EKS

Trainee Name : Chhavi Sharma

Newers ID : 4023

College : UPES

1. Create eks cluster using eksctl
   During creation, Specify
   - Cluster name
   - Kubernetes version
   - Control plane role
   - Subnets for Control Plane
   - Control Plane security Group
   - Add tag: owner, purpose on Control Plane
   - Node Group Name
   - Node Instance Role
   - Subnets for Node Group
   - Node Instance SSH key pair
   - Node Instance Instance Type
   - Node Instance Disk
   - Add tag: owner, purpose on Node Group
   - Node Group Size: min, max

Ans.

Step 1: Install eksctl

```
chhavi@chhavi:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/ek
sctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
chhavi@chhavi:~$ sudo mv /tmp/eksctl /usr/local/bin
chhavi@chhavi:~$ eksctl version
0.16.0
```

Step 2: Write a yaml file for cluster configuration.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
 name: chhavieks
 region: us-east-1

vpc:
 id: "vpc-093a4253d4c9ab207"
 cidr: "10.0.0.0/16"
 subnets:
   public:
     us-east-1a:
         id: "subnet-0af58143cd499547c"
         cidr: "10.0.0.0/24"
     us-east-1b:
         id: "subnet-0342b9c54db410a4e"
         cidr: "10.0.2.0/24"
     us-east-1c:
         id: "subnet-007036a437b689c11"
         cidr: "10.0.4.0/24"
iam:
     serviceRoleARN: "arn:aws:iam::187632318301:role/eks-service-role"
managedNodeGroups:
 - name: mynodegroup
```

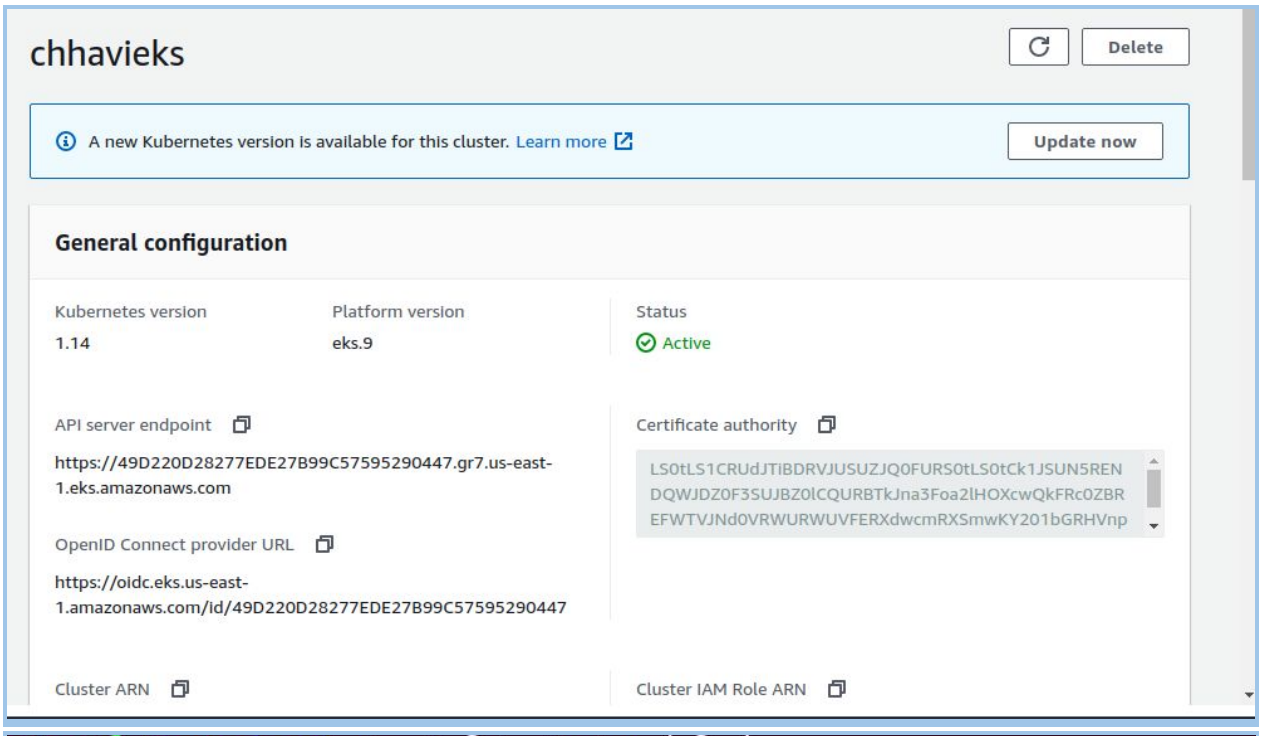
```
managedNodeGroups:
 - name: mynodegroup
   instanceType: t3.medium
   desiredCapacity: 2
   minSize: 1
   maxSize: 3
   availabilityZones: ["us-east-1a", "us-east-1b", "us-east-1c"]
   volumeSize: 15
   tags:
    owner: chhavi
    purpose: ekscluster
   ssh:
    publicKeyName: 'ansiblekeypair'
    allow: true
```

Step 3: Run the command below to create the cluster

```
chhavi@chhavi:~/eks$ eksctl create cluster -f eksconfig.yml
[ℹ]  eksctl version 0.16.0
[ℹ]  using region us-east-1
[✔]  using existing VPC (vpc-093a4253d4c9ab207) and subnets (private:[] public:[subnet-007036a437b689c11 su
bnet-0af58143cd499547c subnet-0342b9c54db410a4e])
[!]  custom VPC/subnets will be used; if resulting cluster doesn't function as expected, make sure to revie
w the configuration of VPC/subnets
[ℹ]  using EC2 key pair "ansiblekeypair"
[ℹ]  using Kubernetes version 1.14
[ℹ]  creating EKS cluster "chhavieks" in "us-east-1" region with managed nodes
[ℹ]  1 nodegroup (mynodegroup) was included (based on the include/exclude rules)
[ℹ]  will create a CloudFormation stack for cluster itself and 0 nodegroup stack(s)
[ℹ]  will create a CloudFormation stack for cluster itself and 1 managed nodegroup stack(s)
[ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --regio
n=us-east-1 --cluster=chhavieks'
[ℹ]  CloudWatch logging will not be enabled for cluster "chhavieks" in "us-east-1"
[ℹ]  you can enable it with 'eksctl utils update-cluster-logging --region=us-east-1 --cluster=chhavieks'
[ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluste
r "chhavieks" in "us-east-1"
[ℹ]  2 sequential tasks: { create cluster control plane "chhavieks", create managed nodegroup "mynodegroup"
 }
[ℹ]  building cluster stack "eksctl-chhavieks-cluster"
[ℹ]  deploying stack "eksctl-chhavieks-cluster"
[ℹ]  building managed nodegroup stack "eksctl-chhavieks-nodegroup-mynodegroup"
[ℹ]  deploying stack "eksctl-chhavieks-nodegroup-mynodegroup"
[✔]  all EKS cluster resources for "chhavieks" have been created
[✔]  saved kubeconfig as "/home/chhavi/.kube/config"
```

```
[ℹ]  building cluster stack "eksctl-chhavieks-cluster"
[ℹ]  deploying stack "eksctl-chhavieks-cluster"
[ℹ]  building managed nodegroup stack "eksctl-chhavieks-nodegroup-mynodegroup"
[ℹ]  deploying stack "eksctl-chhavieks-nodegroup-mynodegroup"
[✔]  all EKS cluster resources for "chhavieks" have been created
[✔]  saved kubeconfig as "/home/chhavi/.kube/config"
[ℹ]  nodegroup "mynodegroup" has 2 node(s)
[ℹ]  node "ip-10-0-0-45.ec2.internal" is ready
[ℹ]  node "ip-10-0-4-66.ec2.internal" is ready
[ℹ]  waiting for at least 1 node(s) to become ready in "mynodegroup"
[ℹ]  nodegroup "mynodegroup" has 2 node(s)
[ℹ]  node "ip-10-0-0-45.ec2.internal" is ready
[ℹ]  node "ip-10-0-4-66.ec2.internal" is ready
[ℹ]  kubectl command should work with "/home/chhavi/.kube/config", try 'kubectl get nodes'
[✔]  EKS cluster "chhavieks" in "us-east-1" region is ready
```

Step 4: Check if the cluster and the nodes are created successfully.




```
chhavi@chhavi:~/eks$ eksctl get cluster
NAME            REGION
chhavieks       us-east-1
chhavi@chhavi:~/eks$ kubectl get nodes
NAME                           STATUS   ROLES    AGE     VERSION
ip-10-0-0-45.ec2.internal      Ready    <none>   8m36s   v1.14.9-eks-1f0ca9
ip-10-0-4-66.ec2.internal      Ready    <none>   8m48s   v1.14.9-eks-1f0ca9
chhavi@chhavi:~/eks$
```

2. Authentication Management
    a. Add new 2 IAM user into the cluster

Ans.

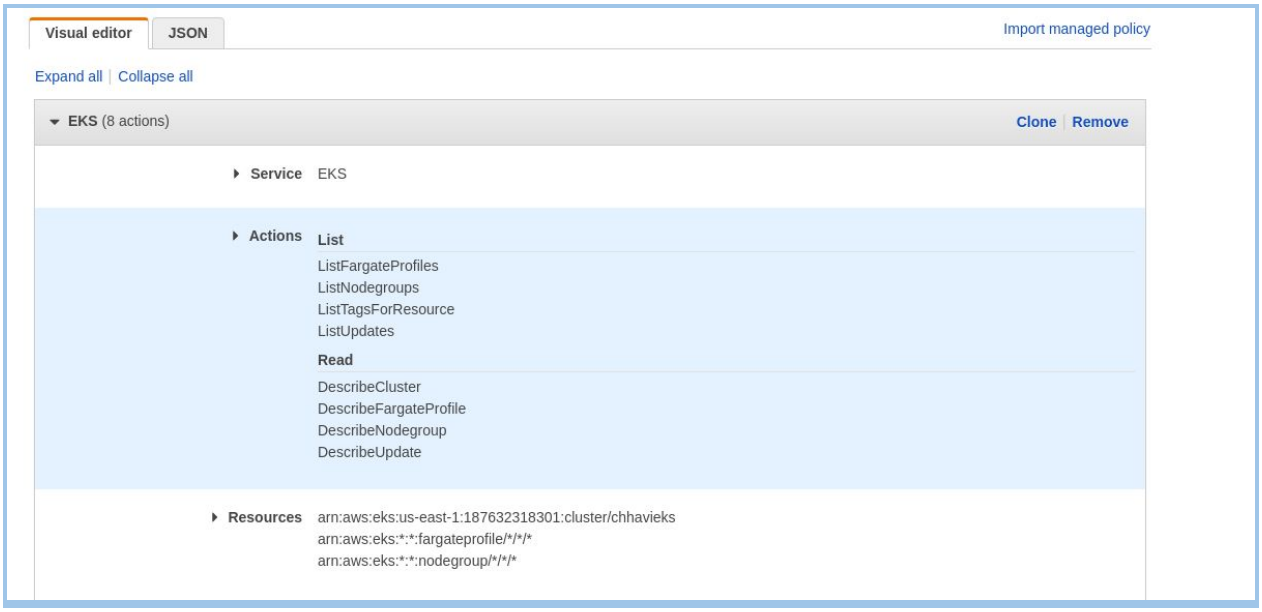
```
chhavi@chhavi:~/eks$ kubectl edit -n kube-system configmap/aws-auth
configmap/aws-auth edited
```

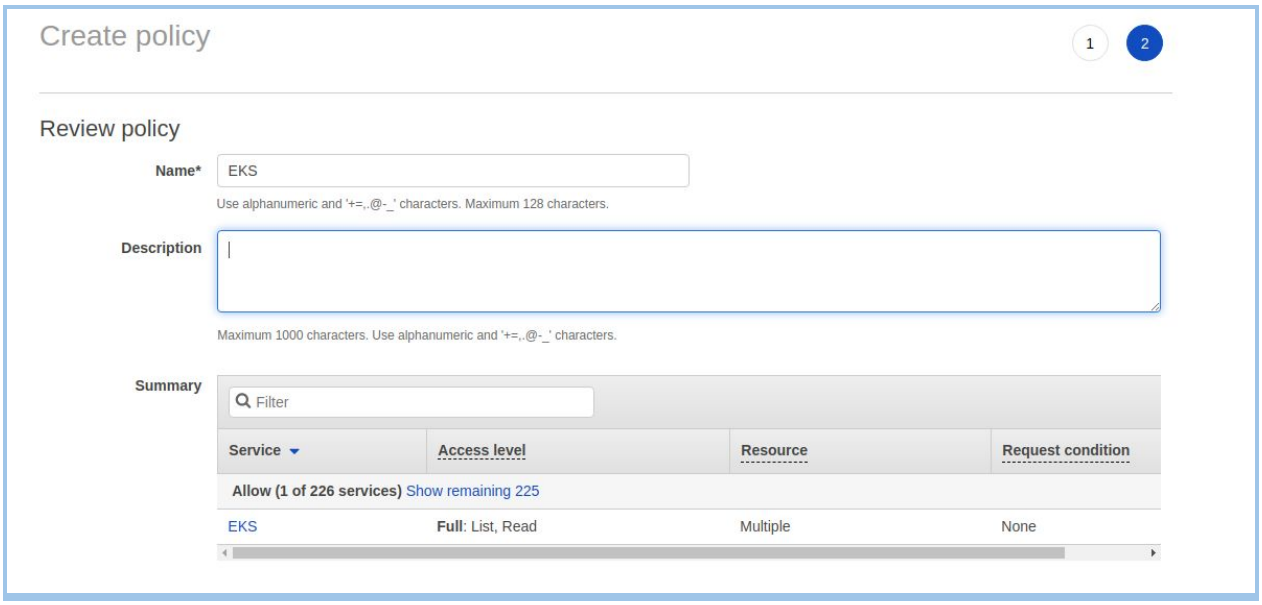
```
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::187632318301:role/eksctl-chhavieks-nodegroup-mynode-NodeInstanceRole-QARWH7TMTW
DP
      username: system:node:{{EC2PrivateDNSName}}

  mapUsers: |
    - userarn: arn:aws:iam::187632318301:user/gargi.sharma@tothenew.com
      username: gargi
      groups:
        - system: masters
    - userarn: arn:aws:iam::187632318301:user/ekanshu.dargan@tothenew.com
      username: ekanshu
      groups:
        - system: masters

kind: ConfigMap
metadata:
  creationTimestamp: "2020-04-06T07:21:44Z"
  name: aws-auth
  namespace: kube-system
-- INSERT --                                                    20,24         57%
```

b. Enable a EC2 server to access Cluster master API without using access/secret key

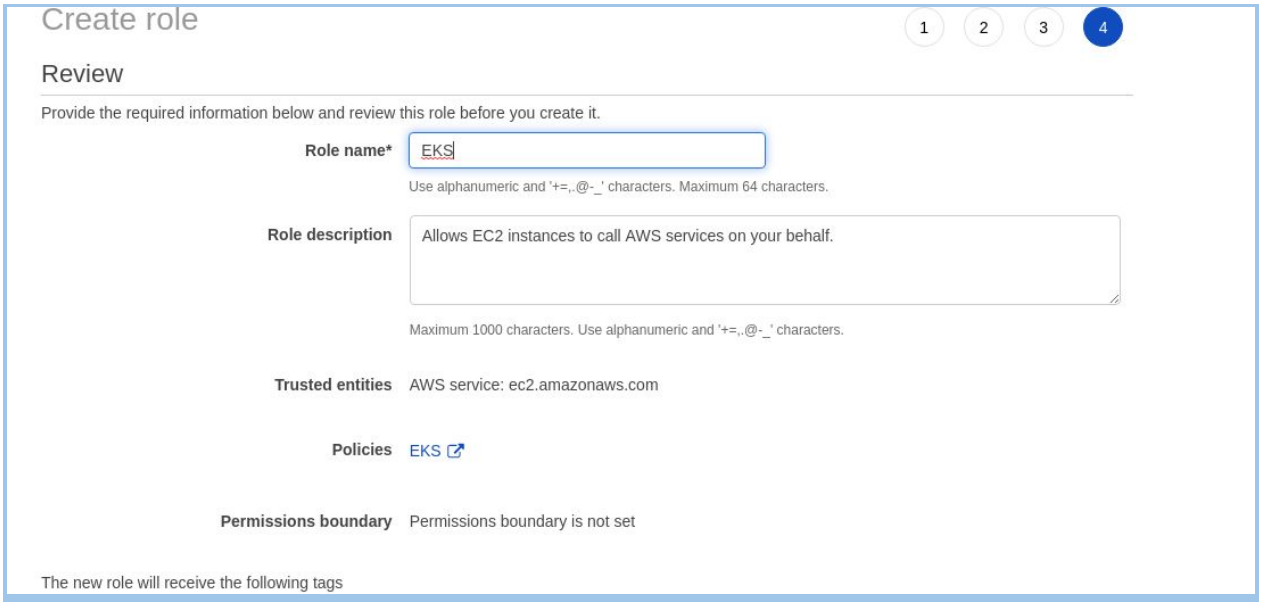
Ans.

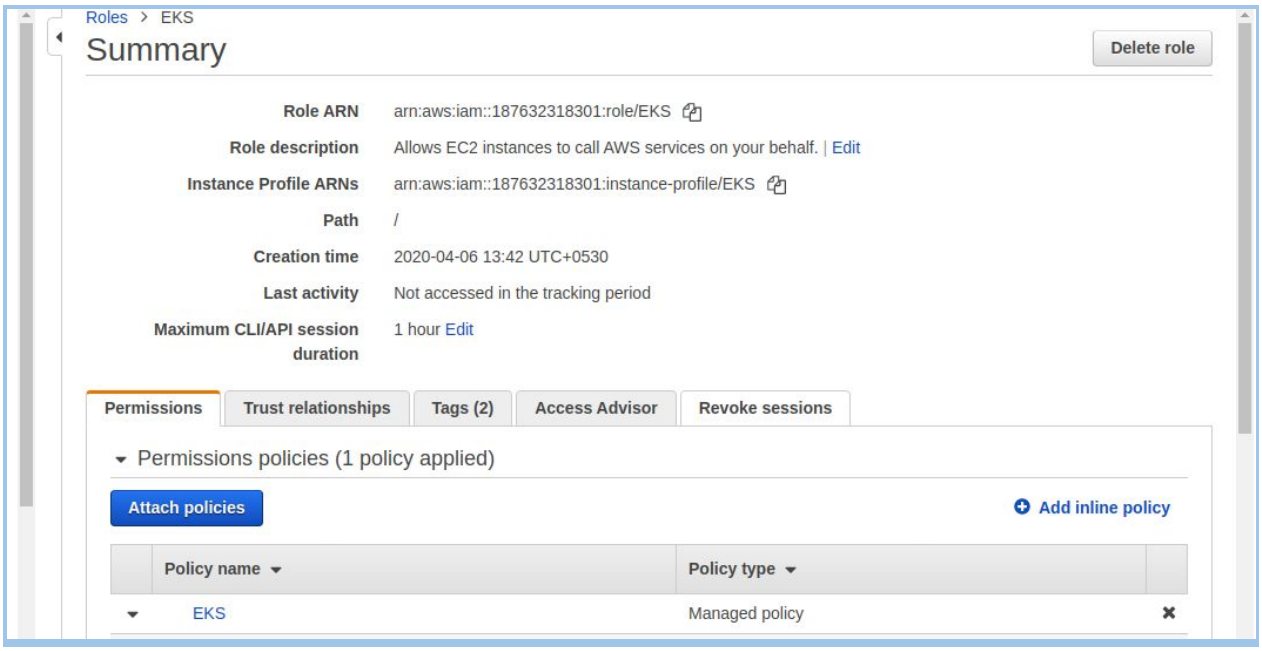
Step 1: Create a policy for service **eks**



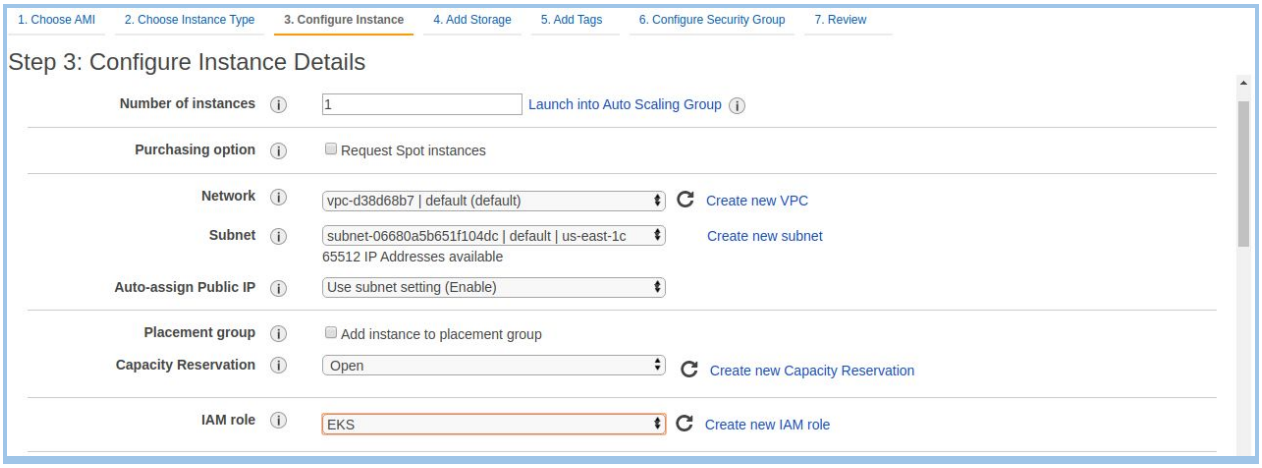


Step 2: Create a new role and attach the above policy.





Step 3: Now launch an instance and attach the role created to it.



Step 4: SSH into the instance and check if you can describe the cluster created.

```
ubuntu@ip-172-31-142-249:~$ aws eks describe-cluster --name chhavieks --region us-east-1
{
    "cluster": {
        "name": "chhavieks",
        "arn": "arn:aws:eks:us-east-1:187632318301:cluster/chhavieks",
        "createdAt": "2020-04-06T07:10:06.993000+00:00",
        "version": "1.14",
        "endpoint": "https://49D220D28277EDE27B99C57595290447.gr7.us-east-1.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::187632318301:role/eks-service-role",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0af58143cd499547c",
                "subnet-0342b9c54db410a4e",
                "subnet-007036a437b689c11"
            ],
            "securityGroupIds": [
                "sg-0f27d9451079fce47"
            ],
            "clusterSecurityGroupId": "sg-023b1f5698945412a",
            "vpcId": "vpc-093a4253d4c9ab207",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": false,
            "publicAccessCidrs": [
                "0.0.0.0/0"
            ]
        },
```

3. Eksctl command to terminate the stack

Ans.

Execute the command below to delete the cluster .

```
chhavi@chhavi:~/eks$ eksctl delete cluster -f eksconfig.yml
[ℹ]  eksctl version 0.16.0
[ℹ]  using region us-east-1
[ℹ]  deleting EKS cluster "chhavieks"
[ℹ]  deleted 0 Fargate profile(s)
[✔]  kubeconfig has been updated
[ℹ]  2 sequential tasks: { delete nodegroup "mynodegroup", delete cluster control plane "chhavieks" [async]
 }
[ℹ]  will delete stack "eksctl-chhavieks-nodegroup-mynodegroup"
[ℹ]  waiting for stack "eksctl-chhavieks-nodegroup-mynodegroup" to get deleted
[ℹ]  will delete stack "eksctl-chhavieks-cluster"
[✔]  all cluster resources were deleted
chhavi@chhavi:~/eks$
```