Enabling IAM user and role access to your cluster

PDF (eks-ug.pdf#add-user-role) | **RSS (doc-history.rss)**

Access to your cluster using AWS Identity and Access Management (IAM); entities is enabled by the AWS IAM Authenticator for Kubernetes (https://github.com/kubernetes-sigs/aws-iam-authenticator#aws-iam-authenticator-for-kubernetes) , which runs on the Amazon EKS control plane. The authenticator gets its configuration information from the `aws-auth ConfigMap`. For all `aws-auth ConfigMap` settings, see Full Configuration Format (https://github.com/kubernetes-sigs/aws-iam-authenticator#full-configuration-format) on GitHub.

Add IAM users or roles to your Amazon EKS cluster

When you create an Amazon EKS cluster, the AWS Identity and Access Management (IAM) entity user or role, such as a federated user (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers.html) that creates the cluster, is automatically granted `system:masters` permissions in the cluster's role-based access control (RBAC) configuration in the Amazon EKS control plane. This IAM entity doesn't appear in any visible configuration, so make sure to keep track of which IAM entity originally created the cluster. To grant additional AWS users or roles the ability to interact with your cluster, you must edit the `aws-auth ConfigMap` within Kubernetes and create a Kubernetes `rolebinding` or `clusterrolebinding` with the name of a `group` that you specify in the `aws-auth ConfigMap`.

Note

For more information about different IAM identities, see Identities (Users, Groups, and Roles) (https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html) in the *IAM User Guide*. For more information on Kubernetes role-based access control (RBAC) configuration, see Using RBAC Authorization (https://kubernetes.io/docs/reference/access-authn-authz/rbac/) .

To add an IAM user or role to an Amazon EKS cluster

1. Determine which credentials `kubectl` is using to access your cluster. On your computer, you can see which credentials `kubectl` uses with the following command. Replace

~/.kube/config with the path to your `kubeconfig` file if you don't use the default path.

```
cat ~/.kube/config
```

Example output:

```
...
contexts:
- context:
    cluster: my-cluster.region-code.eksctl.io
    user: admin@my-cluster.region-code.eksctl.io
  name: admin@my-cluster.region-code.eksctl.io
current-context: admin@my-cluster.region-code.eksctl.io
...
```

In the previous example output the credentials for a user named *admin* are configured for a cluster named *my-cluster*. If this is the user that created the cluster, then it already has access to your cluster. If it's not the user that created the cluster, then you need to complete the remaining steps to enable cluster access for other users. You can see which other roles or users currently have access to your cluster with the following command:

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

Example output:

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws:iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}
```

```
BinaryData
====

Events:  <none>
```

The previous example is a default `aws-auth ConfigMap`. Only the node instance role has access to the cluster.

2. Make sure that you have existing Kubernetes `roles` and `rolebindings` or `clusterroles` and `clusterrolebindings` that you can map IAM users or roles to. For more information about these resources, see Using RBAC Authorization (https://kubernetes.io/docs/reference/access-authn-authz/rbac/) in the Kubernetes documentation.

   a. View your existing Kubernetes `roles` or `clusterroles`. `Roles` are scoped to a `namespace`, but `clusterroles` are scoped to the cluster.

   ```
   kubectl get roles -A
   ```

   ```
   kubectl get clusterroles
   ```

   b. View the details of any `role` or `clusterrole` returned in the previous output and confirm that it has the permissions (`rules`) that you want your IAM users to have in your cluster.

   Replace *`role-name`* with a `role` name returned in the output from the previous command. Replace *`kube-system`* with the namespace of the `role`.

   ```
   kubectl describe role role-name -n kube-system
   ```

   Replace *`cluster-role-name`* with a `clusterrole` name returned in the output from the previous command.

   ```
   kubectl describe clusterrole cluster-role-name
   ```

   c. View your existing Kubernetes `rolebindings` or `clusterrolebindings`. `Rolebindings` are scoped to a `namespace`, but `clusterrolebindings` are scoped to the cluster.

   ```
   kubectl get rolebindings -A
   ```

```
kubectl get clusterrolebindings
```

d. View the details of any `rolebinding` or clusterrolebinding and confirm that it has a `role` or `clusterrole` from the previous step listed as a `roleRef` and a group name listed for `subjects`.

   Replace *`role-binding-name`* with a `rolebinding` name returned in the output from the previous command. Replace *`kube-system`* with the `namespace` of the `rolebinding`.

```
kubectl describe role role-binding-name -n kube-system
```

   Example output:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-
binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

   Replace *`cluster-role-binding-name`* with a `clusterrolebinding` name returned in the output from the previous command.

```
kubectl describe clusterrole cluster-role-binding-name
```

   Example output:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. Edit the `aws-auth ConfigMap`. You can use a tool such as `eksctl` to update the `ConfigMap` or you can update it manually by editing it.

   Important

   We recommend using `eksctl`, or another tool, to edit the `ConfigMap`. For information about other tools you can use, see Use tools to make changes to the `aws-auth ConfigMap` (https://aws.github.io/aws-eks-best-practices/security/docs/iam/#use-tools-to-make-changes-to-the-aws-auth-configmap) in the Amazon EKS best practices guides. An improperly formatted `aws-auth ConfigMap` can cause you to lose access to your cluster.

   eksctl | **Edit ConfigMap manually**

   Prerequisite

   Version 0.99.0 or later of the `eksctl` command line tool installed on your computer or AWS CloudShell. To install or update `eksctl`, see Installing eksctl (./eksctl.html) .

   a. View the current mappings in the `ConfigMap`. Replace *my-cluster* with the name of your cluster. Replace *region-code* with the AWS Region that your cluster is in.

   ```
   eksctl get iamidentitymapping --cluster my-cluster
   --region=region-code
   ```

   Example output:

```
ARN
USERNAME                                GROUPS
ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-
my-nodegroup-NodeInstanceRole-1XLS7754U3ZPA
system:node:{{EC2PrivateDNSName}}
system:bootstrappers,system:nodes
```

b. Add a mapping for a role. Replace *my-role* with your role name. Replace *eks-console-dashboard-full-access-group* with the name of the group specified in your Kubernetes `rolebinding` or `clusterrolebinding`. Replace *111122223333* with your account ID.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-role \
    --group eks-console-dashboard-full-access-group
\
    --no-duplicate-arns
```

Important

The role ARN can't include a path such as `role/my-team/developers/my-role`. The format of the ARN must be `arn:aws:iam::` *111122223333* `:role/` *my-role*. In this example, `my-team/developers/` needs to be removed.

Example output:

```
...
2022-05-09 14:51:20 [ℹ]  adding identity
"arn:aws:iam::111122223333:role/my-role" to auth
ConfigMap
```

c. Add a mapping for a user. Replace *my-user* with your user name. Replace *eks-console-dashboard-restricted-access-group* with the name of the group specified in your Kubernetes `rolebinding` or `clusterrolebinding`. Replace *111122223333* with your account ID.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-
access-group \
    --no-duplicate-arns
```

Example output:

```
...
2022-05-09 14:53:48 [ℹ]  adding identity
"arn:aws:iam::111122223333:user/my-user" to auth
ConfigMap
```

d. View the mappings in the `ConfigMap` again.

```
eksctl get iamidentitymapping --cluster my-cluster
--region=region-code
```

Example output:

```
ARN
USERNAME                                GROUPS
ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-
my-nodegroup-NodeInstanceRole-1XLS7754U3ZPA
system:node:{{EC2PrivateDNSName}}
system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/my-role
eks-console-dashboard-full-access-group
arn:aws:iam::111122223333:user/my-user
eks-console-dashboard-restricted-access-group
```

Apply the `aws-auth` `ConfigMap` to your cluster

The `aws-auth ConfigMap` is automatically created and applied to your cluster when you create a managed node group or when you create a node group using `eksctl`. It is initially created to allow nodes to join your cluster, but you also use this `ConfigMap` to add role-based access control (RBAC) access to IAM users and roles. If you have not launched self-managed nodes and applied the `aws-auth ConfigMap` to your cluster, you can do so with the following procedure.

To apply the `aws-auth ConfigMap` to your cluster

1. Check to see if you have already applied the `aws-auth ConfigMap`.

   ```
   kubectl describe configmap -n kube-system aws-auth
   ```

   If you receive an error stating "`Error from server (NotFound): configmaps "aws-auth" not found`", then proceed with the following steps to apply the stock `ConfigMap`.

2. Download, edit, and apply the AWS authenticator configuration map.

   a. Download the configuration map.

   ```
   curl -o aws-auth-cm.yaml https://s3.us-west-
   2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-
   auth-cm.yaml
   ```

   b. Open the file with a text editor. Replace *`<ARN of instance role (not instance profile)>`* with the Amazon Resource Name (ARN) of the IAM role associated with your nodes, and save the file. Do not modify any other lines in this file.

   ⚠ **Important**

   The role ARN can't include a path such as `role/my-team/developers/my-role`. The format of the ARN must be `arn:aws:iam::`*`111122223333`*`:role/`*`my-role`*. In this example, `my-team/developers/` needs to be removed.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: aws-auth
     namespace: kube-system
   data:
     mapRoles: |
   ```

```
    - rolearn: <ARN of instance role (not instance
profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

You can inspect the AWS CloudFormation stack outputs for your node groups and look for the following values:

- **InstanceRoleARN** – For node groups that were created with `eksctl`
- **NodeInstanceRole** – For node groups that were created with Amazon EKS vended AWS CloudFormation templates in the AWS Management Console

c. Apply the configuration. This command may take a few minutes to finish.

```
kubectl apply -f aws-auth-cm.yaml
```

Note

If you receive any authorization or resource type errors, see Unauthorized or access denied (kubectl) (./troubleshooting.html#unauthorized) in the troubleshooting section.

3. Watch the status of your nodes and wait for them to reach the `Ready` status.

```
kubectl get nodes --watch
```