**SESSION - 59**

--> kubernates is all about how you run the image.  
--> we are buidling images with useing docker.  
--> But running in kubernates.  
--> Priviously our application run in VM. now also running in VM but running like kubernates container.  
--> This is basic **syntax**:  
--> apiVersion: v1  
kind:  
metadata:  
name:  
labels:  
--> annotations are not used as selectors but it will be use to select the external resources outside of kubernates.  
--> There is no limit for the key value pair inside annotations.  
 **Resources**

**----------------------**  
--> You can limit the Resources to containers using soft using resource session requests and unix requests limits is hard.

**Service Mesh**

**-------------------**

--> pod to pod communication, IP address is not useful since it is ephemeral. We have services in kubernetes to acheive

1. pod to pod communication

2. load balancing

--> We are running two carts. Definitely we need load balancing for calling.

--> Load balacer will distribute load.

--> For pod to pod communication called service.



--> cart will directly this DNS.

--> catalogue will distribute load.

--> Kubernates service -- search in google

--> see selector is imp

--> in this pod which one need to select which basis on it will select -- labels

--> labels are used selectors

1. **service.yaml**

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

purpose: service-demo

spec:

selector:

purpose: service-demo

project: roboshop

environment: dev

ports:

- protocol: TCP

port: 80 # service port

targetPort: 80 # container port

---

apiVersion: v1

kind: Pod

metadata:

name: nginx

labels:

purpose: service-demo

project: roboshop

environment: dev

spec:

containers:

- name: nginx

image: nginx

--> Container port is 80.

--> Nginx port no 80 only opening.

--> Now we can run and see

--> Push and pull the code

--> cd k8-resources

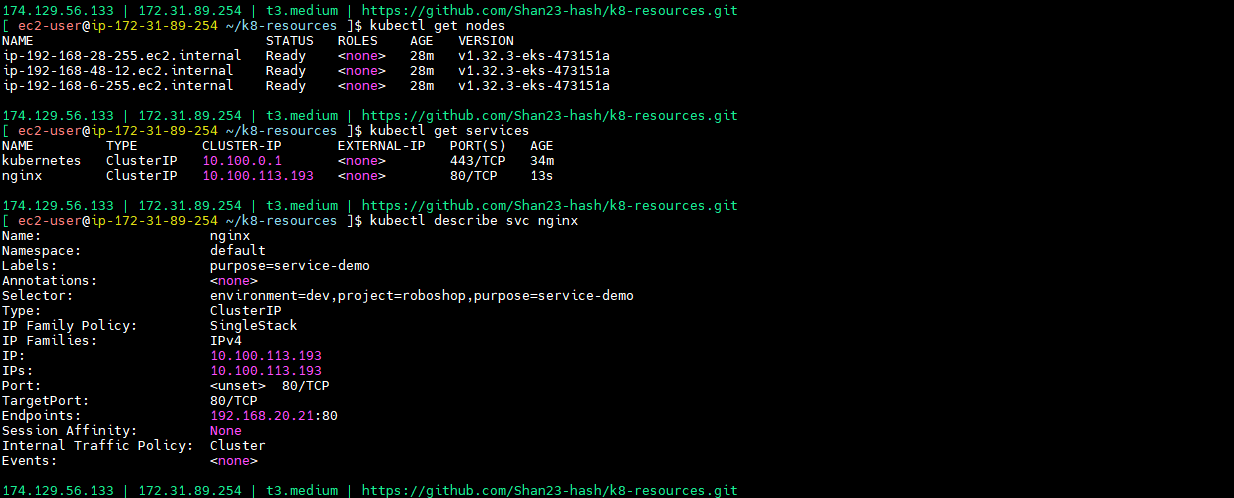
**--> kubectl get nodes**

**--> kubectl apply -f 12-service.yaml**

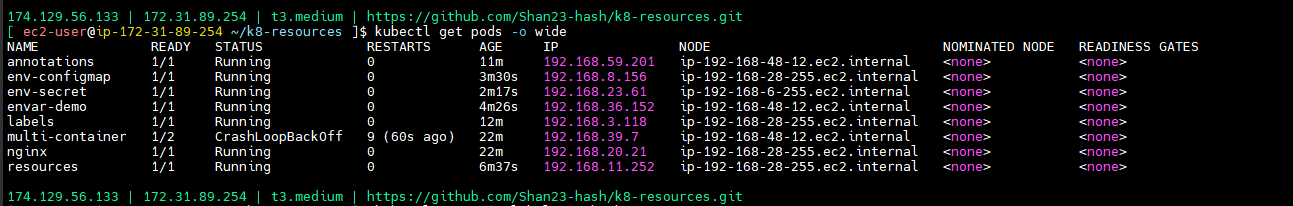
**--> kubectl get nodes**

**--> kubectl get services**

**--> kubectl describe svc nginx**



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





**-->** In services what is the there in endpoint : IP

--> this is nginx services. Inside nginx pod is there. This nginx service IP address.

--> end point is POD ip address.

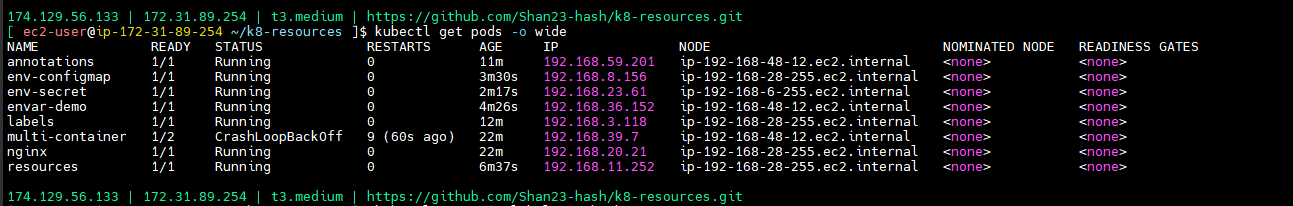
--> if you hit the services, that services have under end points to those service distributed traffic.

--> One only there that’s traffic will send only one, If two are there roudrabbin algorithm.

--> **kubectl apply -f 02-pod.yaml**

**--> kubectl get pods**

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





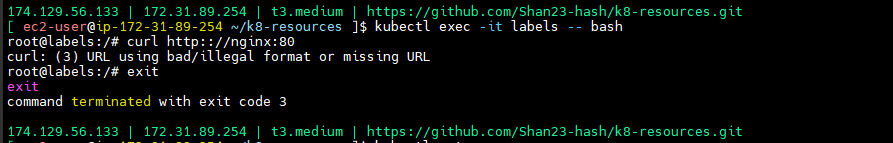
--> labels pod throgh nginx services to connect nginx.

--> **kubectl apply -f 04-labels.yaml**

**--> kubectl get pods**

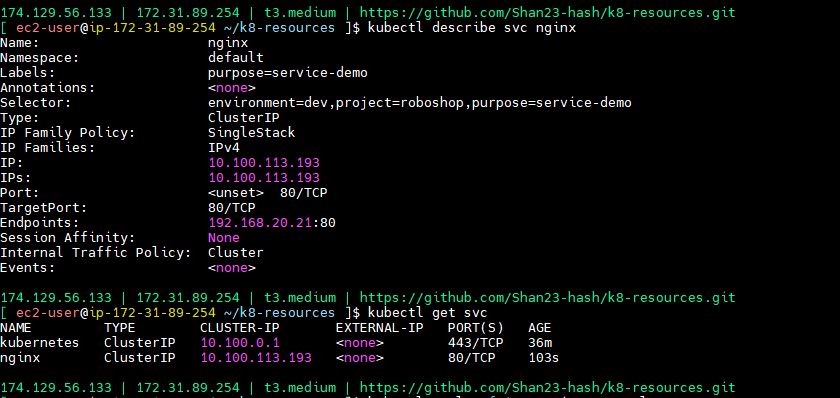
**--> kubectl exec -it labels -- bash**

**--> curl http:://nginx:80** -- got error



**--> kubrctl get svc**

**--> kubectl describe svc nginx**



**--> kubectl apply -f 12-service.yaml**

--> **kubectl describe svc nginx** -- see the end points

**--> kubectl exec -it labels -- bash**

**--> curl http:://nginx:80** -- now you are getting response.

--> DNS will work on service name here.

--> **kubectl get svc --** we got something called clusterIP

1. clusterIP --> internal to the cluster

2. nodePort --> it will open one port called nodePort in every node

3. LoadBalancer --> it will create a loadbalancer and nodePort in all the nodes

**13-service-np.yaml**

apiVersion: v1

kind: Service

metadata:

name: nginx-np

labels:

purpose: service-np-demo

spec:

type: NodePort

selector:

purpose: service-np-demo

project: roboshop

environment: dev

ports:

- protocol: TCP

port: 80 # service port

targetPort: 80 # container port

---

apiVersion: v1

kind: Pod

metadata:

name: nginx-np

labels:

purpose: service-np-demo

project: roboshop

environment: dev

spec:

containers:

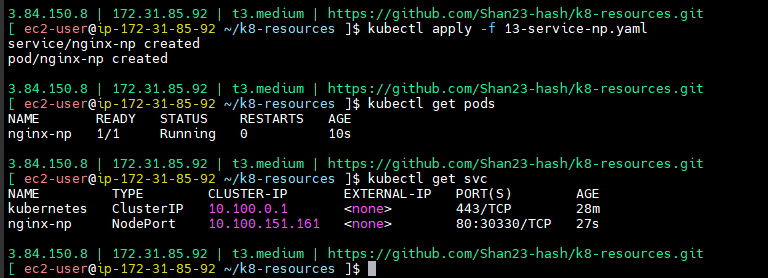
- name: nginx

image: nginx

--> Push and pull the code

--> **kubectl apply -f 13-service-np.yaml**

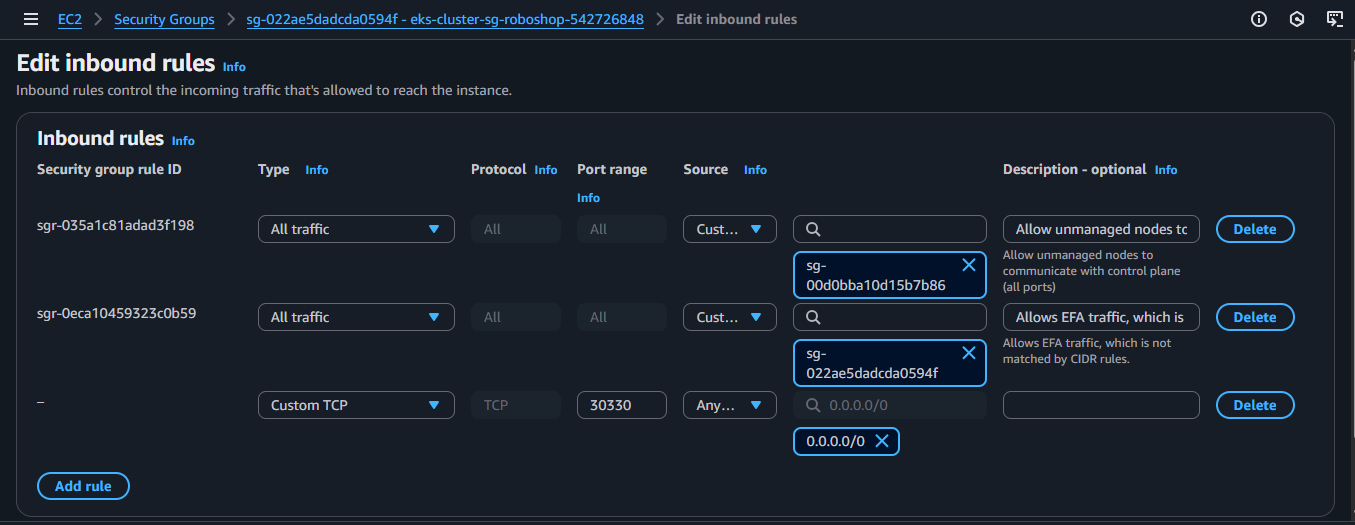
**--> kubectl get svc** -- specially one port is opened.30330



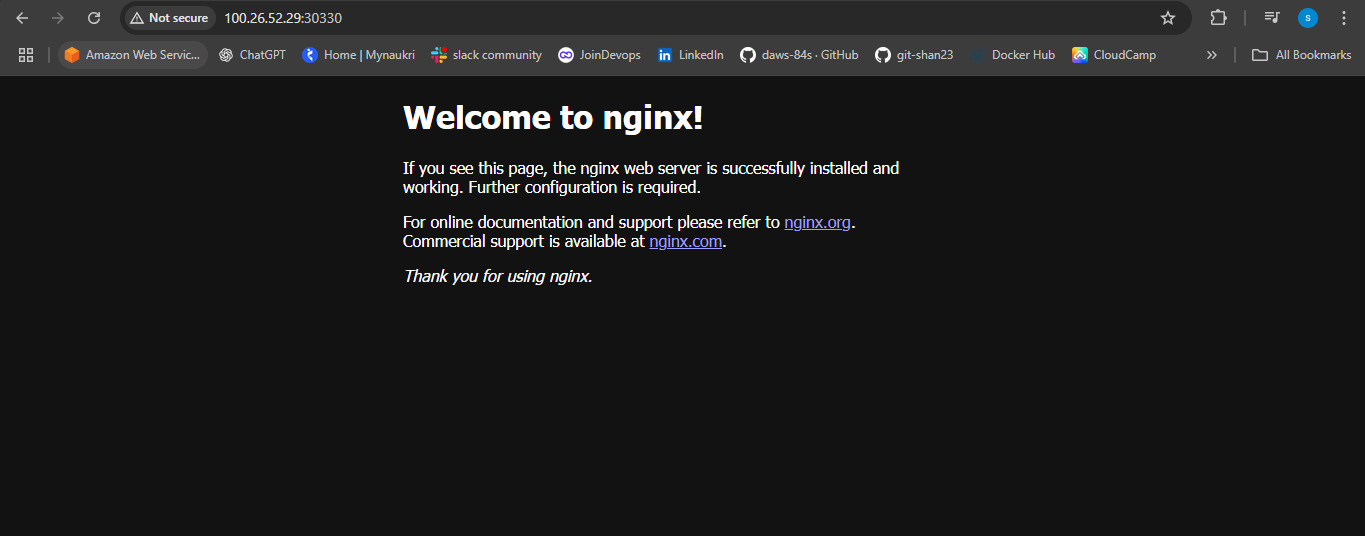
--> allow the security group

--> roboshop-roboshop-node -- security groups --> edit inbound rules --> add rule --> custom tcp - 32752 - anywhere -- > 0.0.0.0/0 --> save rules

--> output: IP:30330



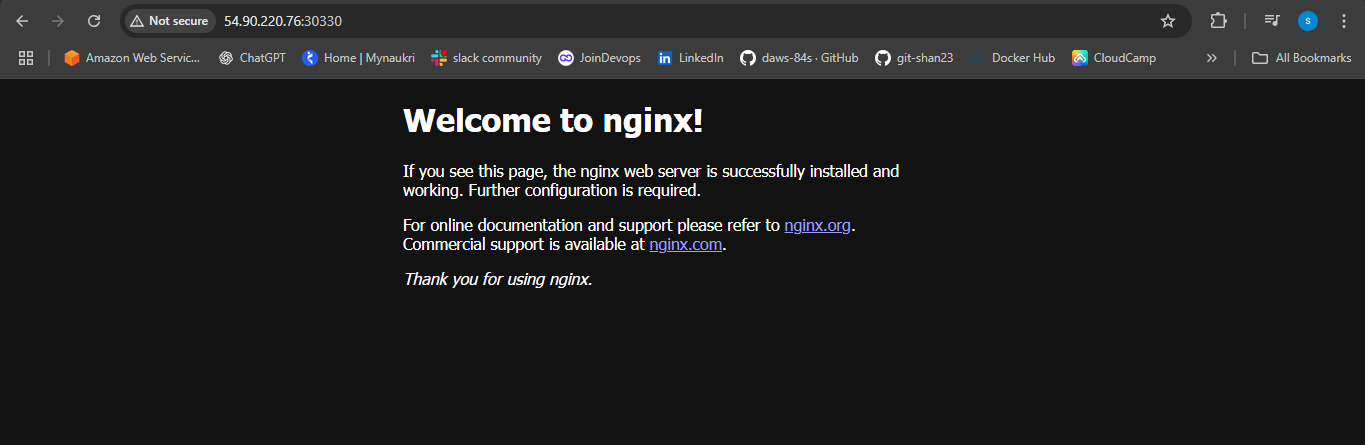
<http://100.26.52.29:30330/>



--> if I taken some other ip address it will come or not -- came

--> try one more ip also -- came

<http://54.90.220.76:30330/>



--> what happening in background. If you mentioned node board kubernates will

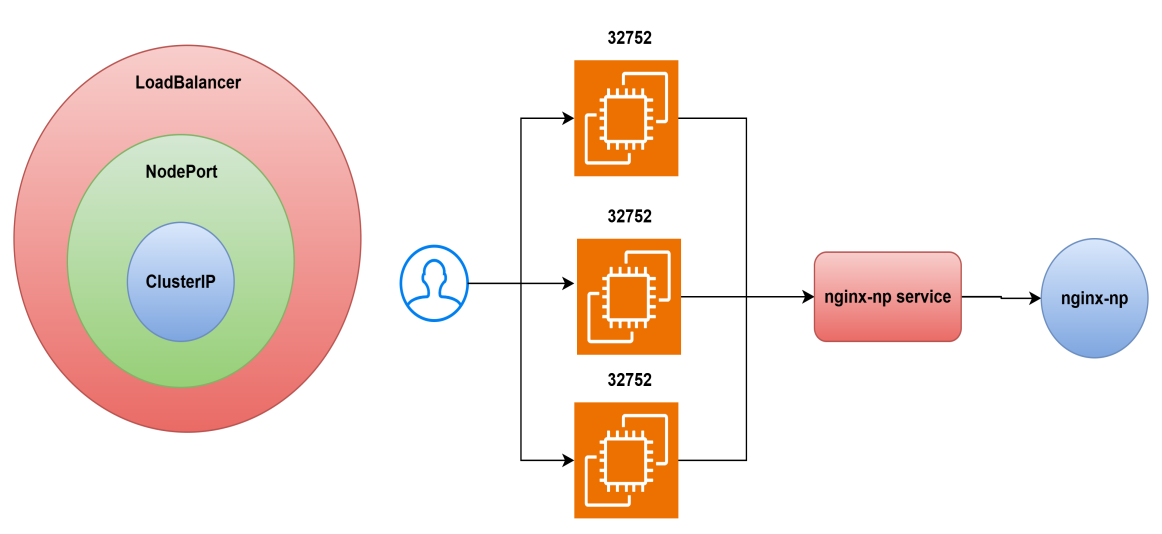
--> Node port ip have cluster ip

--> node port is also a cluster ip but it has some extra capabilities.

--> Cluster ip is the application of internal communication.with in the cluster

--> Node port is capability of external communication. Outside the cluster also.

--> cluster ip is a subset of node pod.



--> Node port will open all nodes. We have to allow security group.

--> Node port service is also a cluster ip for extra capability of opening a port in the worker node. Now this application expose to external

--> Service in kubernates is to achieve to pod to pod communication. Load balancing and you export your application to the outside world.

**LOAD BALANCING**

**14-service-lb.yaml**

apiVersion: v1

kind: Service

metadata:

name: nginx-lb

labels:

purpose: service-lb-demo

spec:

type: LoadBalancer

selector:

purpose: service-lb-demo

project: roboshop

environment: dev

ports:

- protocol: TCP

port: 80 # service port

targetPort: 80 # container port

---

apiVersion: v1

kind: Pod

metadata:

name: nginx-lb

labels:

purpose: service-lb-demo

project: roboshop

environment: dev

spec:

containers:

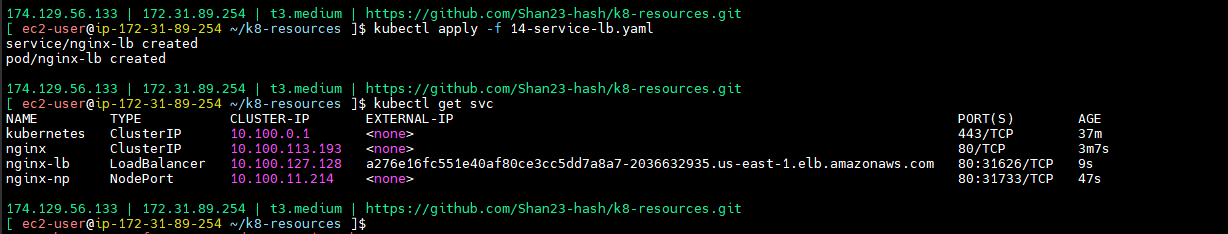
- name: nginx

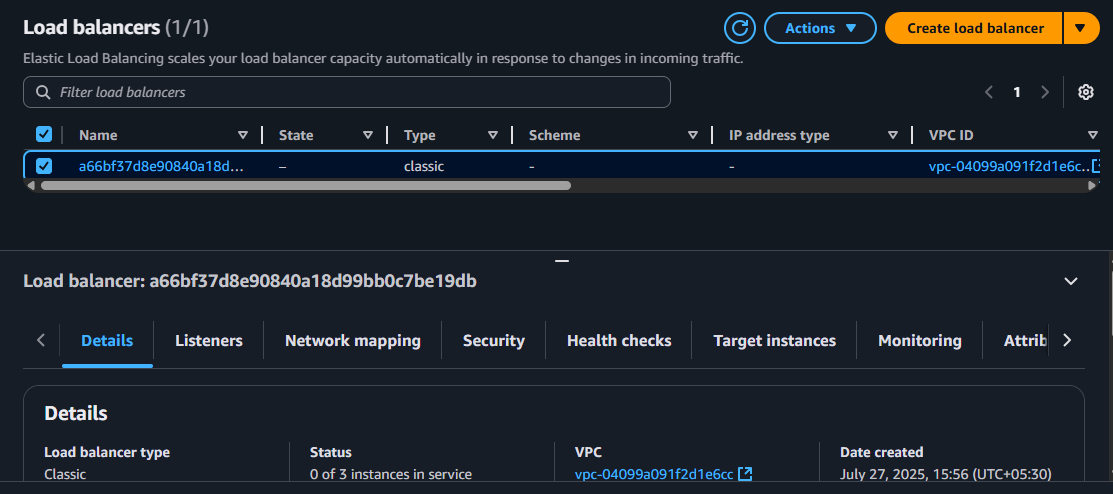
image: nginx

--> push and pull the code

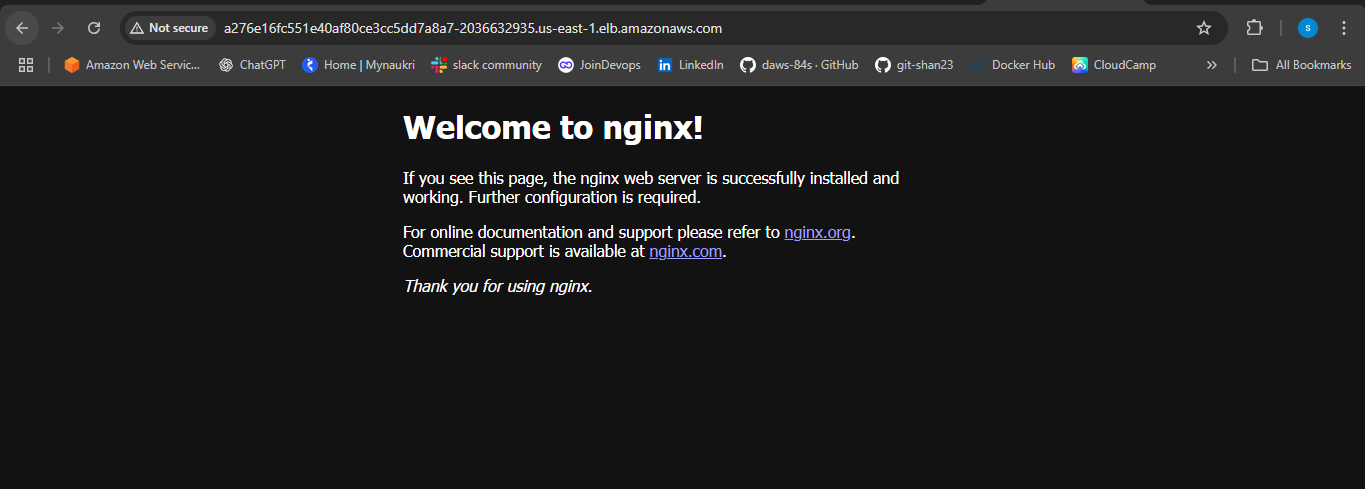
--> **kubectl apply -f 14-service-lb.yaml**

**--> kubectl get svc**





<http://a276e16fc551e40af80ce3cc5dd7a8a7-2036632935.us-east-1.elb.amazonaws.com/>



**-->** load balancer is cluster ip are not

--> so load balancer have cluster ipn node port and also extra one load balacer . this is something real load balancer.

--> Load balancer will have cluster ip,will have node port and it will have an real load balancer created.

--> Take load balancer and hit that one

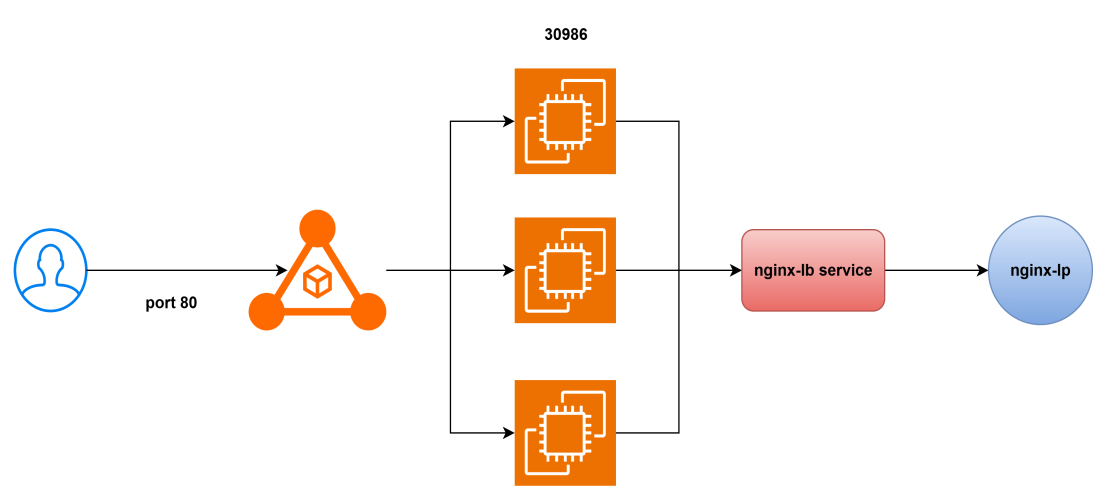
--> OUTPUT - got the response

--> load balancer hit on port no 80

--> node port is 30986

loadbalancer --> node on nodePort --> clusterIP --> pod

clusterIP --> only internal service, means with in the cluster..



--> if you want pod to pod communication it should create services.

--> from catalogue to mongodb is internal traffic.

--> internal port is cluster ip is fine one node port and load balancer?

--> Here extranal traffic needed application is frontend.

--> This pod can be there anywhere.

--> First traffic will hit to node then only it will come to pod.

--> Service means internal DNS. There is nothing run in inside.

--> just for requesting routing.

**How can I create multiple pods to same image**

**----------------------------------------------------------------**

--> we want to run multiple containers to same image need to do multiple docker run commands issued.

--> Here increasing traffic I want multiple pods, I ran multiple times labels.

--> **kubectl apply -f 04-labels.yaml** -- (if you already ran unchanged it will come)

--> **kubectl get pods**

**--> kubectl delete -f 12-service.yaml**

**--> kubectl delete -f 13-service-np.yaml**

**--> kubectl delete -f 14-service-lb.yaml**

**-->** load balance will work in cloud environment like AZURE GCP. On premicious you want to work you have to configuration in background.

--> we have diffrently have replica set

--> replicaset will create multiple replicas.

--> selectors are everything is called labels.

**15-replicaset.yaml**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginx

# replica set labels

labels:

purpose: rs-demo

project: roboshop

app: nginx

spec:

replicas: 2

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

selector:

matchLabels:

purpose: rs-demo

project: roboshop

app: nginx

# This is pod definition

template:

metadata:

labels:

purpose: rs-demo

project: roboshop

app: nginx

spec:

containers:

- name: nginx

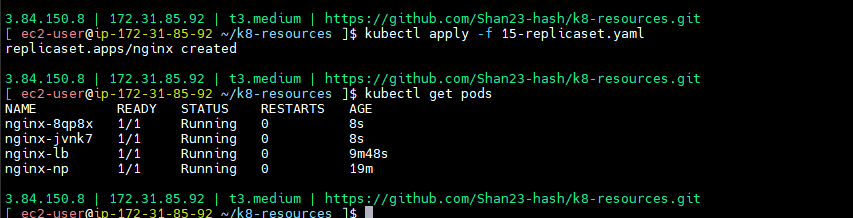
image: nginx:stable-perl

--> pushand pull the code

--> pod is a subset of replica set

--> **kubectl apply -f 15-replicaset.yaml**

--> **kubectl get pods** -- 2 created



--> nginx-c65pj == replicaset-name-<random 5 digit code>

--> we don’t have freedom to select the name. If we selected it will create repos. So replica set will chhose names randomly.

--> if I deleted this one

--> **kubectl delete pod nginx-c65pj**

**--> kubectl get pods --**  within seconds created

--> continuously it should run 3 replicas.

--> by mistakely it will delere crach, immedialy replicaset will create another one.

--> **kubectl delete pod ninx-w2xgp**

**--> kubectl get pods**

**-->** labels will give carefully, replicas and pod replica set will create.

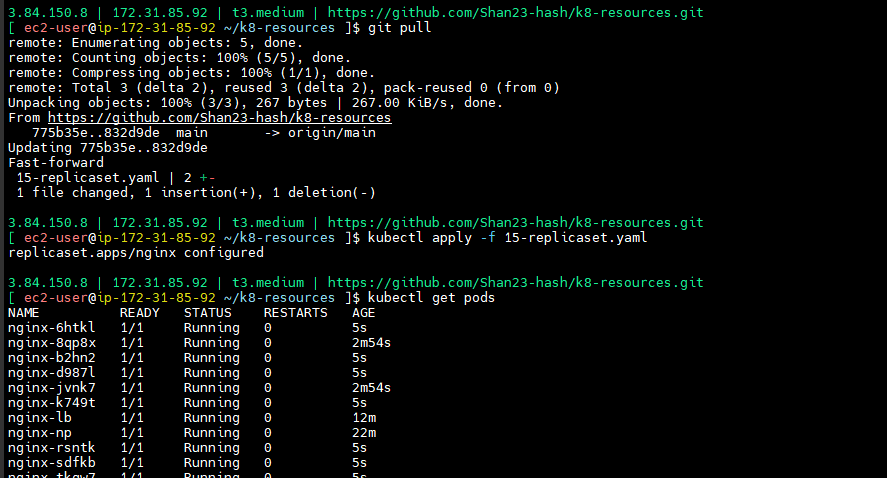
--> this replicas have another applications. That is problem. Now That one also this one will manage.

--> How much you want replicas. For example if you want it will create Immediate 10.

--> mention in code and check

--> **kubectl apply -f 15-replicaset.yaml**

--> **kubectl get pods --** 10 created



**Deployment:**

--> version change is there

--> Version is there in V1, it will some changes code changes, dev changed to v2.

--> Deployment means delete V1 old application

--> download new V2 application

--> restart

--> same like in containers nginx version is v1. you should change to v2.

--> nginx:v1

-->nginx:v2

--> image: nginx:v2

Delete old pods

Create new pods with new image.

--> which version is not mentioned means that one is latest.

--> I changing from latest to another version.

--> I’m changing to new pods.

--> pods are not created.

--> replicas set only responsible in the maintain the no of replicas.

--> that should not respect image was changed.

--> There is another called deployment.deployment it will do all the activates.

--> kubernates deployment -- search in google

--> kind is deployment remaining thing is same.

1. **deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

# deployment labels

labels:

purpose: deployment-demo

project: roboshop

app: nginx

spec:

replicas: 2

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

selector:

matchLabels:

purpose: deployment-demo

project: roboshop

app: nginx

# This is pod definition

template:

metadata:

labels:

purpose: deployment-demo

project: roboshop

app: nginx

spec:

containers:

- name: nginx

image: nginx:stable-perl

--> Push and pull the code

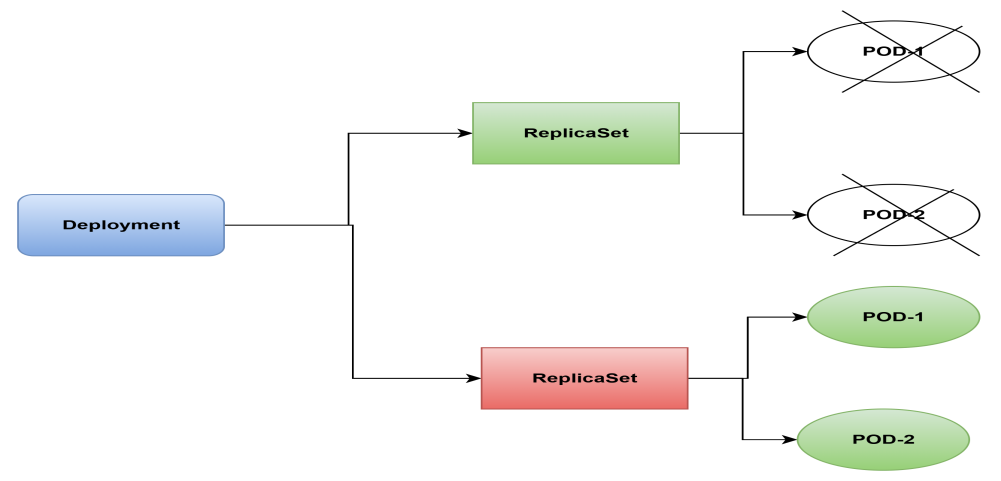
**--> kubectl apply -f 16-deployment.yaml**

--> **kubectl get pods** -- 10 ports are created

--> **kubectl get rs** -- replica set also created.

--> pod is a subset of replica set

--> replica set is a subset of deployment.



--> Kubernates service.

--> deployment created an replica-set, replica set will create pod replica set.

--> here we given two pods.

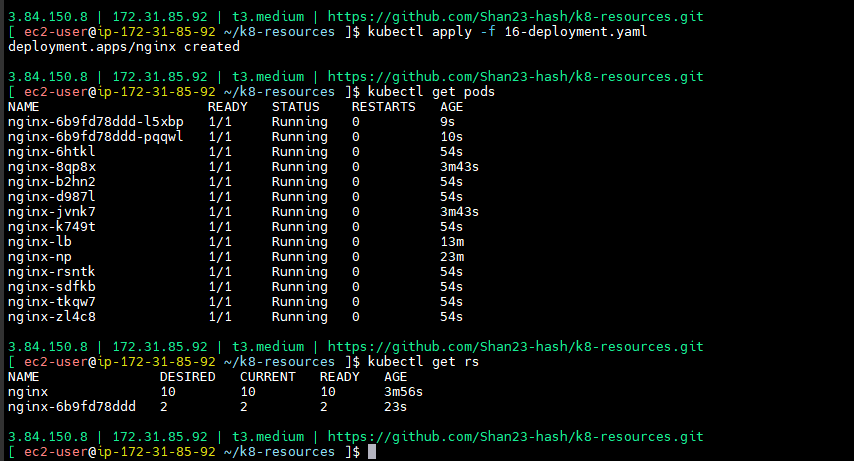
--> replica set created randomly.

--> image: nginx:latest - changes image name

**--> kubectl apply -f 16-deployment.yaml**

--> **kubectl get pods**

--> **kubectl get rs**



**-->** Deployment all about deleting the old application and creating new one.

--> replica will create first new replica.

--> in our application deployment we have to use.because our applications frequently will change.

--> 4 replicas also you can go through deployments it is to update the application.

--> pod to pod application -- services

--> updated time - deployment

--> that why pod is the smallest deployment unit in the kubernates. If you do replicaset or deployment at the end it is nothing but pod.

--> use these resources we can create our application as kubernates application.

--> **cd ../docker**

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

