**CONFIGMAP AND SECRETES**

**# vi cm.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

  name: test-cm

data:

  db-port: "3306"

root@vm:/home/azadmin# kubectl apply -f cm.yaml

configmap/test-cm created

root@vm:/home/azadmin#

root@VM01:/home/azadmin# kubectl describe cm test-cm

Name: test-cm

Namespace: default

Labels: <none>

Annotations: <none>

Data

====

db-port:

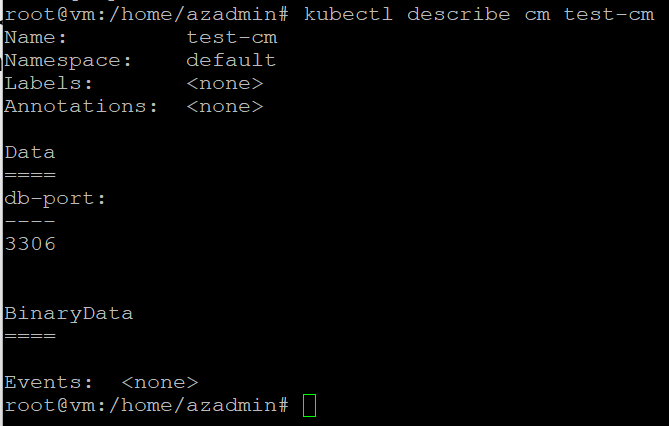
----

3306

BinaryData

====

Events: <none>



# vi deployment.yaml  
  
apiVersion: apps/v1

kind: Deployment

metadata:

name: myapp1-deployment

spec:

replicas: 2

selector:

matchLabels:

app: myapp1

template:

metadata: # Dictionary

name: myapp1-pod

labels: # Dictionary

app: myapp1

spec:

containers: # List

- name: myapp1-container

image: kuchalakantikris/my\_nginx\_image1:v1

ports:

- containerPort: 80

# kubectl get pods

# kubectl exec -it myapp1-deployment-79644dcdf8-8qp28 -- /bin/bash

myapp1-deployment-79644dcdf8-8qp28# env | grep DB

# vi deployment.yaml

spec:

replicas: 2

selector:

matchLabels:

app: myapp1

template:

metadata: # Dictionary

name: myapp1-pod

labels: # Dictionary

app: myapp1

spec:

containers: # List

- name: myapp1-container

image: kuchalakantikris/my\_nginx\_image1:v1

env:

- name: DB-PORT

valueFrom:

configMapKeyRef:

name: test-cm

key: db-port

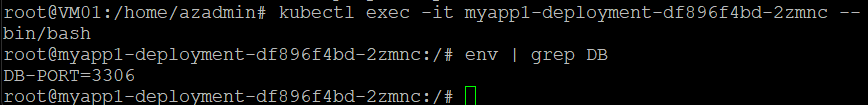
ports:

- containerPort: 80

# kubectl get pods

root@VM01:/home/azadmin# kubectl exec -it myapp1-deployment-df896f4bd-2zmnc -- /bin/bash  
root@myapp1-deployment-df896f4bd-2zmnc:/# env | grep DB

DB-PORT=3306



Now in case DB admin has changed DB port has changed and if you change in config map, how pods will get to know ?

Changing env variable in pods is not possible

You cannot change env value inside the container, as it will not allow, you have to recreate the container.

Solution is => we will use Volume Mounts

Using volume mounts, instead of using them as ENV will use as files, as we are mounting

apiVersion: apps/v1

kind: Deployment

metadata:

name: myapp1-deployment

spec:

replicas: 2

selector:

matchLabels:

app: myapp1

template:

metadata: # Dictionary

name: myapp1-pod

labels: # Dictionary

app: myapp1

spec:

containers: # List

- name: myapp1-container

image: kuchalakantikris/my\_nginx\_image1:v1

volumeMounts:

- name: db-connection

mountPath: /opt

ports:

- containerPort: 80

volumes:

- name: db-connection

configMap:

name: test-cm

# kubectl apply -f deployment.yaml

# kubectl get pods

# kubectl exec -it myapp1-deployment-cf5f5b78c-nhg9f -- /bin/bash

# root@myapp1-deployment--cf5f5b78c-nhg9f:/# env | grep DB

***There will be no environmental variable***

#root@myapp1-deployment-cf5f5b78c-nhg9f:/# ls /opt

db-port

root@myapp1-deployment-cf5f5b78c-nhg9f:/# cat /opt/db-port | more

now change port in cm-yaml

root@VM01:/home/azadmin# vi cm.yml

apiVersion: v1

kind: ConfigMap

metadata:

name: test-cm

data:

db-port: "3307"

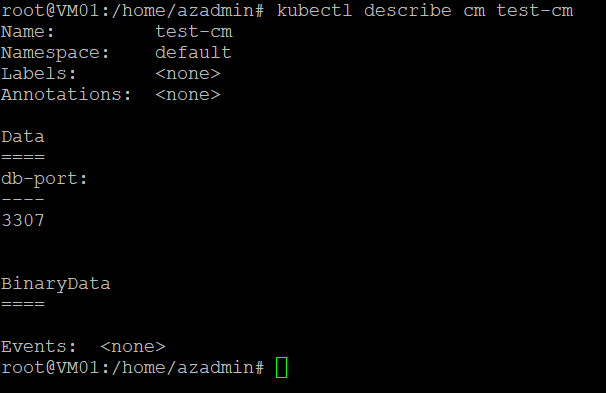
root@VM01:/home/azadmin# kubectl apply -f cm.yml

configmap/test-cm configured

root@VM01:/home/azadmin#

without restarting pod, pod should change the db port

# kubectl describe cm test-cm



kubectl exec -it myapp1-deployment-cf5f5b78c-nhg9f -- /bin/bash

cat /opt/db-port | more

or

kubectl exec -it myapp1-deployment-cf5f5b78c-nhg9f -- cat /opt/db-port

**Secret:**

kubectl create secret generic test-secret --from-literal=db-port="3306"

root@VM01:/home/azadmin# kubectl create secret generic test-secret --from-literal=db-port="3306"

secret/test-secret created  
  
root@VM01:/home/azadmin# kubectl describe secret test-secret

Name: test-secret

Namespace: default

Labels: <none>

Annotations: <none>

Type: Opaque

Data

====

db-port: 4 bytes  
  
#kubectl edit secret test-secret

apiVersion: v1

data:

db-port: MzMwNg==

kind: Secret

metadata:

creationTimestamp: "2025-01-02T23:33:59Z"

name: test-secret

namespace: default

resourceVersion: "12146"

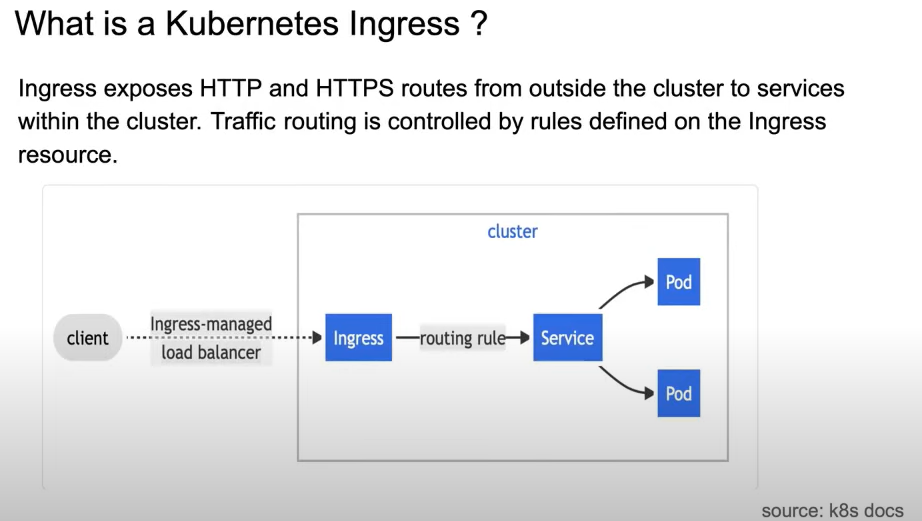
uid: 15e6820b-5244-4fc6-9ae4-7d6f3236b290

type: Opaque

root@VM01:/home/azadmin# echo MzMwNg== | base64 --decode | more

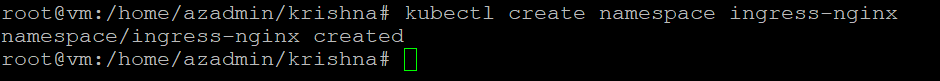
3306

**KUBERENTES INGRESS**

****

Step1: Create Namespace

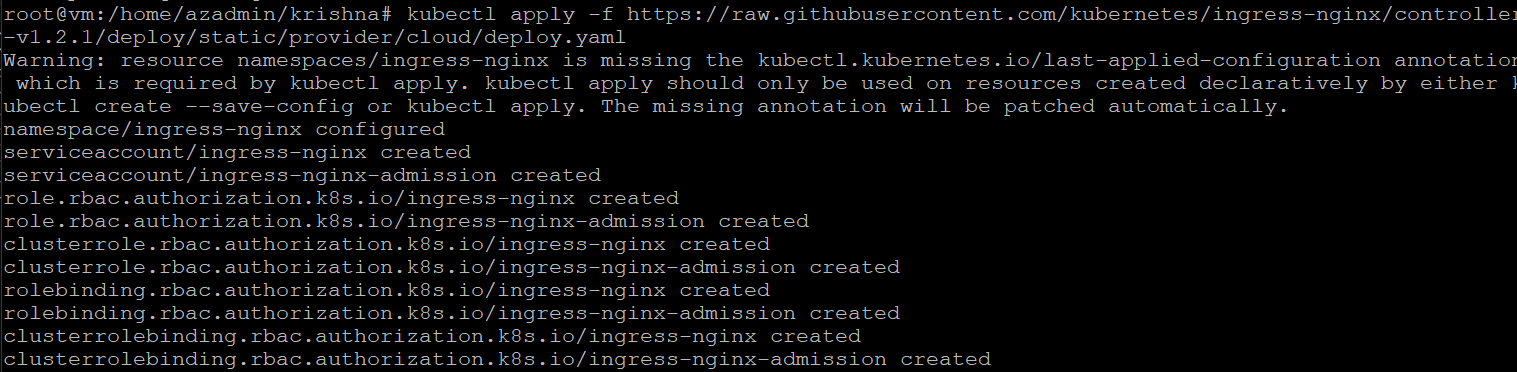
# kubectl create namespace ingress-nginx



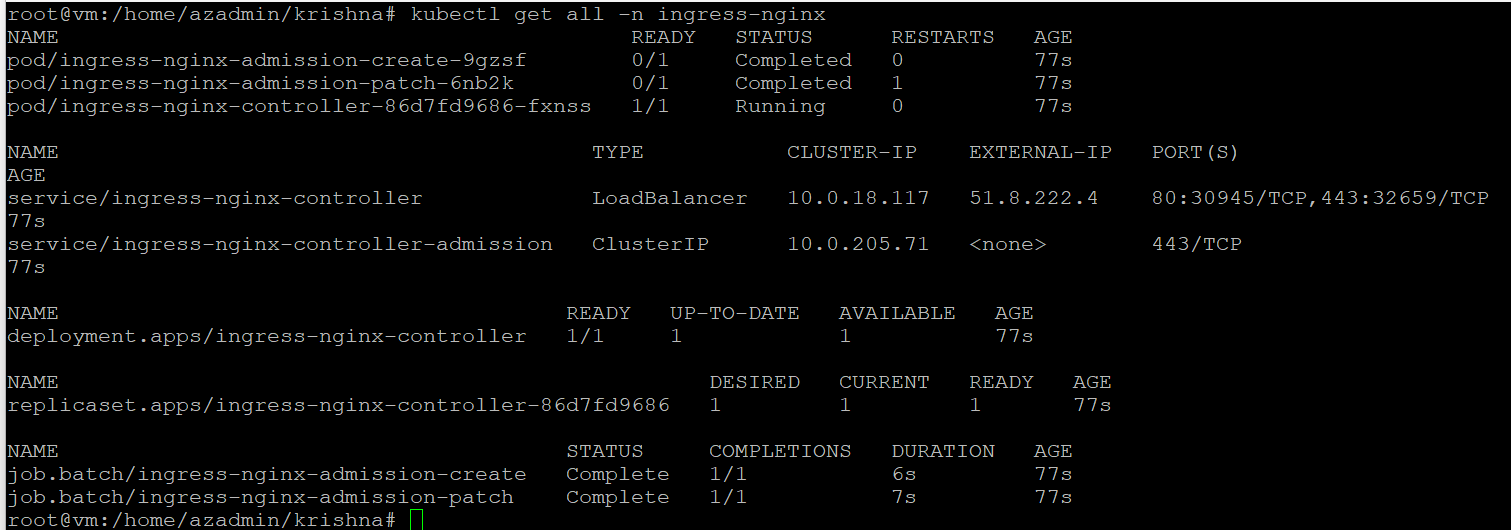
Now create ingress resource

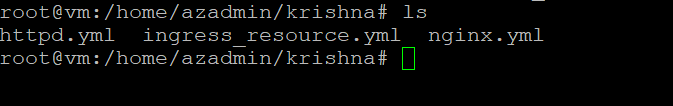
<https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.2.1/deploy/static/provider/cloud/deploy.yaml>

kubectl apply -f <aboveurl>

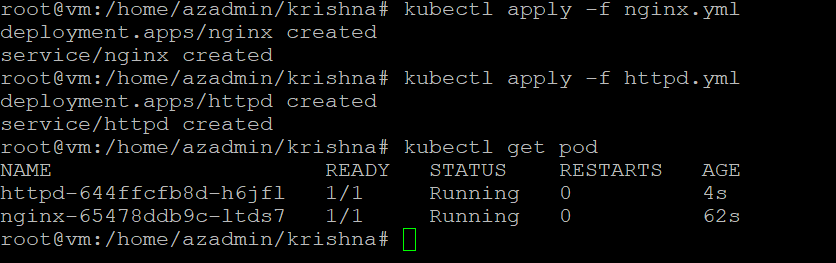


# kubectl get all -n ingress-nginx





Now deploy nginx , http and ingress yaml



Now we need to install ingress resource as we have already created ingress controller above

root@vm:/home/azadmin/krishna# kubectl apply -f ingress\_resource.yml

ingress.networking.k8s.io/k8s-ingress created

root@vm:/home/azadmin/krishna# kubectl get ingress

NAME CLASS HOSTS ADDRESS PORTS AGE

k8s-ingress nginx \* 80 8s

root@vm:/home/azadmin/krishna# kubectl get ingress

NAME CLASS HOSTS ADDRESS PORTS AGE

k8s-ingress nginx \* 51.8.222.4 80 40s

