Note:before start project you must knowledge of basic aws & docker like create ecr & secrate key & how to install aws-cli & how to push image on ecr & tag image as requirement. So i put some documents first read this before start

https://docs.aws.amazon.com/eks/latest/userguide/aws-load-balancer-controller.html

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

https://www.geeksforgeeks.org/how-to-create-aws_access_key-and-seceret-key/ -for configure & make secrate key

Step 1: Create IAM Role using `eksctl`

curl -O
https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.0/docs/install/iam_policy.json

Step2 Create an IAM policy using the policy downloaded in the previous step.
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json

Step3-install eksctl
curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz"
tar -xzf eksctl_$(uname -s)_amd64.tar.gz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
Step-craete ocl
 eksctl utils associate-iam-oidc-provider --region=ap-south-1 --cluster=learning (chnage cluster name & region as requirment
Step4-
Replace the values for cluster name, region code, and account ID.

```
eksctl create iamserviceaccount \
    --cluster=learning \ (change name as requirement)
    --namespace=kube-system \
    --name=aws-load-balancer-controller \

--attach-policy-arn=arn:aws:iam::<AWS_ACCOUNT_ID>:policy/AWSLoadBalancerCont
rollerIAMPolicy \
    --override-existing-serviceaccounts \
    --region ap south-1 \
    --approve
```

Step 2: Install AWS Load Balancer Controller

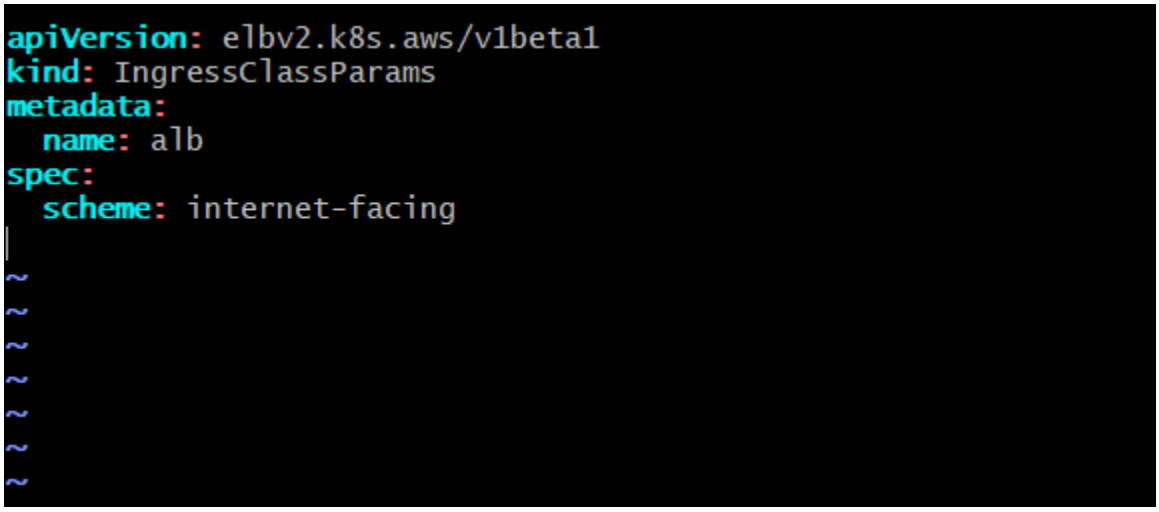
```
helm repo add eks https://aws.github.io/eks-charts

helm repo update eks

helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --version 1.13.0
  --set vpcId=

#install crds

wget
https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-cont
roller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Step 3: Verify that the controller is installed

kubectl get deployment -n kube-system aws-load-balancer-controller

Step4- now make 2 files for implementing aws controller on eks for setup ingress service with aws controller.
vim IngressClassParamsbase.yaml

apiVersion: elbv2.k8s.aws/v1beta1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing

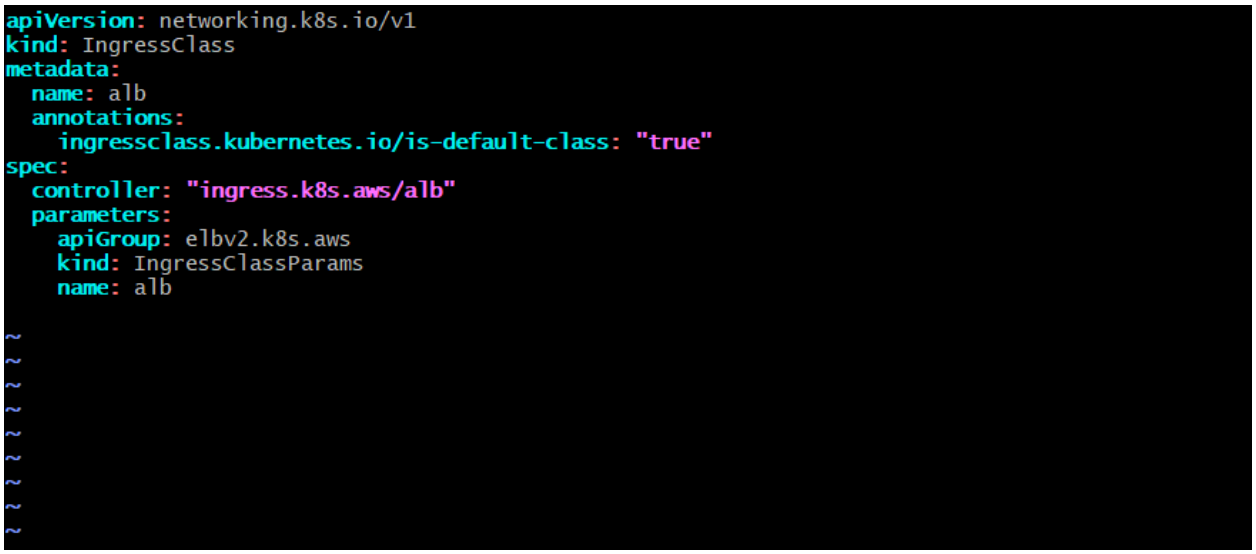


vim IngressClassParams.yaml

apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb

```
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: "ingress.k8s.aws/alb"
  parameters:
    apiGroup: elbv2.k8s.aws
    kind: IngressClassParams
    name: alb
```

*Make sure kubectl apply -f file name 2 times for apply this



Now make 3 services first for deployment

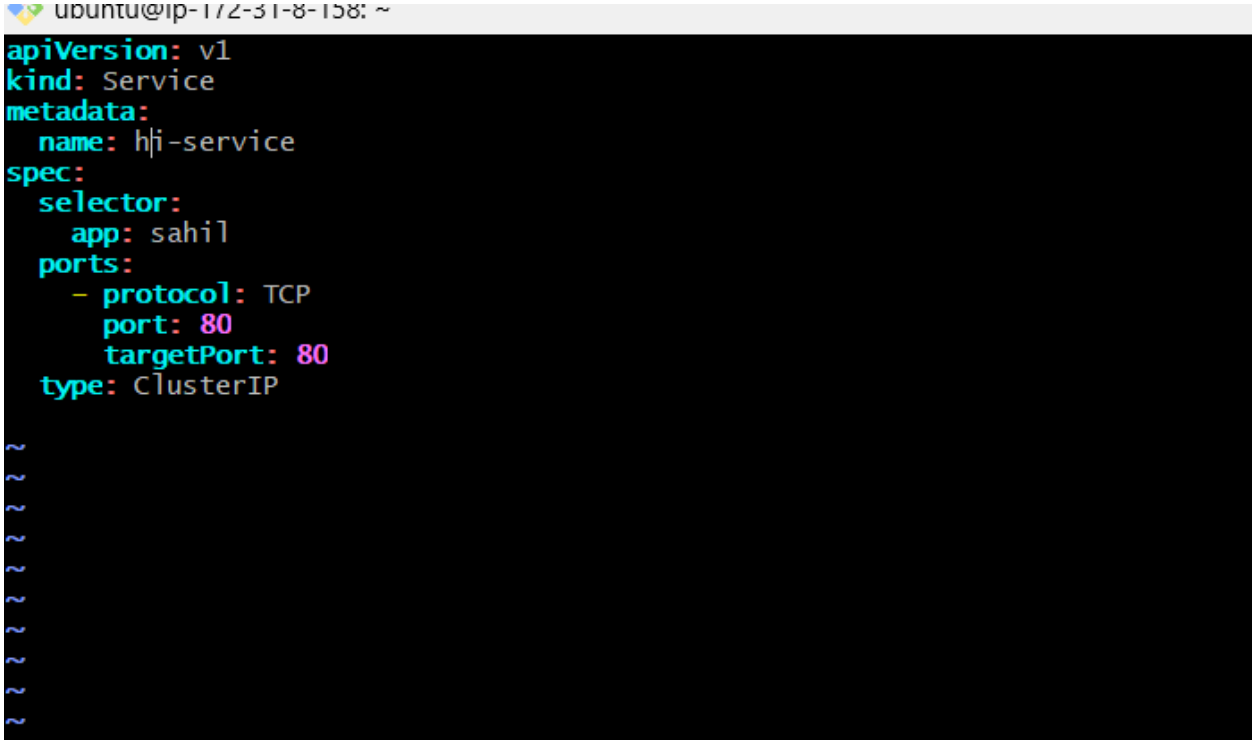
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sahil-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sahil
  template:
    metadata:
      labels:
        app: sahil
    spec:
      imagePullSecrets:
        - name: sahil
```

```
    containers:
      - name: sahil-container
        image: 947557066309.dkr.ecr.ap-south-1.amazonaws.com/sahil10:latest
        ports:
          - containerPort: 80
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sahil-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sahil
  template:
    metadata:
      labels:
        app: sahil
    spec:
      imagePullSecrets:
        - name: sahil
      containers:
        - name: sahil-container
          image: 947557066309.dkr.ecr.ap-south-1.amazonaws.com/sahil10:latest
          ports:
            - containerPort: 80

~
~
~
~
```

Kubectl apply -f default-deployment.yaml
Now make service file
vim ingreseservice.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: hi-service
spec:
  selector:
    app: sahil
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
```



kubectl apply -f default-deployment.yaml
Now i will share docker file

```
FROM ubuntu:latest
RUN apt-get update && \
    apt-get install -y nginx && \
    rm -rf /var/lib/apt/lists/*

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

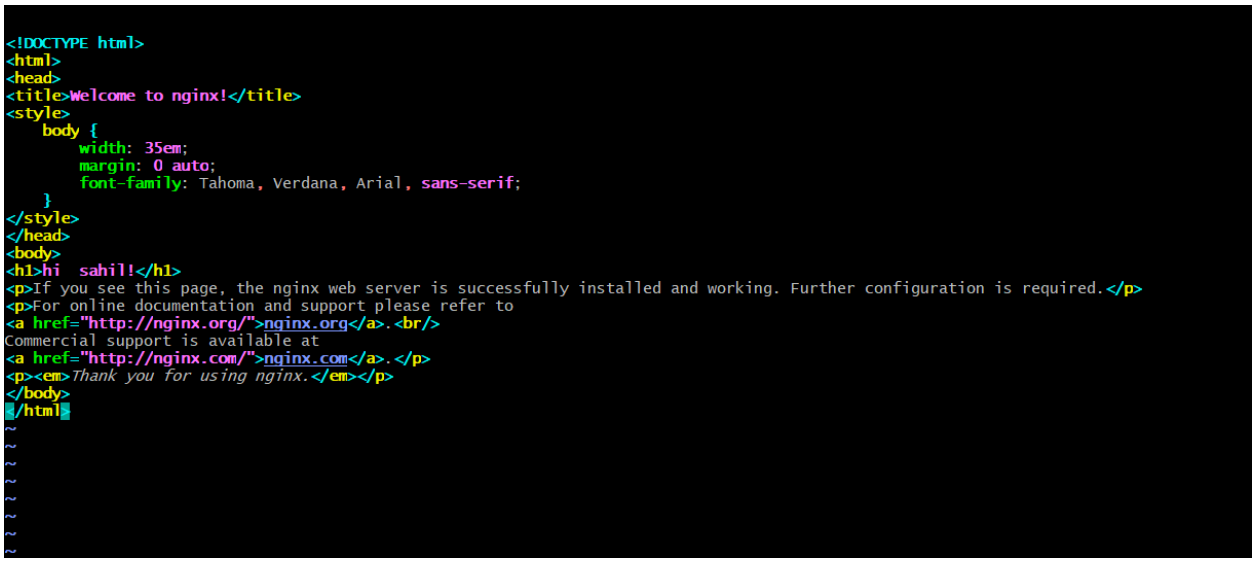
Note:in the default path we will use the default nginx page so just a simple docker file.

Now we make 2 service hi-service

vim index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
```

```
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>hi  sahil!</h1>
<p>If you see this page, the nginx web server is successfully installed and working.
Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```



Now make image of this but first need to write custom nginx.conf for run on

```
events {}

http {
    server {
        listen 80;
        server_name localhost;

        location /hi {
```

```
        root /var/www/html;
        index index.html;
    }
  }
}
```

```
events {}

http {
    server {
        listen 80;
        server_name localhost;

        location /hi {
            root /var/www/html;
            index index.html;
        }
    }
}
~
~
~
~
~
~
~
```

Vim Docker file

```
FROM ubuntu:latest
RUN apt-get update && \
   apt-get install -y nginx && \
   rm -rf /var/lib/apt/lists/*


COPY index.html /usr/share/nginx/html/index.html

RUN rm -f /var/www/html/index.nginx-debian.html


RUN rm /etc/nginx/nginx.conf


COPY nginx.conf /etc/nginx/nginx.conf
```

COPY index.html /var/www/html/hi/index.html
EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

```
FROM ubuntu:latest
RUN apt-get update && \
    apt-get install -y nginx && \
    rm -rf /var/lib/apt/lists/*

COPY index.html /usr/share/nginx/html/index.html

RUN rm -f /var/www/html/index.nginx-debian.html

RUN rm /etc/nginx/nginx.conf

COPY nginx.conf /etc/nginx/nginx.conf

COPY index.html /var/www/html/hi/index.html
EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

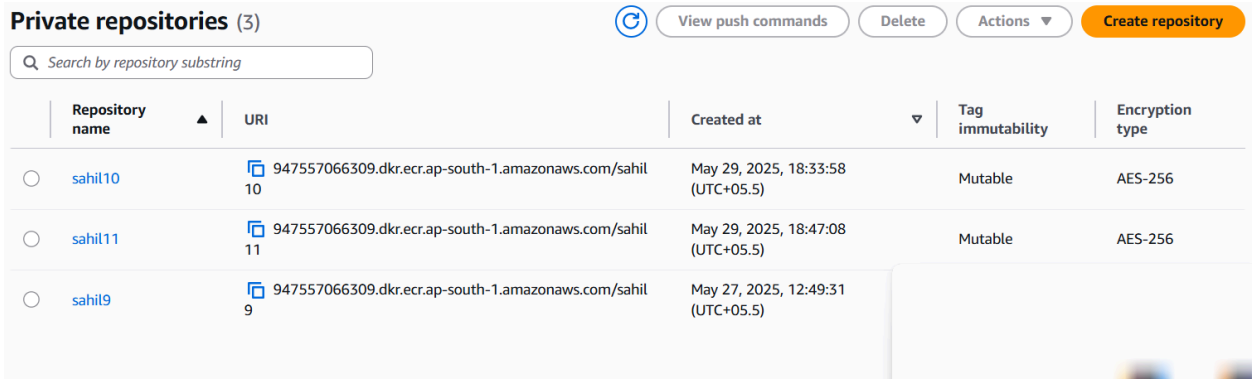
Now make an image using
docker build -t sahil1 . (you must change name as your ecr repo just give me for demo)
Now after build image push on ecr/docker hub & make secrate of kubernetes of id & password.
But first need to install aws-cli & configure secrate key i will give you command for in ubuntu
Do this

aws update -y
Snap install aws-cli
aws configure

Example

```
ubuntu@ip-172-31-8-158:~$ vim Dockerfile
ubuntu@ip-172-31-8-158:~$ kubectl create secret docker-registry ecr-registry-secret \
  --docker-server=307302069467.dkr.ecr.ap-southeast-2.amazonaws.com \
  --docker-username=AWS \
  --docker-password="$(aws ecr get-login-password --region ap-southeast-2)"
```

Make sure you change docker server with account id mentioned in aws ecr & region as ecr present. I share image of ecr as example

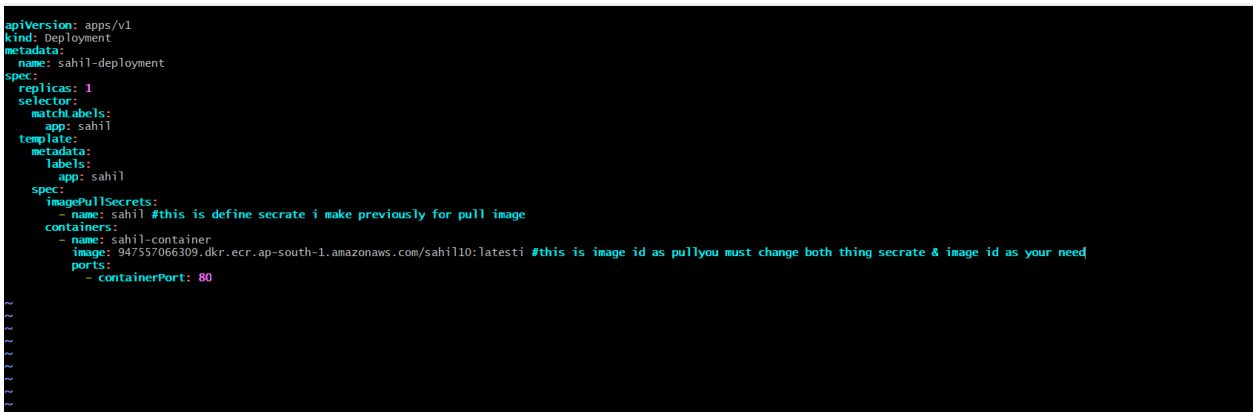


Now make deployment file

vim ingrese1deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sahil-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sahil
  template:
    metadata:
      labels:
        app: sahil
    spec:
      imagePullSecrets:
```

```
      - name: sahil
    containers:
      - name: sahil-container
        image: 947557066309.dkr.ecr.ap-south-1.amazonaws.com/sahil10:latest
        ports:
          - containerPort: 80
```



kubectl apply -f ingrese1deployment.yaml

#now make service of this
Vim ingreseservice.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: hi-service
spec:
  selector:
    app: sahil
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
```

```
ubuntu@ip-172-31-8-158: ~
apiVersion: v1
kind: Service
metadata:
  name: hi-service
spec:
  selector:
    app: sahil
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
~
```

kubectl apply -f ingreseservice.yaml

#now make a bye service
Vim 2index.html # we have already make index.html page on same server so assign this name 2index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>bye  sahil!</h1>
<p>If you see this page, the nginx web server is successfully installed and working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
~
~
~
~
```

Vim 2nginx.conf
events {}

```
http {
    server {
        listen 80;
        server_name localhost;

        location /bye {
            root /var/www/html;
```

```
        index index.html;
      }
    }
}
```

ubuntu@ip-172-31-8-158: ~

```
events {}

http {
    server {
        listen 80;
        server_name localhost;

        location /bye {
            root /var/www/html;
            index index.html;
        }
    }
}
~
~
~
```

#now make a Docker file

```
FROM ubuntu:latest
RUN apt-get update && \
    apt-get install -y nginx && \
    rm -rf /var/lib/apt/lists/*


COPY 2index.html /usr/share/nginx/html/index.html

RUN rm -f /var/www/html/index.nginx-debian.html


RUN rm /etc/nginx/nginx.conf


COPY 2nginx.conf /etc/nginx/nginx.conf

COPY 2index.html /var/www/html/bye/index.html
```

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

```
FROM ubuntu:latest
RUN apt-get update && \
    apt-get install -y nginx && \
    rm -rf /var/lib/apt/lists/*

COPY 2index.html /usr/share/nginx/html/index.html

RUN rm -f /var/www/html/index.nginx-debian.html

RUN rm /etc/nginx/nginx.conf

COPY 2nginx.conf /etc/nginx/nginx.conf

COPY 2index.html /var/www/html/bye/index.html
EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
~
```

Docker build image name .& push the image
vim bye-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bye-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bye
  template:
    metadata:
      labels:
        app: bye
```

```
    spec:
      imagePullSecrets:
        - name: sahil
      containers:
      - name: bye-container
        image: 947557066309.dkr.ecr.ap-south-1.amazonaws.com/sahil11:latest
        ports:
          - containerPort: 80
~
~
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bye-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bye
  template:
    metadata:
      labels:
        app: bye
    spec:
      imagePullSecrets:
        - name: sahil ##change this as you write secrate name
      containers:
      - name: bye-container
        image: 947557066309.dkr.ecr.ap-south-1.amazonaws.com/sahil11:latest #change this as your requirment if you use docker so set image name according
        ports:
          - containerPort: 80
~
~
~
~
~
```

Now make a service file
Vim bye-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: bye-service
spec:
  selector:
    app: bye
  ports:

    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP
```

ubuntu@ip-172-31-8-158: ~

```
apiVersion: v1
kind: Service
metadata:
  name: bye-service
spec:
  selector:
    app: bye
  ports:

    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP

~
~
~
~
```

kubectl apply -f bye-deployment.yaml
kubectl apply -f bye-service.yaml

Done now make sure all deployments & service work fine
Kubectl get deployments

```
ubuntu@ip-172-31-8-158:~$ kubectl get deployments
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
bye-deployment           1/1     1            1           2d22h
default-nginx            1/1     1            1           2d20h
nginx-deploynodeselector 0/3     3            0           2d20h
sahil-deployment         1/1     1            1           18h
ubuntu@ip-172-31-8-158:~$
```

Kubectl get svc #we have 3 services hi bye & sahil we make

```
ubuntu@ip-172-31-8-158:~$ kubectl get svc
NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
bye-service           ClusterIP   10.100.153.204   <none>        80/TCP    2d22h
default-nginx-service ClusterIP   10.100.143.92    <none>        80/TCP    2d20h
hi-service            ClusterIP   10.100.251.165   <none>        80/TCP    19h
kubernetes            ClusterIP   10.100.0.1       <none>        443/TCP   7d
sahil-service         ClusterIP   10.100.97.255    <none>        80/TCP    2d23h
ubuntu@ip-172-31-8-158:~$
```

Now write ingress file

Vim ingress.yaml

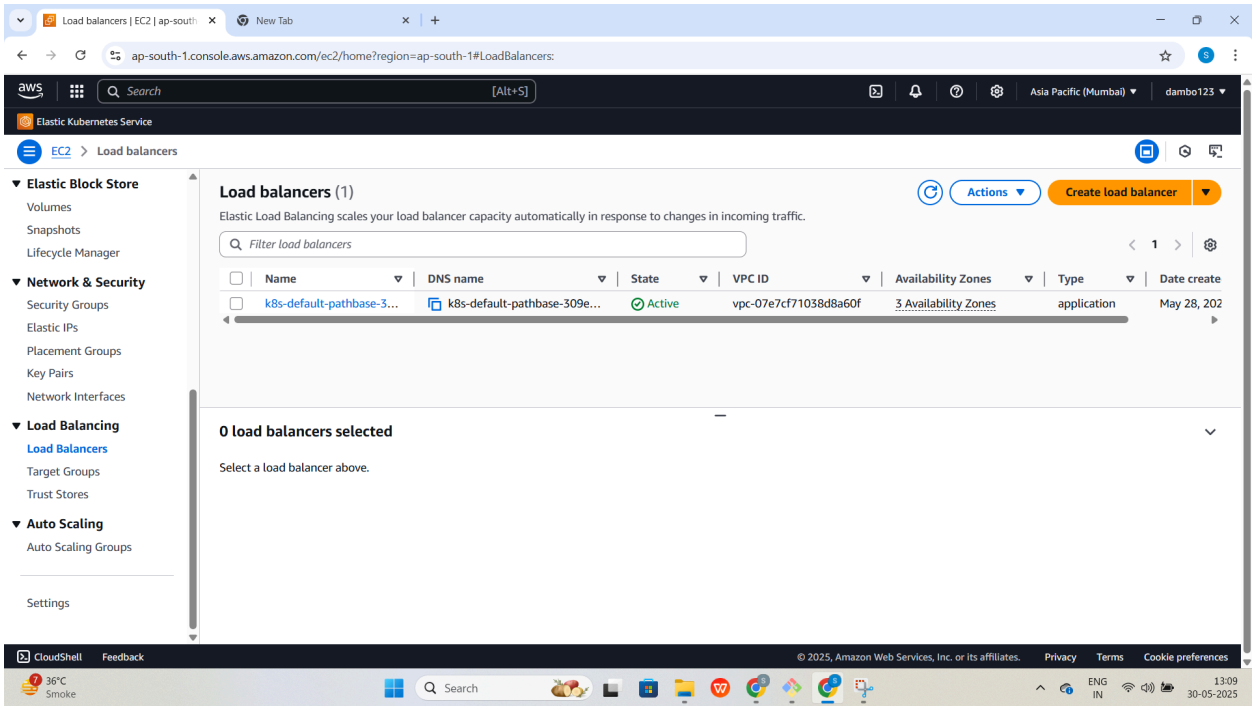
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: path-based-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}]'
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /hi
            pathType: Prefix
            backend:
              service:
                name: hi-service
                port:
                  number: 80
          - path: /bye
            pathType: Prefix
            backend:
              service:
                name: bye-service
                port:
                  number: 80
          - path: /*
            pathType: Prefix
            backend:
              service:
                name: default-nginx-service
                port:
                  number: 80
~
~
~
```

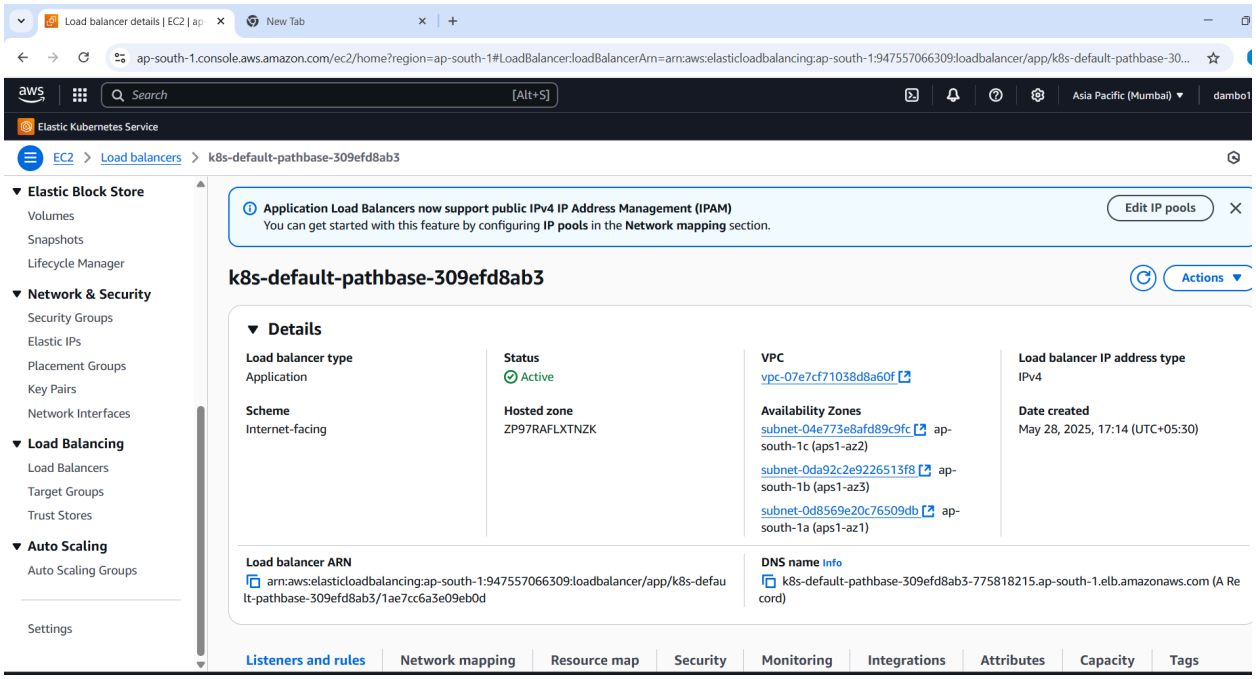
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: path-based-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}]'
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /hi
            pathType: Prefix
            backend:
              service:
                name: hi-service
                port:
                  number: 80
          - path: /bye
            pathType: Prefix
            backend:
              service:
                name: bye-service
                port:
                  number: 80
          - path: /*
            pathType: Prefix
            backend:
              service:
                name: default-nginx-service
                port:
                  number: 80
~
~
~
```

kubectl apply -f ingress.yaml
ubuntu@ip-172-31-8-158:~$ kubectl get ingress

```
ubuntu@ip-172-31-8-158:~$ kubectl get ingress
NAME                CLASS   HOSTS   ADDRESS                                                                  PORTS   AGE
path-based-ingress   alb     *       k8s-default-pathbase-309efd8ab3-775818215.ap-south-1.elb.amazonaws.com   80      2d21h
ubuntu@ip-172-31-8-158:~$
```

Now go on loadbalancer section on aws & check this.





now use dns name & check paths

http://k8s-default-pathbase-309efd8ab3-775818215.ap-south-1.elb.amazonaws.com
http://k8s-default-pathbase-309efd8ab3-775818215.ap-south-1.elb.amazonaws.com/hi
http://k8s-default-pathbase-309efd8ab3-775818215.ap-south-1.elb.amazonaws.com/bye

Everything is working fine

note :if you face 404 our 503 eror so you must mistake on nginx.conf file define path our make mistake in DOcker file at write commands to create path in /var/www/html/hi our bye directory for put index.html page so fix this issue by verified check logs of pods of deployment.

Example
Suppose face eror /hi so hi deployment is sahil-deployment so check logs of pod