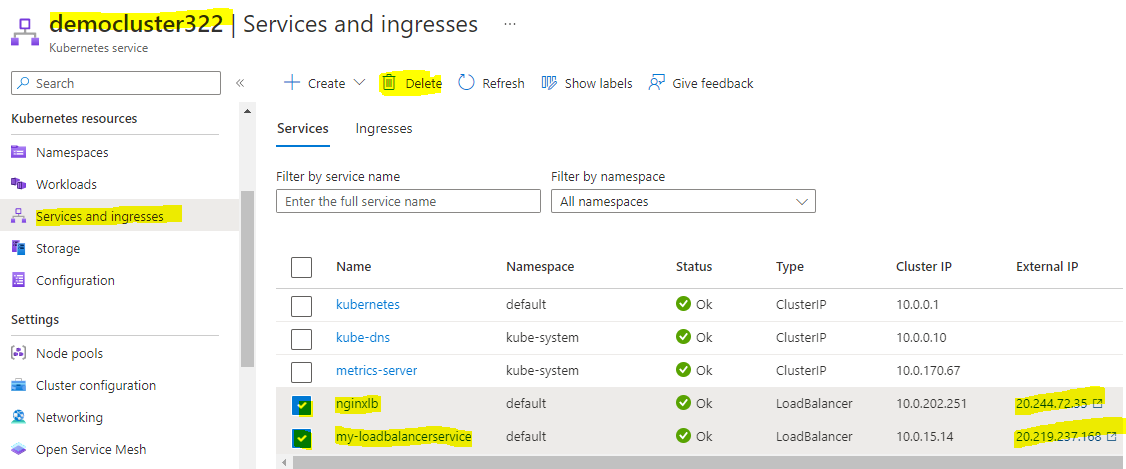
* Now run below command to check the IP of our just installed loadbalancer
  + CMD: kubectl get svc



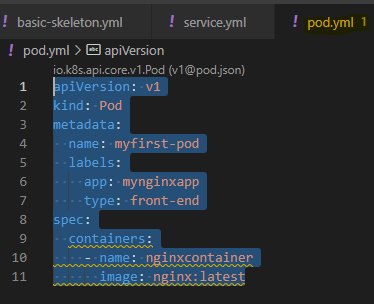
* Now we can see without IP address without running “expose” command.
* We can see the same in portal also.
* Go to cluster
* Services and ingresses
* Then we can see the loadbalancer service



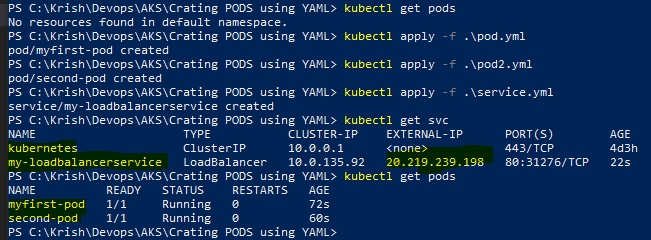
* Now if you want to reach to nginx we need to go through the LoadBalancer. If POD goes down we can’t able to reach the webpage. So we need to install multipler pods on same Node. So if one POD goes down then LoadBalancer will send to other POD.
* No we need to delete the existing POD and LoadBalancer.
  + Go to Cluster
  + Click on services and Ingesses
  + Now remove POD and Loadbalancer



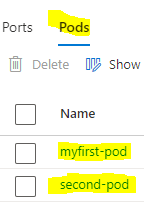
* Now check PODs using below commands, there won’t be any POD.
  + CMD: kubectl get pods
* If any pod not deleting run below command to remove
  + CMD: kubectl delete pod myfirst-pod
* Now we can create multiple PODS using YML script
* Now we can use earlier writen POD.YML, service.yml and we need to 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 thew script as pod2.yml



* 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 attache 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.

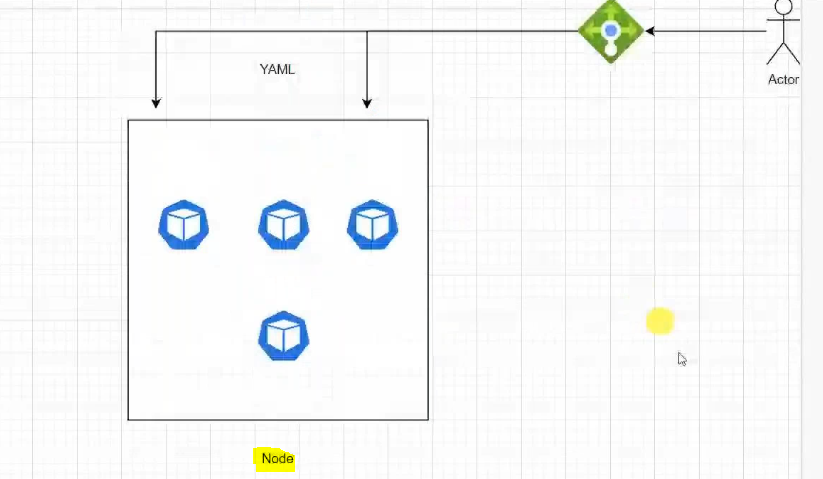


* Now we can see that loadbalancer is handling 2 pods in poratl
* Go to cluser
* Click on services and ingresses
* Select loadbalancer
* Now click on pods
* Then it will show the both pods which it is handling



**Replication Set**

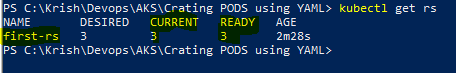
* Earlier we have deployed the PODs on Node directly.
* For high avialability we have created multiple PODs and attached to loadbalancer. But while deploying PODs it will be so manual using earlier method. If we need 1 or 2 pods then we can create the yaml script for each pod. But if we need many the 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.

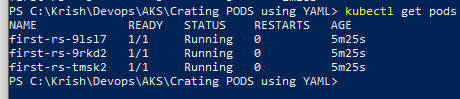


* Before we deploying the replication set, we ensure to delete POD and loadbalancer which we have earlier deployed.
* Now create the new file as “replicaset.yml” in Visual studio code.
* Write below script on replicaset file
* apiVersion: apps/v1
* kind: ReplicaSet
* metadata:
* name: first-rs
* labels:
* 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

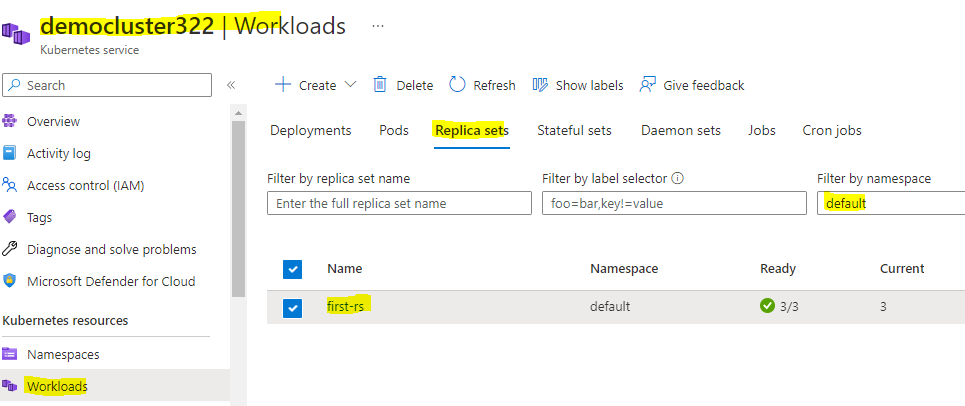


* As per above we can see our replica set has created. To check our replica set run below command
  + **CMD: kubectl get rs**
* As per YML script we have given 3 replicas, and it has deployed the same as below

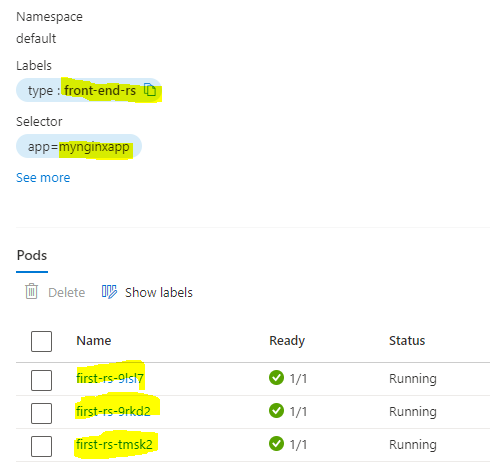




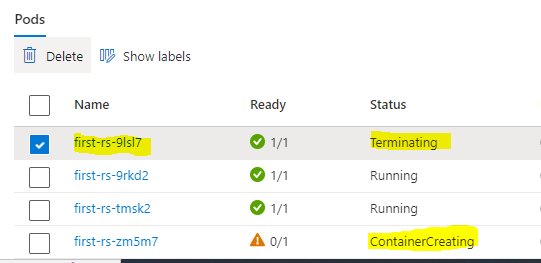
* We can see the same in portal, this pods under replicaset.
* Now go to cluser in portal
* Click on workloads on blade
* Click on replicaset on top
* Filter by default
* Now click on our replicaset



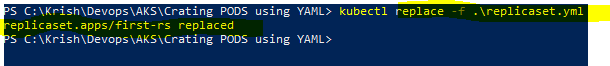
* Now it will show the pods.



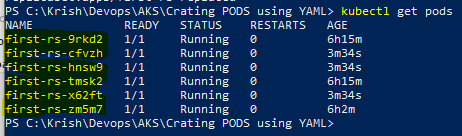
* If our pod geos down or deleted it auto matically create one more pod. In below we tried to delete the one pod and it is terminating one and creating one as below.



* 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 chage 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 ren below command to excute 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.



* To check the pods run below command

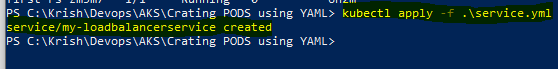


* Above are the 6 pods which we have deployed on the above replace command.

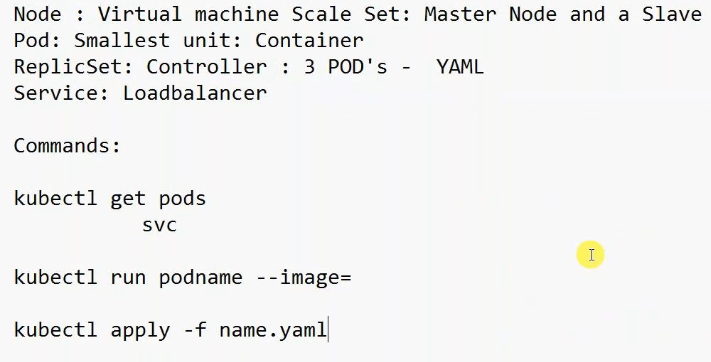
**Orchestration:**

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

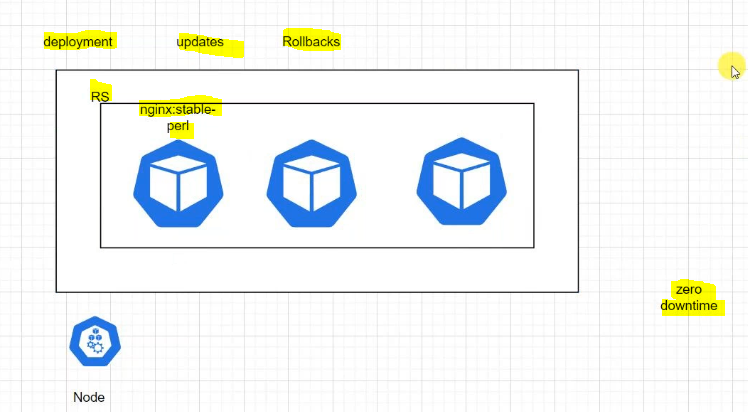
* Now we can attach Loadbalancer using our “service.yml” file
* The script as below.
* apiVersion: v1
* kind: Service
* metadata:
* name: my-loadbalancerservice
* spec:
* type: LoadBalancer
* ports:
* - port: 80
* selector:
* app: mynginxapp
* The LoadBalancer will attach to the PODS because we are giving selector as “mynginxapp”



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



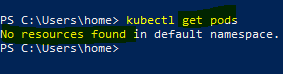
**Deplioyment**



* Deployment having advantages ove replicaset. In above diagram “RS” is nothing but Replicaset.
* When we do deployment it will alos install Replicaset as shown in above picture.
* If we are using deployment if we want to update the container image, kubernetes will do in increamental manner. It will update the 1 POD then sencond 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 increamental manner.
* Kubernetes always zero down time

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

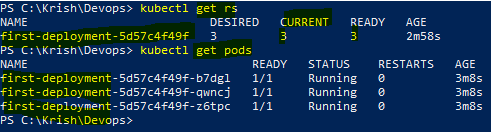
* + CMD: kubectl get pods



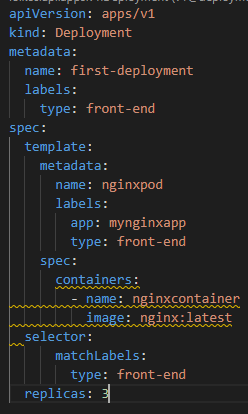
* Now write below script and save as “deployment.yml”
* We can use replicaset and need to do few changes, major change in kind has to change Replicaset to deployment.
* The code as below
* apiVersion: apps/v1
* kind: Deployment
* metadata:
* name: first-deployment
* labels:
* type: front-end
* spec:
* 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



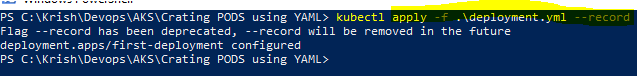
* It has created deployment in that it has created Replicaset and pods as shown in below.



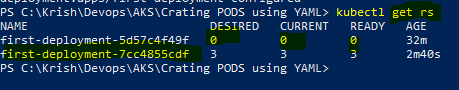
* As shown above deployment script has created the one Replicaset and 3 pods.
* Erlier we have created nginx stable perl version now we wan to updat to latest. So we need to do small change in yml script at image



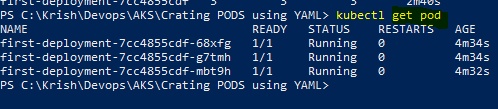
* Now run below command.



* It has created new replicaset as shown in below.

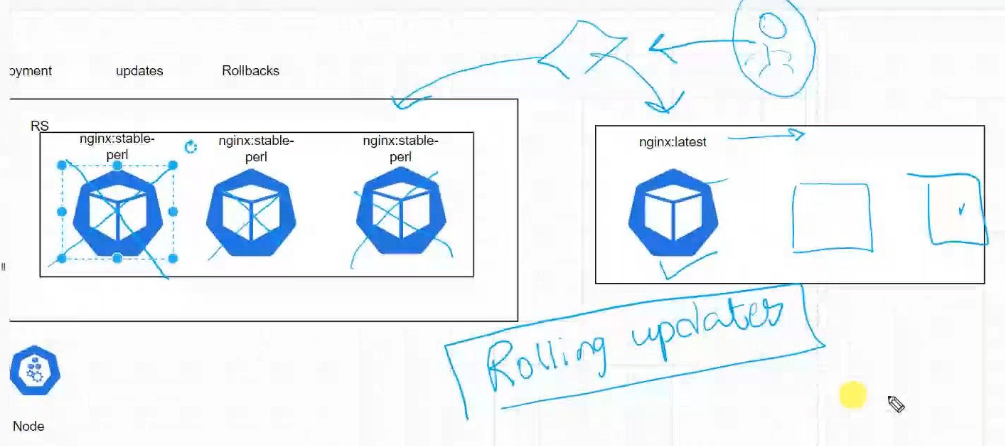


* As shown in above picture PODs removed from earlier Replicaset and created new replicaset inside that new 3 pods created.

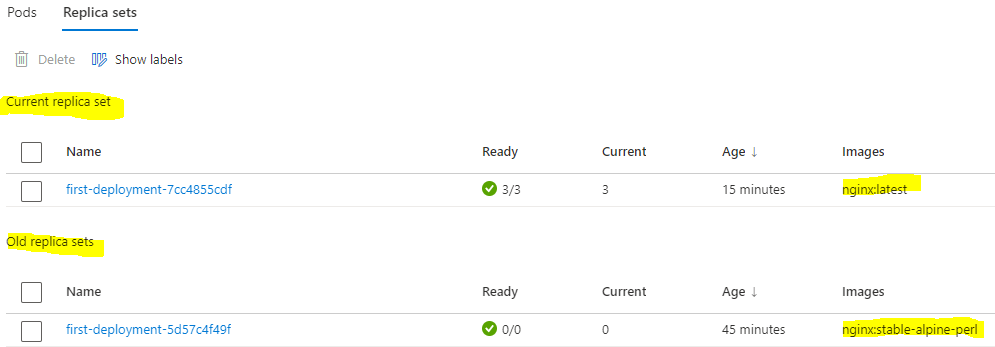


* Above are the 3 new pods.
* Note: the new replicaset created in same node as where our 1st replicaset created.

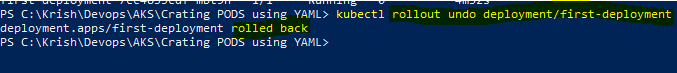
**Rolling out stratagy**



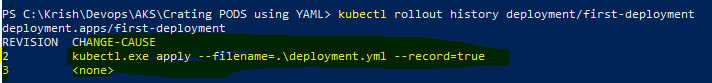
* As shown in above diagram rolling out will happen.
* If one POD goes down it will create one more POD one by one. It will not allow down time
* If you are updating the container image it will rollout the image one by one. 1st it will create new container image in new Replicaset and then it will destroy the old container.
* Then it will update the second POD as like abobe. 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.

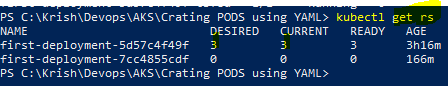


* If we want to go back to orl version of “alpine-perl” we need to run below command
  + CMD: kubectl rollout undo deployment/first-deployment

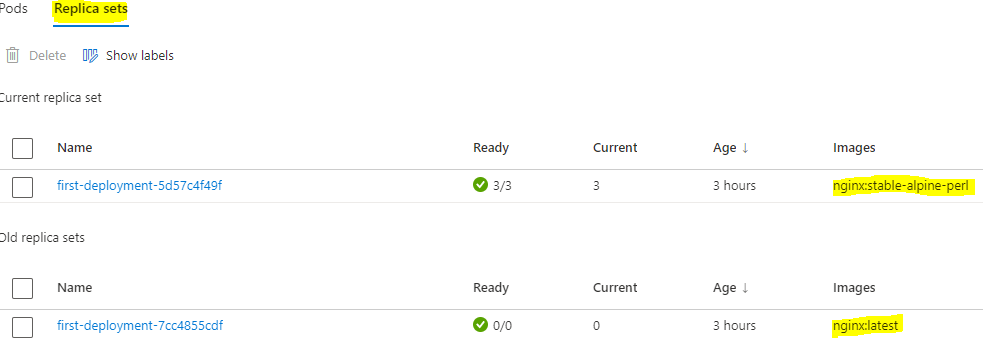


* To check revisions history we can run below command
  + CMD: kubectl rollout history deployment/first-deployment

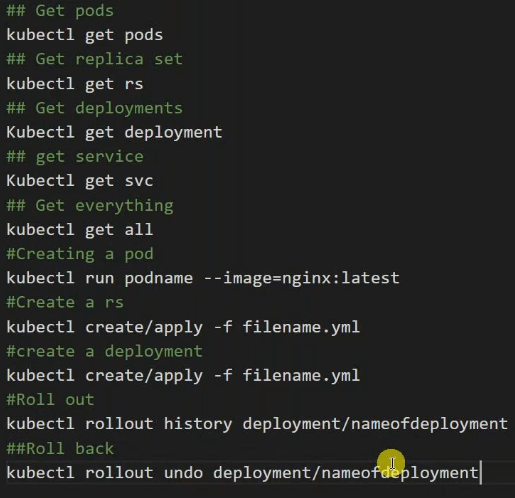




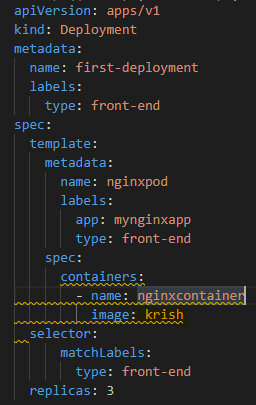
* Now we can go to portal and see that our perl version shows in current version and latest version shows in older version as shown in below picture.



* 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



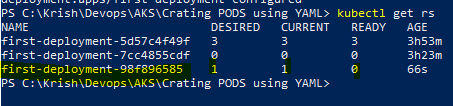
* Note: While writing yml, if write wrong in image it will show error.
* Now below we are using existing deployment.yml script and giving wrong image name.



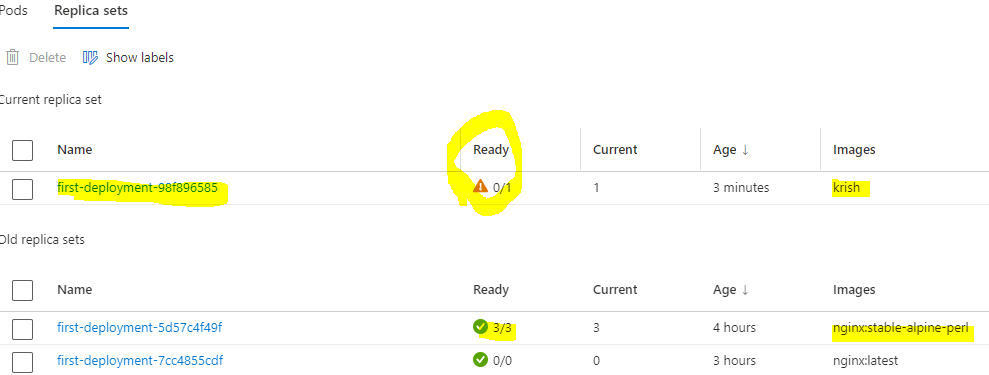
* Image name we have given as krish that doesn’t exist and it will fail in pulling image.
* Now run below command to excute the above script.
  + CMD: kubectl rollout history deployment/first-deployment



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



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



* As per above picture the script trying to pull the image krish but failed because krish image doesn’t exist. So it’s fialed and the kubernetes are divirting 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.