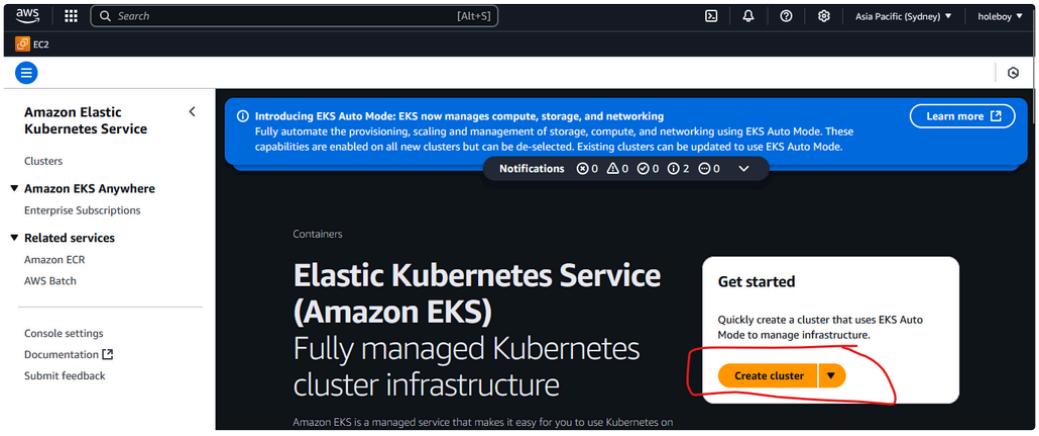
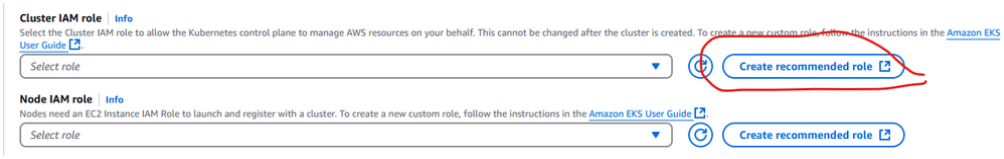
AWS Training: Basic AWS EKS deployment with NGINX

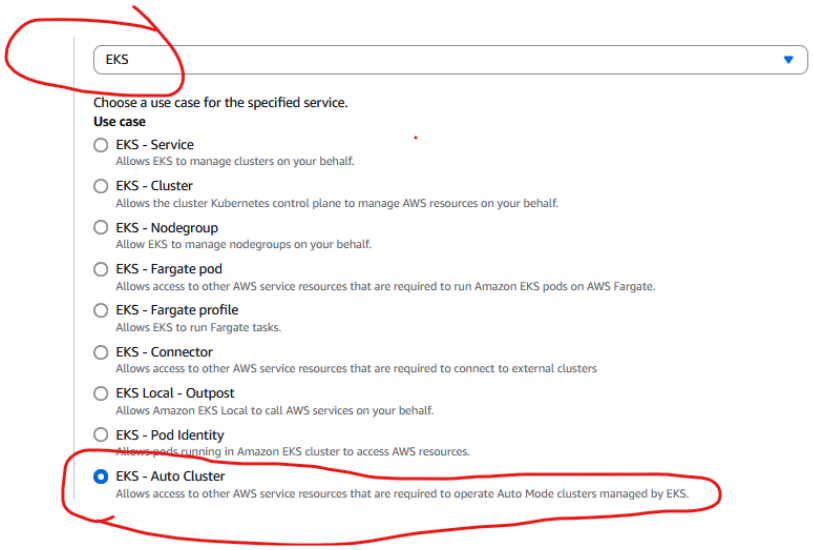
Create Kubernetes cluster

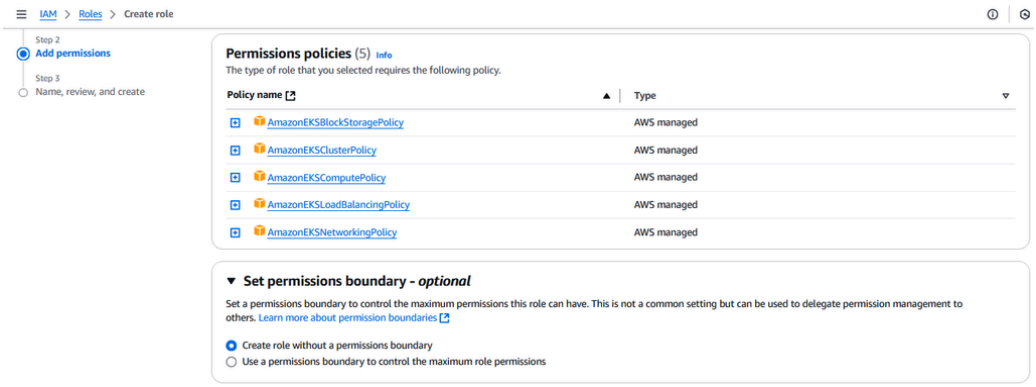
Create cluster



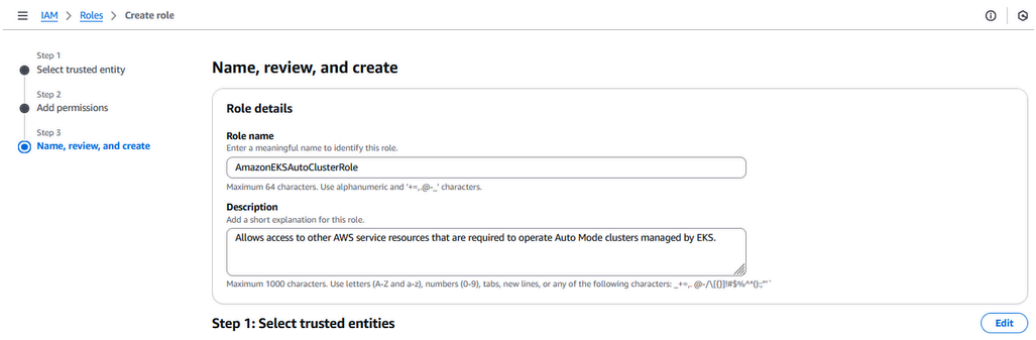
Create cluster IAM role



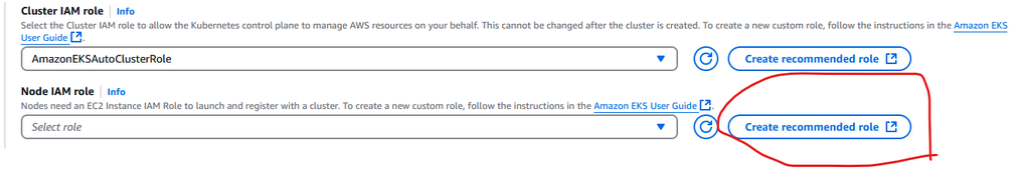


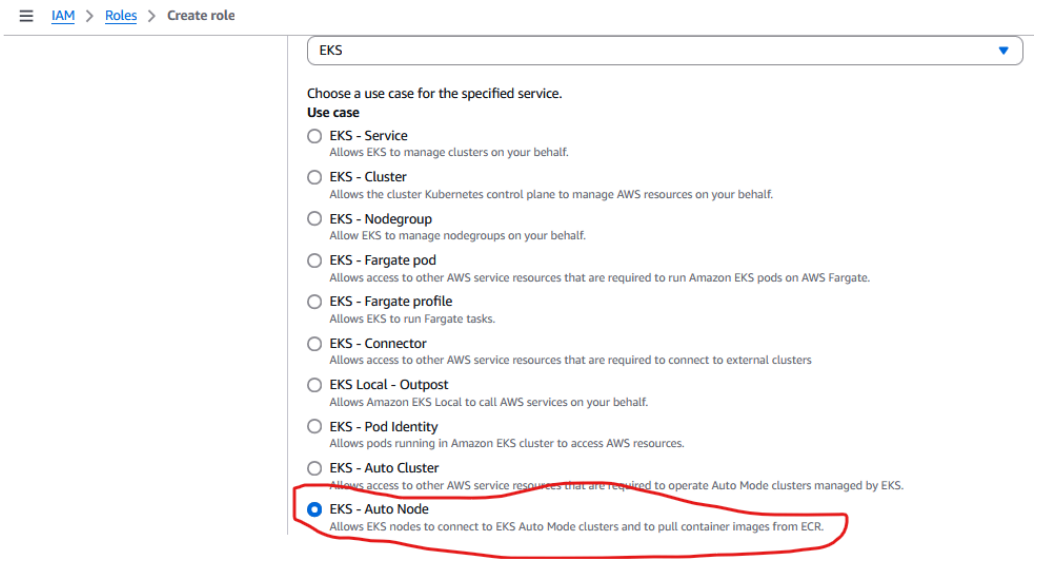


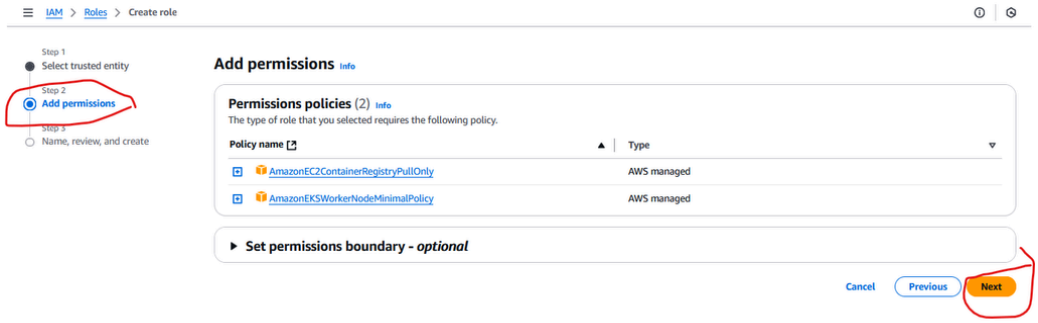
Use defaults if you don’t know what you’re doing

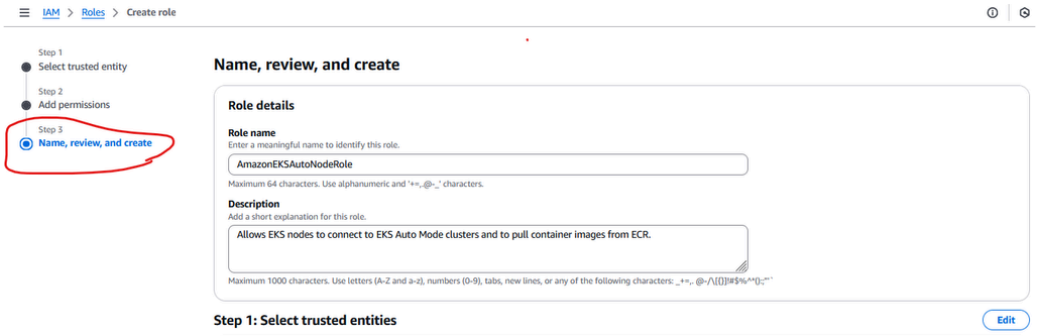


Create node IAM role

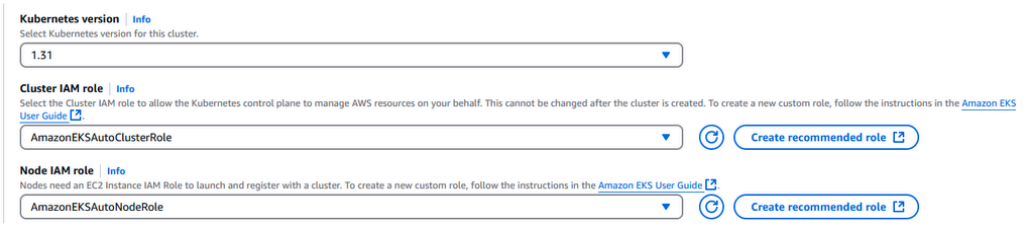


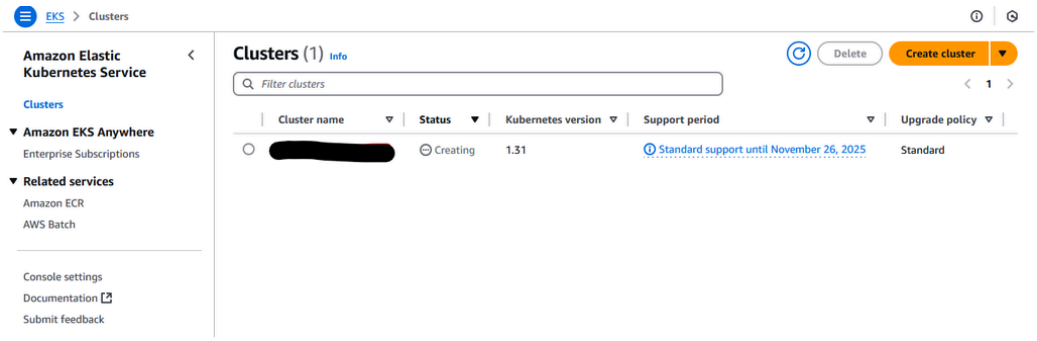




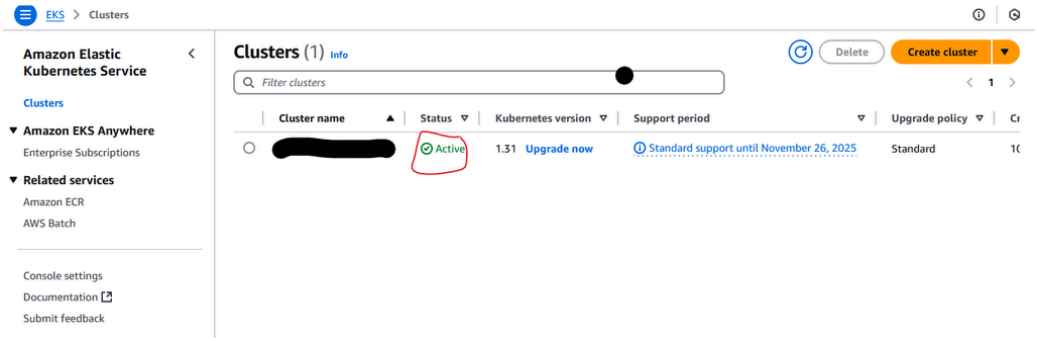


Create a cluster



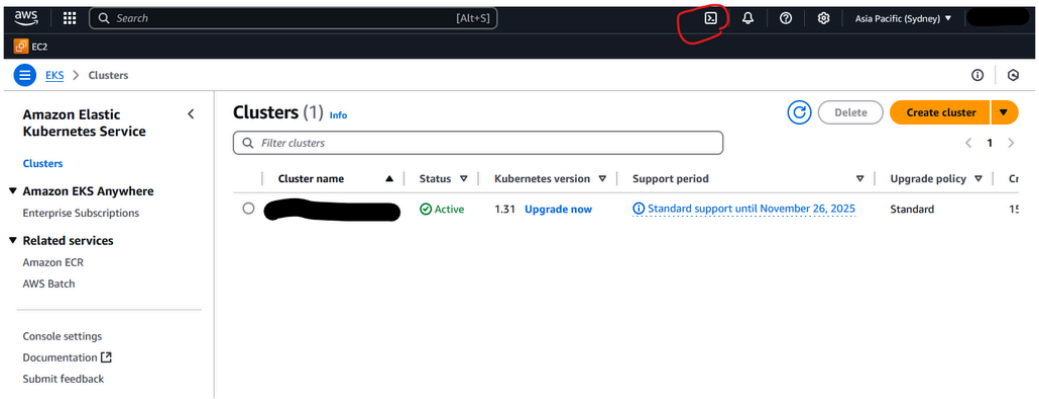


It can take up to 10-15 mins to provision a Kubernetes cluster



Create kubeconfig

Enter AWS cloudshell



Create a kubeconfig file using the AWS CLI

```
# Change region-code and my-cluster values
aws eks update-kubeconfig --region region-code --name my-cluster

# Test the connection
kubectl get svc
```

NGINX manifest (ClusterIP)

See Deploy a sample application on Linux - Amazon EKS

Create namespace:

```
kubectl create namespace eks-example
```

nginx-deployment.yaml

```
cat <<EOF > nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: eks-example
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
EOF

kubectl apply -f nginx-deployment.yaml
kubectl get pods -l 'app=nginx' -o wide -n eks-example
```

nginx-service.yaml

```
cat <<EOF > nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: eks-example
  labels:
    app: nginx
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
```

```
      port: 80
      targetPort: 80
EOF

kubectl apply -f nginx-service.yaml
kubectl get service/nginx-service -n eks-example

kubectl -n default describe service nginx-service

kubectl get pods -l 'app=nginx' -o wide -n eks-example # Get a POD_ID
kubectl -n eks-example describe pod nginx-deployment-${POD_ID}

kubectl get all -n eks-example
```

Enter the pod

```
kubectl exec -it nginx-deployment-${POD_ID} -n eks-example -- /bin/bash

# Inside the pod
curl nginx-service

cat /etc/resolv.conf
```

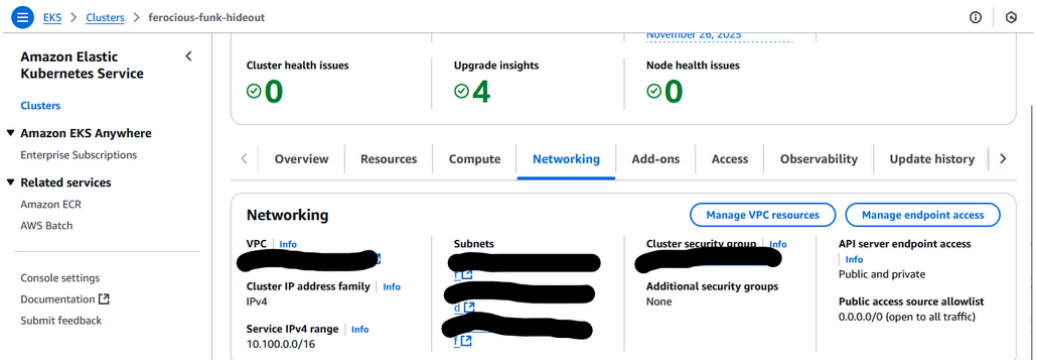
Clean up everything

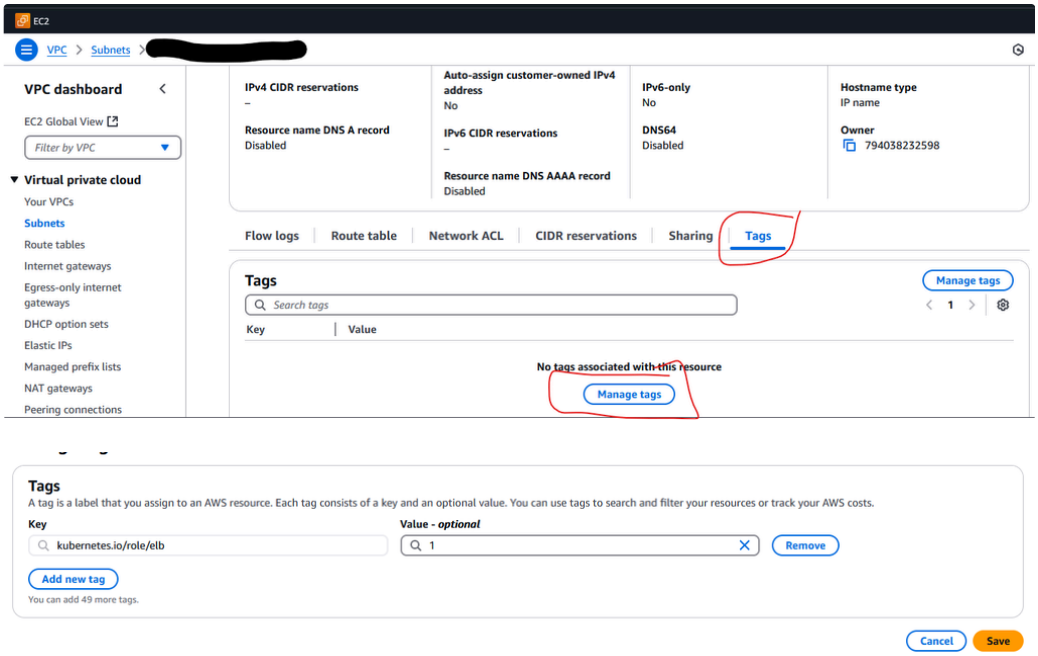
```
exit # If still inside the pod
kubectl delete ns eks-example
```

NGINX manifest (LoadBalancer)

See Route TCP and UDP traffic with Network Load Balancers - Amazon EKS

Tag subnets if needed





- For public subnets:
  - Key: `kubernetes.io/role/elb`
  - Value: `1`
- For private subnets:
  - Key: `kubernetes.io/role/internal-elb`
  - Value: `1`

NGINX Config

Create namespace:

```
kubectl create namespace eks-example
```

nginx-deployment.yaml

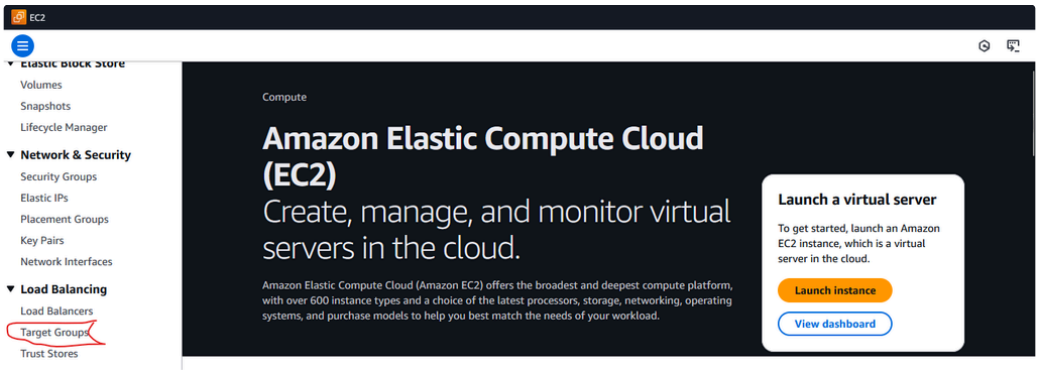
```
cat <<EOF > nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: eks-example
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
```

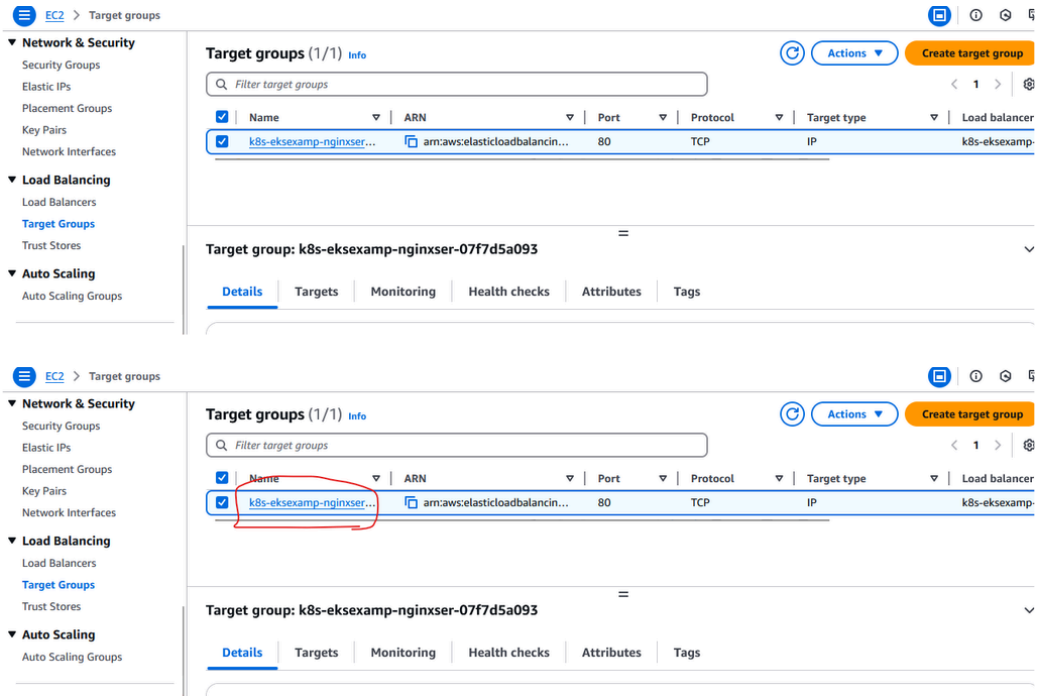
```
          ports:
            - name: tcp
              containerPort: 80
EOF

kubectl apply -f nginx-deployment.yaml
kubectl get pods -l 'app=nginx' -o wide -n eks-example
```

nginx-service.yaml

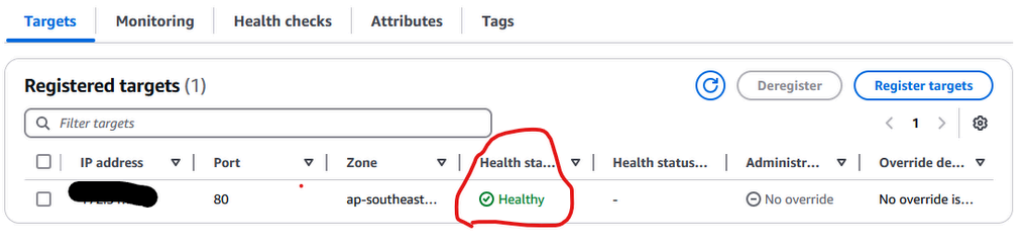
```
cat <<EOF > nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: eks-example
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
EOF

kubectl apply -f nginx-service.yaml
kubectl get all -n eks-example
kubectl get svc nginx-service -n eks-example
```

Open EC2 AWS Management console and select Target Groups under Load Balancing:
https://console.aws.amazon.com/ec2





Visit Registered targets, and wait until the status is healthy:



Then run the following in the CloudShell console:

```
curl k8s-eksexamp-nginxser-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.${REGION_CODE}.amazonaws.com
```

You can also visit the above link in your browser

Clean up everything

```
kubectl delete ns eks-example
```

Don’t forget to delete your cluster after testing