Project: K8s on EC2 & Deploying via AWS ELB
(EC2에 K8s 설치 및 ELB와 연동)

Environment(환경): Ubuntu 20.04(EC2)

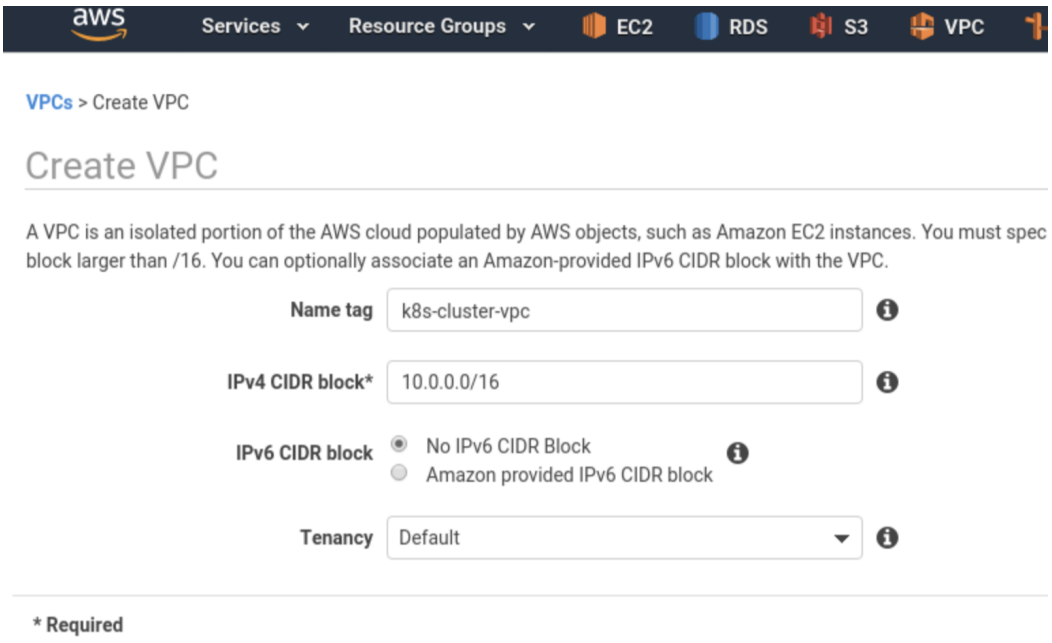
List to Create during Hands-on-Lab(오늘 Hands-on-Lab에서 생성할 목록)

- 1. VPC
- 2. Subnet(서브넷)
- 3. Internet Gateway(인터넷 게이트웨이)
- 4. Route Table(라우팅 테이블)
- 5. IAM Roles(IAM 역할)
    - Master Node Role
    - Worker Node Role
- 6. Running EC2(EC2 생성)
    - K8s Cluster Setup(쿠버네티스 클러스터 설정)
        - K8s installation(쿠버네티스 설치)
            - Hostname(호스트명)
            - Cluster Setup(클러스터 설정)
            - Kubeadm Reset(Kubeadm 초기화)
            - Flannel CNI Installation(Flannel 네트워크 플러그인 설치)
            - Attaching the Worker Node(Worker Node 연동)
            - Load Balancer Creation via K8s Service(쿠버네티스 서비스를 통한 AWS 로드밸런서 생성)
- 7. Troubleshooting(문제 해결)
    - AWS Load Balancer
        - When no Worker Node is added to the ELB(AWS 로드밸런서가 생성됐지만 Worker Node가 연결안되어 있을 때)

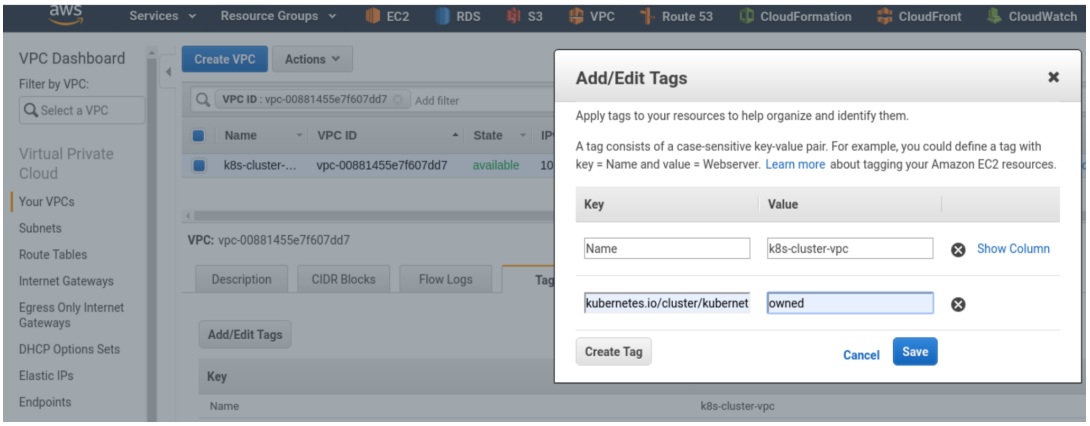
---

1. VPC

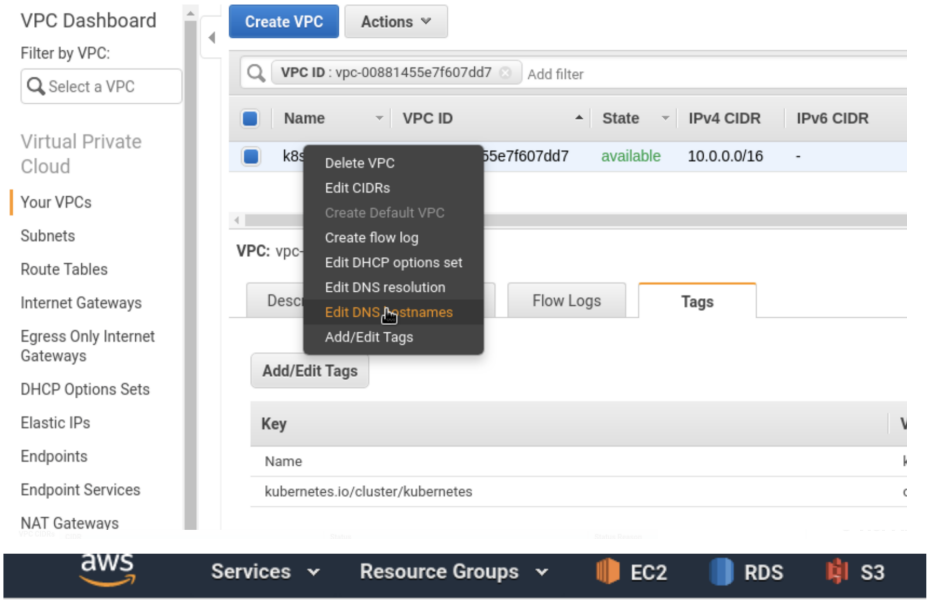
- Create a VPC with the 10.0.0.0/16 CIDR(10.0.0.0/16 대역대를 사용하는 VPC를 생성)

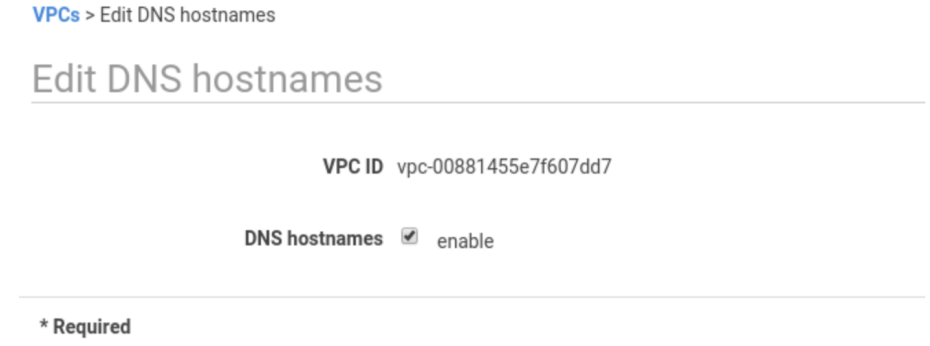


- Add a tag named ***kubernetes.io/cluster/kubernetes*** with the ***owned*** value - it will be used by K8s for AWS resources auto-discovery related to the Kubernetes stack, also it will add such a tag itself during creating new resources(쿠버네티스가 AWS 자원을 자동으로 발견할 수 있도록 태그 및 값을 지정
  태그: ***kubernetes.io/cluster/kubernetes***
  값: ***owned***
  추가적으로 새로운 리소스를 생성할 때 해당 태그를 생성)



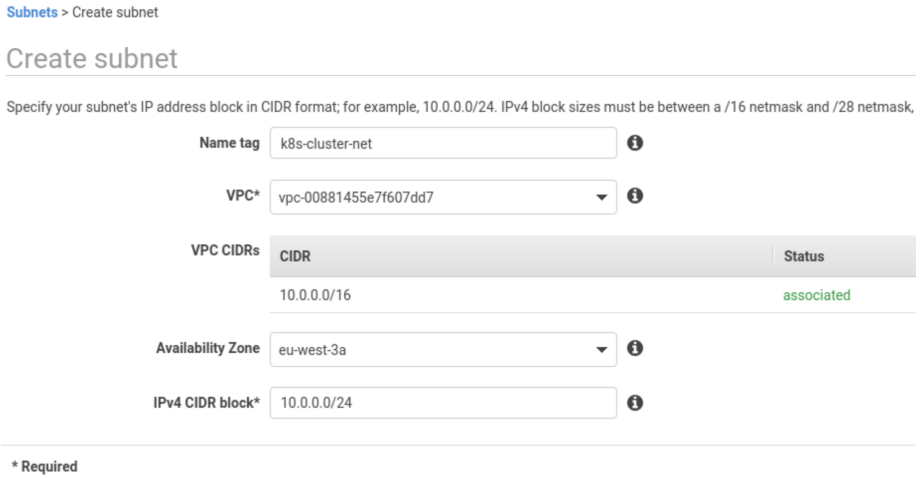
- Enable DNS hostname(DNS 호스트명 활성화)



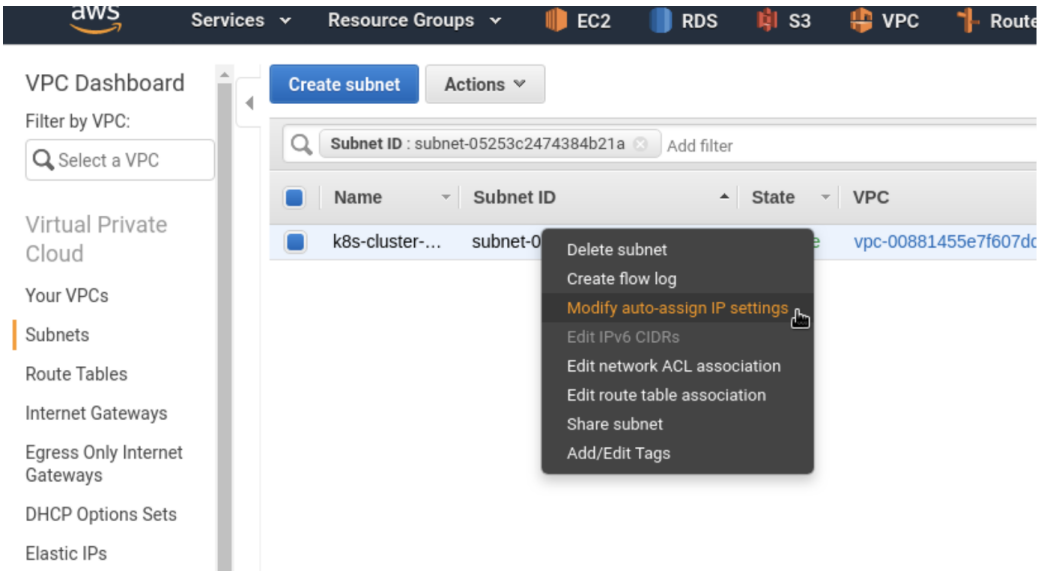


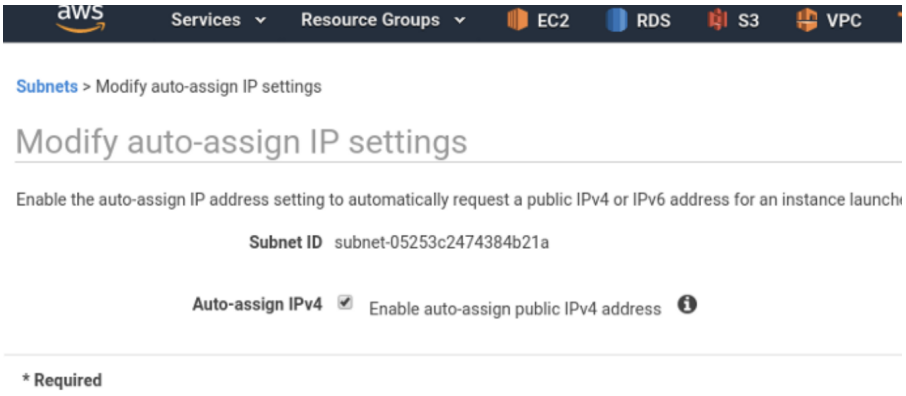
2. Subnet

- Create a new subnet in this VPC(VPC에 새로운 서브넷 생성)

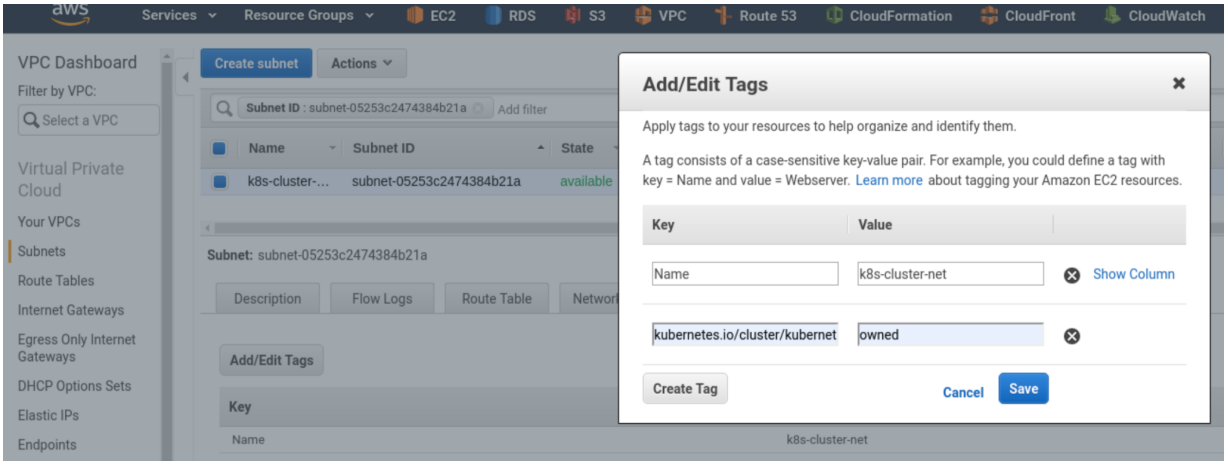


- Enable Public IPs for EC2 instances which will be placed in this subnet(해당 서브넷에 생성될 EC2 인스턴스들에 대해 공인 IP 부여를 허용)



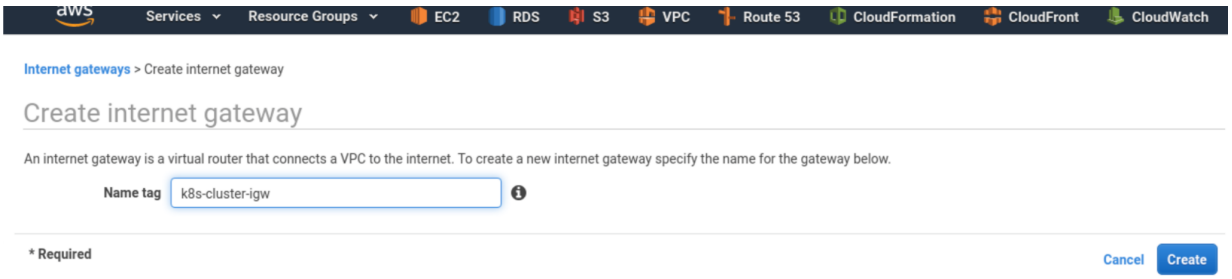


- Add the tag(태그 추가)

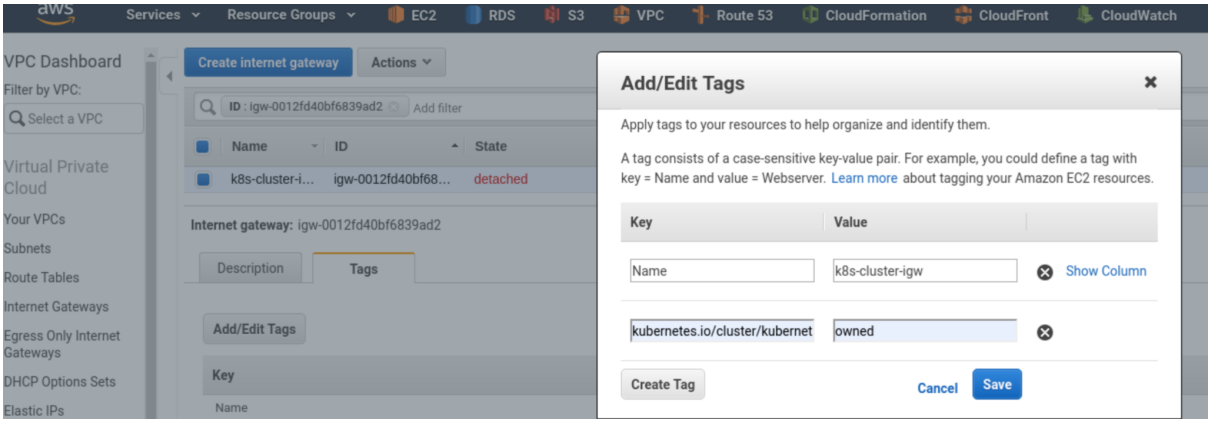


3. Internet Gateway

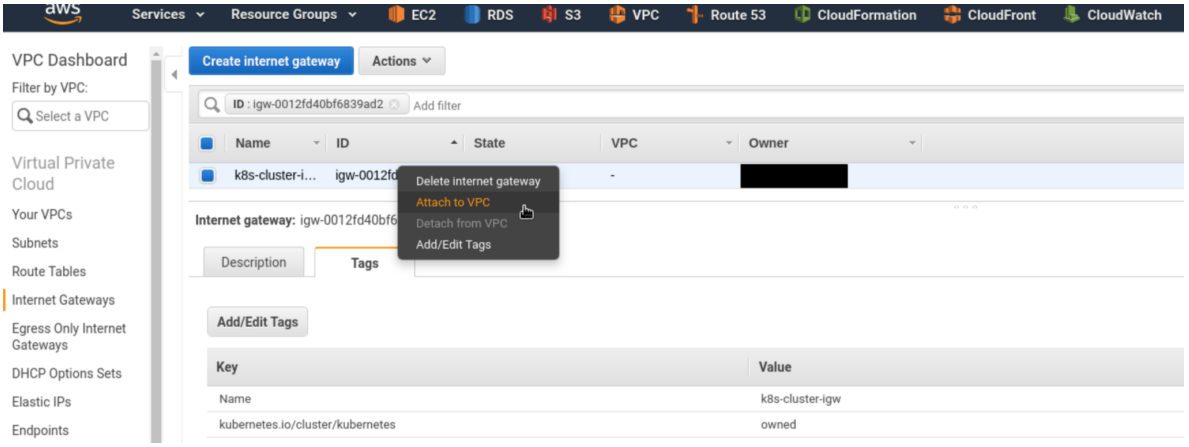
- Create an IGW to route traffic from the subnet into the Internet(서브넷에서 인터넷 통신을 받기 위해 인터넷 게이트웨이 생성)



- For IGW, add the tag as well, just in case(인터넷 게이트웨이에도 태그를 추가)



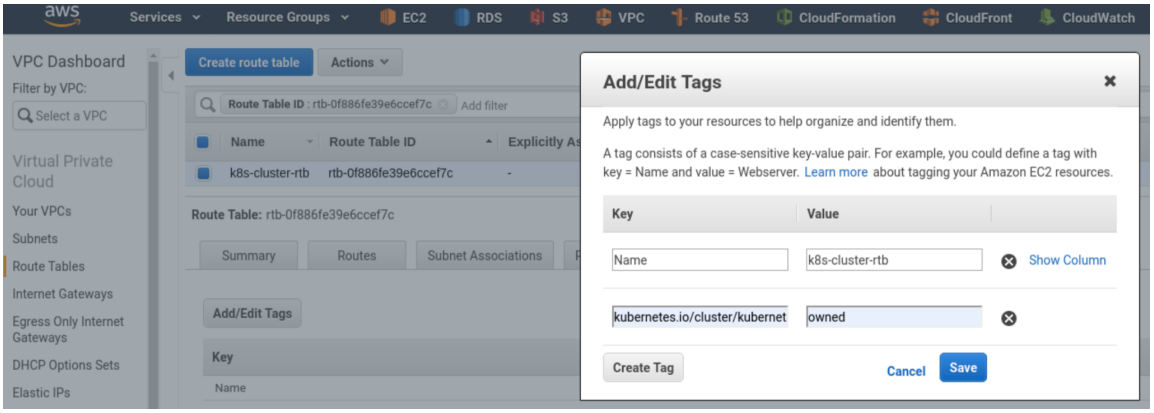
- Attach this IGW to your VPC(1번 단계에서 생성한 VPC에 인터넷 게이트웨이를 추가)



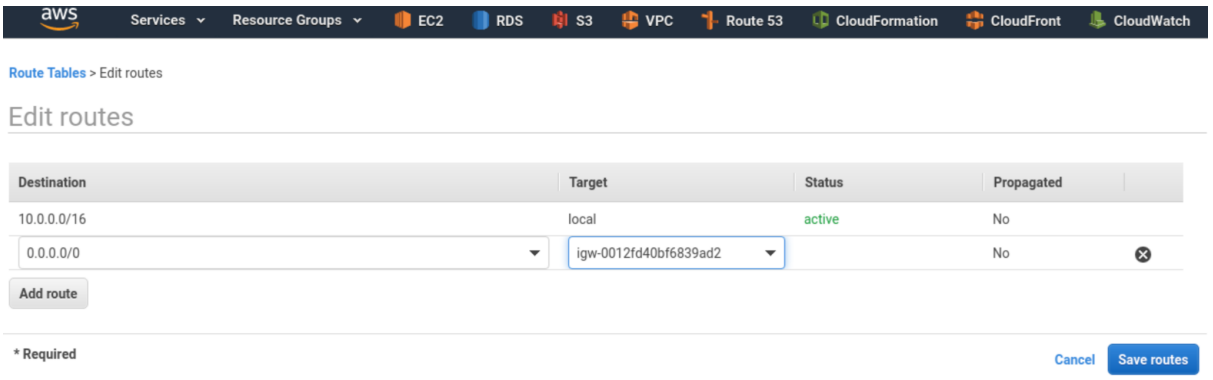
4. Route Table

- Create a routing table(라우팅 테이블 생성)

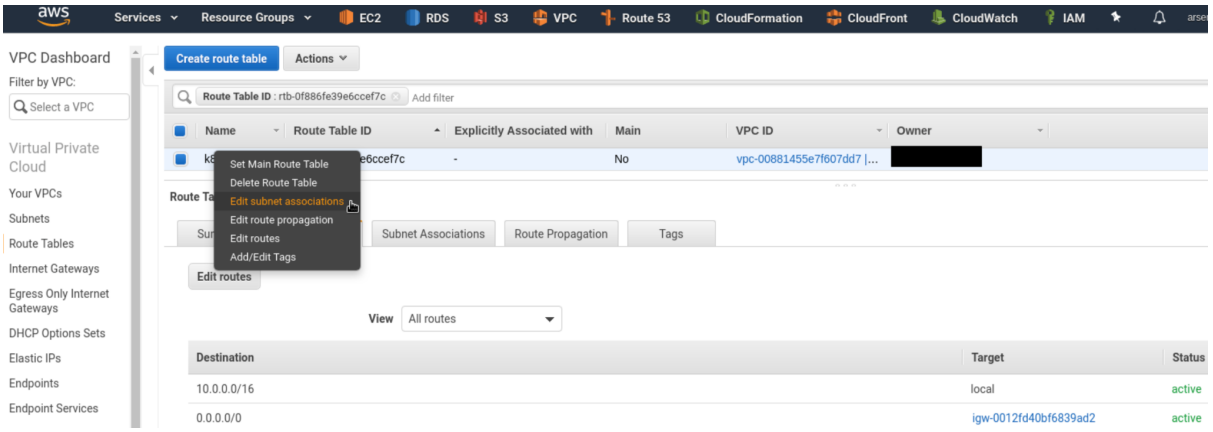
- Add the tag here(태그를 추가)



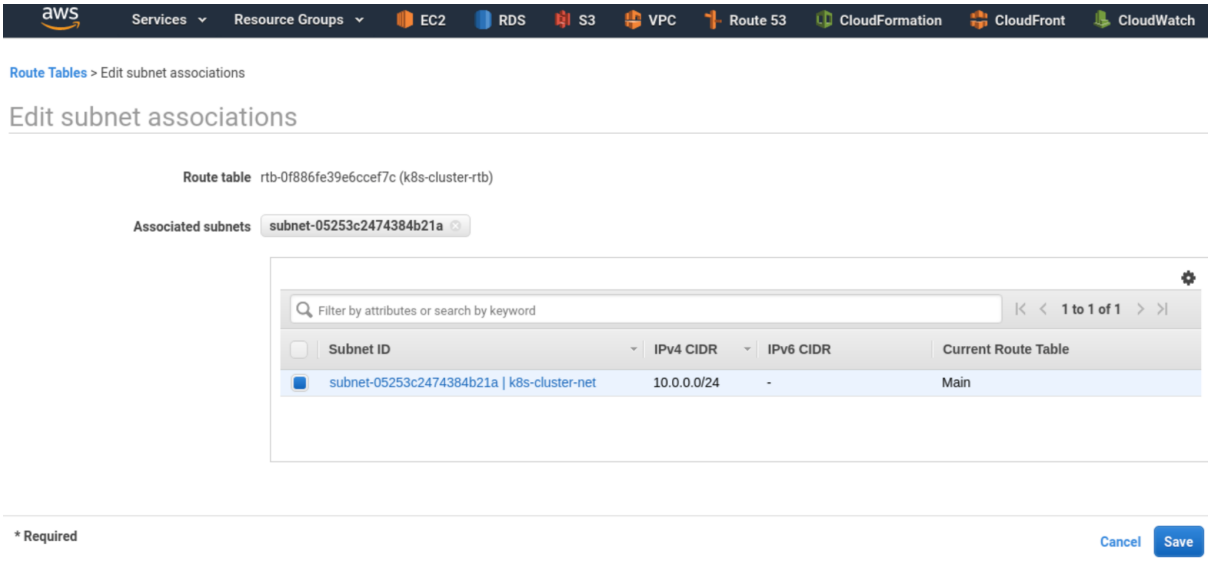
- Click on the Routes tab, add a new route to the 0.0.0.0/0 network via the IGW we created above(라우팅 테이블 탭으로 가서 IGW을 통해 0.0.0.0/0으로 오는 트래픽 허용)



- Attach this table to the subnet & Edit subnet association(생성하신 서브넷에 테이블 추가 및 서브넷 설정 편집)



- Choose the subnet you created earlier(조금 전에 생성한 서브넷 선택)



5. IAM Role

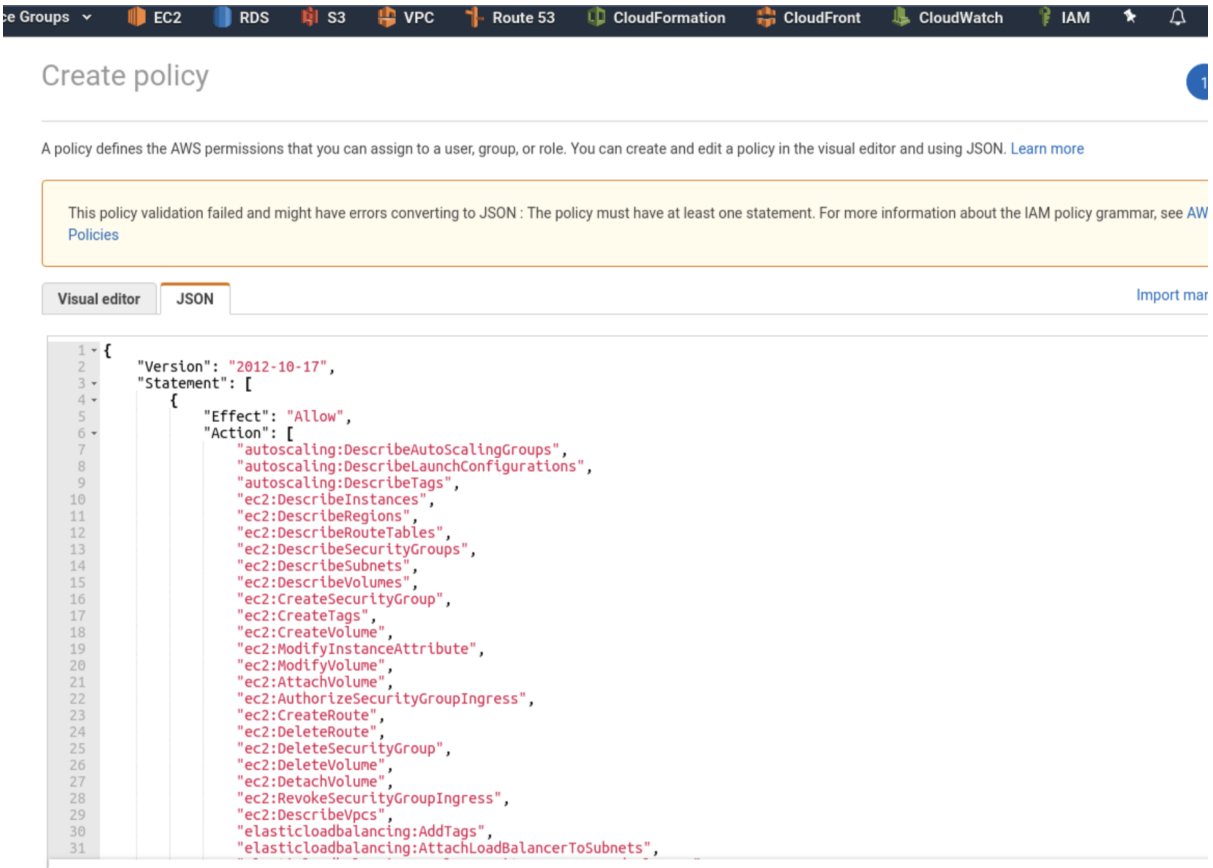
- To make Kubernetes working with AWS to create two IAM EC2 roles for master and worker(EC2의 K8s가 정상적으로 구동하려면 Master와 Worker로 사용될 EC2에 역할을 부여)

5.1. IAM Master Role

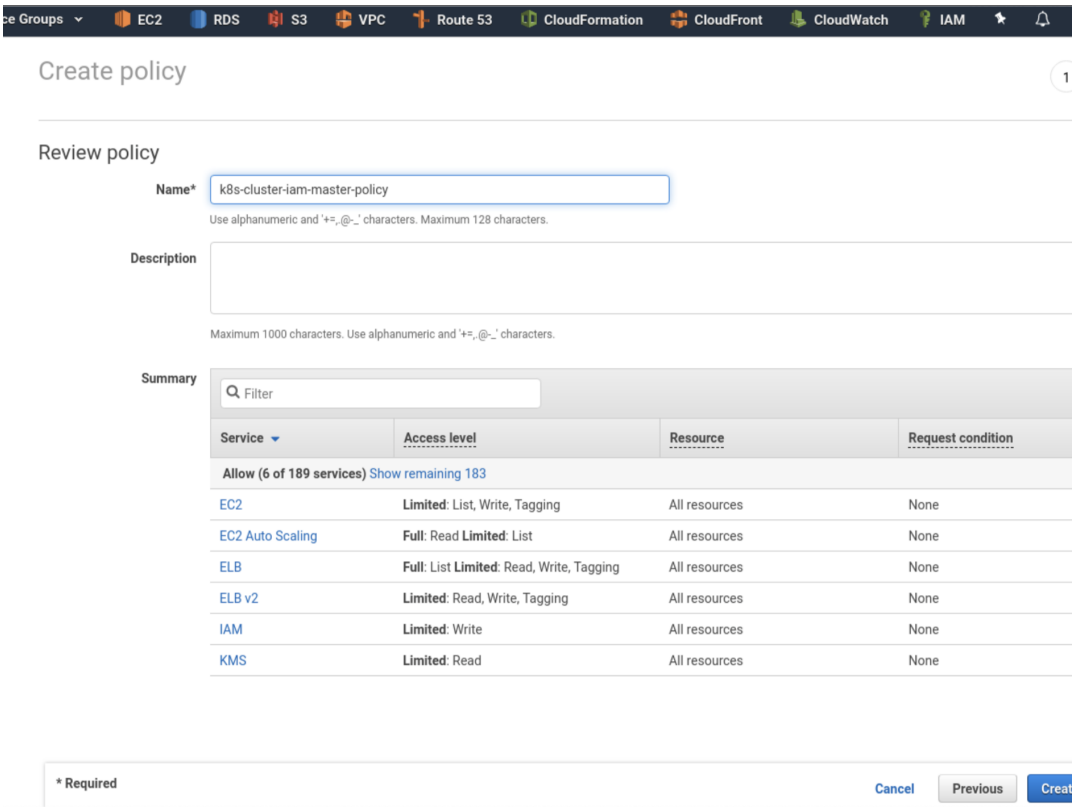
- Go to IAM -> Policies, click Create policy, into the JSON and add a new policy description(IAM 서비스 선택 후 정책을 누른 후 정책 생성을 선택합니다. 그리고 JSON을 눌러서 하기에 있는 코드를 복사붙여넣기)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLaunchConfigurations",
        "autoscaling:DescribeTags",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVolumes",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:ModifyInstanceAttribute",
        "ec2:ModifyVolume",
        "ec2:AttachVolume",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateRoute",
        "ec2:DeleteRoute",
        "ec2:DeleteSecurityGroup",
        "ec2:DeleteVolume",
        "ec2:DetachVolume",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:DescribeVpcs",
        "elasticloadbalancing:AddTags",
        "elasticloadbalancing:AttachLoadBalancerToSubnets",
        "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
        "elasticloadbalancing:CreateLoadBalancer",
        "elasticloadbalancing:CreateLoadBalancerPolicy",
        "elasticloadbalancing:CreateLoadBalancerListeners",
        "elasticloadbalancing:ConfigureHealthCheck",
        "elasticloadbalancing:DeleteLoadBalancer",
        "elasticloadbalancing:DeleteLoadBalancerListeners",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DetachLoadBalancerFromSubnets",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:ModifyLoadBalancerAttributes",
```

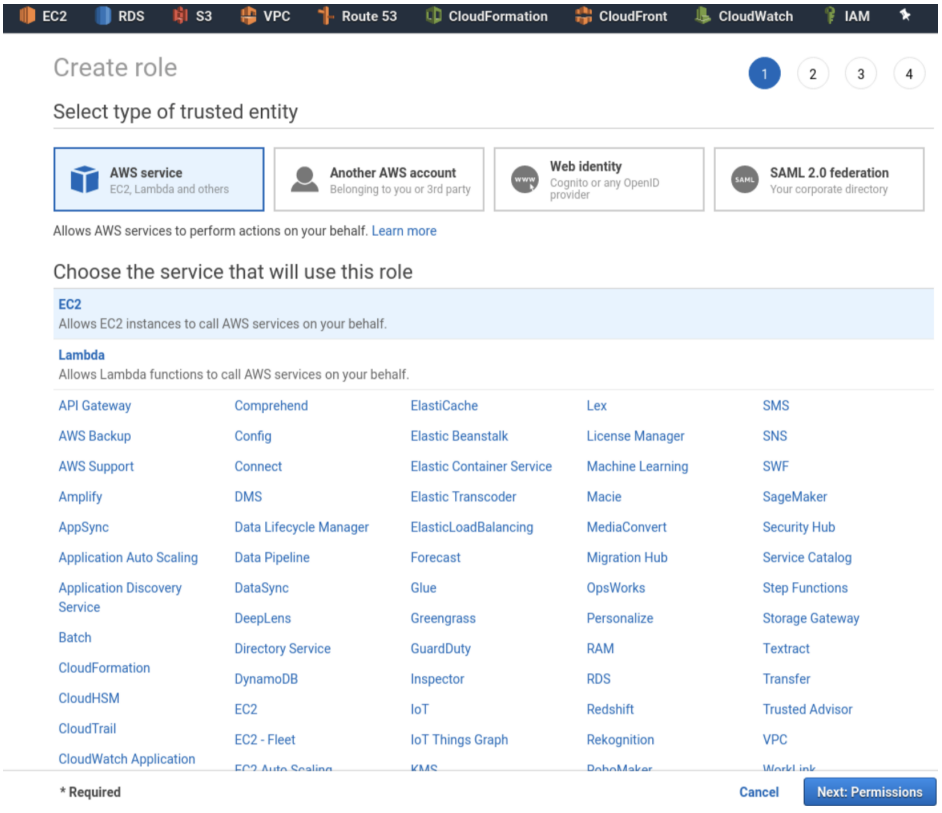
```
            "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
            "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
            "elasticloadbalancing:AddTags",
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:CreateTargetGroup",
            "elasticloadbalancing:DeleteListener",
            "elasticloadbalancing:DeleteTargetGroup",
            "elasticloadbalancing:DescribeListeners",
            "elasticloadbalancing:DescribeLoadBalancerPolicies",
            "elasticloadbalancing:DescribeTargetGroups",
            "elasticloadbalancing:DescribeTargetHealth",
            "elasticloadbalancing:ModifyListener",
            "elasticloadbalancing:ModifyTargetGroup",
            "elasticloadbalancing:RegisterTargets",
            "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
            "iam:CreateServiceLinkedRole",
            "kms:DescribeKey"
        ],
        "Resource": [
            "*"
        ]
    }
  ]
}
```



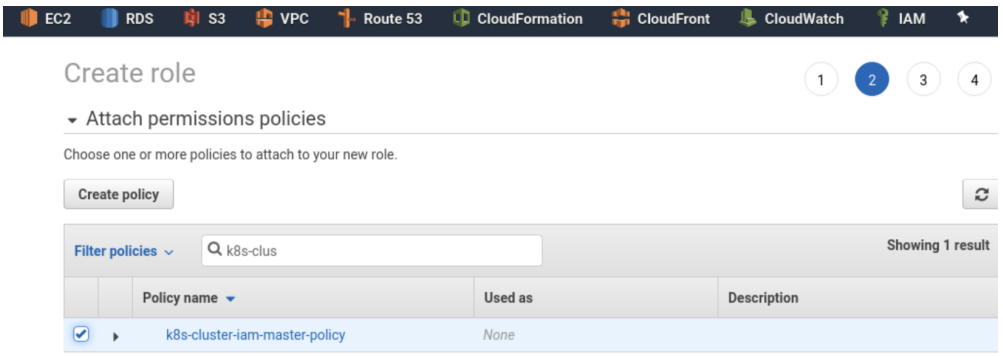
- Save it(저장)

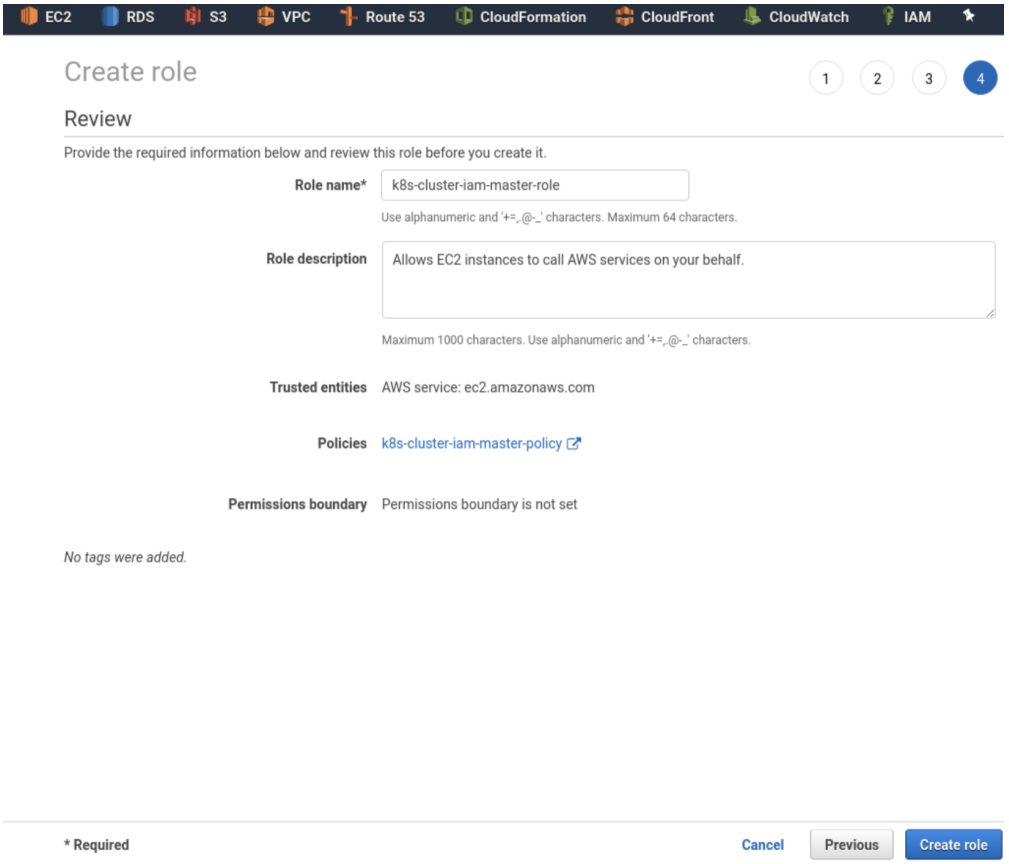


- Go to the *Roles*, create a role using the EC2 type(역할로 들어가서 EC2를 선택)



- Click on the *Permissions*, find and attack the policy added above(권한 정책 부분에 위에 만들었던 정책 이름 기입 후 정책을 역할에 추가)



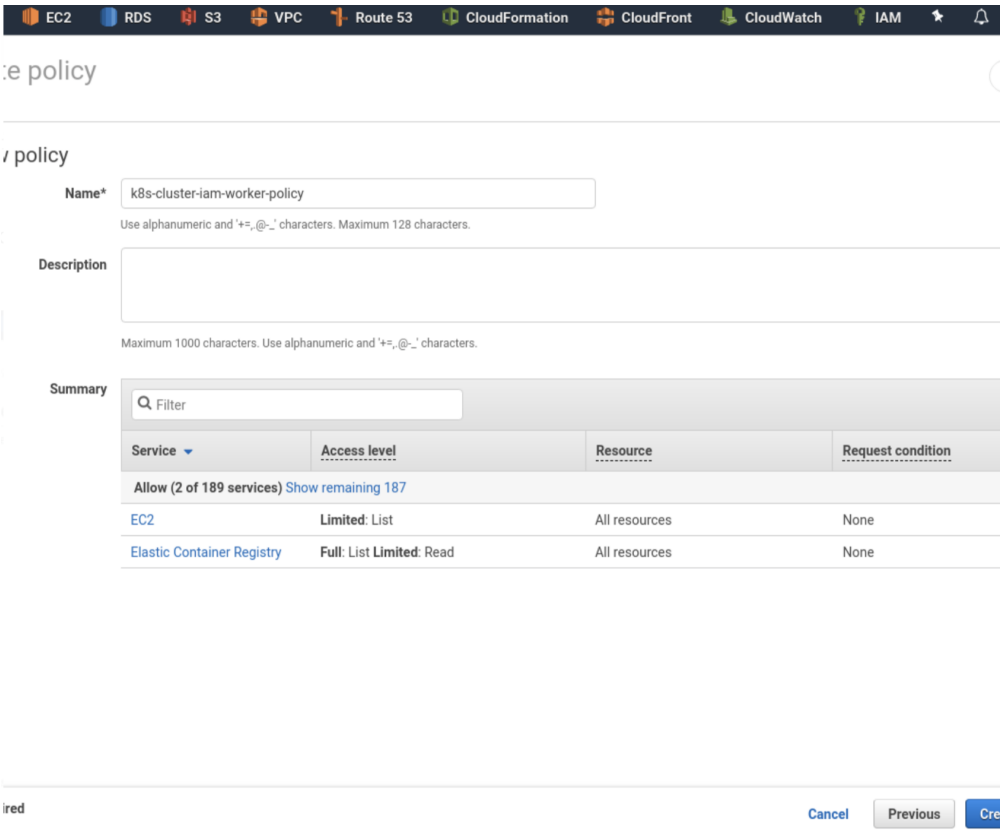


5.2. IAM Worker Role

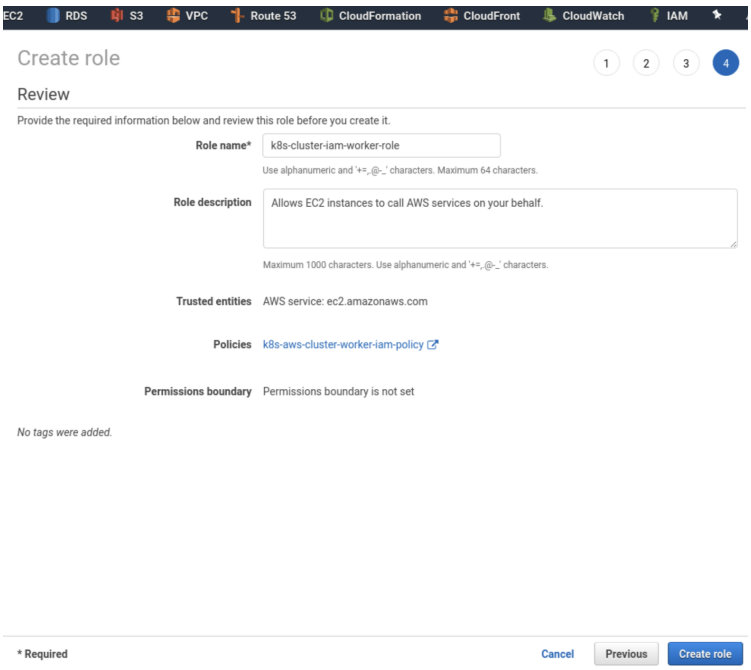
- In the same way, create another policy for worker nodes(Master 역할을 만든것처럼 똑같이 Worker 역할을 만들기)

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "ec2:DescribeInstances",
              "ec2:DescribeRegions",
              "ecr:GetAuthorizationToken",
              "ecr:BatchCheckLayerAvailability",
              "ecr:GetDownloadUrlForLayer",
              "ecr:GetRepositoryPolicy",
              "ecr:DescribeRepositories",
              "ecr:ListImages",
              "ecr:BatchGetImage"
          ],
          "Resource": "*"
      }
  ]
}
```

- Save it as k8s-cluster-iam-worker-policy(k8s-cluster-iam-worker-policy라는 이름으로 저장)

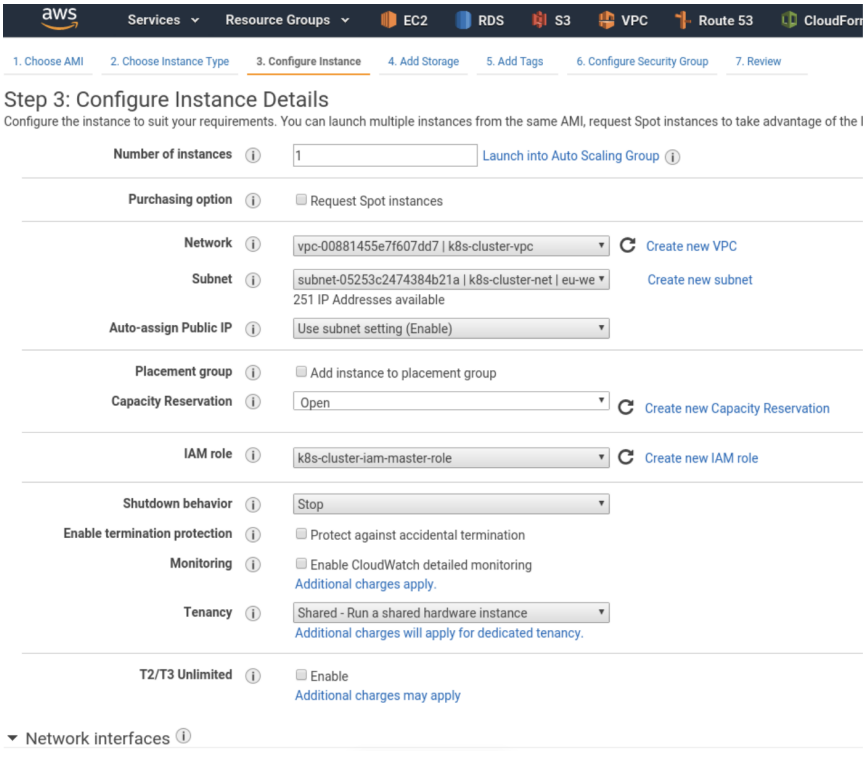


- And create a k8s-cluster-iam-worker-role(그리고 k8s-cluster-iam-worker-role을 생성)

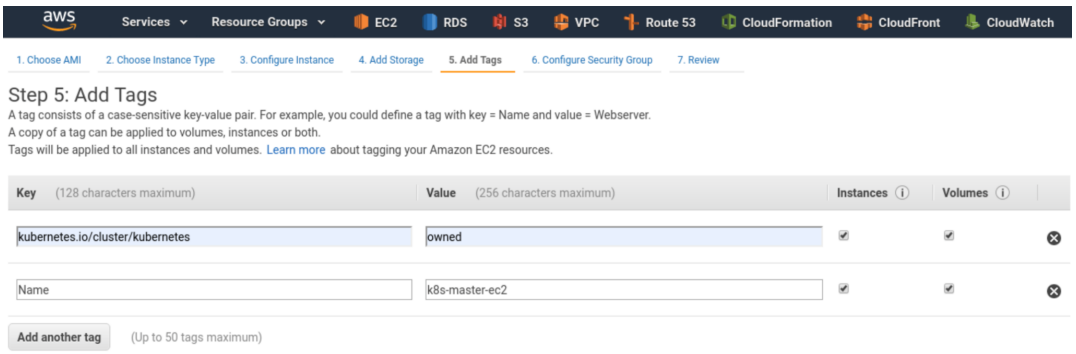


6. Running EC2

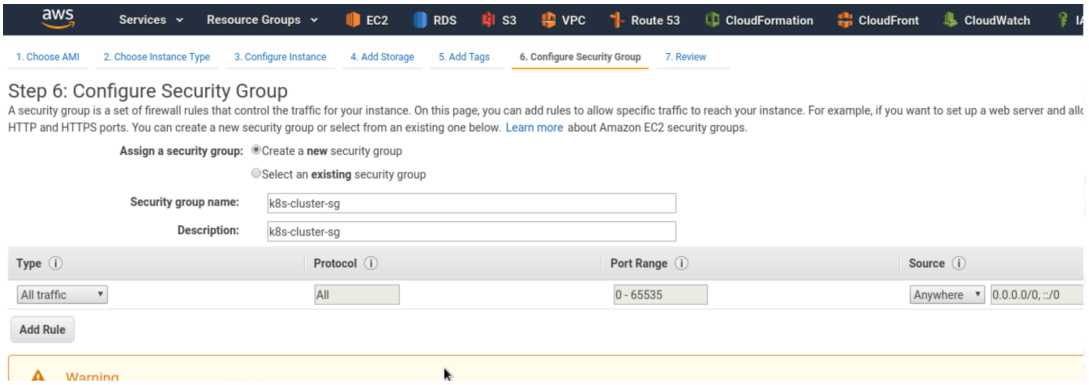
- Create an EC2 using t2.medium type (minimal type as Kubernetes master needs to have at least 2 CPU cores and 2 GB of RAM), using your VPC and set k8s-cluster-iam-master-role as the IAM role(쿠버네티스 Master 노드는 최소 2개의 CPU와 2GB의 램이 필요하기 때문에 EC2의 t2.medium 타입을 선택하고 이전에 생성한 VPC 선택 후 IAM 역할을 추가)



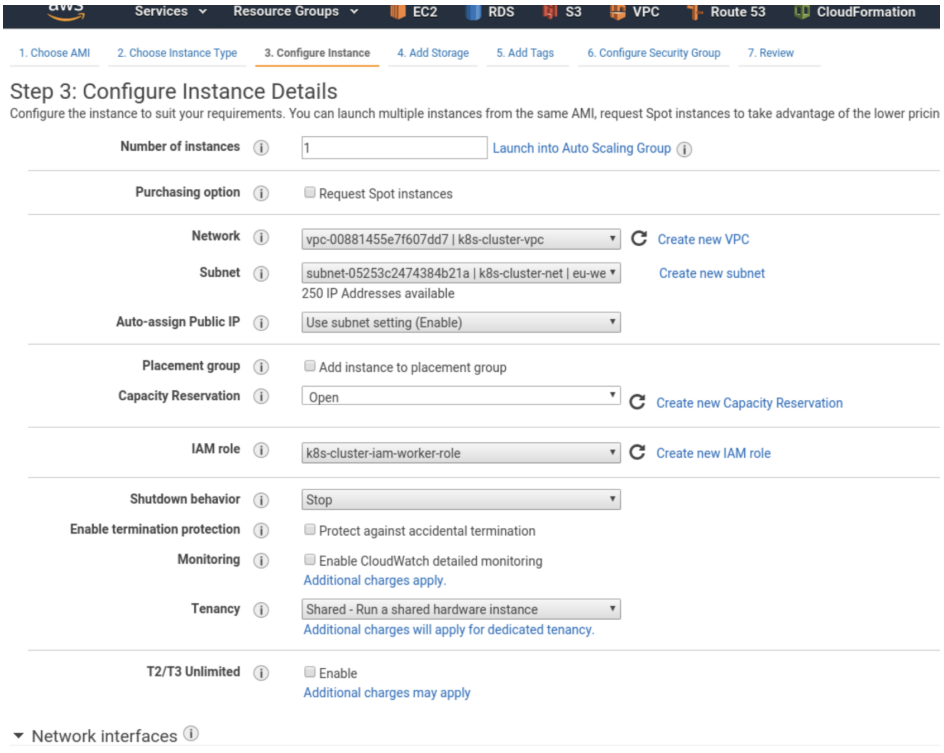
- Add tags(태그를 추가)



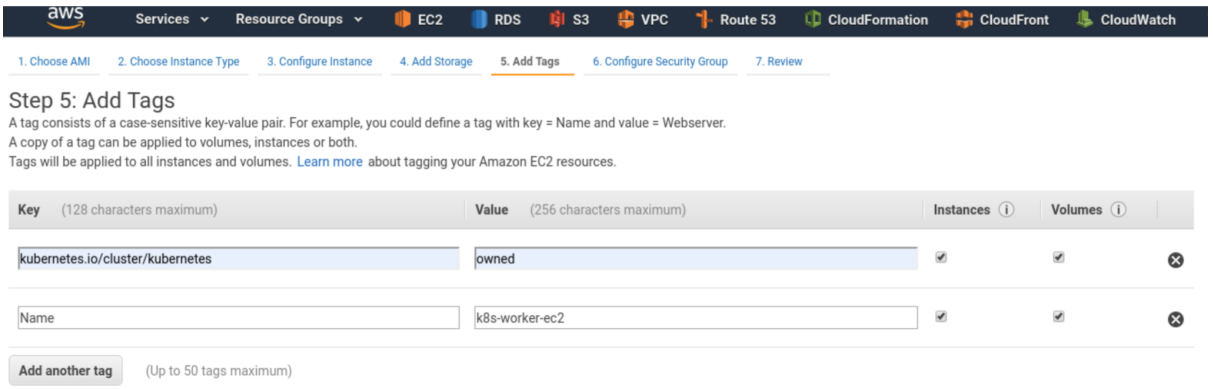
- Create a *Security Group*(보안 그룹을 생성)



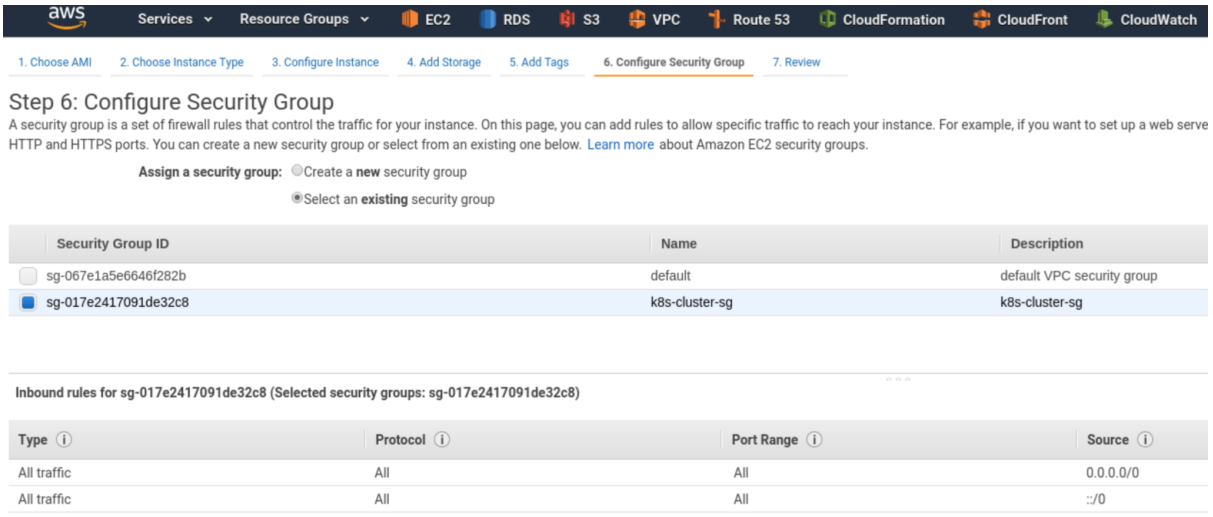
- While Master is spinning up, create a Worker Node in the same way and change the IAM role only which is *k8s-cluster-iam-worker-role*(Master Node의 생성을 기다리는 동안 똑같이 Worker Node를 생성하지만 IAM 역할을 k8s-cluster-iam-worker-role로 변경)



- Add the tags(태그 추가)



- Attach the existing *Security Group*(Master Node를 생성할 때 만들었던 보안 그룹 똑같이 사용)



- Perform the next steps on both EC2. Update packages list and installed packages(하기 단계들을 생성한 2대의 EC2에 동일하게 진행)

```
# apt update && apt -y upgrade
```

- Add Docker and Kubernetes Repositories(도커 및 쿠버네티스 리포를 추가)

```
# apt-get install -y gnupg lsb-release
# mkdir -p /etc/apt/keyrings
# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
# echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
# apt-get update
# apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
# docker info | grep Cgroup
```

vim /etc/docker/daemon.json(**cgroup**이 **cgroupfs**일때만 실행)
{ "exec-opts": ["native.cgroupdriver=systemd"] }
systemctl daemon-reload
systemctl restart docker
systemctl status docker(docker daemon이 active 상태인지 확인)
docker info | grep Cgroup

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

- Get it as a fully qualified domain name(FQDN 형식의 도메인명으로 조회)

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

- Set the hostname as FQDN(호스트명을 FQDN으로 변경)

hostnamectl set-hostname “FQDN”

- Check the hostname and do the same step on the Worker(호스트명 확인 및 호스트명 변경 단계를 Worker에서도 똑같이 진행)

hostname

apt-get update
apt-get install -y kubelet=1.23.0-00 kubeadm=1.23.0-00 kubectl=1.23.0-00
apt-mark hold kubelet kubeadm kubectl

//Run this command on Master Node only(Master 노드에서만 실행**)**
swapoff -a && sed -i '/swap/s/&/#/' /etc/fstab
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

vim /etc/kubernetes/aws.yaml
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
networking:
  serviceSubnet: "10.100.0.0/16"
  podSubnet: "10.244.0.0/16"
apiServer:
  extraArgs:
    cloud-provider: "aws"
controllerManager:
  extraArgs:
    cloud-provider: "aws"

kubeadm init --config /etc/kubernetes/aws.yaml

//After the init command, you'll get a kubeadm join command like below(init 명령어 이후 kubeadm join 명령어 확인)//

ex. kubeadm join 10.0.10.238:6443 --token ftknf4.yez7x0g7wxo8vt66 \
        --discovery-token-ca-cert-hash sha256:4c2fe2bd01da2a47be7ba1ad6b774f84e3188cff25ccddcadf8fa55e7e688a22

mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown ubuntu:ubuntu $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

//Run this command on Worker Node only(Worker 노드에서만 실행)

```
# vim /etc/kubernetes.node.yaml
apiVersion: kubeadm.k8s.io/v1beta2
kind: JoinConfiguration
discovery:
  bootstrapToken:
    token: "ftknf4.yez7x0g7wxo8vt66"
    apiServerEndpoint: "10.0.10.238:6443"
    caCertHashes:
      - "sha256:4c2fe2bd01da2a47be7ba1ad6b774f84e3188cff25ccddcadf8fa55e7e688a22"
nodeRegistration:
  name: ip-10-0-10-123.ap-northeast-2.compute.internal(this should be the worker nodes FQDN, worker node의 FQDN 적요)
  kubeletExtraArgs:
    cloud-provider: aws
```

//fyi. match the colors from kubeadm init command. pink=apiServerEndpoint, orange=token and green=caCertHashes(kubeadm init 명령어에서 받은 핑크값이 apiServerEndpoint, 오렌지값이 token 그리고 그린값이 caCertHashes//

kubeadm join --config /etc/kubernetes/node.yaml

//Back on Master Node(Master Node로 다시 접속)

kubectl get nodes -o wide

```
root@ip-10-0-10-238:/home/ubuntu# kubectl get node -o wide
NAME                                              STATUS   ROLES                  AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
ip-10-0-10-123.ap-northeast-2.compute.internal   Ready    <none>                 15m   v1.23.0   10.0.10.123   3.39.250.33   Ubuntu 20.04.5 LTS
ip-10-0-10-238.ap-northeast-2.compute.internal   Ready    control-plane,master   32m   v1.23.0   10.0.10.238   <none>        Ubuntu 20.04.5 LTS
```

- Load Balancer Creation by creating a simple nginx container(nginx 컨테이너를 로드 밸런서로 배포)

vim /home/ubuntu/lb-deploy.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      containers:
        - name: hello
          image: nginx
```

```
# vim /home/ubuntu/lb-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: hello
spec:
  type: LoadBalancer
  selector:
    app: hello
  ports:
    - name: http
      protocol: TCP
      port: 80
```

```
# kubectl create -f /home/ubuntu/lb-deploy.yaml
# kubectl create -f /home/ubuntu/lb-svc.yaml
```

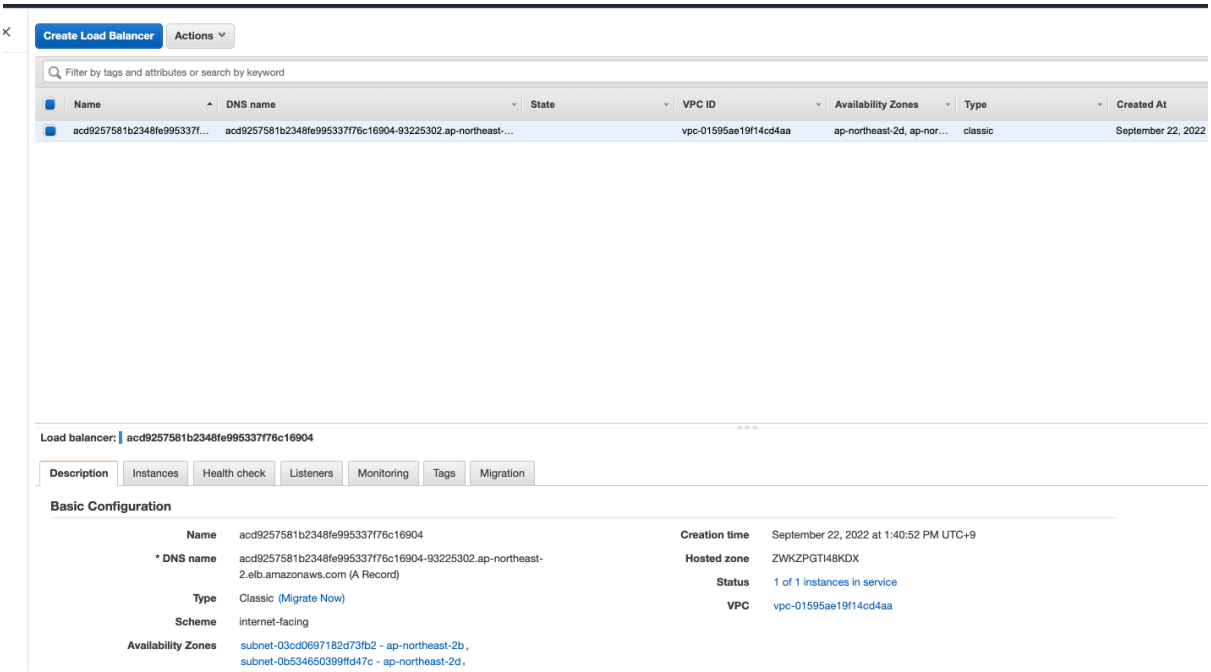
kubectl get deploy

```
root@ip-10-0-10-238:/home/ubuntu# kubectl get node -o wide
NAME                                              STATUS   ROLES                  AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
ip-10-0-10-123.ap-northeast-2.compute.internal   Ready    <none>                 15m   v1.23.0   10.0.10.123   3.39.250.33   Ubuntu 20.04.5 LTS
ip-10-0-10-238.ap-northeast-2.compute.internal   Ready    control-plane,master   32m   v1.23.0   10.0.10.238   <none>        Ubuntu 20.04.5 LTS
```

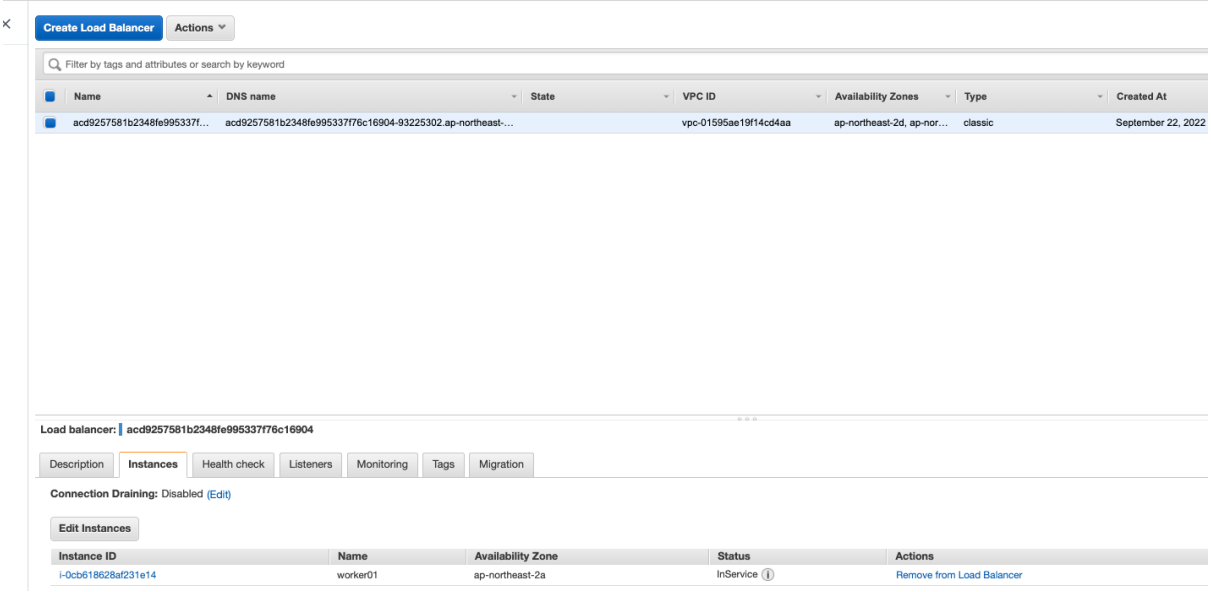
kubectl get svc

```
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP                                                                  PORT(S)
hello        LoadBalancer   10.100.29.240   acd9257581b2348fe995337f76c16904-93225302.ap-northeast-2.elb.amazonaws.com   80:30839/TCP
kubernetes   ClusterIP      10.100.0.1      <none>                                                                       443/TCP
```

- Check an ELB in AWS Console(ELB가 정상적으로 생성되었는지 AWS 콘솔에서 확인)

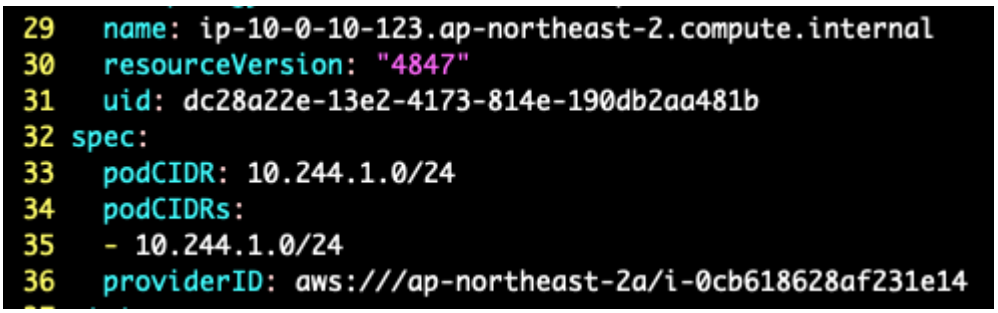


- Check if Worker node is assigned properly to the ELB(Worker node가 ELB에 정상적으로 연동이 되었는지 확인)



7. AWS Load Balancer(TroubleShooting) - no Worker Node added(k8s에서 서비스 배포 시 로드밸런서에 worker node가 연동이 안되었을 때 해결방법)

kubectl edit node “worker-nodes-FQDN”



// If you don’t see providerID like the image, add one and just change the last part to your worker node’s instance ID and save the file(위 사진과 같이 providerID가 생성이 안되어있다면 추가 및 마지막 부분 자신의 worker node 인스턴스 ID로 변경 후 파일 저장)//