**SESSION - 63**

**-->** Volume is most import in the kubernates

--> Volumes should be outside of cluster

--> Outside of cluster means marked node or worked node it should not store.

--> We have options in AWS -- EFS,EBS

--> it can be static or EBS AND EFS.

**EBS vs EFS static**

**================**

1. Install drivers

2. Give permissions in EC2 role

3. create volume

4. create PV(physical representation of volume)

5. create PVC

6. volume mount to pod

if EBS volume should be in the same az as in instance

if EFS SG should allow port 2049

**EBS of EFS dynamic**

**=================**

1. Install drivers

2. Give permissions in EC2 role

3. create storage class

4. create PVC with SC name, volume and PV will be created automatically

5. volume mount to pod

in case of dynamic pod pvc creates volume, so it creates in the same az where ec2 instance is there

if EFS SG should allow port 2049

**Deployment vs StatefulSet**

**===========================**

1. Deployment is for stateless applications like frontend and backend

2. StatefulSet is for DB related applications like MongoDB, MySQL, Redis, RabbitMQ, Prometheus, Grafana, ELK, etc.

3. Deployment pods can share PV and PVC, but statefulset each pod creates its own volume.

4. StatefulSet needs headless service i.e clusterIP none, because in DB environments pods should find other pods for data sync

5. Deployment pods not follows orderly manner, statefulset pods follows orderly manner. it creates <statefulset-name>-0 <statefulset-name>-1....

6. pods in statefulset keeps their identity.

**why I took nginx as example in statefulset, even it is frontend server?**

Ans)) just for prove simple example to show the data exists

--> this is heavy configuration

--> no one will keep end user in kubernates.

--> all databases we have set in our stateful set.

--> same like deployment need to create databases

**--> cd /c/devops/daws-84s/repos/docker**

**--> terraform apply -auto-approve**

**--> sudo usermod -aG docker ec2-user**

**--> aws configure**

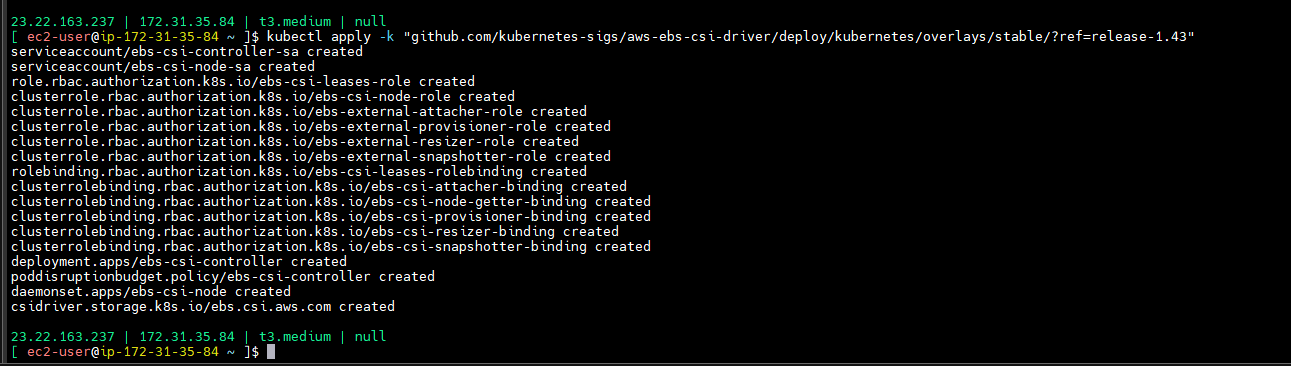
**--> git clone <https://github.com/Shan23-hash/docker.git>**

**--> eksctl create cluster --config-file=eks.yaml**

**--> kubectl get nodes**

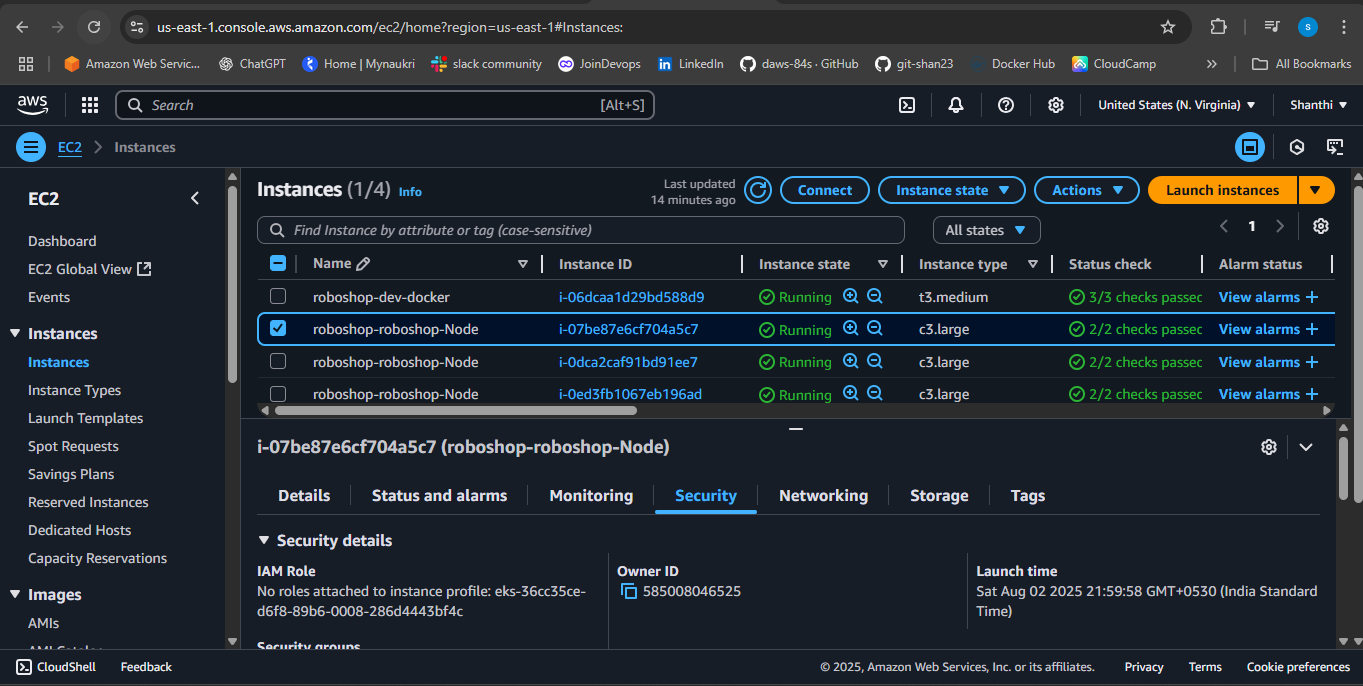
--> This is EBS

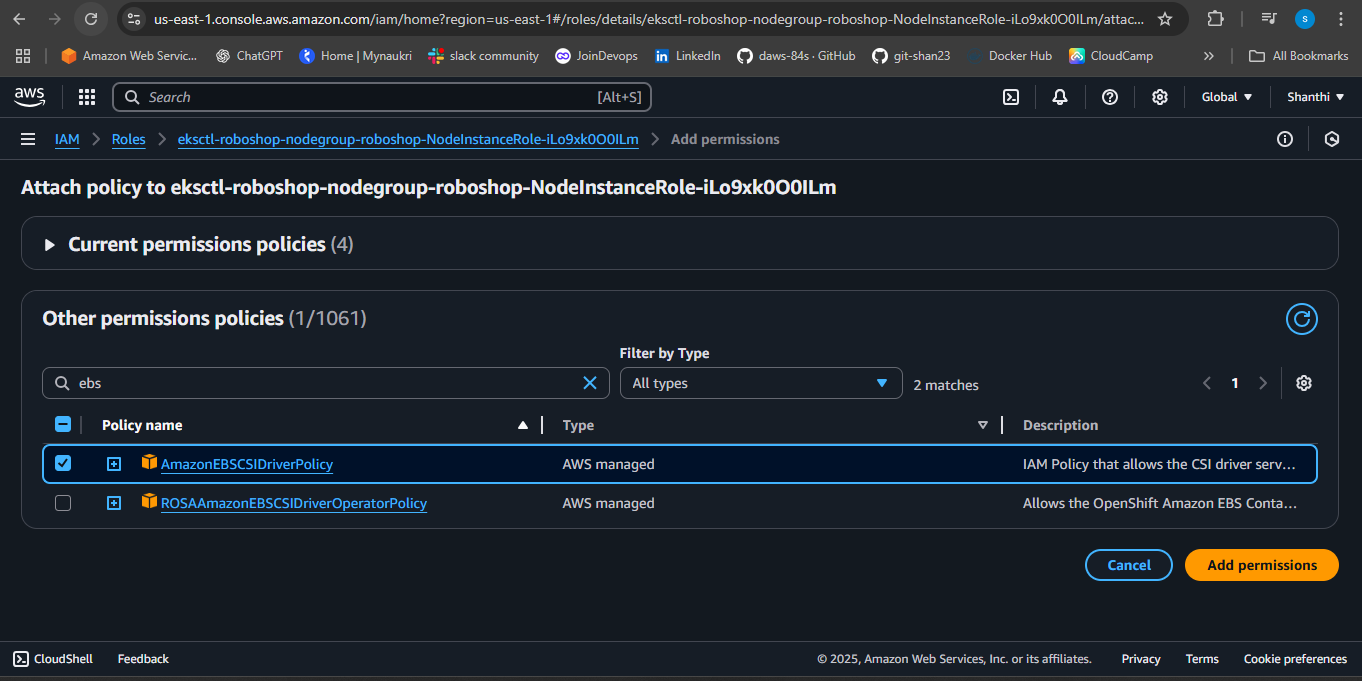
**kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-1.43"**



--> In instance need to add the permissions.

--> Take one instance --> Go to security --> open IAM --> Go to add permissions --> search ebs (AmazonEBSCSIDriverPolicy) select this one --> add permissions.





--> Create storage class

**k8-resources/ebs-sc.yaml**

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: roboshop-ebs

reclaimPolicy: Retain

provisioner: ebs.csi.aws.com

volumeBindingMode: WaitForFirstConsumer # disk will be created when pod is getting created

--> Create k8-databases repo

--> push the code

**mongodb/manifest.yaml**

---

apiVersion: v1

kind: Service

metadata:

name: mongodb-headless

namespace: roboshop

labels:

component: mongodb

project: roboshop

tier: database

spec:

clusterIP: None

selector:

component: mongodb

project: roboshop

tier: database

ports:

- protocol: TCP

port: 27017 # service port

targetPort: 27017 # container port

---

apiVersion: v1

kind: Service

metadata:

name: mongodb

namespace: roboshop

labels:

component: mongodb

project: roboshop

tier: database

spec:

selector:

component: mongodb

project: roboshop

tier: database

ports:

- protocol: TCP

port: 27017 # service port

targetPort: 27017 # container port

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: mongodb

namespace: roboshop

labels:

component: mongodb

project: roboshop

tier: database

spec:

selector:

matchLabels:

component: mongodb

project: roboshop

tier: database # has to match .spec.template.metadata.labels

serviceName: "mongodb-headless" # this should be headless service

replicas: 3 # by default is 1

template:

metadata:

labels:

component: mongodb

project: roboshop

tier: database

spec:

containers:

- name: mongodb

image: shan2324/mongodb:v1

volumeMounts:

- name: mongodb-data

mountPath: /data/db

# This is nothing but PVC, you can directly create here

volumeClaimTemplates:

- metadata:

name: mongodb-data

spec:

accessModes: [ "ReadWriteOnce" ]

storageClassName: "roboshop-ebs"

resources:

requests:

storage: 2Gi

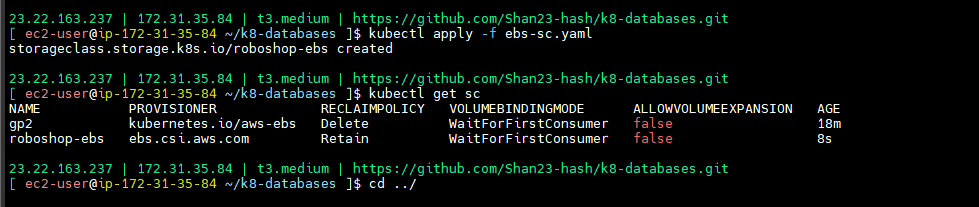
--> push and pull the code

--> **git clone https://github.com/Shan23-hash/k8-databases.git**

--> **cd /k8-databases**

**--> kubectl apply -f ebs-sc.yaml**

**--> kubectl get sc**



**01-namespace.yaml**

apiVersion: v1

kind: Namespace

metadata:

name: roboshop

labels:

project: roboshop

environment: dev

--> Push and Pull the code

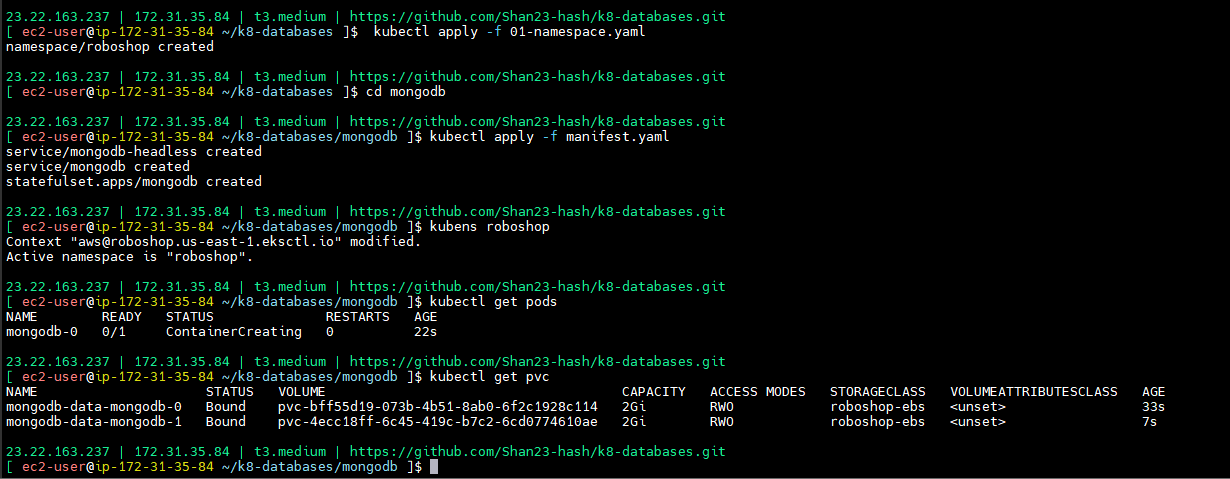
**--> kubectl apply -f 01-namespace.yaml**

**--> cd /mongodb**

**--> kubectl apply -f manifest.yaml**

**--> kubens roboshop**

**--> kubectl get pods**



**-->** given three replicas it’s creating one by one

**--> kubectl get pvc**

**--> kubectl get pods**

**--> kubectl delete pod mongodb-0**

**--> kubectl get pods**

--> why after deleting also creating because use names pv and pvc will attach one to one. If 3 deleted also again same will create.

**REDIS**

**redis/manifest.yaml**

apiVersion: v1

kind: Service

metadata:

name: redis-headless

namespace: roboshop

labels:

component: redis

project: roboshop

tier: database

spec:

clusterIP: None

selector:

component: redis

project: roboshop

tier: database

ports:

- protocol: TCP

port: 6379 # service port

targetPort: 6379 # container port

---

apiVersion: v1

kind: Service

metadata:

name: redis

namespace: roboshop

labels:

component: redis

project: roboshop

tier: database

spec:

selector:

component: redis

project: roboshop

tier: database

ports:

- protocol: TCP

port: 6379 # service port

targetPort: 6379 # container port

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: redis

namespace: roboshop

labels:

component: redis

project: roboshop

tier: database

spec:

selector:

matchLabels:

component: redis

project: roboshop

tier: database # has to match .spec.template.metadata.labels

serviceName: "redis-headless" # this should be headless service

replicas: 2 # by default is 1

template:

metadata:

labels:

component: redis

project: roboshop

tier: database

spec:

containers:

- name: redis

image: redis:7.0

volumeMounts:

- name: redis-data

mountPath: /data

# This is nothing but PVC, you can directly create here

volumeClaimTemplates:

- metadata:

name: redis-data

spec:

accessModes: [ "ReadWriteOnce" ]

storageClassName: "roboshop-ebs"

resources:

requests:

storage: 2Gi

--> keep services on first,create headless service

--> replace deployment stateful set.

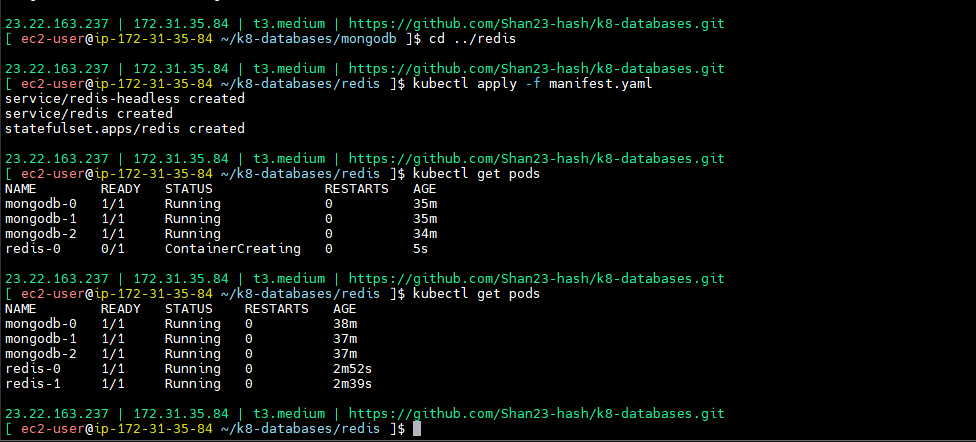
--> here I’m giving two replicas

--> push and pull the code

**--> cd ../redis**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



**MYSQL**

**mysql/manifest.yaml**

apiVersion: v1

kind: Secret

metadata:

name: mysql

labels:

component: mysql

project: roboshop

tier: database

type: Opaque

data:

MYSQL\_ROOT\_PASSWORD: "Um9ib1Nob3BAMQ=="

---

apiVersion: v1

kind: Service

metadata:

name: mysql-headless

namespace: roboshop

labels:

component: mysql

project: roboshop

tier: database

spec:

clusterIP: None

selector:

component: mysql

project: roboshop

tier: database

ports:

- protocol: TCP

port: 3306 # service port

targetPort: 3306 # container port

---

apiVersion: v1

kind: Service

metadata:

name: mysql

namespace: roboshop

labels:

component: mysql

project: roboshop

tier: database

spec:

selector:

component: mysql

project: roboshop

tier: database

ports:

- protocol: TCP

port: 3306 # service port

targetPort: 3306 # container port

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: mysql

namespace: roboshop

labels:

component: mysql

project: roboshop

tier: database

spec:

selector:

matchLabels:

component: mysql

project: roboshop

tier: database # has to match .spec.template.metadata.labels

serviceName: "mysql-headless" # this should be headless service

replicas: 2 # by default is 1

template:

metadata:

labels:

component: mysql

project: roboshop

tier: database

spec:

containers:

- name: mysql

image: shan2324/mysql:v1

envFrom:

- secretRef:

name: mysql

volumeMounts:

- name: mysql-data

mountPath: /var/lib/mysql

# This is nothing but PVC, you can directly create here

volumeClaimTemplates:

- metadata:

name: mysql-data

spec:

accessModes: [ "ReadWriteOnce" ]

storageClassName: "roboshop-ebs"

resources:

requests:

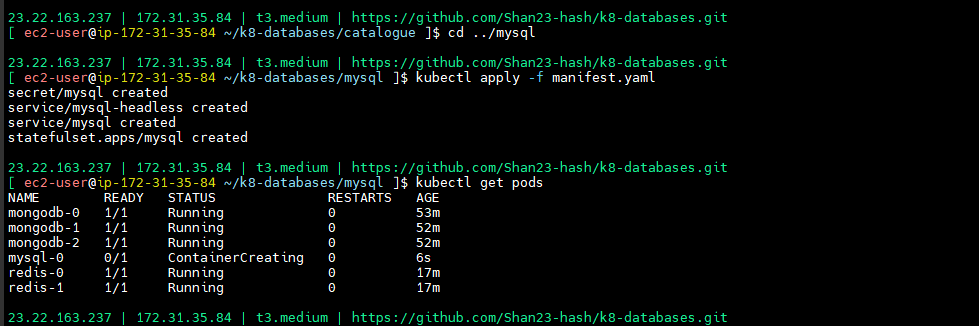
storage: 2Gi

--> push and pull the code

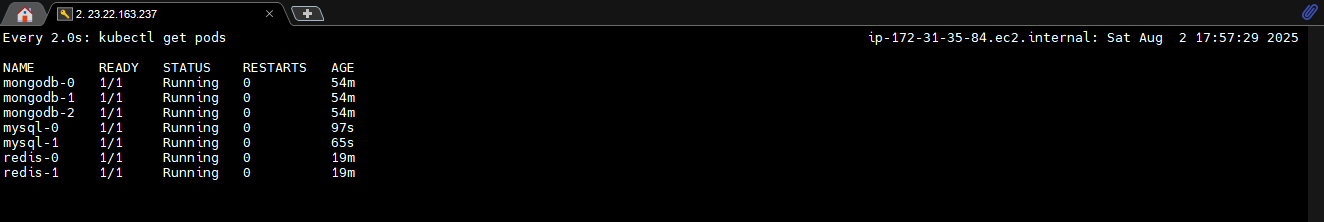
**--> cd ../mysql**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



**--> watch kubectl get pods**



**RABBITMQ**

**rabbitmq/manifest.yaml**

apiVersion: v1

kind: Secret

metadata:

name: rabbitmq

labels:

component: rabbitmq

project: roboshop

tier: database

type: Opaque

data:

RABBITMQ\_DEFAULT\_USER: "cm9ib3Nob3A="

RABBITMQ\_DEFAULT\_PASS: "cm9ib3Nob3AxMjM="

---

apiVersion: v1

kind: Service

metadata:

name: rabbitmq-headless

namespace: roboshop

labels:

component: rabbitmq

project: roboshop

tier: database

spec:

clusterIP: None

selector:

component: rabbitmq

project: roboshop

tier: database

ports:

- protocol: TCP

port: 5672 # service port

targetPort: 5672 # container port

---

apiVersion: v1

kind: Service

metadata:

name: rabbitmq

namespace: roboshop

labels:

component: rabbitmq

project: roboshop

tier: database

spec:

selector:

component: rabbitmq

project: roboshop

tier: database

ports:

- protocol: TCP

port: 5672 # service port

targetPort: 5672 # container port

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: rabbitmq

namespace: roboshop

labels:

component: rabbitmq

project: roboshop

tier: database

spec:

selector:

matchLabels:

component: rabbitmq

project: roboshop

tier: database # has to match .spec.template.metadata.labels

serviceName: "rabbitmq-headless" # this should be headless service

replicas: 2 # by default is 1

template:

metadata:

labels:

component: rabbitmq

project: roboshop

tier: database

spec:

containers:

- name: rabbitmq

image: rabbitmq:3

envFrom:

- secretRef:

name: rabbitmq

volumeMounts:

- name: rabbitmq-data

mountPath: /var/lib/rabbitmq

# This is nothing but PVC, you can directly create here

volumeClaimTemplates:

- metadata:

name: rabbitmq-data

spec:

accessModes: [ "ReadWriteOnce" ]

storageClassName: "roboshop-ebs"

resources:

requests:

storage: 2Gi

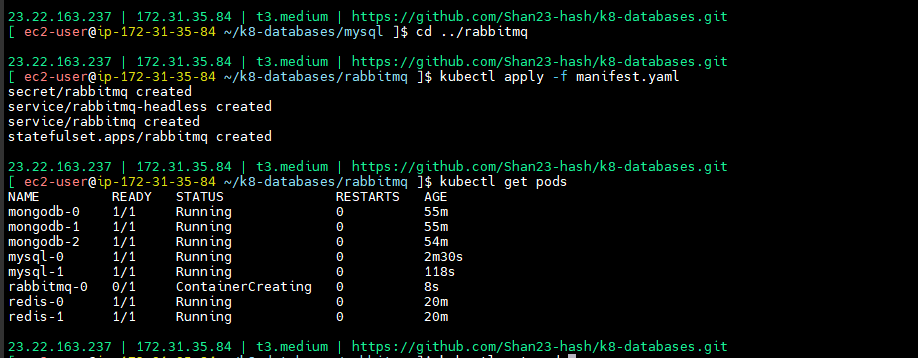
--> change port number

--> push and pull the code

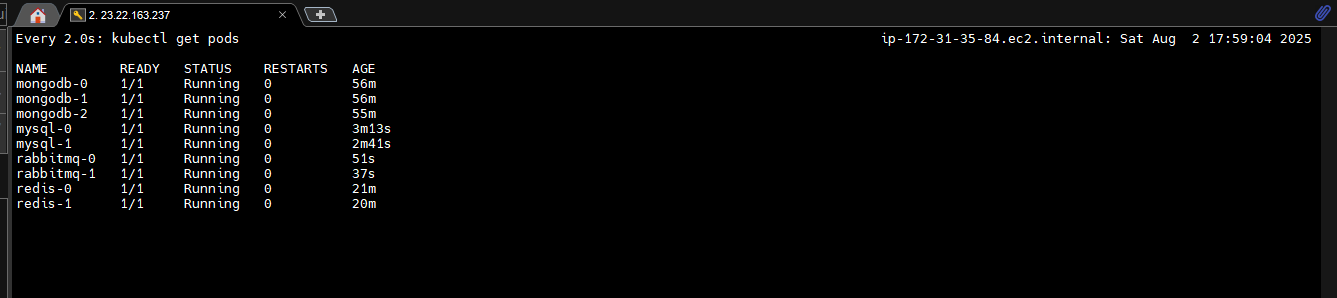
**--> cd ../rabbitmq**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



**--> watch kubectl get pods**



**-->** see now regarding roboshop volumes

--> through mongo

--> if you kept mongo gb in kubernates yiu have run in cluster mode.

--> We don’t have problem

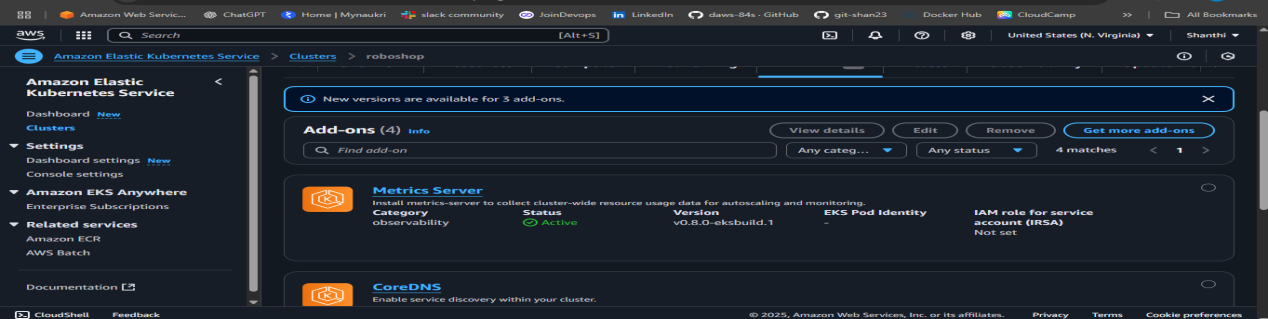
--> real e-coomerce project

--> Data is not in perfect sync

--> Users there in mongodb

--> you can run in cluster mode

--> this is only for demo session purpose.



**horizontal autoscaling vs vertical autoscaling**

**==============================================**

--> hirizontal autoscaling it’s incresing the replicas.

--> vertical autoscaling means incresing the resources.

--> single point of failure there vertical autoscaling

--> horizontal autoscaling is safe.

Cpu utilisation

1. make sure metrics server is running

2. we should mention resources requests and limits

current usage

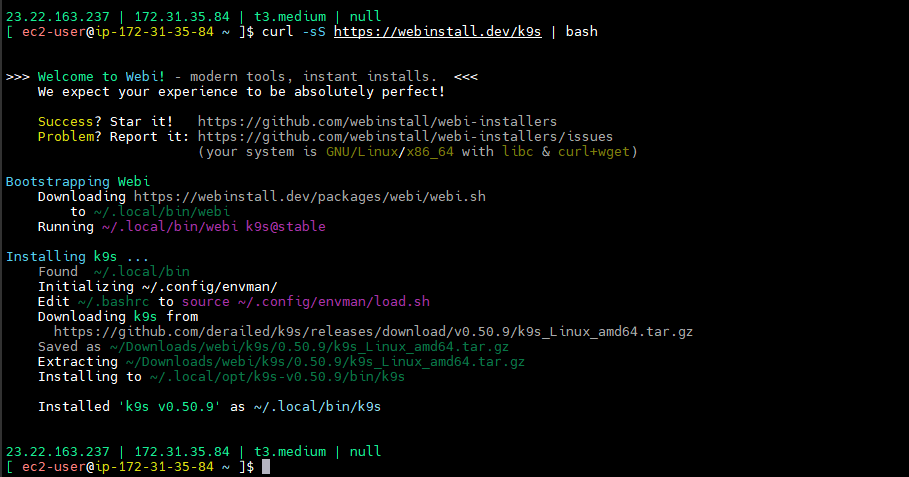
=============X100

max usage limit

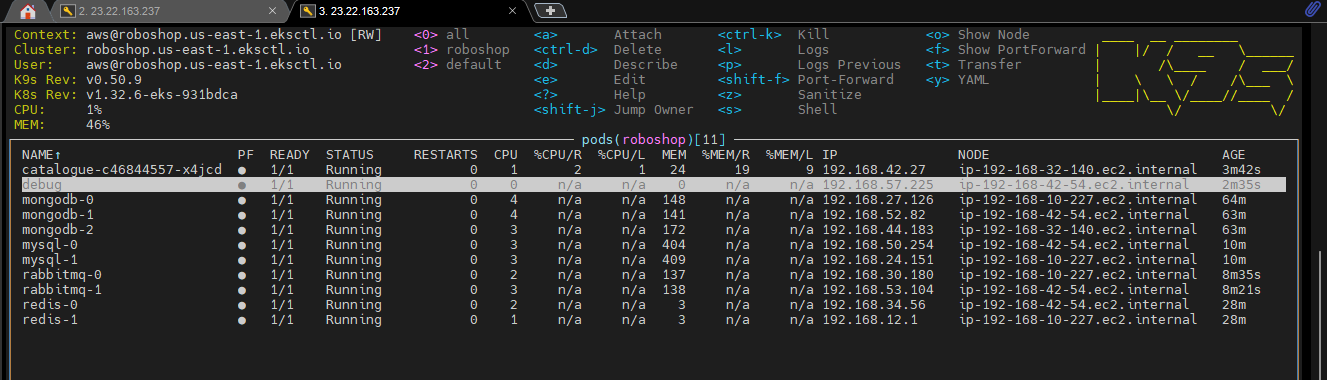
--> while creating horizontal autoscaling you should have something to measure how much percanrage is used in you pod nodes resources also need to mention.

--> so now use k9s - it will show percentages.

--> **curl -sS <https://webinstall.dev/k9s> | bash** -- install this one and give



--> k9s



--> Data bases scaling is different

**CATALOGUE**

**catalogue/manifest.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: catalogue

namespace: roboshop

labels:

component: catalogue

project: roboshop

tier: app

data:

MONGO: "true"

MONGO\_URL: "mongodb://mongodb:27017/catalogue"

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: catalogue

namespace: roboshop

# deployment labels

labels:

component: catalogue

project: roboshop

tier: app

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: catalogue

project: roboshop

tier: app

# This is pod definition

template:

metadata:

labels:

component: catalogue

project: roboshop

tier: app

spec:

containers:

- name: catalogue

image: joindevops/catalogue:v1

resources:

requests:

cpu: "50m"

memory: "128Mi"

limits:

cpu: "100m"

memory: "256Mi"

envFrom:

- configMapRef:

name: catalogue

---

apiVersion: v1

kind: Service

metadata:

name: catalogue

namespace: roboshop

labels:

component: catalogue

project: roboshop

tier: app

spec:

selector:

component: catalogue

project: roboshop

tier: app

ports:

- protocol: TCP

port: 8080 # service port

targetPort: 8080 # container port

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: catalogue

namespace: roboshop

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: catalogue

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

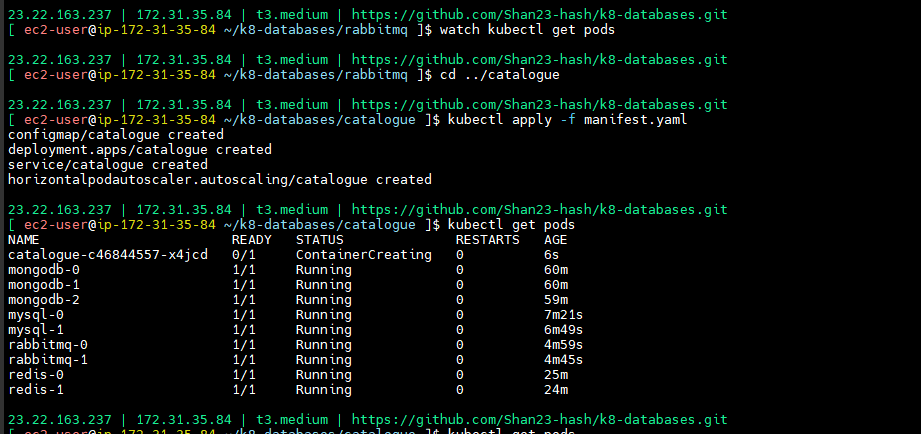
averageUtilization: 10

--> push and pull the code

**--> cd ../catalogue**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



--> go to catalogue inside and check

**DEBUG**

**debug/Dockerfile**

FROM almalinux:9

RUN dnf install telnet net-tools iputils -y

RUN useradd roboshop

USER roboshop

CMD ["sleep","100000"]

**debug/manifest.yaml**

apiVersion: v1

kind: Pod

metadata:

name: debug

namespace: roboshop

spec:

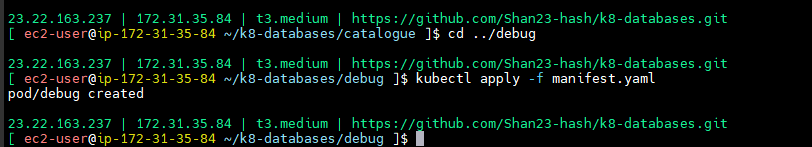
containers:

- name: debug

image: joindevops/debug:v1

--> push and pull the code

**--> cd ../debug**



**--> kubectl apply -f manifest.yaml**

**-->** select dubug and give esc+s for going to shell script

**--> curl http://catalogue:8080/health**

Now see uasge it’s showing

--> hpi reference is deplyment.

--> deployment will watch continously

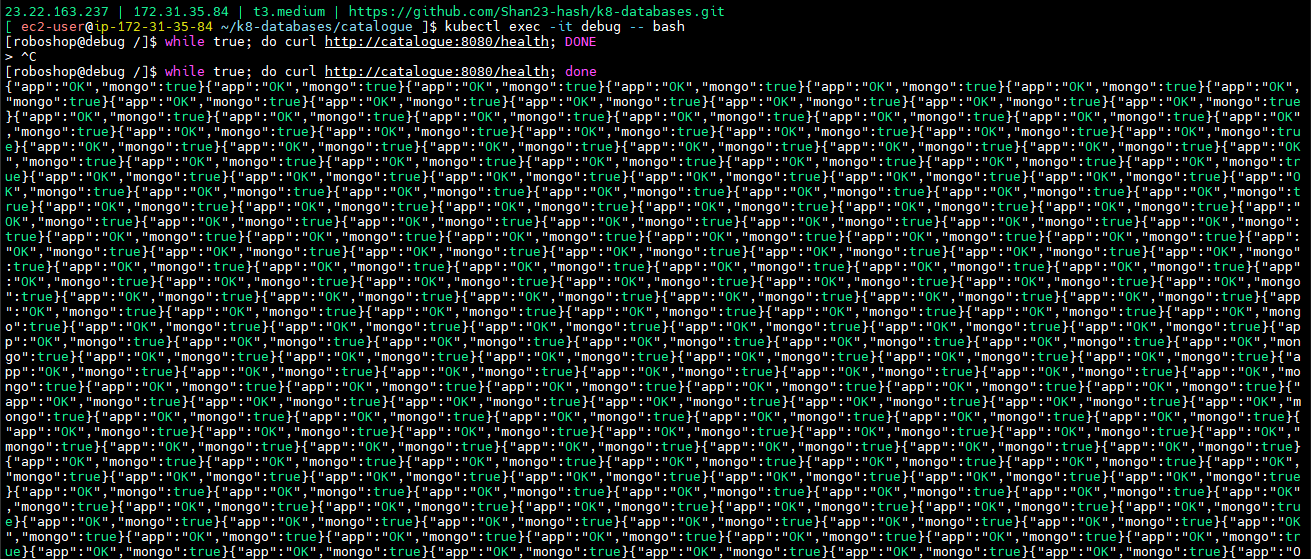
--> increase the load

--> go to debug pod

--> **cd ../catalogue**

**--> kubectl exec -it debug -- bash**

**--> while true; do curl <http://catalogue>:8080/health; done**



**-->** contionously it will send output

--> pods it’s get increased automatically

--> maximum 10 it’s reached

--> exit out then it will decrease load.

--> pods will delete one by one

**CART**

**cart/manifest.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: cart

namespace: roboshop

labels:

component: cart

project: roboshop

tier: app

data:

REDIS\_HOST: "redis"

CATALOGUE\_HOST: "catalogue"

CATALOGUE\_PORT: "8080"

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: cart

namespace: roboshop

# deployment labels

labels:

component: cart

project: roboshop

tier: app

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: cart

project: roboshop

tier: app

# This is pod definition

template:

metadata:

labels:

component: cart

project: roboshop

tier: app

spec:

containers:

- name: cart

image: shan2324/cart:v1

resources:

requests:

cpu: "50m"

memory: "128Mi"

limits:

cpu: "100m"

memory: "256Mi"

envFrom:

- configMapRef:

name: cart

---

apiVersion: v1

kind: Service

metadata:

name: cart

namespace: roboshop

labels:

component: cart

project: roboshop

tier: app

spec:

selector:

component: cart

project: roboshop

tier: app

ports:

- protocol: TCP

port: 8080 # service port

targetPort: 8080 # container port

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: cart

namespace: roboshop

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: cart

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

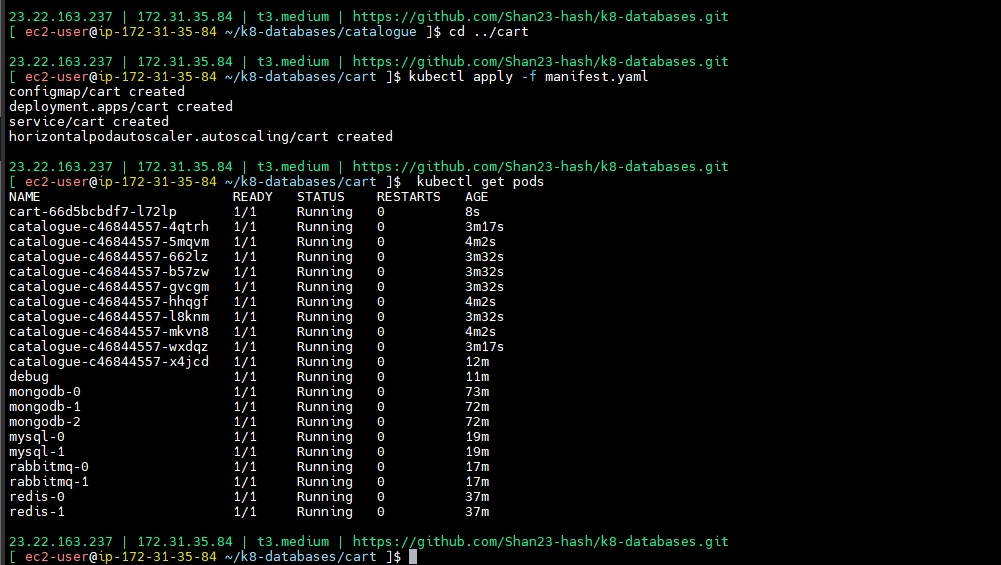
averageUtilization: 10

--> push and pull the code

**--> cd ../cart**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



**USER**

**user/manifest.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: user

namespace: roboshop

labels:

component: user

project: roboshop

tier: app

data:

MONGO\_URL: "mongodb://mongodb:27017/users"

REDIS\_URL: "redis://redis:6379"

MONGO: "true"

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: user

namespace: roboshop

# deployment labels

labels:

component: user

project: roboshop

tier: app

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: user

project: roboshop

tier: app

# This is pod definition

template:

metadata:

labels:

component: user

project: roboshop

tier: app

spec:

containers:

- name: user

image: shan2324/user:v1

resources:

requests:

cpu: "50m"

memory: "128Mi"

limits:

cpu: "100m"

memory: "256Mi"

envFrom:

- configMapRef:

name: user

---

apiVersion: v1

kind: Service

metadata:

name: user

namespace: roboshop

labels:

component: user

project: roboshop

tier: app

spec:

selector:

component: user

project: roboshop

tier: app

ports:

- protocol: TCP

port: 8080 # service port

targetPort: 8080 # container port

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: user

namespace: roboshop

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: user

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

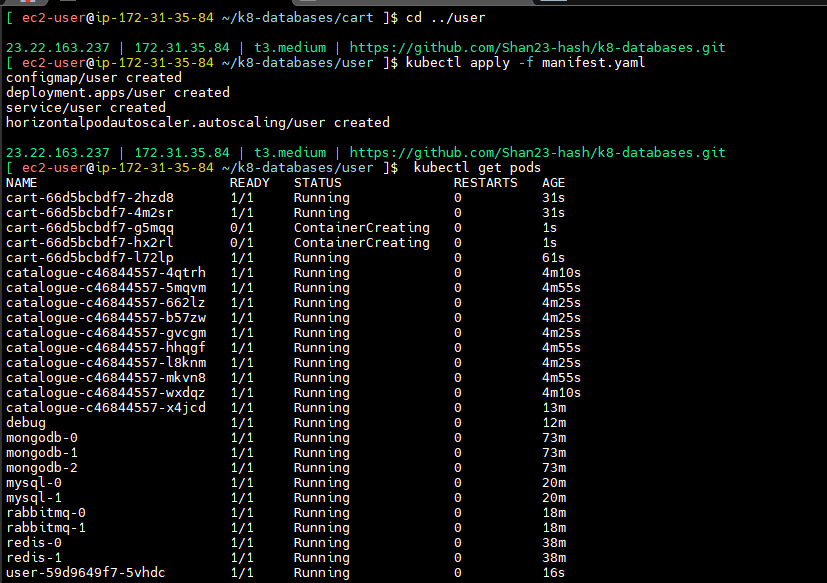
averageUtilization: 10

--> Push and pull the code

**--> cd ../user**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



**SHIPPING**

**shipping/manifest.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: shipping

namespace: roboshop

labels:

component: shipping

project: roboshop

tier: app

data:

CART\_ENDPOINT: "cart:8080"

DB\_HOST: "mysql"

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: shipping

namespace: roboshop

# deployment labels

labels:

component: shipping

project: roboshop

tier: app

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: shipping

project: roboshop

tier: app

# This is pod definition

template:

metadata:

labels:

component: shipping

project: roboshop

tier: app

spec:

containers:

- name: shipping

image: joindevops/shipping:v1

resources:

requests:

cpu: "100m"

memory: "256Mi"

limits:

cpu: "200m"

memory: "512Mi"

envFrom:

- configMapRef:

name: shipping

---

apiVersion: v1

kind: Service

metadata:

name: shipping

namespace: roboshop

labels:

component: shipping

project: roboshop

tier: app

spec:

selector:

component: shipping

project: roboshop

tier: app

ports:

- protocol: TCP

port: 8080 # service port

targetPort: 8080 # container port

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: shipping

namespace: roboshop

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: shipping

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 10

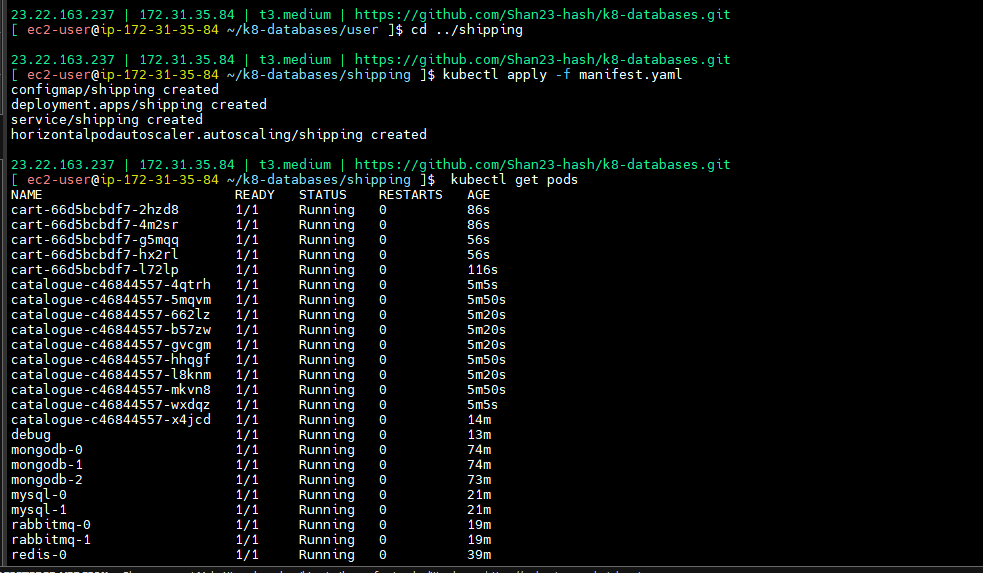
--> shipping java replated application need to give extra gb

--> push and pull the code

--> **cd ../shipping**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



**PAYMENT**

**payment/manifest.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: payment

namespace: roboshop

labels:

component: payment

project: roboshop

tier: app

data:

CART\_HOST: "cart"

CART\_PORT: "8080"

USER\_HOST: "user"

USER\_PORT: "8080"

AMQP\_HOST: "rabbitmq"

---

apiVersion: v1

kind: Secret

metadata:

name: payment

labels:

component: payment

project: roboshop

tier: app

type: Opaque

data:

AMQP\_USER: "cm9ib3Nob3A="

AMQP\_PASS: "cm9ib3Nob3AxMjM="

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: payment

namespace: roboshop

# deployment labels

labels:

component: payment

project: roboshop

tier: app

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: payment

project: roboshop

tier: app

# This is pod definition

template:

metadata:

labels:

component: payment

project: roboshop

tier: app

spec:

containers:

- name: payment

image: shan2324/payment:v1

resources:

requests:

cpu: "50m"

memory: "128Mi"

limits:

cpu: "100m"

memory: "256Mi"

envFrom:

- configMapRef:

name: payment

- secretRef:

name: payment

---

apiVersion: v1

kind: Service

metadata:

name: payment

namespace: roboshop

labels:

component: payment

project: roboshop

tier: app

spec:

selector:

component: payment

project: roboshop

tier: app

ports:

- protocol: TCP

port: 8080 # service port

targetPort: 8080 # container port

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: payment

namespace: roboshop

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: payment

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

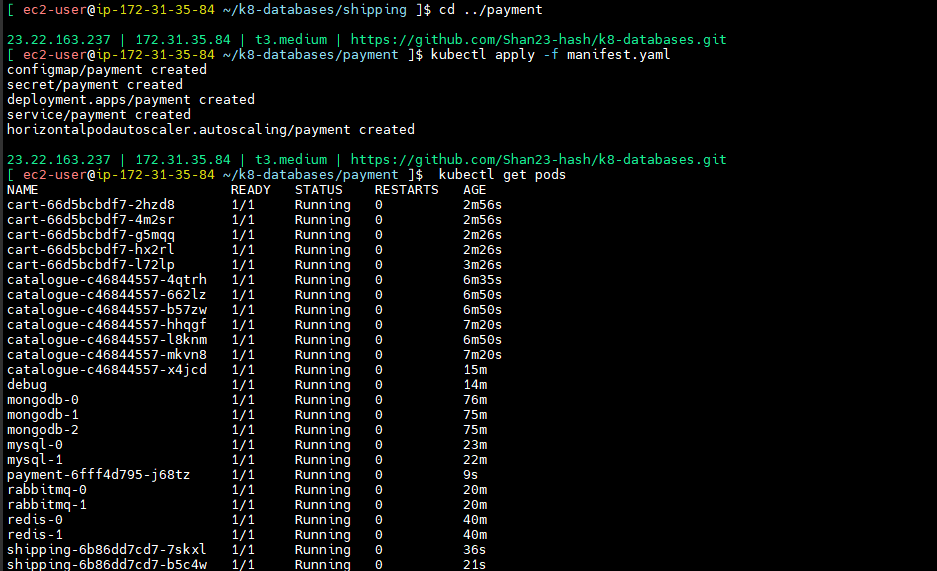
averageUtilization: 10

--> push and pull the code

--> **cd ../payment**

**--> kubectl apply -f manifest.yaml**

**--> kubectl get pods**



**FRONTEND**

**frontend/manifest.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: nginx-conf

namespace: roboshop

labels:

component: frontend

project: roboshop

tier: web

data:

nginx.conf: |

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log notice;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

keepalive\_timeout 65;

types\_hash\_max\_size 4096;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

include /etc/nginx/conf.d/\*.conf;

server {

listen 8080;

listen [::]:8080;

server\_name \_;

root /usr/share/nginx/html;

include /etc/nginx/default.d/\*.conf;

error\_page 404 /404.html;

location = /404.html {

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

}

location /images/ {

expires 5s;

root /usr/share/nginx/html;

try\_files $uri /images/placeholder.jpg;

}

location /api/catalogue/ { proxy\_pass http://catalogue:8080/; }

location /api/user/ { proxy\_pass http://user:8080/; }

location /api/cart/ { proxy\_pass http://cart:8080/; }

location /api/shipping/ { proxy\_pass http://shipping:8080/; }

location /api/payment/ { proxy\_pass http://payment:8080/; }

location /health {

stub\_status on;

access\_log off;

}

}

}

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend

namespace: roboshop

# deployment labels

labels:

component: frontend

project: roboshop

tier: web

spec:

replicas: 1

# These are the labels replica set use to create pod replicas, this should match pod labels

selector:

matchLabels:

component: frontend

project: roboshop

tier: web

# This is pod definition

template:

metadata:

labels:

component: frontend

project: roboshop

tier: web

spec:

containers:

- name: frontend

image: shan2324/frontend:v1

resources:

requests:

cpu: "50m"

memory: "128Mi"

limits:

cpu: "100m"

memory: "256Mi"

volumeMounts:

- name: nginx-conf

mountPath: /etc/nginx/nginx.conf

subPath: nginx.conf

readOnly: true

volumes:

- name: nginx-conf

configMap:

name: nginx-conf

items:

- key: nginx.conf

path: nginx.conf

---

apiVersion: v1

kind: Service

metadata:

name: frontend

namespace: roboshop

labels:

component: frontend

project: roboshop

tier: web

spec:

type: LoadBalancer

selector:

component: frontend

project: roboshop

tier: web

ports:

- protocol: TCP

port: 80 # service port

targetPort: 8080 # container port

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: frontend

namespace: roboshop

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: frontend

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 10

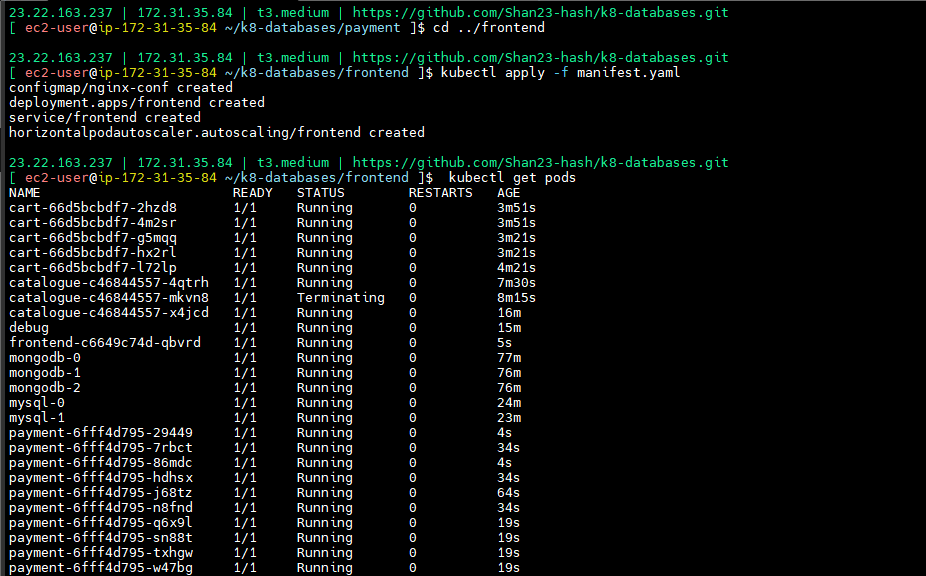
--> next horizontal pod auto scaling

--> push and pull the code

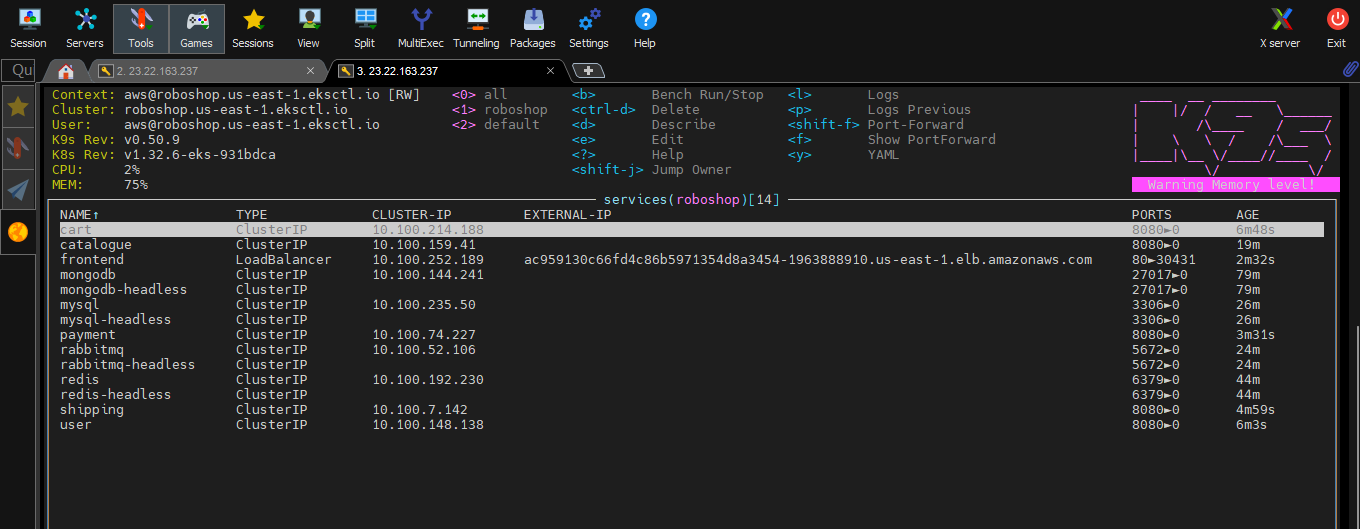
--> **cd ../frontend**

**--> kubectl apply -f manifest.yaml**

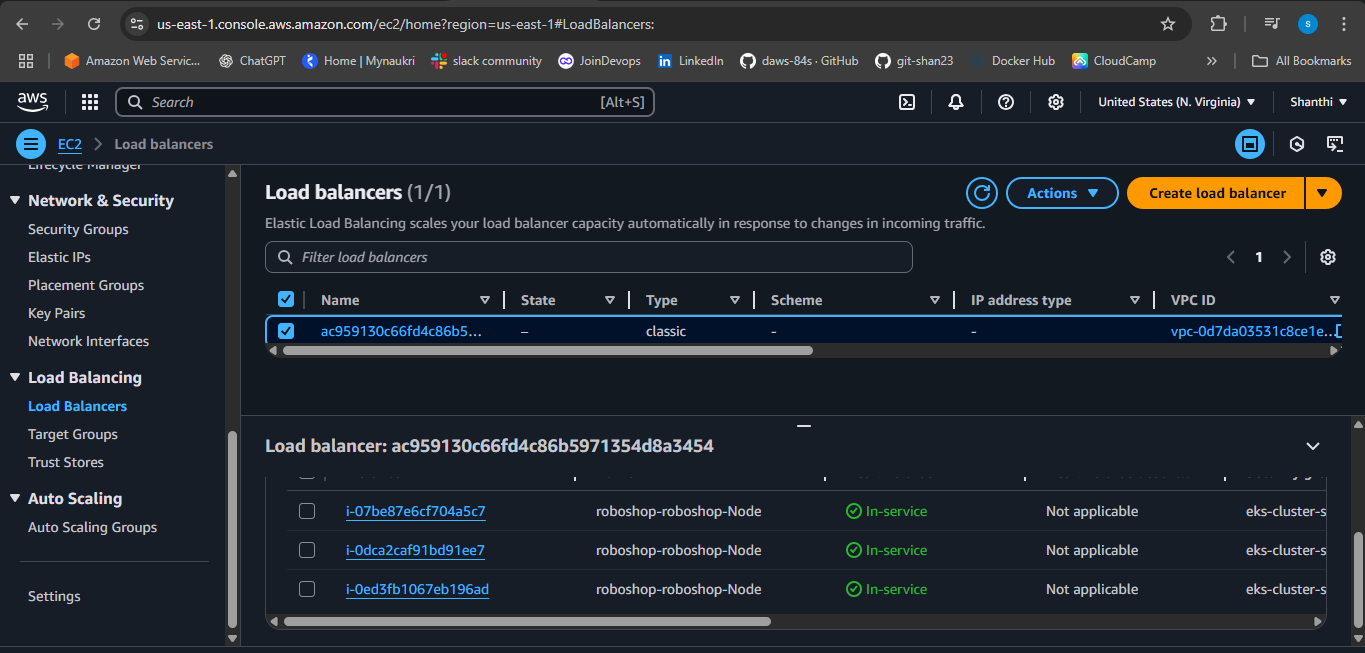
**--> kubectl get pods**



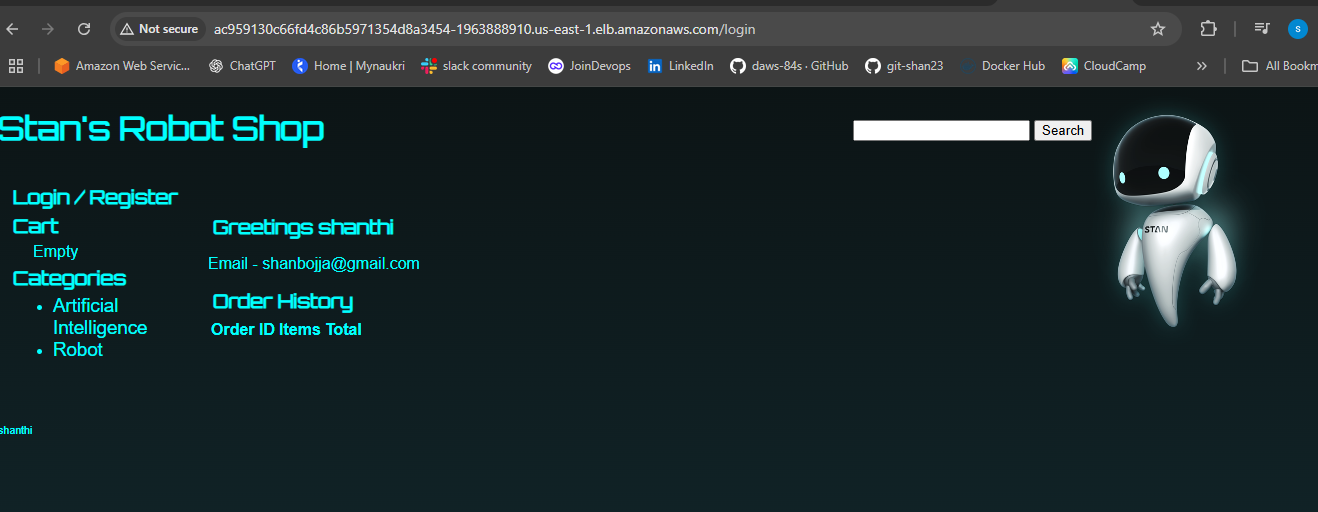
Shift+; type services

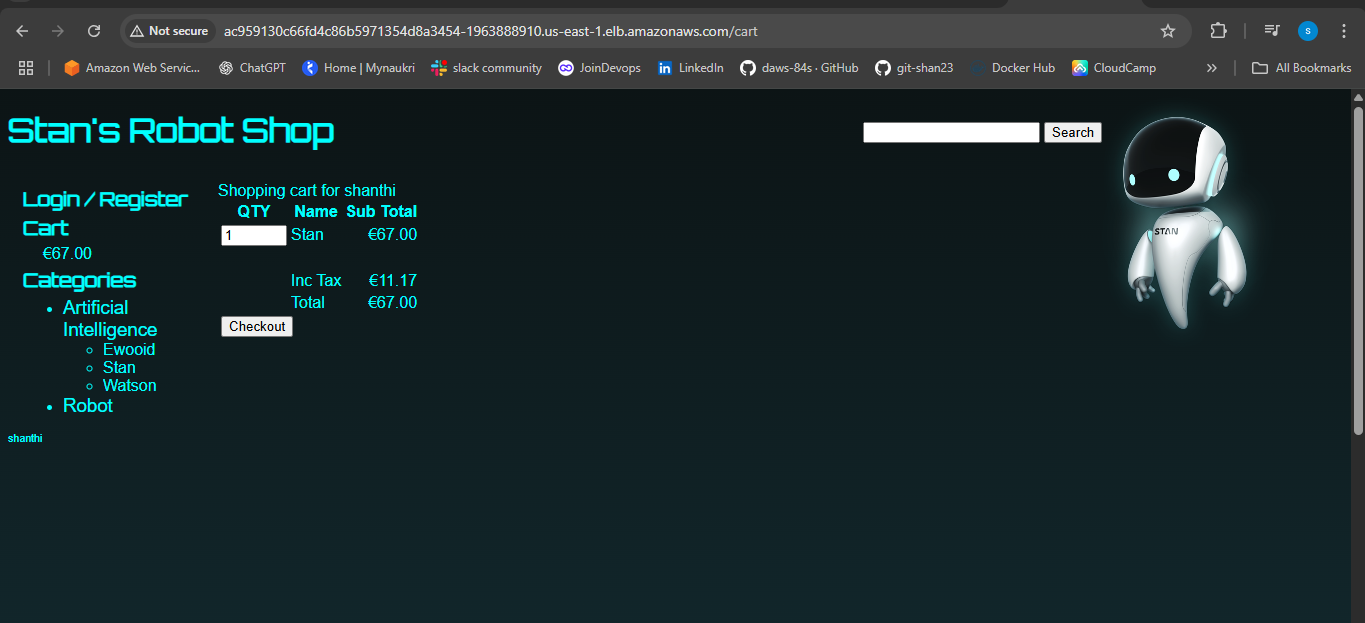


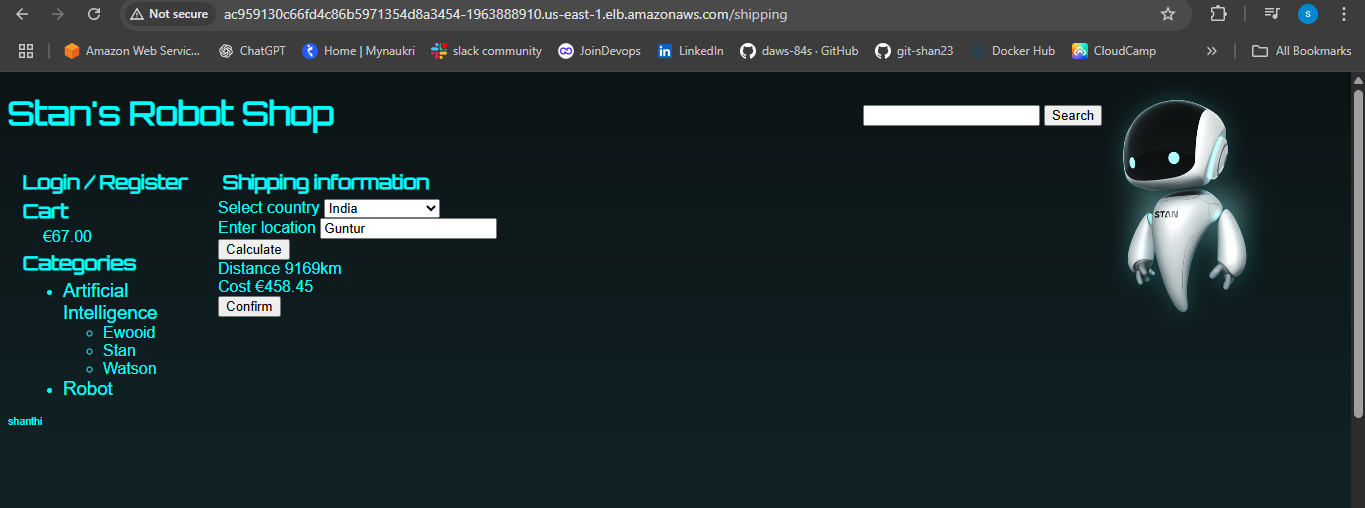
**-->** check status whether everything is running or not.

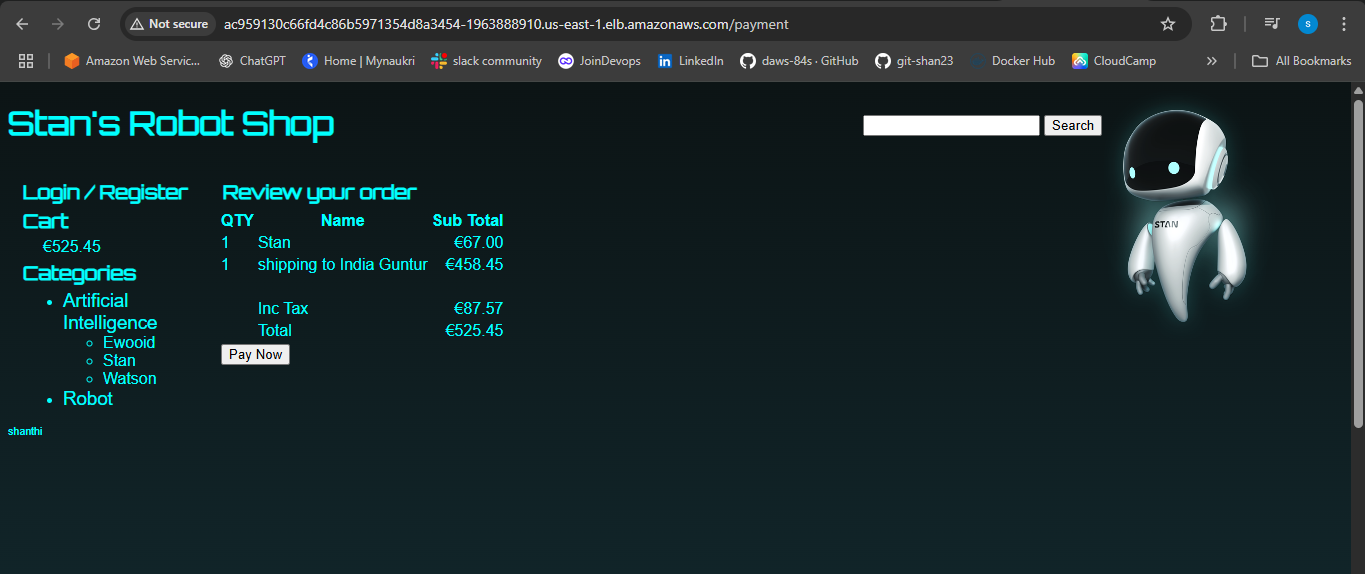


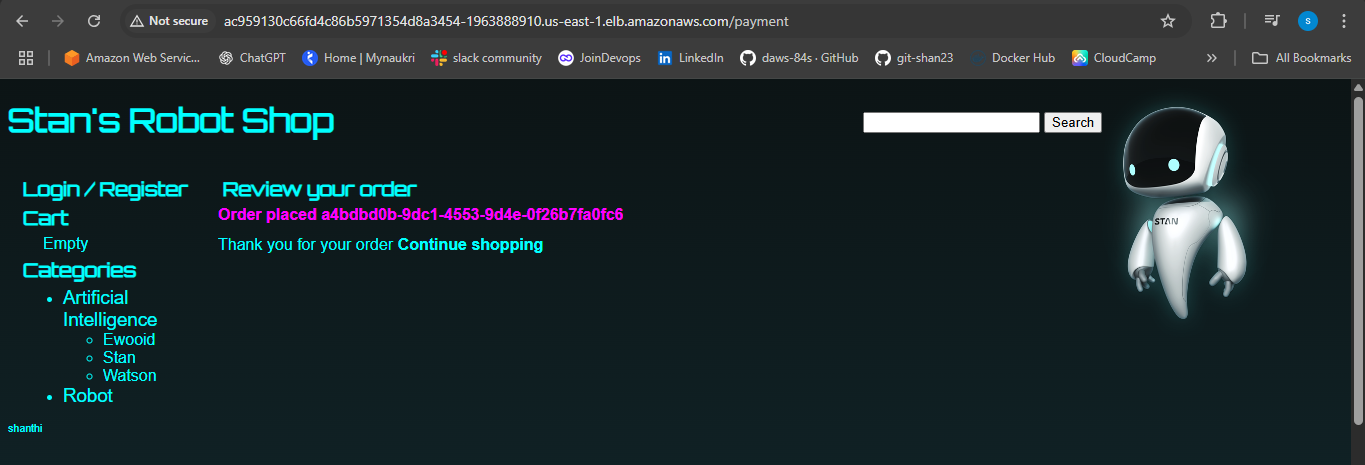
<http://ac959130c66fd4c86b5971354d8a3454-1963888910.us-east-1.elb.amazonaws.com/>











Delete deployments. All available comes in volumes.

Delete all avilable volumes

**taints and tolerations**

**======================**

nodeSelector:

az: 1b

--> this is what we are controlled still now.

--> kubernates show labels - search in google

--> add labels and give node selectros.

**01-node-selector.yaml**

apiVersion: v1

kind: Pod

metadata:

name: node-selector-demo

labels:

env: test

spec:

containers:

- name: nginx

image: nginx

imagePullPolicy: Always

nodeSelector:

disktype: ssd

--> image pull policy is always so better policy is always

--> push and pull the code

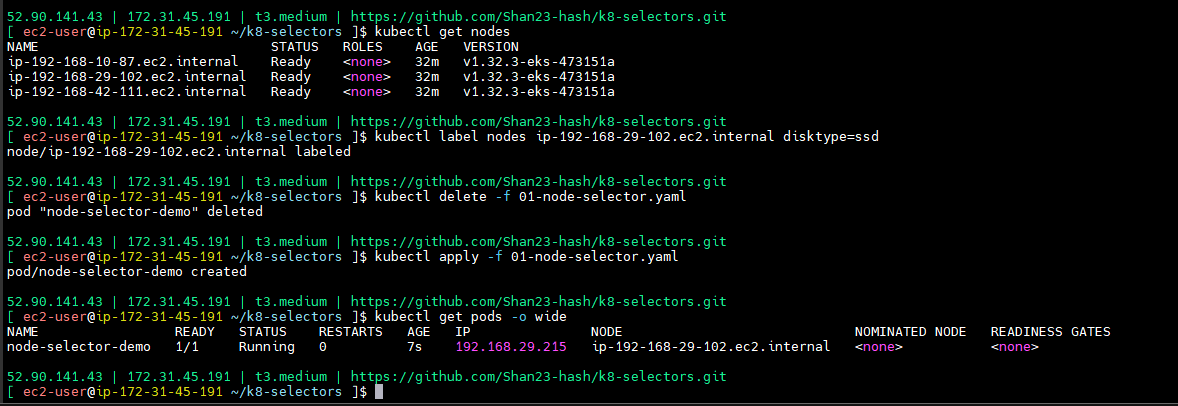
**--> git clone <https://github.com/Shan23-hash/k8-selectors.git>**

**--> cd ~/k8-selectors**

**--> kubectl label nodes ip-192-168-32-140.ec2.internal disktype=ssd**

**--> kubectl apply -f 01-node-selector.yaml**

**--> kubectl get pods -o wide**



catalogue:v1

node pulled image catalogue:v1

taint --> is like paint

Taints and tolerations kubernates

**kubectl taint nodes ip-192-168-32-140.ec2.internall key1=value1:NoSchedule**

**02-node-selector.yaml**

apiVersion: v1

kind: Pod

metadata:

name: node-selector-demo-1

labels:

env: test

spec:

containers:

- name: nginx

image: nginx

imagePullPolicy: Always

nodeSelector:

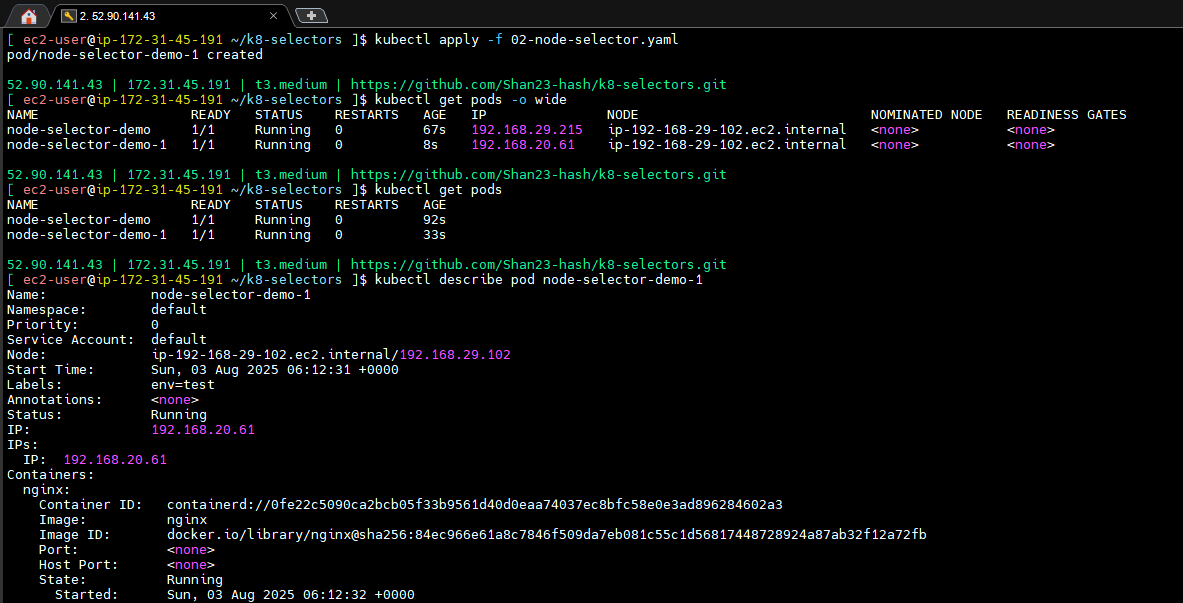
disktype: ssd

--> push and pull the code

**--> kubectl apply -f 02-node-selector.yaml**

**--> kubectl get pods**

**--> kubectl describe pod node-selector-demo-1**



**02-node-selector.yaml**

apiVersion: v1

kind: Pod

metadata:

name: node-selector-demo-1

labels:

env: test

spec:

containers:

- name: nginx

image: nginx

imagePullPolicy: Always

nodeSelector:

disktype: ssd

tolerations:

- key: "key1"

operator: "Equal"

value: "value1"

effect: "NoExecute"

- key: "key1"

operator: "Equal"

value: "value1"

effect: "NoSchedule"

**--> kubectl apply -f 02-node-selector.yaml**

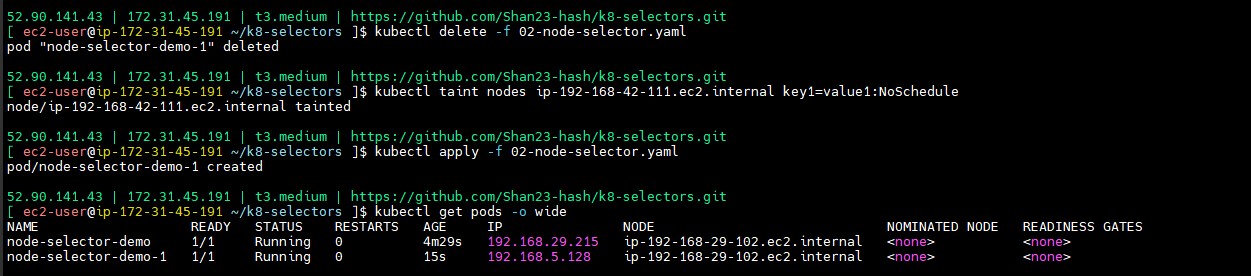
**--> kubectl get pods**

**--> kubectl describe pod node-selector-demo-1**

**--> kubectl delete -f 02-node-selector.yaml**

**--> kubectl apply -f 02-node-selector.yaml**

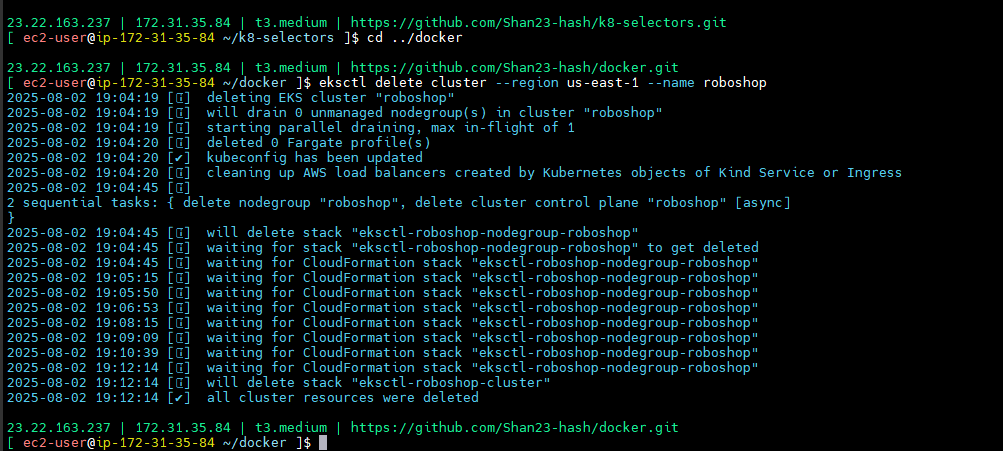
--> **kubectl taint nodes ip-192-168-32-140.ec2.internall key1=value1:NoSchedule**



--> **kubectl describe pod ip-192-168-32-140.ec2.internall**

Scheduler --> Control plane component in kubernates

**eksctl delete cluster --region us-east-1 --name roboshop**



**why taint?**

project specific hardware, seperate firewalls, performance reasons