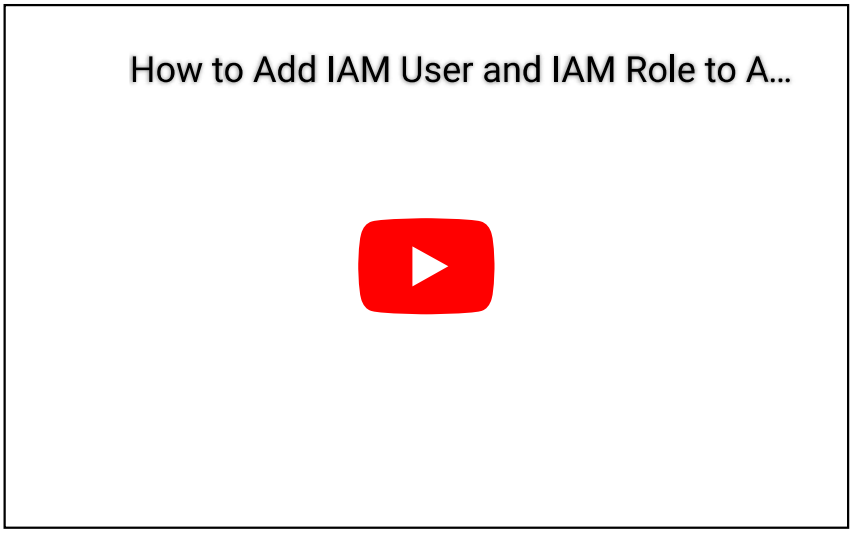
How to Add IAM User and IAM Role to AWS EKS Cluster?



- You can find the source code for this video in my GitHub Repo.
- If you want to create EKS cluster using terraform, you can follow this tutorial.

Add IAM User to EKS Cluster

- Create `ClusterRole` with read-only access to the Kubernetes cluster and bind it to the `reader` group via `ClusterRoleBinding`. Name the file `read-group.yaml`.

read-group.yaml

```yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: reader
rules:
```

```
- apiGroups: ["*"]
  resources: ["deployments", "configmaps", "pods", "secrets", "services"]
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: reader
subjects:
- kind: Group
  name: reader
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: reader
  apiGroup: rbac.authorization.k8s.io
```

- Apply RBAC policies with kubectl.

```
kubectl apply -f k8s/read-group.yaml
```

- Create IAM policy to let users view nodes and workloads for all clusters in the AWS Management Console. Give it a name `AmazonEKSViewNodesAndWorkloadsPolicy`.

AmazonEKSViewNodesAndWorkloadsPolicy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:DescribeCluster",
                "eks:ListClusters",
                "eks:AccessKubernetesApi",
                "ssm:GetParameter",
                "eks:ListUpdates",
                "eks:ListFargateProfiles"
            ],
            "Resource": "*"
        }
    ]
}
```

- We can attach the IAM policy directly to the IAM user or follow the best practice and create an IAM group first. Let's call it `developers` and attach `AmazonEKSViewNodesAndWorkloadsPolicy` IAM policy.
- Then, we need an IAM user. Let's create one and call it `developer`. We're going to place it in `developers` IAM grop with read only access. Don't forget to download credentials, we will use them to configure aws cli.
- Now, we need to create local aws profile using `developer`'s user credentials. To do that simplly add `--profile` flag to `aws configure` command.

```
aws configure --profile developer
```

- To map IAM user with Kubernetes RBAC system, we need to modify `aws-auth` configmap. Open the config map and add arn of the IAM user under `mapUsers` key.

```
kubectl edit -n kube-system configmap/aws-auth
```

```
...
  mapUsers: |
    - userarn: arn:aws:iam::424432388155:user/developer
      username: developer
      groups:
      - reader
...
```

- Now, we need to switch to the `developer` user. We need to update Kubernetes context using the `developer` profile. Don't forget to update region and the cluster name.

```
aws eks update-kubeconfig \
  --region us-east-1 \
  --name demo \
  --profile developer
```

- You can verify the Kubernetes context that you use `developer` profile to interact with the EKS cluster.

```
kubectl config view --minify
```

- By now, we have created RBAC policy with read only access and maped it to the IAM `developer` user. Let's see what we can do in our cluster now. You can run `kubectl auth can-i <object>` to verify access. First let's check if we can get pods in Kubernetes cluster. The response should be `yes`.

```
kubectl auth can-i get pods
```

- Then, let's check if we can create pods in the Kuberentes. The response should be `no`.

```
kubectl auth can-i create pods
```

- You can also try to create the pod. You should get `Forbidden` error.

```
kubectl run nginx --image=nginx
```

Add IAM Role to EKS Cluster

- First of all, let's create IAM policy with admin access to EKS clusters. Give it a name `AmazonEKSAdminPolicy`. To view information on the Nodes and Workloads in the AWS Management Console, you need additional IAM permissions, as well as Kubernetes permissions.

AmazonEKSAdminPolicy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

- Then, create `eks-admin` IAM role and attach `AmazonEKSAdminPolicy` policy that you just created. Select `Another AWS account` and enter your account id.
- Optionally, you can describe `eks-admin` role to check who can use it. Potentially, any IAM user can assume this role if they have an appropriate policy in place.

```
aws iam get-role --role-name eks-admin
```

```
...
"Principal": {
    "AWS": "arn:aws:iam::424432388155:root"
},
...
```

- For any user that wants to use `eks-admin` IAM role, we need to create an additional `AmazonEKSAssumeEKSAdminPolicy` policy, that allows to assume the role.

AmazonEKSAssumeEKSAdminPolicy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::424432388155:role/eks-admin"
        }
    ]
}
```

- To test it, we need another IAM user. Let's create `manager` user and allow it to use `eks-admin` IAM role. In this case, just attach `AmazonEKSAssumeEKSAdminPolicy` directly to the user.
- The same thing for this user, we need to create a `manager` aws profile.

```
aws configure --profile manager
```

- You can check if the `manager` user can assume the `eks-admin` role by running the following command. If we get credentials back, it means we can use it.

```
aws sts assume-role \
  --role-arn arn:aws:iam::424432388155:role/eks-admin \
```

```
  --role-session-name manager-session \
  --profile manager
```

- Now, we need to switch back to the user that created the EKS cluster. When you omit the profile, aws will use the `default` one.

```
aws eks --region us-east-1 update-kubeconfig --name demo
```

- It's a very similar process to add an IAM role. You also need to update the `aws-auth` configmap. In this case, we will use Kubernetes RBAC group `system:masters` that ships with the cluster.

```
kubectl edit -n kube-system configmap/aws-auth
```

```
...
  mapRoles: |
    - rolearn: arn:aws:iam::424432388155:role/eks-admin
      username: eks-admin
      groups:
      - system:masters
...
```

- Finally, to test the IAM role, we need to create an `eks-admin` profile config to assume the role by the `manager` user. You need to add it to `~/.aws/config`.

```
vim ~/.aws/config
```

config

```
...
[profile eks-admin]
role_arn = arn:aws:iam::424432388155:role/eks-admin
source_profile = manager
```

- Update Kubernetes context to automatically assume `eks-admin` role by `manager` user.

```
aws eks update-kubeconfig \
  --region us-east-1 \
  --name demo \
  --profile eks-admin
```

- You can check the context with the following command.

```
kubectl config view --minify
```

- Check if `manager` user has admin access to the EKS cluster. The response should be `yes.`

```
kubectl auth can-i "*" "*"
```

- You can also try to create a pod using the previous command.

```
kubectl run nginx --image=nginx
```

Clean

- Delete IAM policies
  - `AmazonEKSViewNodesAndWorkloadsPolicy`
  - `AmazonEKSAssumeEKSAdminPolicy`
  - `AmazonEKSAdminPolicy`
- Delete IAM roles
  - `eks-admin`
- Delete IAM groups
  - `eks-admin`
  - `developers`
- Delete IAM users
  - `manager`
  - `developer`
- Clean UP `~/.aws/config` and `~/.aws/credentials`