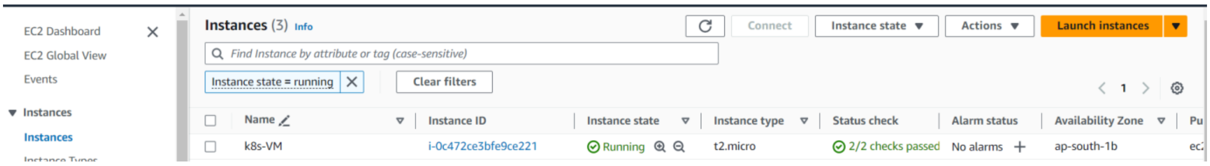
K8s Setup and deployment of our web application

Step-1) Create a new Ec2-instance with ubuntu VM



□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pu
□	k8s-VM	i-0c472ce3bfe9ce221	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms	ap-south-1b	ec

Step-2) Install Kubectl in the Ubuntu VM which you have created just now

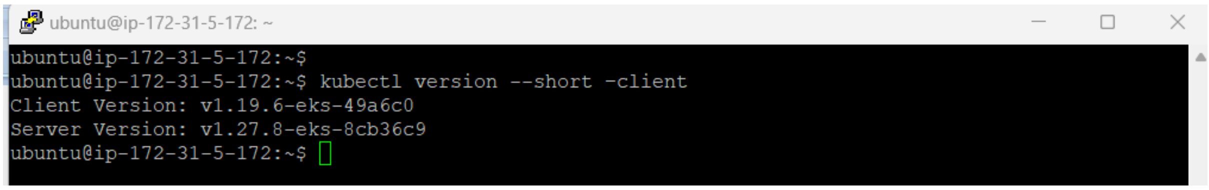
To install kubectl, run the following command one by one

$ curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl

$ chmod +x ./kubectl

$ sudo mv ./kubectl /usr/local/bin

$ kubectl version --short –client



```
ubuntu@ip-172-31-5-172:~$
ubuntu@ip-172-31-5-172:~$ kubectl version --short -client
Client Version: v1.19.6-eks-49a6c0
Server Version: v1.27.8-eks-8cb36c9
ubuntu@ip-172-31-5-172:~$
```

Step-3) Install AWS-CLI

To install Aws-Cli, run the following command one by one

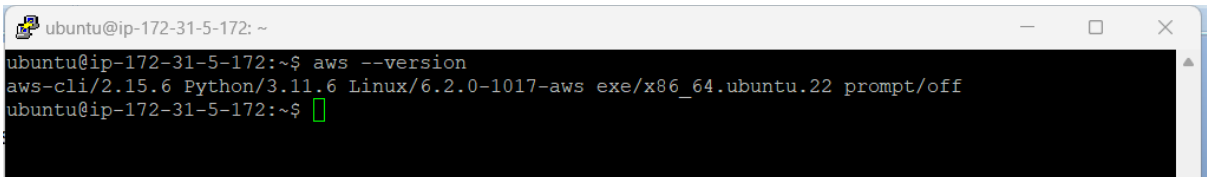
$ sudo apt install unzip

$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

$ unzip awscliv2.zip

$ sudo ./aws/install

$ aws –version



```
ubuntu@ip-172-31-5-172:~$ aws --version
aws-cli/2.15.6 Python/3.11.6 Linux/6.2.0-1017-aws exe/x86_64.ubuntu.22 prompt/off
ubuntu@ip-172-31-5-172:~$
```

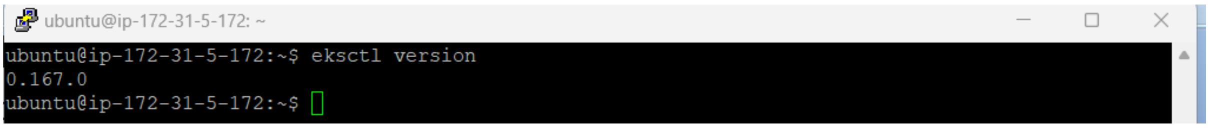
Step-4) Install eksctl

To install eksctl, run the following command one by one

$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

$ sudo mv /tmp/eksctl /usr/local/bin

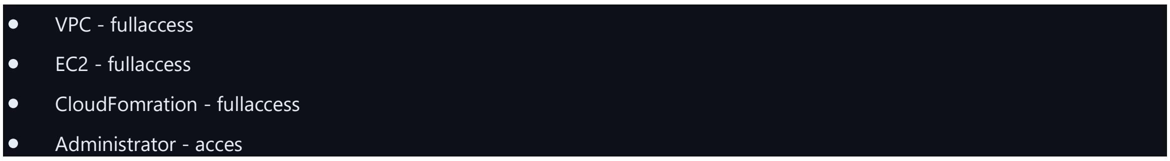
$ eksctl version



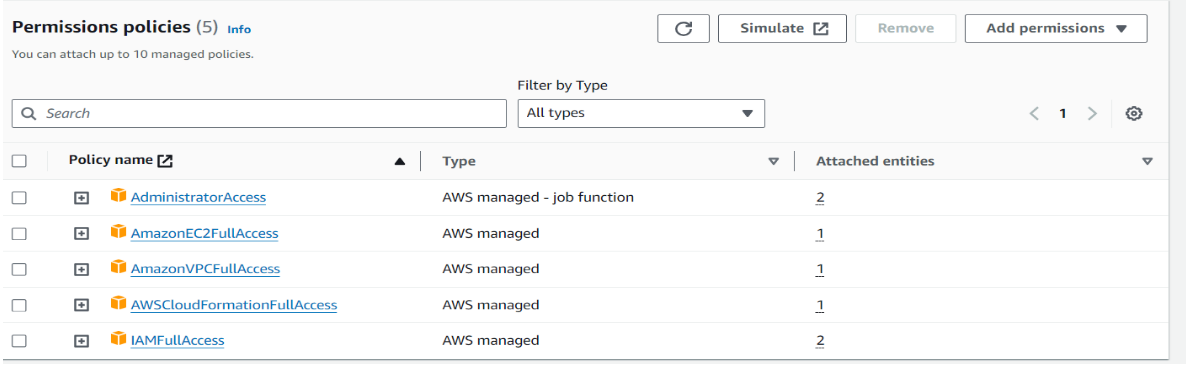
```
ubuntu@ip-172-31-5-172:~$ eksctl version
0.167.0
ubuntu@ip-172-31-5-172:~$
```

Step-5) Create Iam role

Create an Iam Role with the below permission and attach that role to Ubuntu-VM IAM - fullaccess



- VPC - fullaccess
- EC2 - fullaccess
- CloudFomration - fullaccess
- Administrator - acces



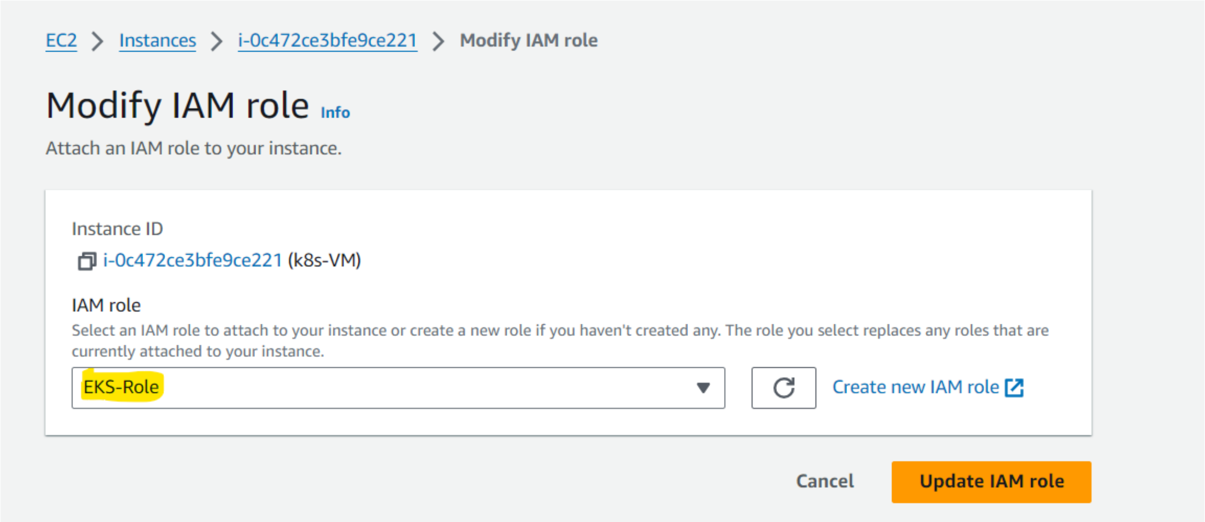
Permissions policies (5) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Search

□	Policy name	Type	Attached entities
□	AdministratorAccess	AWS managed - job function	2
□	AmazonEC2FullAccess	AWS managed	1
□	AmazonVPCFullAccess	AWS managed	1
□	AWSCloudFormationFullAccess	AWS managed	1
□	IAMFullAccess	AWS managed	2



EC2 > Instances > i-0c472ce3bfe9ce221 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-0c472ce3bfe9ce221 (k8s-VM)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

EKS-Role

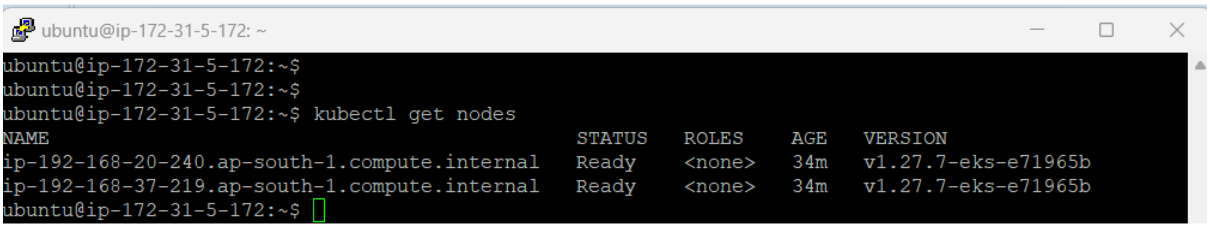
Step-6) Create Eks Cluster

$ eksctl create cluster --name ashray-cluster4 --region ap-south-1 --node-type t2.medium --zones ap-south-1a,ap-south-1b

Note: Cluster Creation may take 15-20 mins

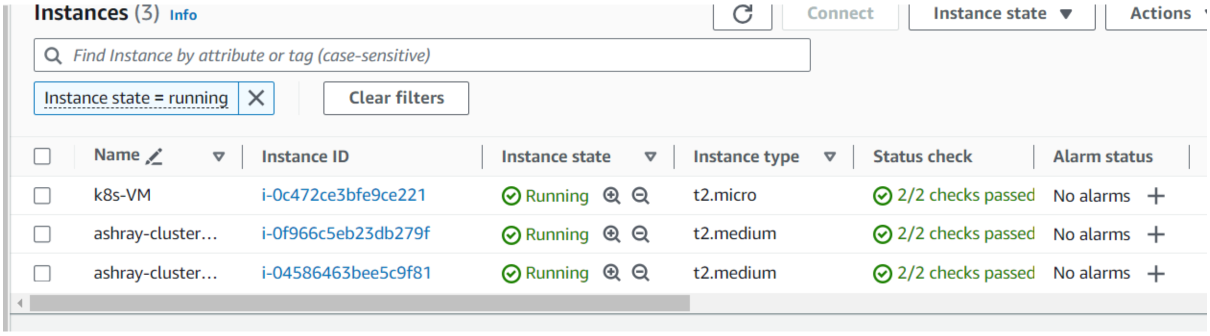
To check if cluster got created run the below command

$ Kubectl get node



```
ubuntu@ip-172-31-5-172:~$
ubuntu@ip-172-31-5-172:~$
ubuntu@ip-172-31-5-172:~$ kubectl get nodes
NAME                                            STATUS   ROLES    AGE   VERSION
ip-192-168-20-240.ap-south-1.compute.internal   Ready    <none>   34m   v1.27.7-eks-e71965b
ip-192-168-37-219.ap-south-1.compute.internal   Ready    <none>   34m   v1.27.7-eks-e71965b
ubuntu@ip-172-31-5-172:~$
```

Once the Cluster get created, check if you have two nodes created by cluster.
This is default number of node created by cluster. We can increase the number of instance if needed.



□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
□	k8s-VM	i-0c472ce3bfe9ce221	Running	t2.micro	2/2 checks passed	No alarms
□	ashray-cluster…	i-0f966c5eb23db279f	Running	t2.medium	2/2 checks passed	No alarms
□	ashray-cluster…	i-04586463bee5c9f81	Running	t2.medium	2/2 checks passed	No alarms

Step-7) Create pod

Now to create a pod we have to write a pod-manifest.yam

It should be written in Yaml just like we write ansible-playbook

Pod manifest should always start with --- and end with …

Vi my-pod.yml

```
---

apiVersion: v1

kind: Pod

metadata:

  name: mywebapppod
```

labels:

app: mywebapp

spec:

containers:

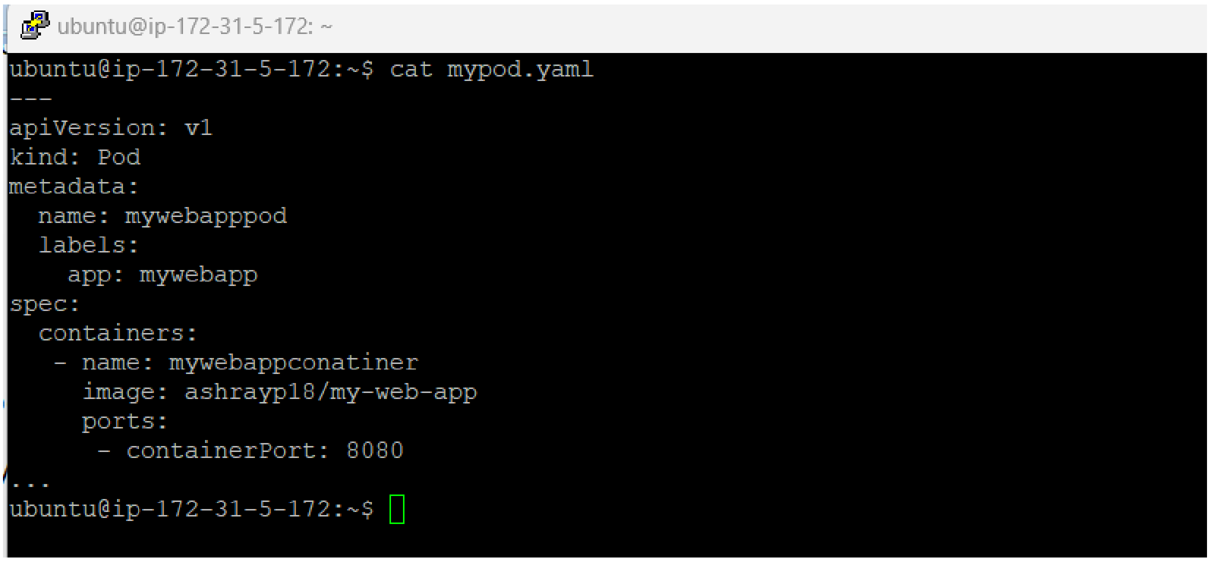
- name: mywebappconatiner

image: ashrayp18/my-web-app

ports:

- containerPort: 8080

...



```
ubuntu@ip-172-31-5-172:~$ cat mypod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: mywebapppod
  labels:
    app: mywebapp
spec:
  containers:
   - name: mywebappconatiner
     image: ashrayp18/my-web-app
     ports:
      - containerPort: 8080
...
ubuntu@ip-172-31-5-172:~$
```

Step-8) Creating Service manifest

Now we have created a Pod, but when we will run our pod we will not be able to access our website as our container image will be inside a pod.

So to make our instance communicate with pod it is necessary to write a service manifest.yaml file.

Service-manifest.yaml file will help us to access our application from outside world.

Vi svc-manifest.yaml

---

apiVersion: v1

kind: Service

metadata:

name: mywebappsvc

spec:

type: NodePort

selector:

app: mywebapp

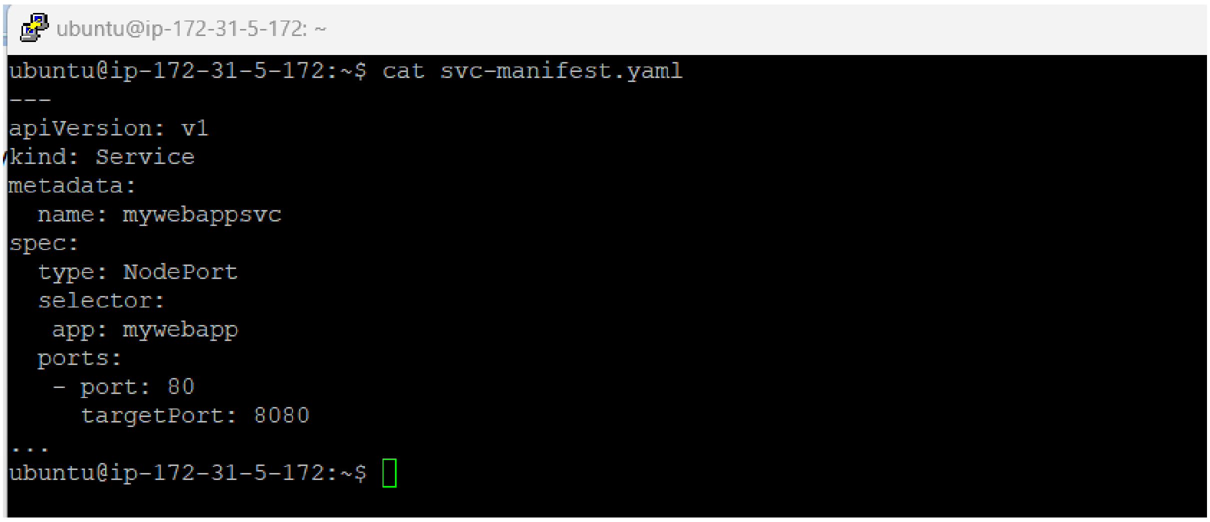
ports:

- port: 80

targetPort: 8080

...

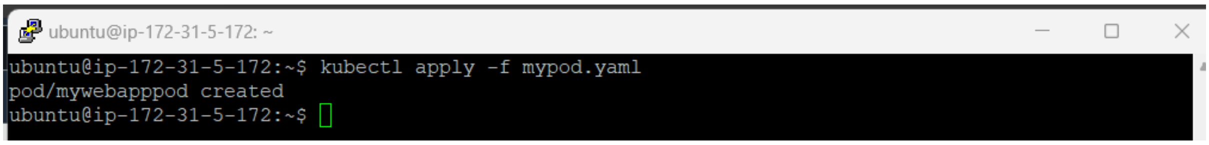
~



```
ubuntu@ip-172-31-5-172:~$ cat svc-manifest.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: mywebappsvc
spec:
  type: NodePort
  selector:
   app: mywebapp
  ports:
   - port: 80
     targetPort: 8080
...
ubuntu@ip-172-31-5-172:~$
```

Step-9) Run the pod

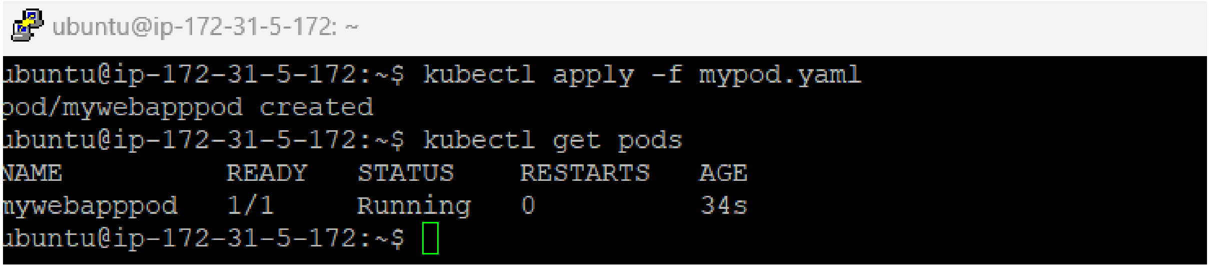
$ Kubectl apply -f podname.yaml



```
ubuntu@ip-172-31-5-172:~$ kubectl apply -f mypod.yaml
pod/mywebapppod created
ubuntu@ip-172-31-5-172:~$
```

Check it:

$ Kubectl get pods



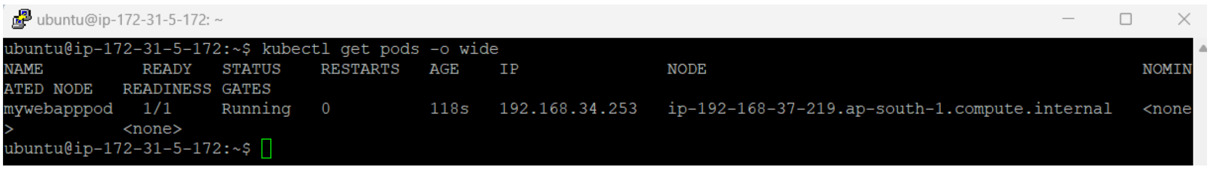
```
ubuntu@ip-172-31-5-172:~$ kubectl apply -f mypod.yaml
pod/mywebapppod created
ubuntu@ip-172-31-5-172:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
mywebapppod   1/1     Running   0          34s
ubuntu@ip-172-31-5-172:~$
```

For more info:

$ Kubectl describe pods

$ Kubectl get pods –o wide

This is very Imp Command as it is used to check on which node our pod is running.



ubuntu@ip-172-31-5-172: ~

```
ubuntu@ip-172-31-5-172:~$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE    IP               NODE                                            NOMIN
ATED NODE   READINESS GATES
mywebapppod   1/1     Running   0          118s   192.168.34.253   ip-192-168-37-219.ap-south-1.compute.internal   <none
>           <none>
ubuntu@ip-172-31-5-172:~$
```

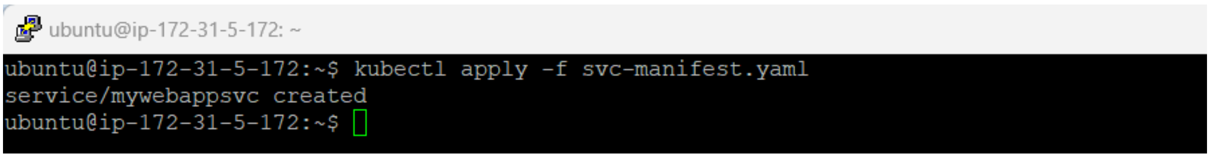
As seen below our pod is running on node with ip 192.68.37.219.

Check the IP and note down the Node-instance-name for this IP.

Step-10) Now we have to run the service

Run the service manifest.yaml file using the below command

$ Kubectl apply –f svc-manifest.yaml

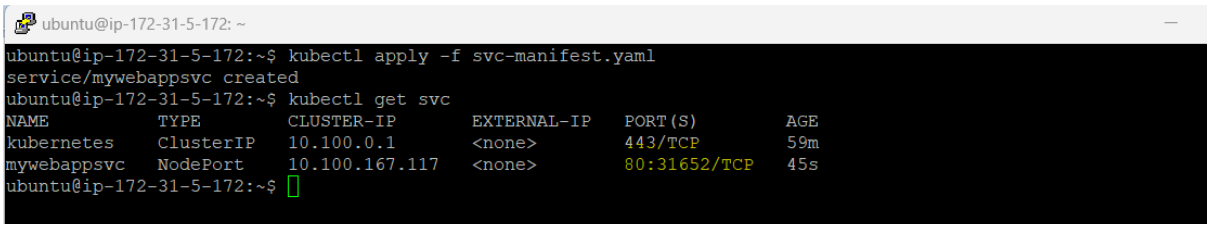


ubuntu@ip-172-31-5-172: ~

```
ubuntu@ip-172-31-5-172:~$ kubectl apply -f svc-manifest.yaml
service/mywebappsvc created
ubuntu@ip-172-31-5-172:~$
```

Service manifest has been created, now check it

$ Kubectl get svc



ubuntu@ip-172-31-5-172: ~

```
ubuntu@ip-172-31-5-172:~$ kubectl apply -f svc-manifest.yaml
service/mywebappsvc created
ubuntu@ip-172-31-5-172:~$ kubectl get svc
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes    ClusterIP   10.100.0.1       <none>        443/TCP        59m
mywebappsvc   NodePort    10.100.167.117   <none>        80:31652/TCP   45s
ubuntu@ip-172-31-5-172:~$
```

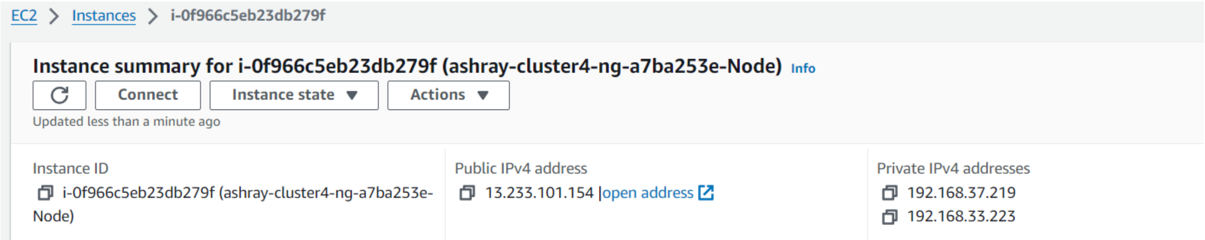
As we can see our svc are up and running with port number 31652.

Step-11) Edit Security Group

Now go to security group of node with IP (192.68.37.219) (This we got from step number 9 on which node our pod is running)

When the cluster was created, node with SG have been created as well automatically.

Now edit the inbound rule of SG attached to Node1 (192.168.37.219) and add the above port number.



EC2 > Instances > i-0f966c5eb23db279f

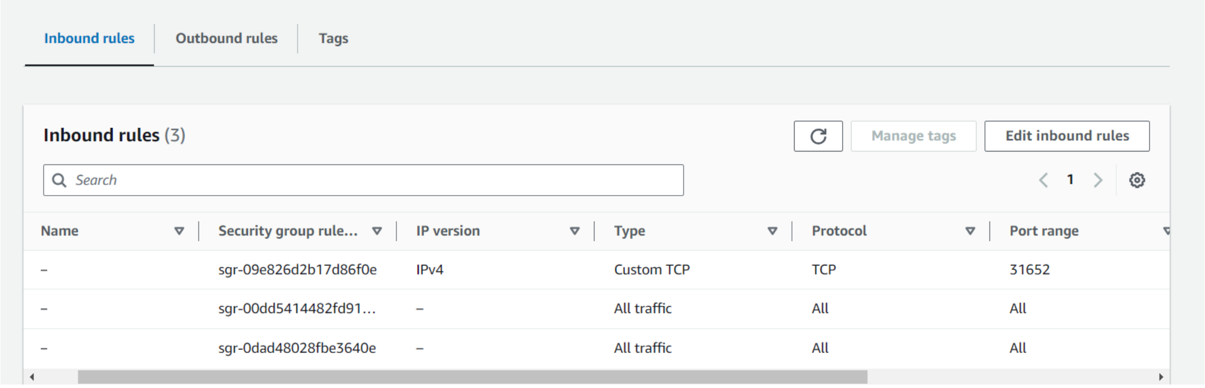
Instance summary for i-0f966c5eb23db279f (ashray-cluster4-ng-a7ba253e-Node) Info

Updated less than a minute ago

Instance ID: i-0f966c5eb23db279f (ashray-cluster4-ng-a7ba253e-Node)

Public IPv4 address: 13.233.101.154 |open address

Private IPv4 addresses: 192.168.37.219, 192.168.33.223

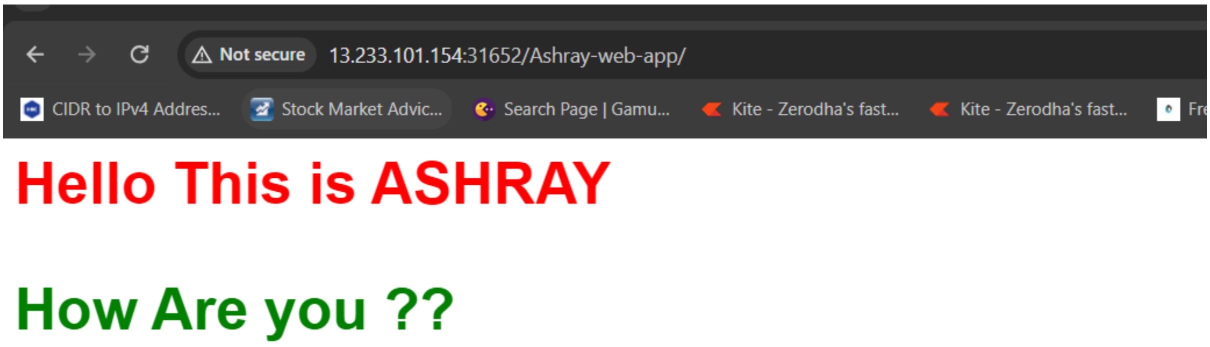


Inbound rules (3)

Name	Security group rule...	IP version	Type	Protocol	Port range
–	sgr-09e826d2b17d86f0e	IPv4	Custom TCP	TCP	31652
–	sgr-00dd5414482fd91...	–	All traffic	All	All
–	sgr-0dad48028fbe3640e	–	All traffic	All	All

Step-12) now access the URL

url: node-instance-public-ip:above-node-port/appname



⚠ Not secure 13.233.101.154:31652/Ashray-web-app/

Hello This is ASHRAY

How Are you ??

Bingo!!!! Our app is up and running using k8s.