How to manage only specific namespaces with IAM users in Amazon EKS

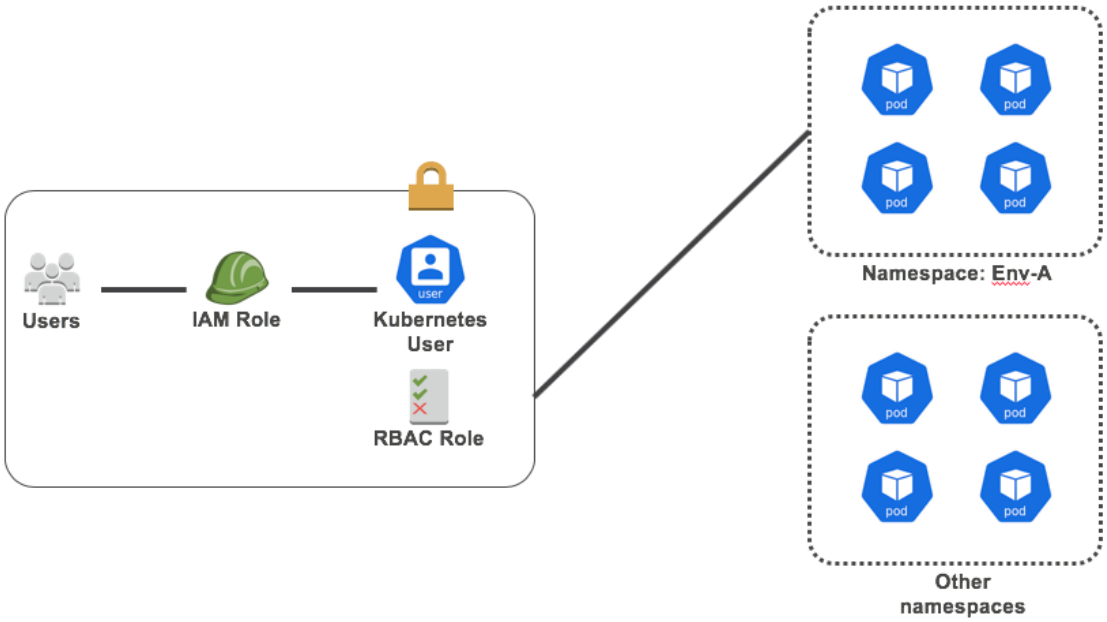
Namespaces allow to create virtual clusters backed by the same physical cluster. A popular feature of namespaces is to create resource quotas where you can limit the amount of resources assigned.

You may want to allow specific IAM users to manage only a single Kubernetes namespace, preventing them from interacting with other namespaces.

Amazon EKS uses aws-iam-authenticator for authentication and Kubernetes Role Based Access Control (RBAC) for authorization. In our example we will set up an IAM role for authentication and assign a RBAC role to scope the API calls allowed.

Summary:

- Create namespace: env-a.
- Create IAM role: eks-role-env-a.
- Add policy to existing users to assume role.
- Create Kubernetes user: admin-env-a.
- Configure Kubeconfig with the new role.



---

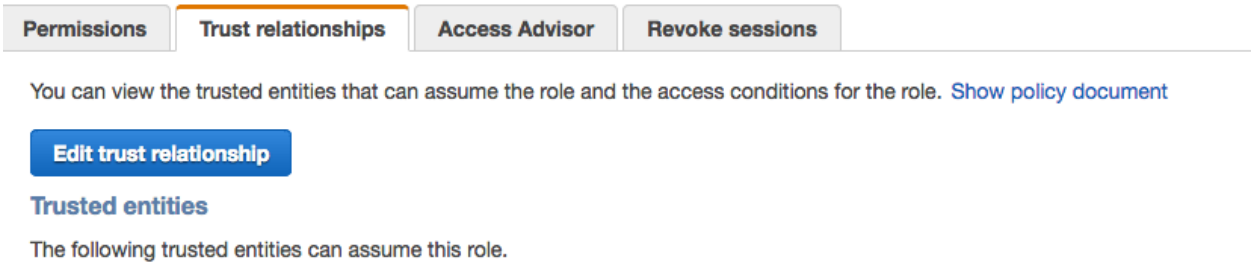
1. **Create namespace:**

$ kubectl create namespace env-a

namespace "env-a" created

2. Create IAM role:

In the IAM console, create a role: eks-role-env-a. There is no need to add IAM permissions. You will only need to add a Trust Relationship:



Add a trust relationship to allow who should be able to assume the IAM role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::ACCOUNT ID HERE:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Note: this code allows all users from an account ID to assume role. You can customize this policy. Read more in the documentation.

3. Allow users/roles to assume the IAM role to interact with the namespace.

Check what IAM user/role the AWS CLI is using in the host **from where you want to manage your cluster with "kubectl"**.

In case you do not know/remember your user/role, check the file $HOME/.aws/credentials to find the access key ID you are using. Use the search box in the IAM console to find the user.

If you manage your cluster from an instance which uses an EC2 Instance role get the IAM role name from the instance details in the EC2 console.

Go to the permissions tab and add this policy:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::ACCOUNT ID HERE:role/eks-role-env-a"
  }
}
```

4. Edit the aws-auth configmap to map users to the cluster.

Add the lines marked in bold.

```
$ kubectl edit -n kube-system configmap/aws-authapiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::123456789:role/EKS-node
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
```

```
    - system:nodes
  - rolearn: arn:aws:iam::XXXXXXXXX:role/eks-role-env-a
    username: admin-env-a
```

5. Create an RBAC role and map it to the RBAC user.

```
$ vi role.yaml
####
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: env-a
  name: env-a-full-right
rules:
- apiGroups: [""]
  resources: ["*"]
  verbs: ["*"]
- apiGroups: ["extensions"]
  resources: ["*"]
  verbs: ["*"]
```

Apply the role:

```
$ kubectl apply -f role.yaml
```

6. Create role binding.

Role Bindings bind roles to user or groups.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
```

```
metadata:
  name: env-a-full-right-binding
  namespace: env-a
subjects:
- kind: User
  name: admin-env-a
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: env-a-full-right
  apiGroup: rbac.authorization.k8s.io
```

7. Edit Kubeconfig.

Add the lines in bold so that iam-authenticator assumes the role.

```
apiVersion: v1
clusters:
- cluster:
    server: "https://xxxxxxxxxx.sk1.us-east-1.eks.amazonaws.com"
    certificate-authority-data: "XXXXXXXX"
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: aws
  name: aws
current-context: aws
kind: Config
```

```
preferences: {}
users:
- name: aws
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      command: aws-iam-authenticator
      args:
        - "token"
        - "-i"
        - "EKS"
        - "-r"
        - "arn:aws:iam::xxxxxxx:role/eks-role-env-a"
```

8. Test your set up.

Pod is launched in the default namespace and fails:

```
$ kubectl run nginx5 --image=nginx
Error from server (Forbidden): deployments.extensions is forbidden: User "admin-env-a"
cannot create deployments.extensions in the namespace "default"
```

In env-a namespace works:

```
$ kubectl run nginx5 --image=nginx -n env-a
deployment.apps "nginx5" created
```

9. References / considerations.

- This is a basic sample. It can definitely be improved with RBAC groups and more accurate rules for the API calls required by

your users.
Refer to RBAC documentation for learn more.

- In the above example, many users assume a role, the role is mapped to a Kubernetes user. Unfortunately, IAM policies do not support Groups in principals which would make easier to manage the mappings.
- This example is AWS cross-account compatible.