<u>/var/log/diego</u> <u>Github Twitter</u>

- airflow
- alb
- ansible
- aws-auth
- aws-console
- bash
- ⊘ bludit
- configmap
- docker
- eks
- git
- gpu
- https
- ingress
- ingress-controller
- init-container
- kops
- ✓ kubectl
- kubernetes
- / Luck and at a con-
- kubernetes-api
- multi-zone
- persistent-volume
- pods
- raspberry-pi
- scheduler
- service

## EKS: Your current user or role does not have access to Kubernetes objects on this EKS cluster.

August 20, 2021 - Last updated: May 12, 2022

When you install EKS for the first time, you receive the following message in the AWS Console UI.

Your current user or role does not have access to Kubernetes objects on this EKS cluster. This may be due to the current user or role not having Kubernetes RBAC permissions to describe cluster resources or not having an entry in the cluster's auth config map.

This happen because your AWS user account doesn't have access to the Kubernetes control plane.

When you deploy a new EKS cluster create a config-map called aws-auth in the namespace kube-system to configure a relation between an AWS IAM user/role and Kubernetes user/group.

If you are the administrator of the EKS cluster you can bind your AWS IAM user/role with the Kubernetes group called system:masters; This is an special group hardcoded into the Kubernetes API with unrestricted rights to the Kubernetes API (the group is bound with the Kubernetes cluster-role cluster-admin).

Depending on your authentication system to AWS will be different the configuration, for example, at my company we use SSO, the user personalizes an AWS IAM role, for that I need to include the AWS IAM role to the list mapRoles inside the config-map aws-auth, but if you have an AWS IAM user account you can modify the list mapUsers.

There is a second option that is more secure and you can segregate more the access, perhaps give read-only access to the Developers and full access to the DevOps.

The idea is create a Kubernetes cluster-role that allows only access to the resources needs it for AWS console UI. The following cluster-role gives read-only (get and list) access to the resources need it for the AWS Console UI.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks-console
rules:
  - apiGroups:
    resources:
      - nodes
     - namespaces
      - pods
    verbs:
      - get
      - list
  - apiGroups:
      - apps
   resources:

    deployments

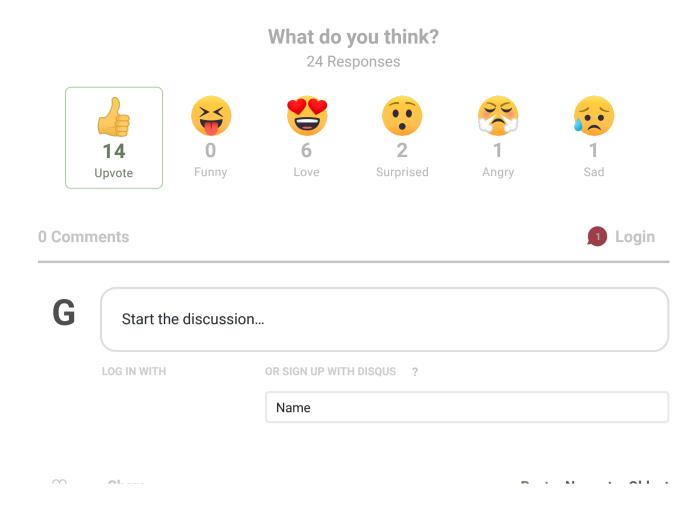
     - daemonsets
      - statefulsets
      - replicasets
    verbs:
      - get
      - list
  - apiGroups:
      - batch
    resources:
      - jobs
    verbs:
      - get
      - list
```

Next, you have to bind the cluste-role with a group.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: eks-console
subjects:
    - kind: Group
    name: eks-console
    apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: eks-console
    apiGroup: rbac.authorization.k8s.io
```

And finally edit the Kubernetes config-map aws-auth and link the AWS IAM role arn:aws:iam::123456789:role/developers with the Kubernetes group eks-console.

```
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
apiVersion: v1
data:
  mapUsers:
    - userarn: arn:aws:iam::123456789:user/diego
     username: diego
     groups:
        - system:masters
  mapRoles: |
   - rolearn: arn:aws:iam::123456789:role/developers
      username: developers
     groups:
       - eks-console
    - rolearn: arn:aws:iam::123456789:role/devops
      username: devops
      groups:
        - system:masters
```



My logs as DevOps Engineer. Technical blog with nerd stuff By Diego Najar. Copyright © 2019-2021. Powered by <u>Gris CMS</u>