Kubernetes namespace:

Namespace is virtual cluster withing our cluster.

Name of the resource should be unique within namespace, but not across namespace.

e.g. POD name should be unique in DEV namespace, in UAT namespace same name of POD can be created.

When to use?

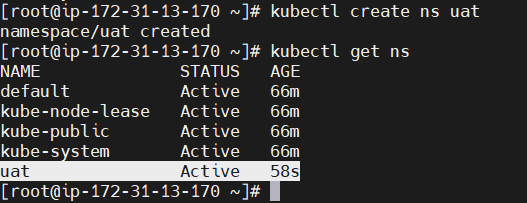
When we have multiple envs.

Kubectl get ns 🡪 displays the namespace

By default it use default namespace.

Create namespace:

Kubectl create ns uat

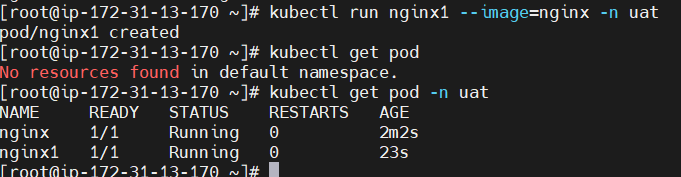


Create pod in namespace

kubectl run nginx1 --image=nginx -n uat

Check that pod in that namespace

kubectl get pod -n uat



Note: if you search pod in default namespace you will not get that pod. Also you cannot create same name pod in same namespace.





Second way:

Same can be achieve using our manifest file.

vim pod1.yaml

cat pod1.yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx12

namespace: uat

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

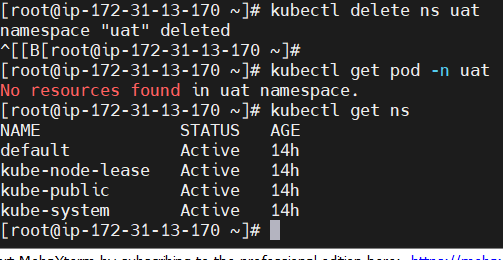
- containerPort: 80

kubectl apply -f pod1.yaml

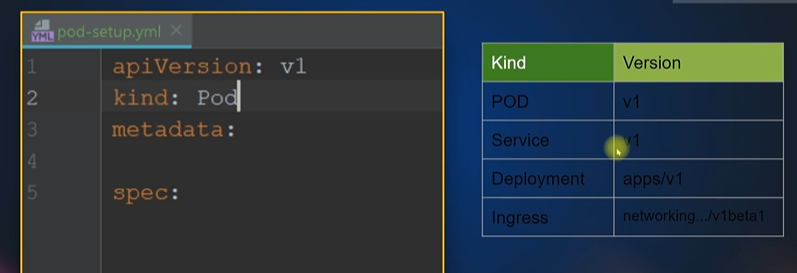
kubectl get pod -n uat

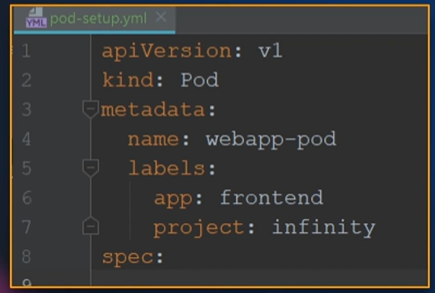
Delete namespace

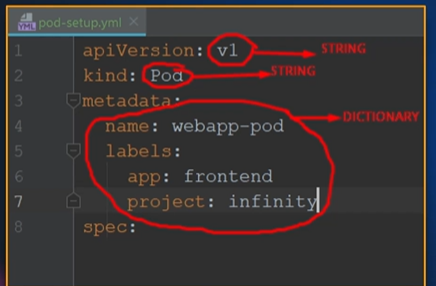
kubectl delete ns uat

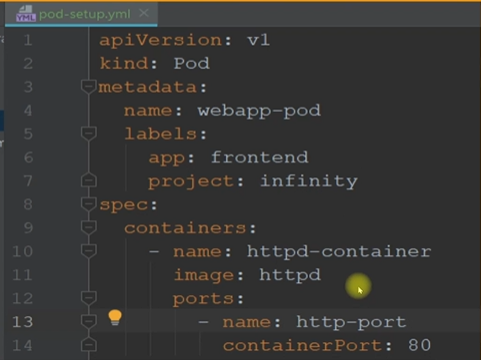


PODS









How to create pod using definition file.

cat pod.yml

apiVersion: v1

kind: Pod

metadata:

name: demo-pod

labels:

app: demo-app

spec:

containers:

- name: demo-nginx

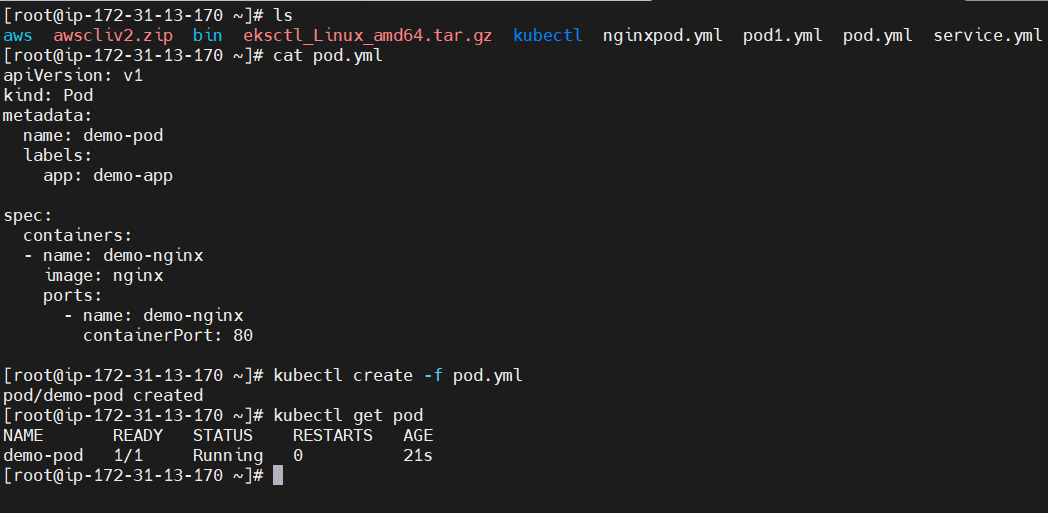
image: nginx

ports:

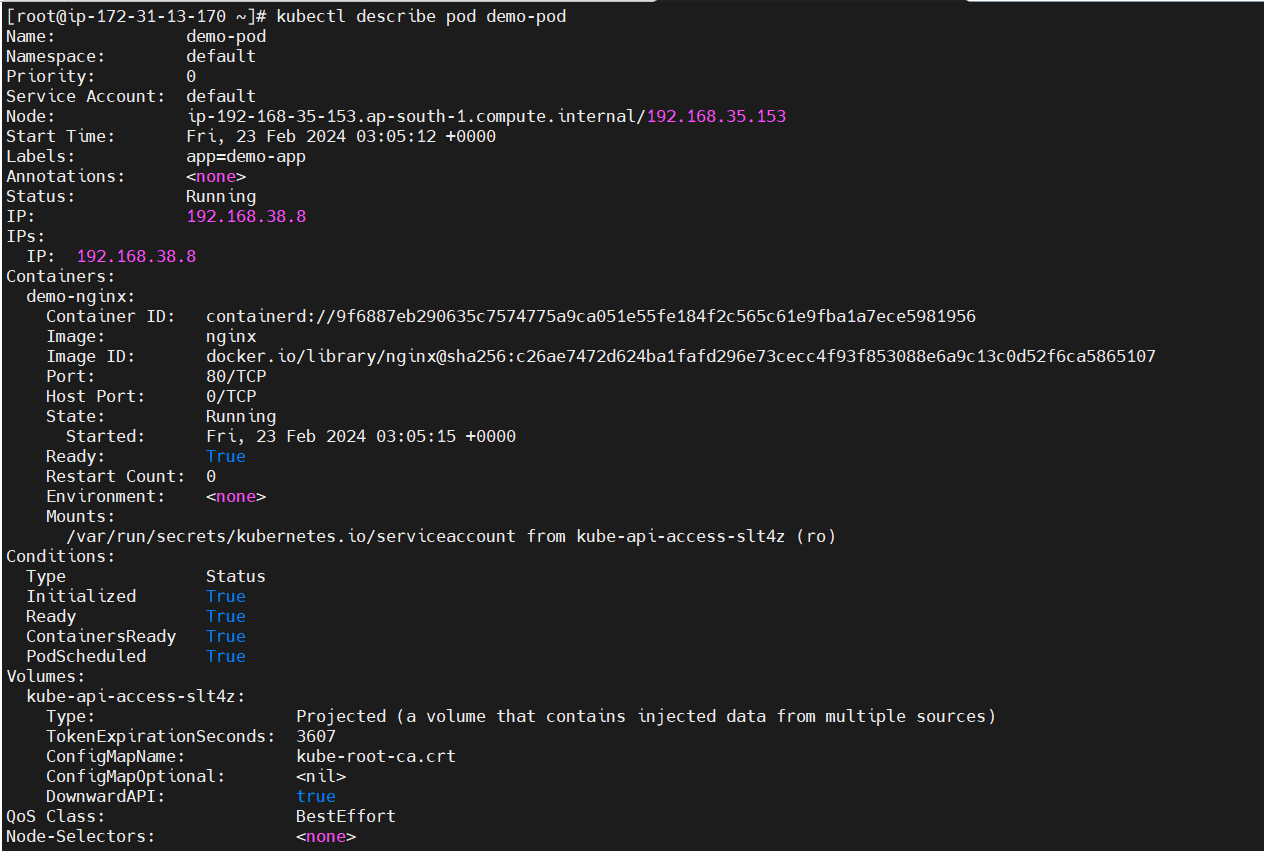
- name: demo-nginx

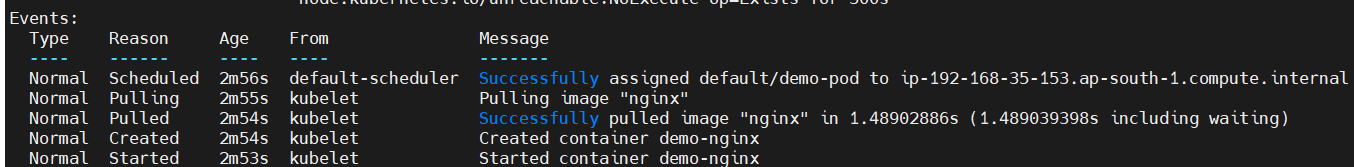
containerPort: 80

kubectl create -f pod.yml



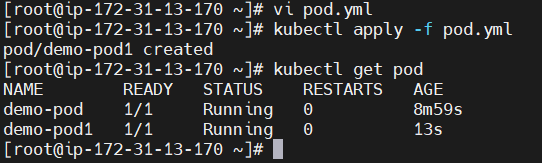
If you want to see detail information of pod you can use describe pod.





You can use apply command also to create pod.

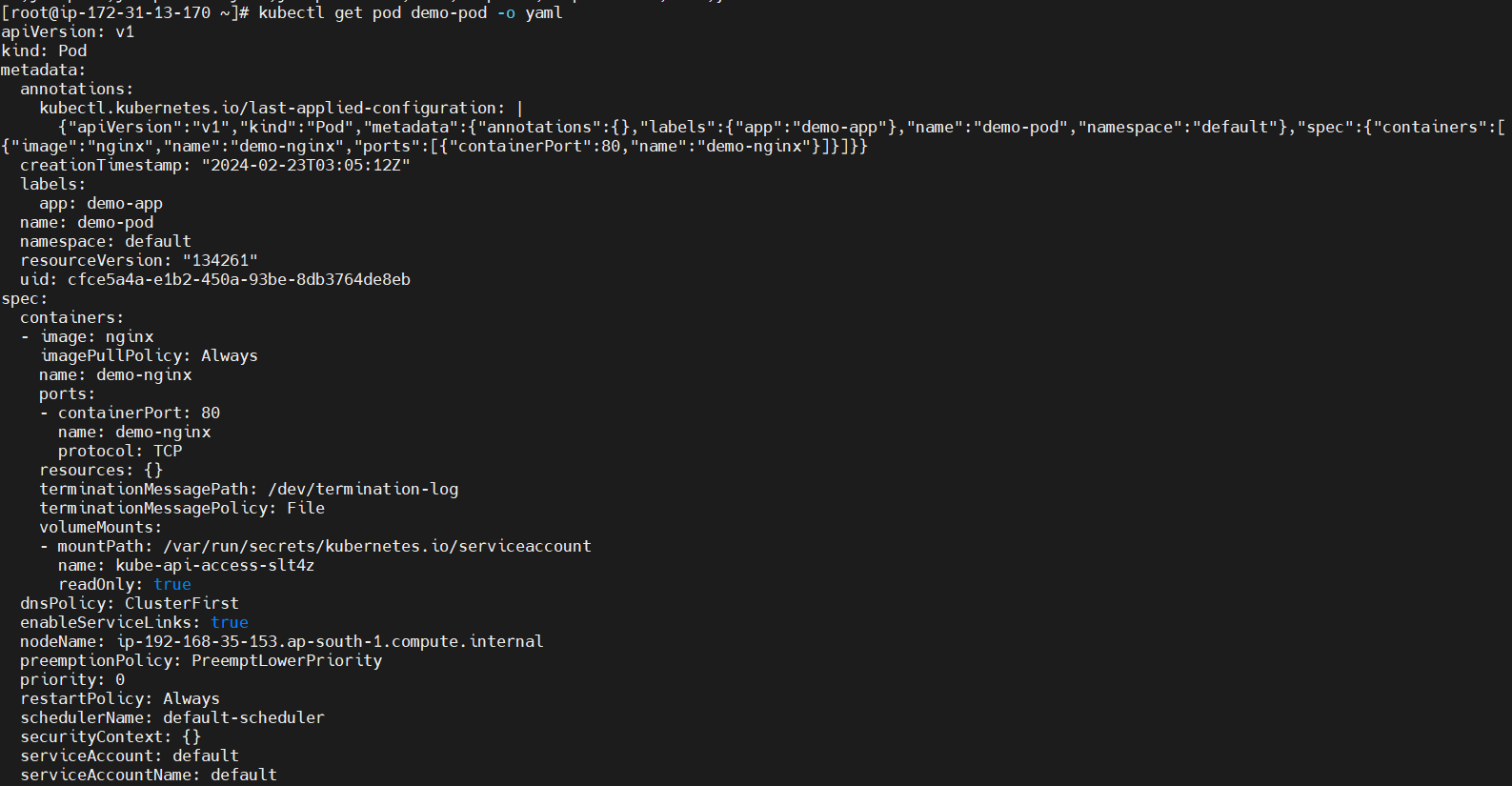
kubectl apply -f pod.yml



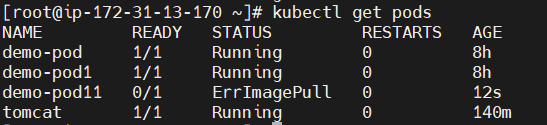
Kubectl apply is used for modification/updates in current resource.

Kubectl Create is used to create new resource.

To get detailed information about pod  
kubectl get pod demo-pod -o yml

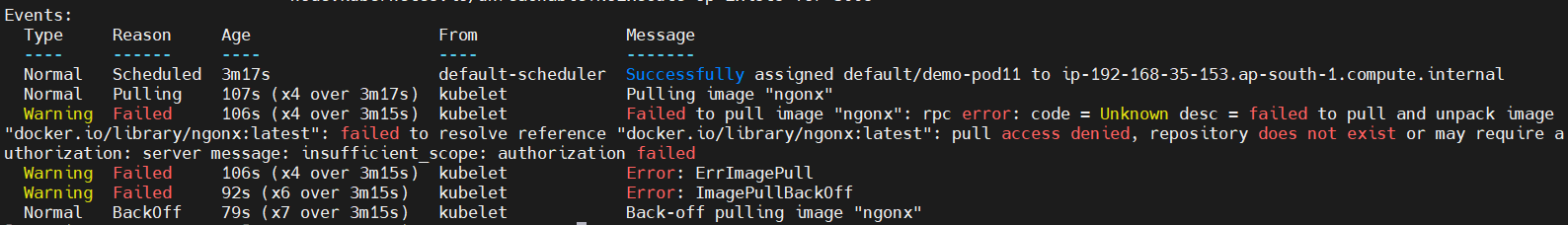


If you give image name wrong, you will see below error

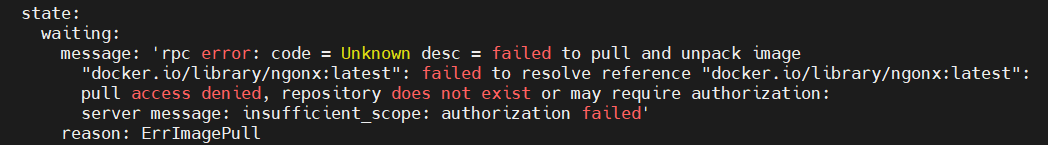


How to debug pod error?

kubectl describe pod demo-pod11



kubectl get pod demo-pod11 -o yaml

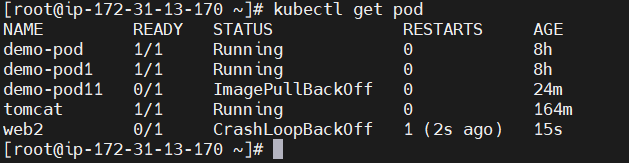


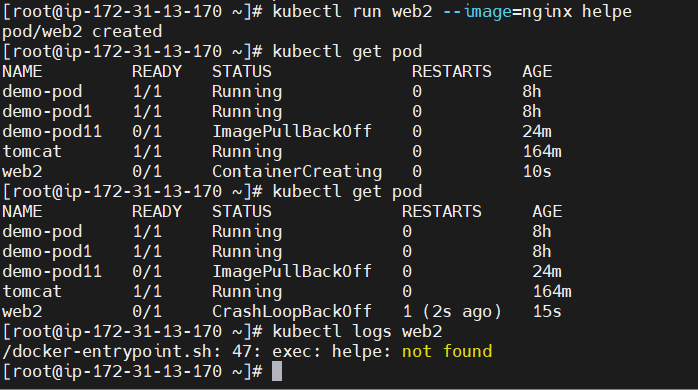
Kubectl logs Pod\_name



Now create a pod with command and give wrong paramaters to docker container then it will crash the POD, as you can see below

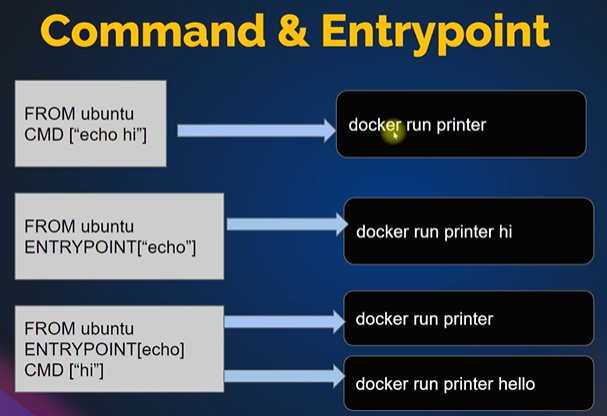
kubectl run web2 --image=nginx helpe







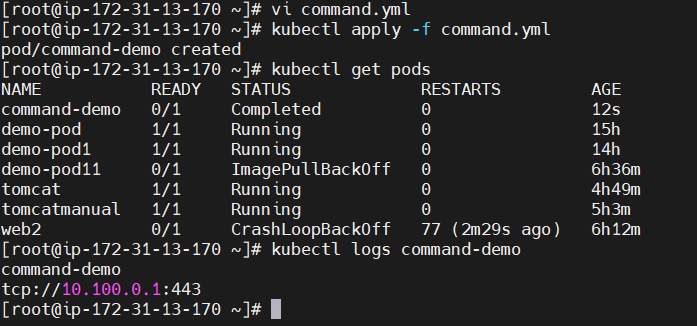
How to Pass command or any argument to container?



When we use both (in last example) entrypoint will override the cmd and print echo hello

https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/

Here you can see that status is completed as container is not running because this is not continuously run like other, it performs some action and return back normal.

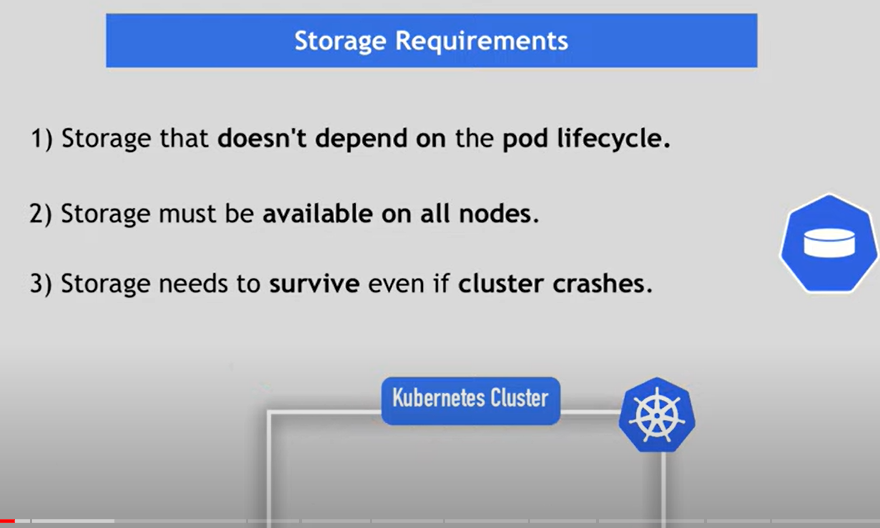


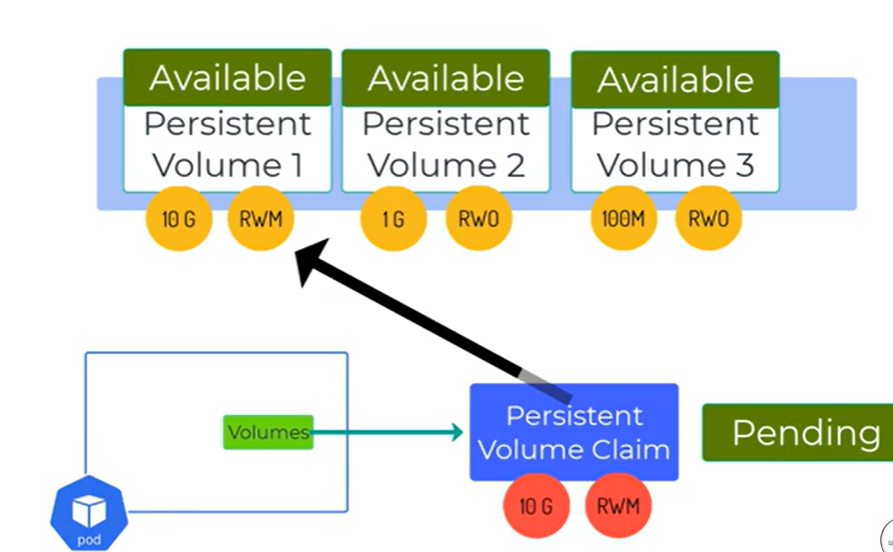
Volume:

<https://kubernetes.io/docs/concepts/storage/volumes/>

Types of Volumes:

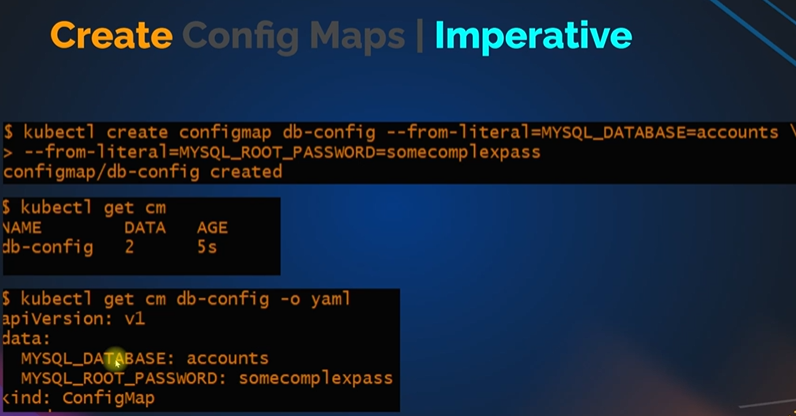
1. awsElasticblockstore
2. hostpath
3. local
4. pv(It is separate volume created and then it mount in POD)
5. azure

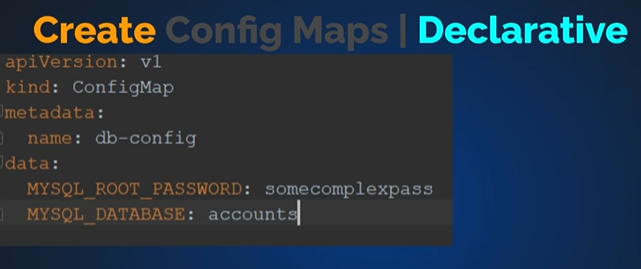


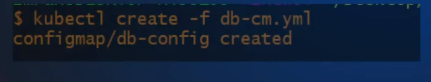


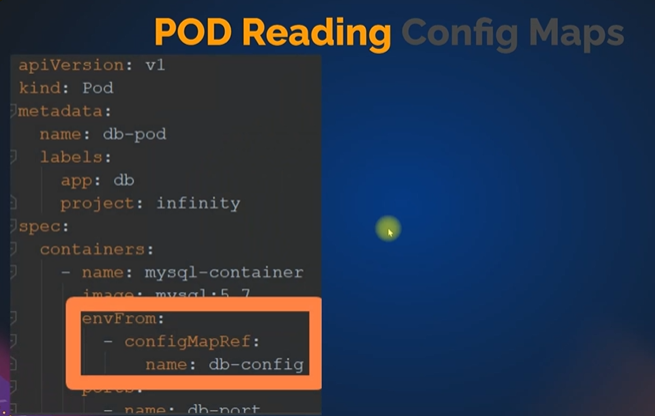
Config maps:

Two ways to implement configmap:

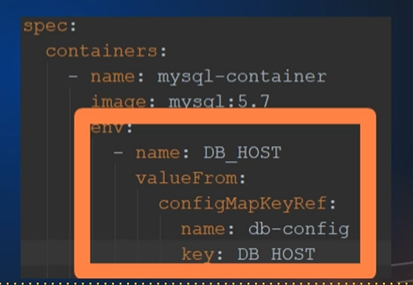








If you want spefic vaule from config map use below



Practical:

Create configmap yaml

<https://kubernetes.io/docs/concepts/configuration/configmap/>

**apiVersion**: v1

**kind**: ConfigMap

**metadata**:

**name**: game-demo

**data**:

*# property-like keys; each key maps to a simple value*

**player\_initial\_lives**: "3"

**ui\_properties\_file\_name**: "user-interface.properties"

*# file-like keys*

**game.properties**: |

*enemy.types=aliens,monsters*

*player.maximum-lives=5*

**user-interface.properties**: |

*color.good=purple*

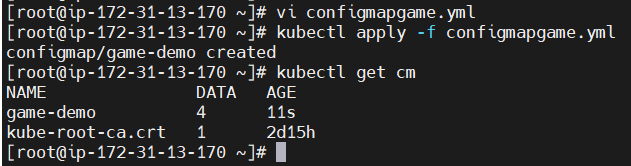
*color.bad=yellow*

*allow.textmode=true*

kubectl apply -f configmapgame.yml

kubectl get cm

kubectl get cm game-demo -o yml



Vi configure-pod.yml

apiVersion: v1

kind: Pod

metadata:

name: configmap-demo-pod

spec:

containers:

- name: demo

image: alpine

command: ["sleep", "3600"]

env:

# Define the environment variable

- name: PLAYER\_INITIAL\_LIVES # Notice that the case is different here

# from the key name in the ConfigMap.

valueFrom:

configMapKeyRef:

name: game-demo # The ConfigMap this value comes from.

key: player\_initial\_lives # The key to fetch.

- name: UI\_PROPERTIES\_FILE\_NAME

valueFrom:

configMapKeyRef:

name: game-demo

key: ui\_properties\_file\_name

volumeMounts:

- name: config

mountPath: "/config"

readOnly: true

volumes:

# You set volumes at the Pod level, then mount them into containers inside that Pod

- name: config

configMap:

# Provide the name of the ConfigMap you want to mount.

name: game-demo

# An array of keys from the ConfigMap to create as files

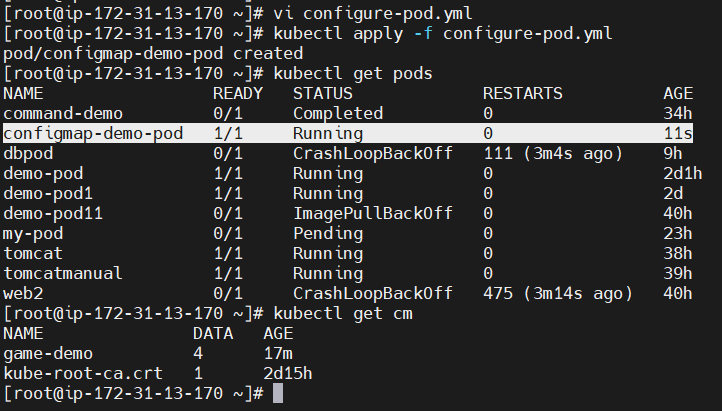
items:

- key: "game.properties"

path: "game.properties"

- key: "user-interface.properties"

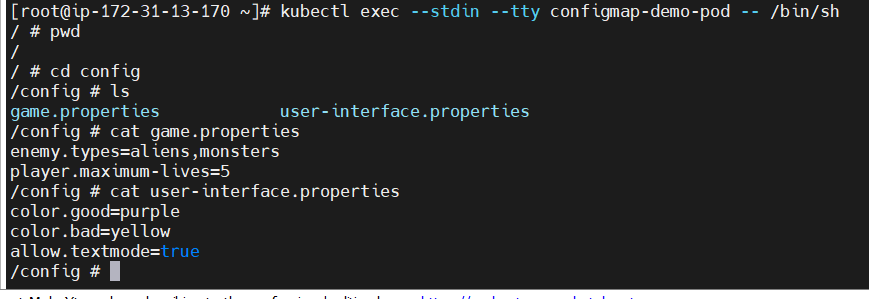
path: "user-interface.properties"

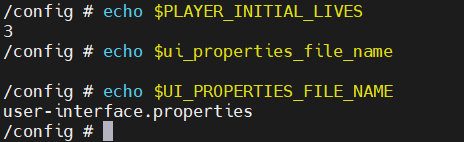


Login into pod

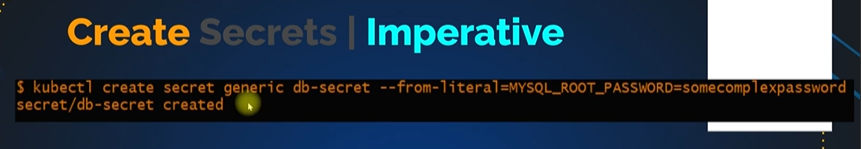
kubectl exec --stdin --tty configmap-demo-pod -- /bin/sh

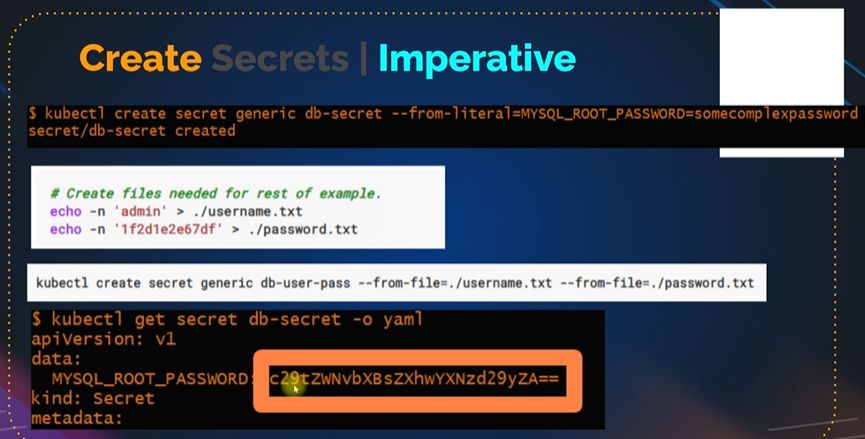


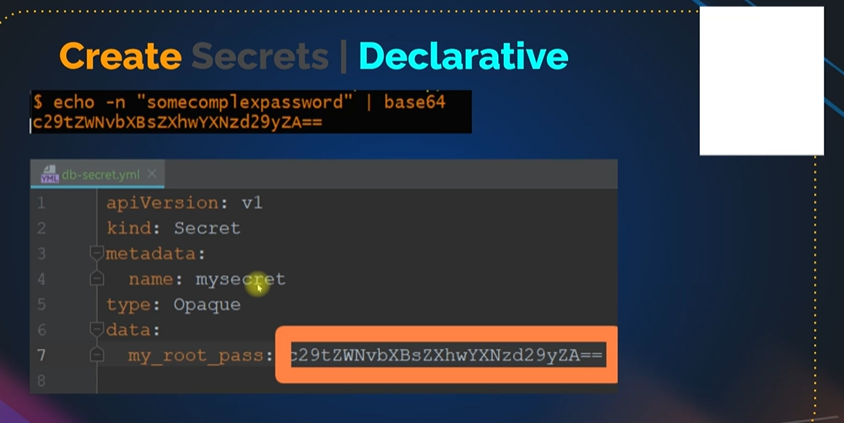


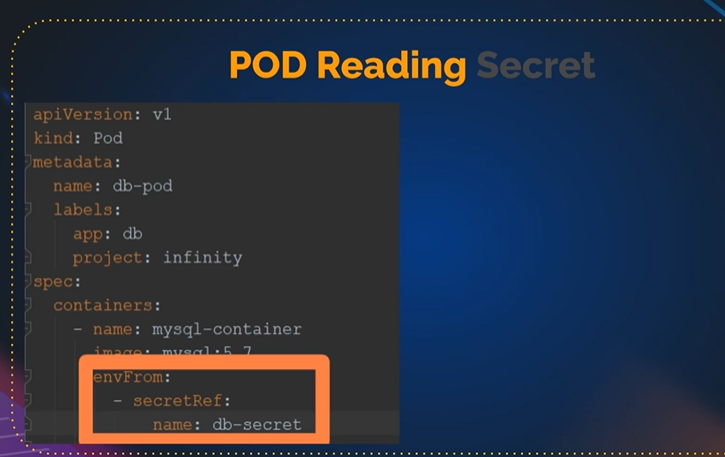


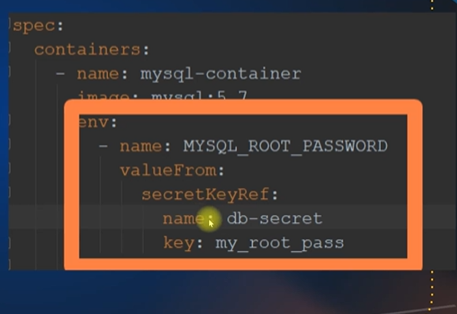
Secrets:











# Encode text

echo -n "admin" | base64

echo -n "mysecretpass" | base64

# Create Secret

vim mysecret.yaml

apiVersion: v1

kind: Secret

metadata:

name: mysecret

data:

username: YWRtaW4=

password: bXlzZWNyZXRwYXNz

type: Opaque

kubectl create -f mysecret.yaml

# Create Pod to read secret keys

vim readsecret.yaml

apiVersion: v1

kind: Pod

metadata:

name: secret-env-pod

spec:

containers:

- name: mycontainer

image: redis

env:

- name: SECRET\_USERNAME

valueFrom:

secretKeyRef:

name: mysecret

key: username

optional: false # same as default; "mysecret" must exist

# and include a key named "username"

- name: SECRET\_PASSWORD

valueFrom:

secretKeyRef:

name: mysecret

key: password

optional: false # same as default; "mysecret" must exist

# and include a key named "password"

restartPolicy: Never

kubectl create -f readsecret.yaml

kubectl get pod

# Login to Pod echo print variables

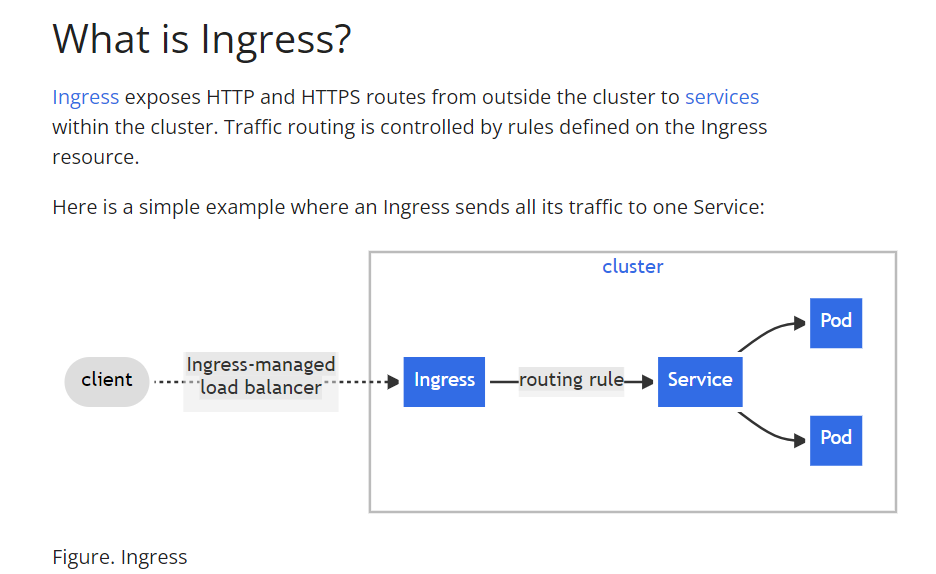
kubectl exec --stdin --tty secret-env-pod -- /bin/bash

echo $SECRET\_USERNAME

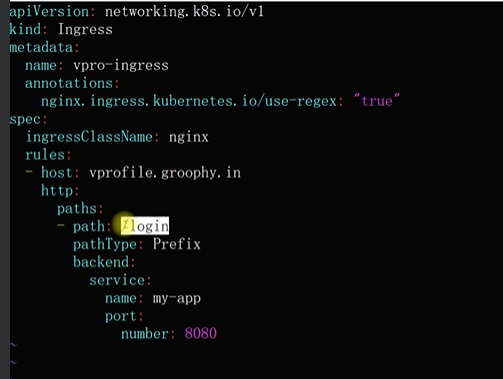
echo $SECRET\_PASSWORD

Ingress:

Ingress is for external access to application which comes from internet http



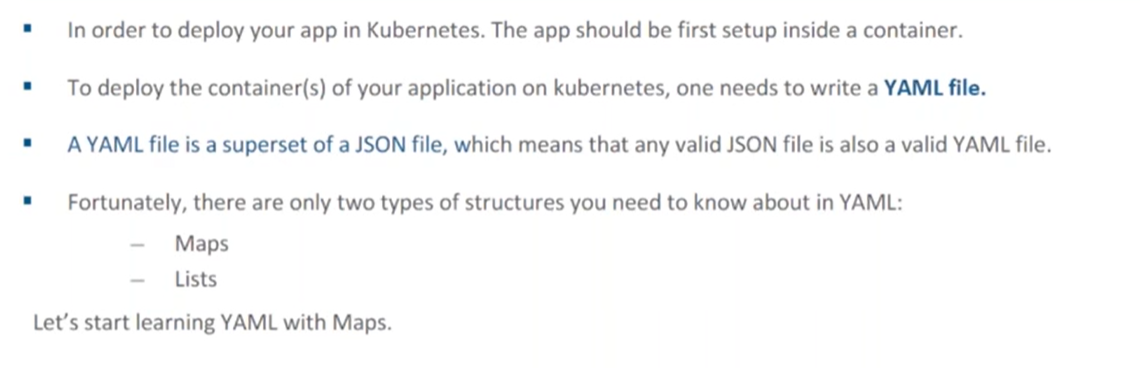
<https://kubernetes.github.io/ingress-nginx/deploy/#aws>

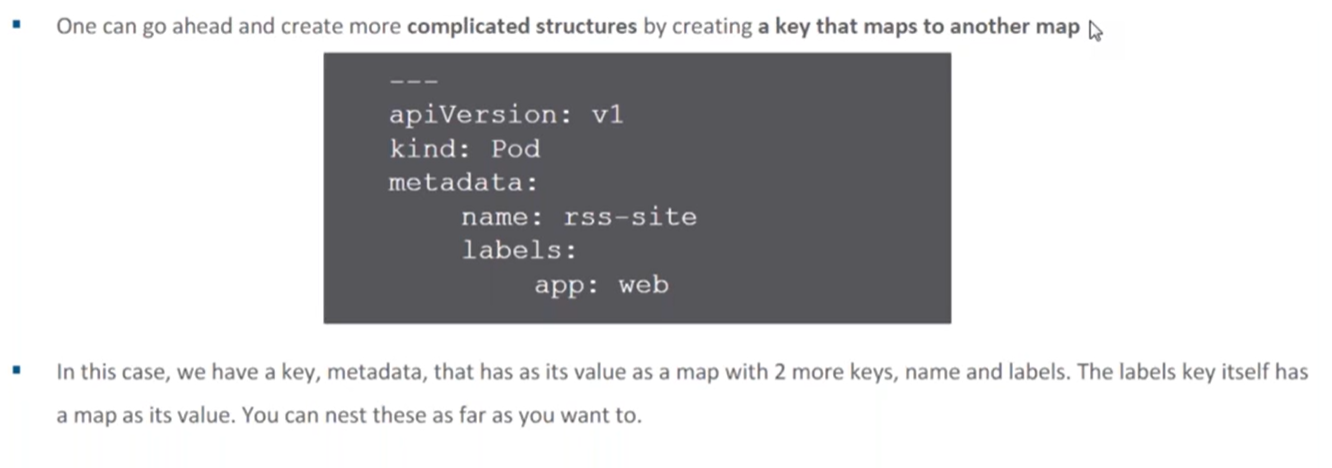


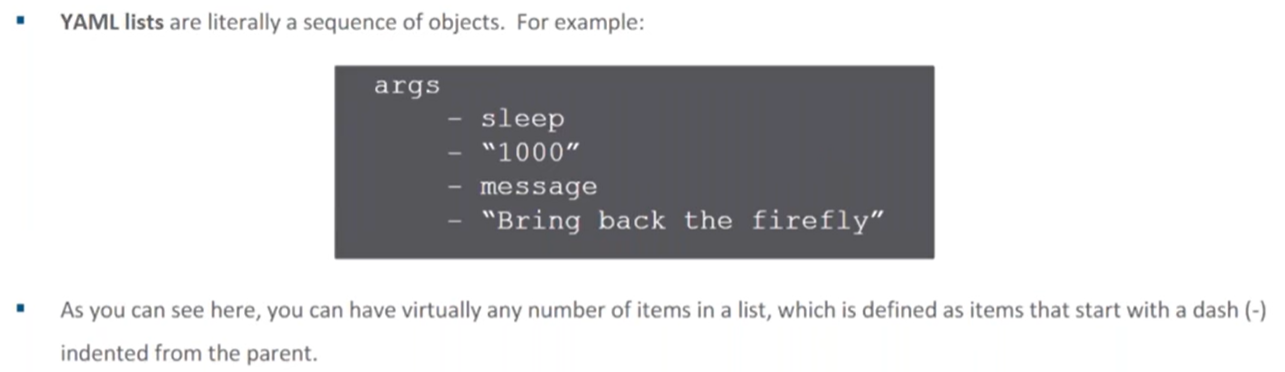




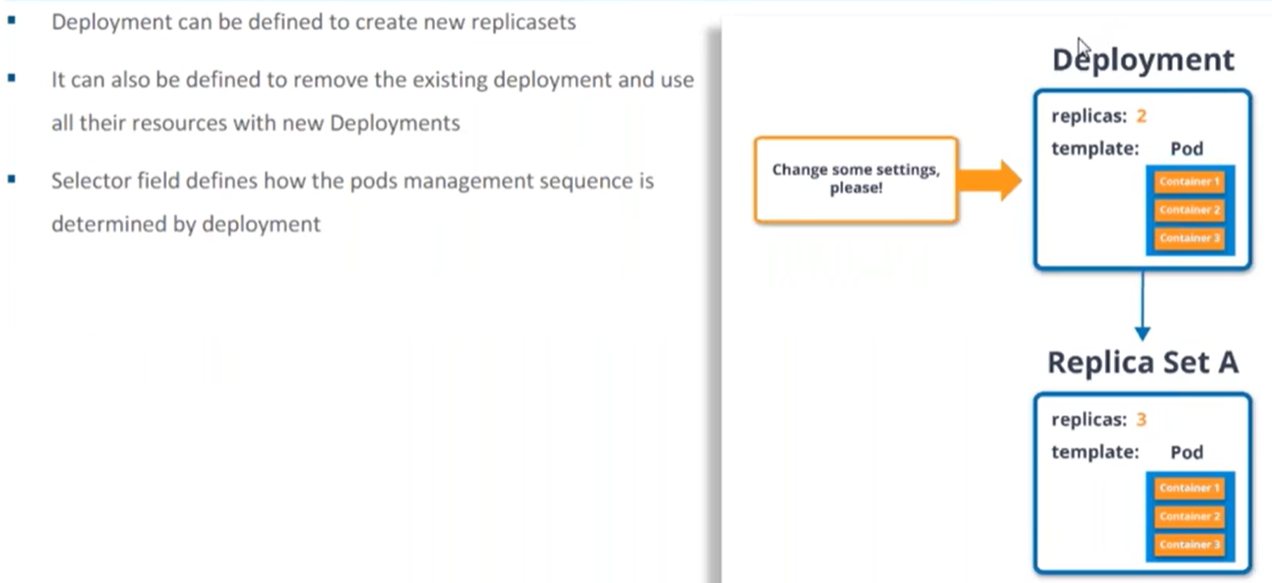
How to deploy apps in k8s

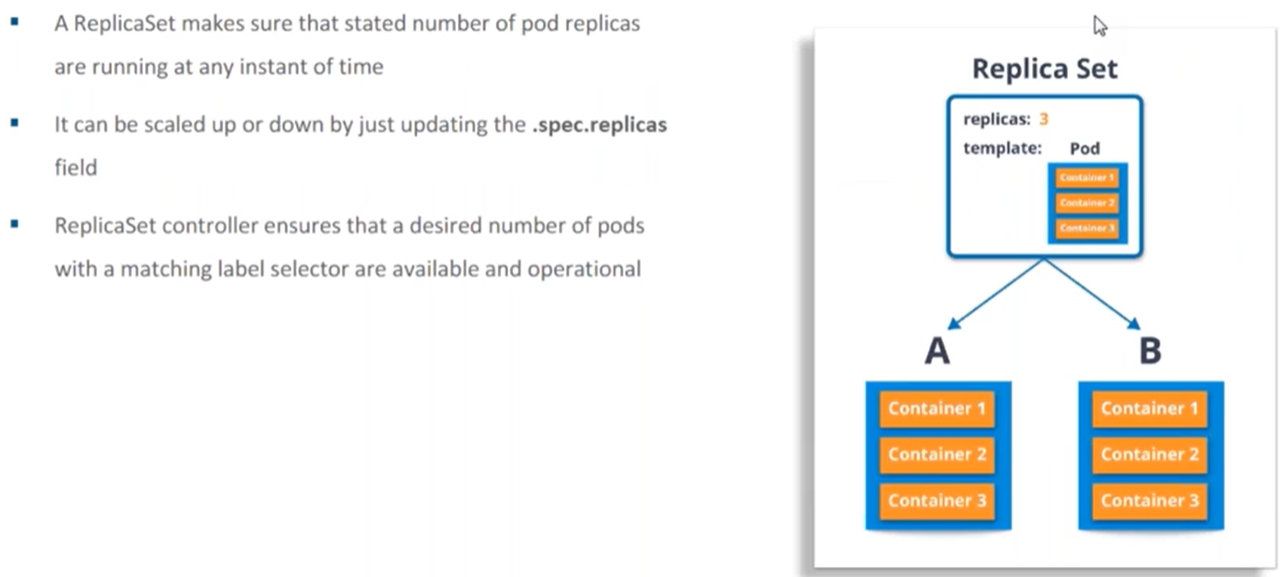


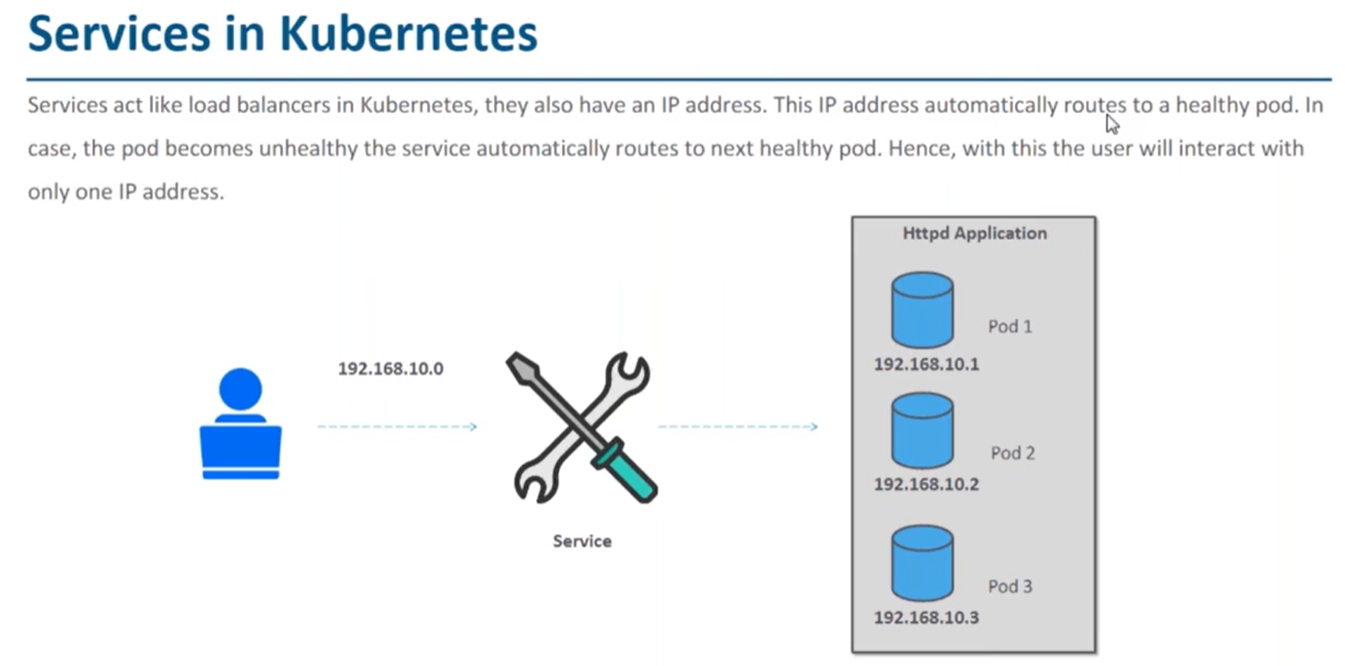




Deployment of apps:







How to create service

Kubectl create service nodeort httpd –tcp=80:80

How to see them

Kubectl get svc httpd

Dashboard

