**Step-1**: IAM role/user with EC2, S3, ELB, ESB, etc

**Step-2**: Create EC2 instances with IAM role

In this example, I am choosing t2.medium for master, t2.micro for nodes.

**Step-3**: connect to all EC2 instances and update hostname

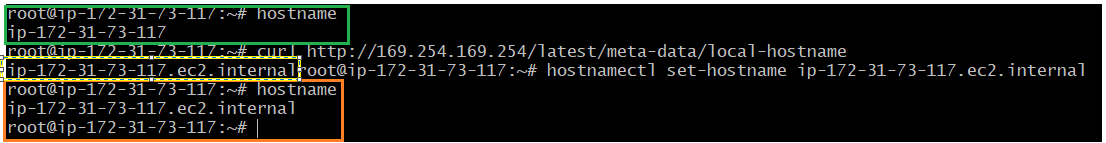


|  |
| --- |
| hostnamectl set-hostname <hostname> |

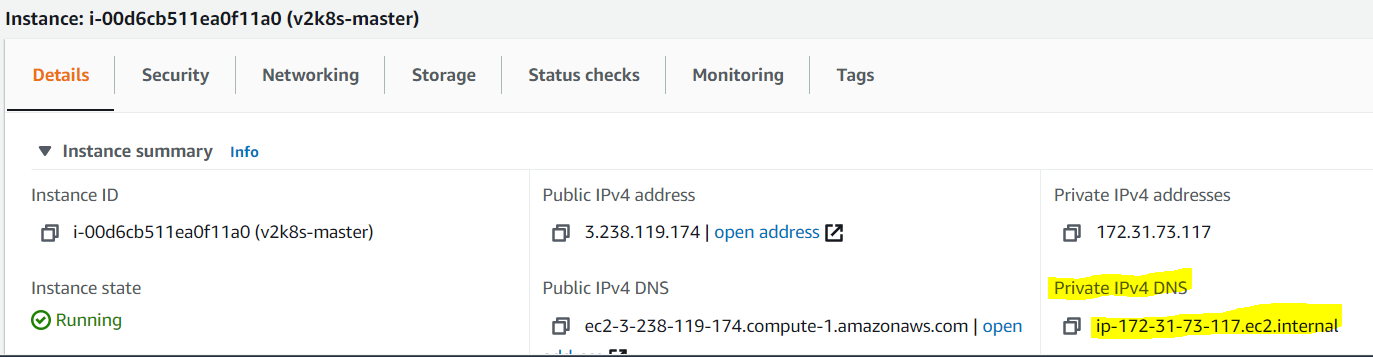
here hostname = Private IPv4 DNS ex: ip-172-31-59-217.ec2.internal

run the below curl command to get "Private IPv4 DNS"

|  |
| --- |
| curl http://169.254.169.254/latest/meta-data/local-hostname |



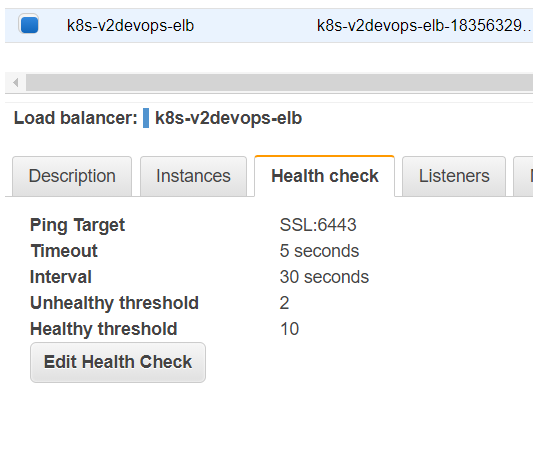
Or you can copy the “Private IPv4 DNS” from EC2 console also.

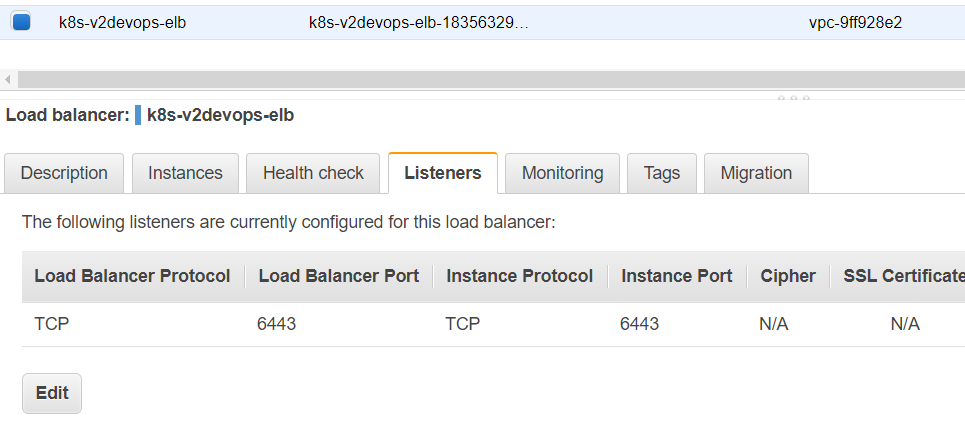


Repeat the above step-3 for other instances also.

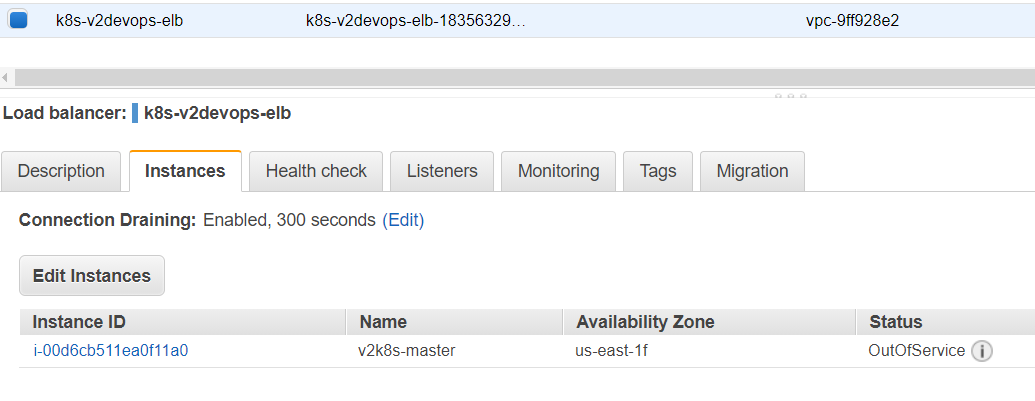
Step-4: Create a loadbalancer (classic) for master (control-plane)

k8s-v2devops-elb-1835632935.us-east-1.elb.amazonaws.com





As of now, Status is OutOfService. Once k8s cluster setup ready, status will be changed.

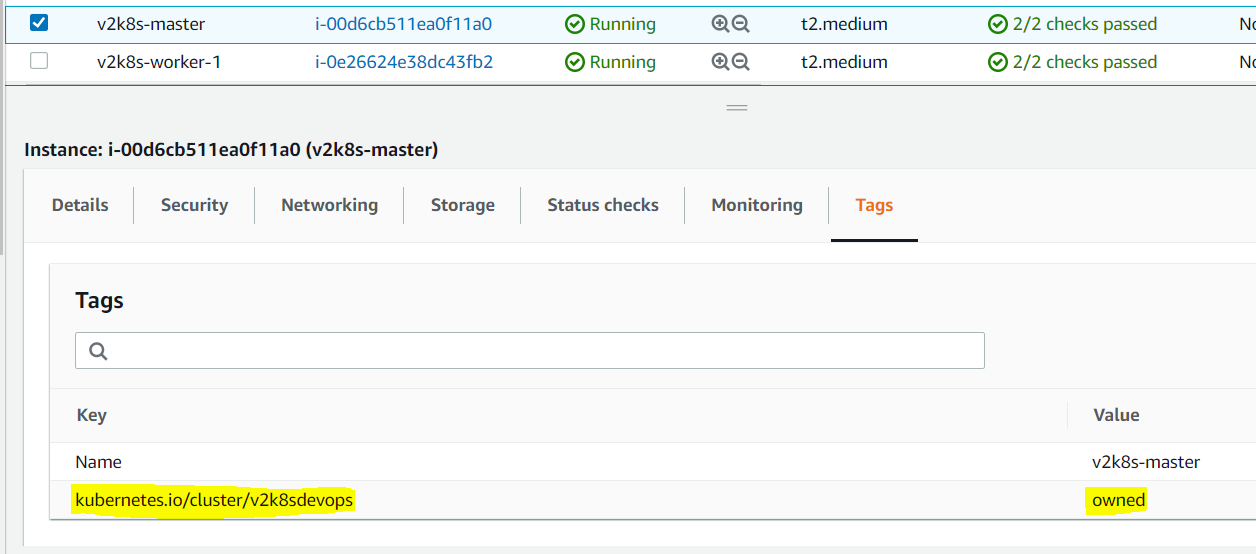


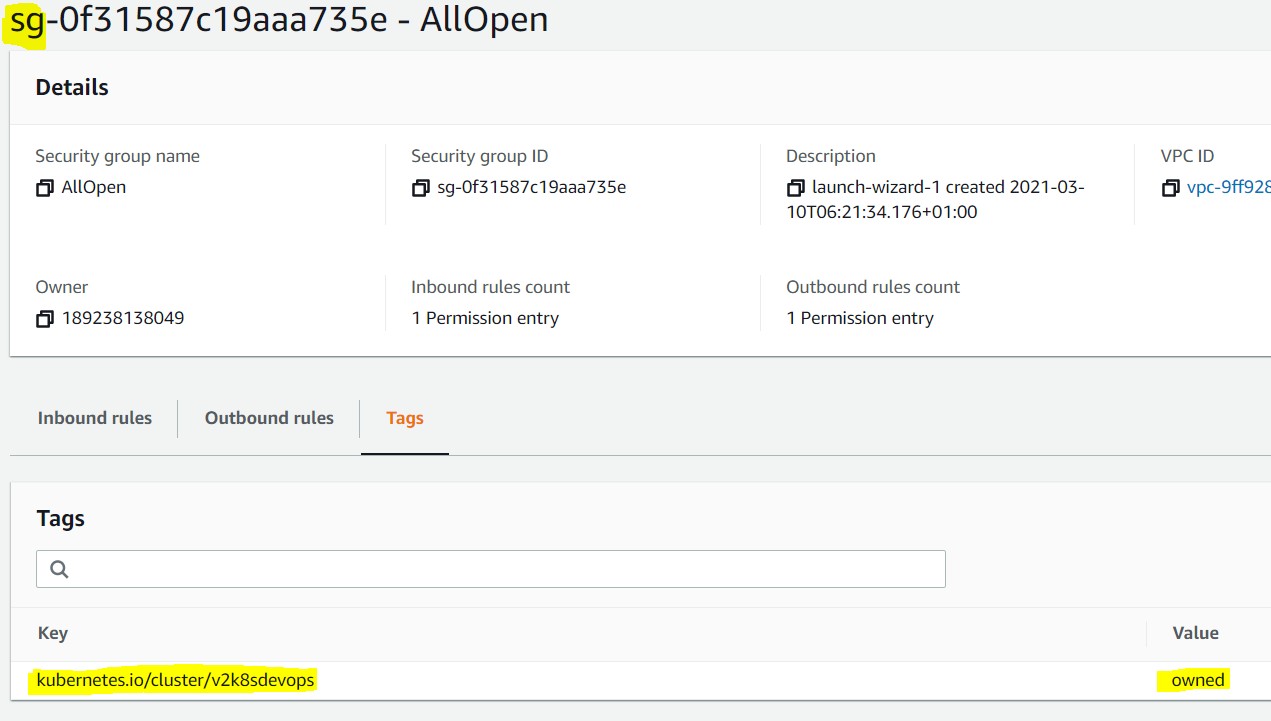
Step-5:

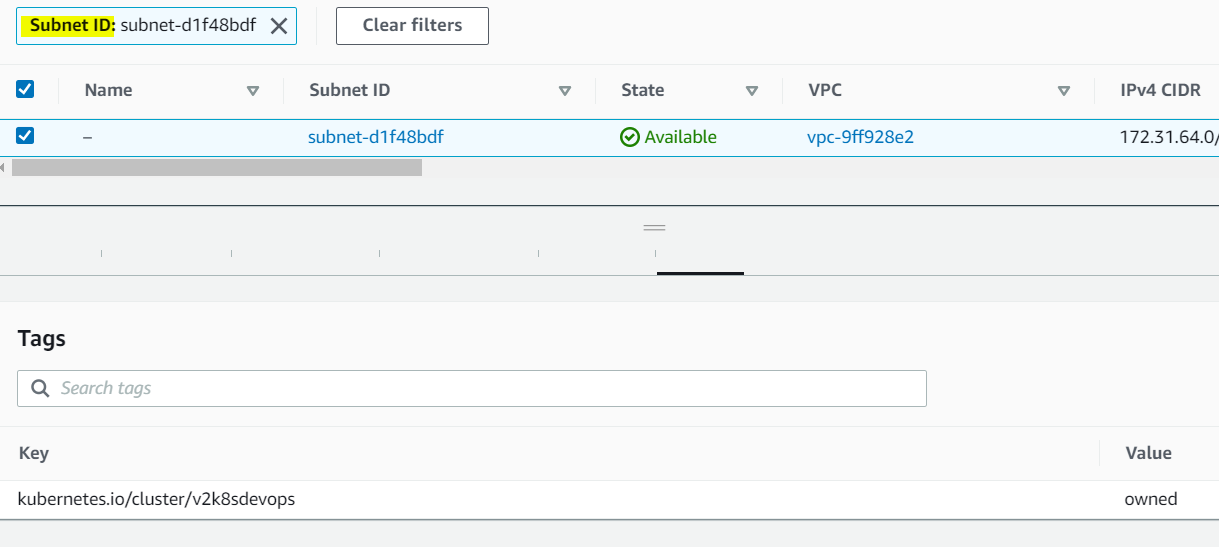
add a new tag to EC2, SG, Subnect, Role, VPC, ELB

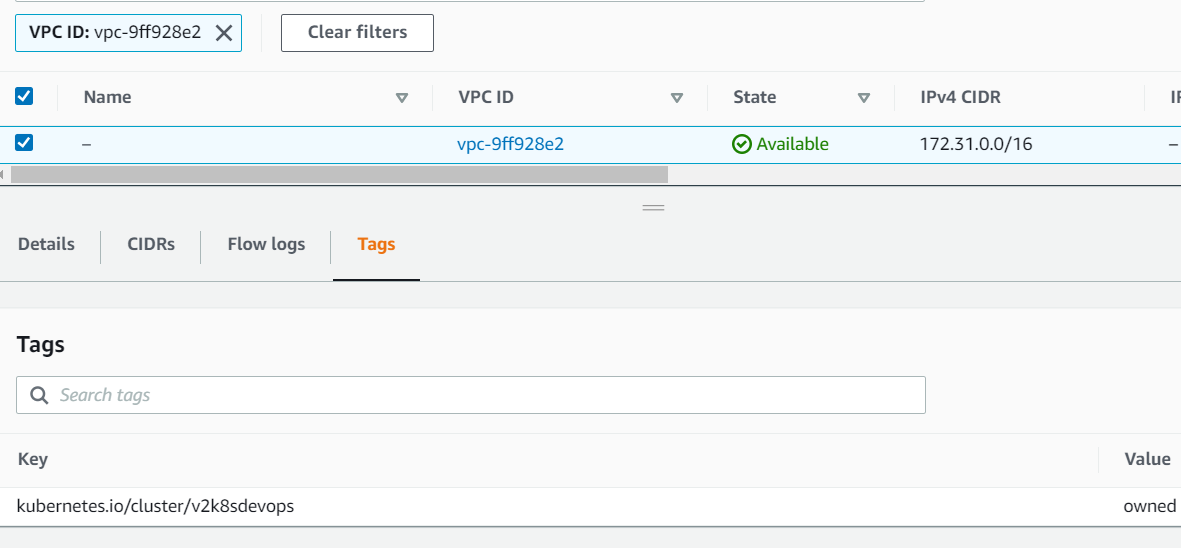
name: kubernetes.io/cluster/<CLUSTERNAME> ex: kubernetes.io/cluster/v2k8sdevops

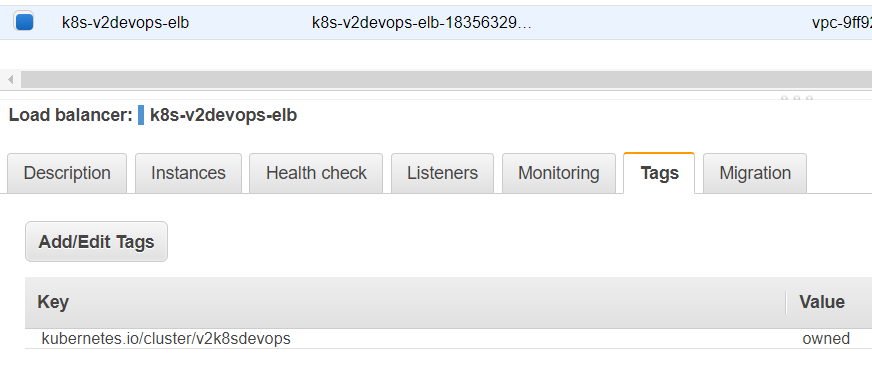
value: owned











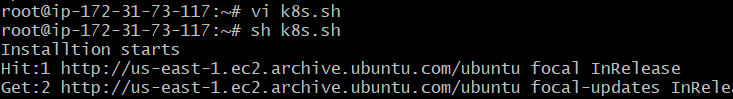
Step-6: Install the required tools: docker, kubelet, kubeadm, kubectl

Copy the commands from the section “Installation” from the GitHub file [1.Kubernetes-installation-examples.md](https://github.com/DevOpsPlatform/Phase-2/blob/master/kubernetes/multi-node-cluster/kubeadm/1.Kubernetes-installation-examples.md)

Create sh file and run it. (do the same this on all the EC2 instances which will be part of the k8s cluster

vi k8s.sh

paste the copied commands from the section “Installation” from the GitHub file [1.Kubernetes-installation-examples.md](https://github.com/DevOpsPlatform/Phase-2/blob/master/kubernetes/multi-node-cluster/kubeadm/1.Kubernetes-installation-examples.md)



Check all the tools are working or not

docker version

kubeadm version

kubectl version

Step-8: change the cgroup driver to system after docker installed (perform this on all EC2 instances)

cat > /etc/docker/daemon.json <<EOF

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

sudo systemctl restart docker

sudo systemctl enable docker

Step-8: Update the configuration of the kubelet service so that it knows about the AWS environment as well. Edit the /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

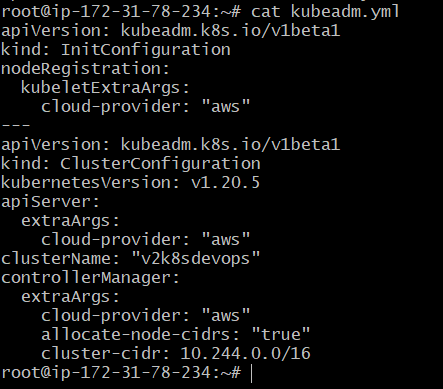
add --cloud-provider=aws as new line

systemctl daemon-reload

Step-9: Prepare kubeadm.yml configuration file. (you can copy the below file and update the cluster name and ELB name with your resource details.



|  |
| --- |
| apiVersion: kubeadm.k8s.io/v1beta1  kind: InitConfiguration  nodeRegistration:  kubeletExtraArgs:  cloud-provider: "aws"  ---  apiVersion: kubeadm.k8s.io/v1beta1  kind: ClusterConfiguration  kubernetesVersion: v1.20.5  apiServer:  extraArgs:  cloud-provider: "aws"  clusterName: "v2k8sdevops"  controllerManager:  extraArgs:  cloud-provider: "aws"  allocate-node-cidrs: "true"  cluster-cidr: 10.244.0.0/16 |



Step-10: Cluster setup

kubeadm config images pull

kubeadm init --config /path/kubeadm.yml

kubeinit failed with this reason and displaye the below command, generate new yml and run kubeadm init again.

kubeadm config migrate --old-config old.yaml --new-config new.yaml

ex: kubeadm config migrate --old-config /path/kubeadm.yml --new-config /path/kubeadm\_new.yml

kubeadm init --config /path/kubeadm\_new.yml



if anything goes wrong, try: kubeadm reset -f

sysctl net.bridge.bridge-nf-call-iptables=1

export kubever=$(kubectl version | base64 | tr -d '\n')

kubectl apply -f <https://cloud.weave.works/k8s/net?k8s-version=$kubever>

kubectl get nodes

kubectl get pods --all-namespaces

Join the nodes into the cluster using kubeadm join command. Use this command to create and print the join command : kubeadm token create --print-join-command

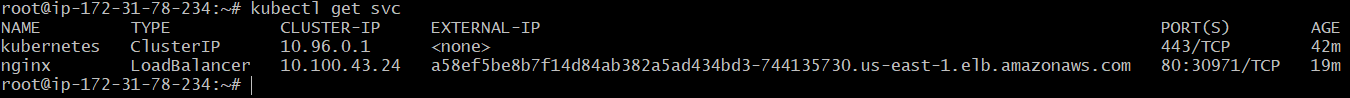
Step-11: deployment and create service

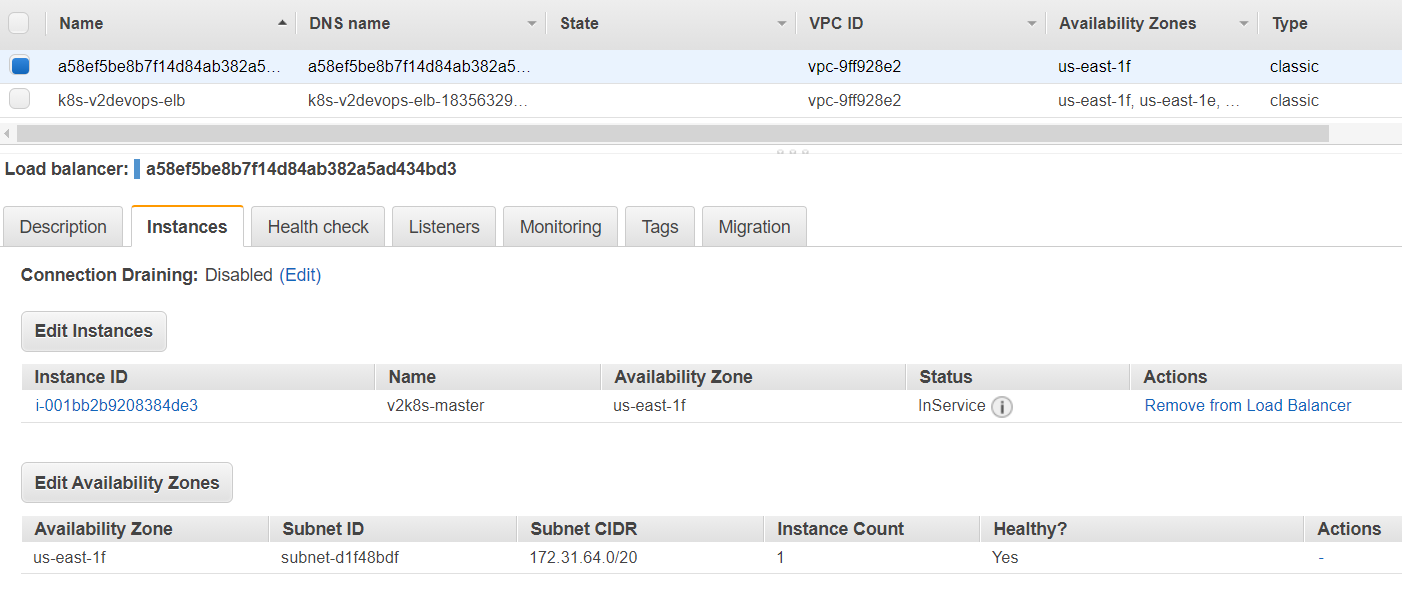
kubectl create deployment nginx --image=nginx

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name: nginx  spec:  selector:  app: nginx  ports:  - port: 80  targetPort: 80  type: LoadBalancer |

kubectl get svc: You noticed the ELB automatically created, Go to AWS and add any instance id from the k8s cluster to ELB - a58ef5be8b7f14d84ab382a5ad434bd3-744135730.us-east-1.elb.amazonaws.com

Or you can use AWSCLI.





Go to AWS and add any instance id from the k8s cluster to ELB - a58ef5be8b7f14d84ab382a5ad434bd3-744135730.us-east-1.elb.amazonaws.com

NO NEED TO UPDATE SECURITY GROUP with new outbound rules, AWS automatically set the rule as to open the TCP port at 80.

Then launch the URL in any browser

