ROLE-BASED ACCESS CONTROL (RBAC)

About RBAC

Cloud provider will be having cloud level user access control, which they restrict access and in Kubernetes we can set limit access to users. In Kubernetes cluster Role-based access control, we can define the user access inside the cluster by using RBAC method. Under cloud, Kubernetes cluster service will be there, so first we need to have access to cloud and then second inside Kubernetes to find the data access control is called RBAC.

Four concepts are there:

Rules:- permission can be done with objects in API. inside objects what can be done is called rules.

Subject:- three types:-

user - normal user (like deepan)

group - group of user, in gmail how we get gsuit, with gsuit group will get created and with that we can enforce

system - kubernetes internal components to work within service, its required this subjects.

Roles:- giving access to namespace separately and also give only access to particular namespace for user.

clusterrole:- entire cluster administrations, we can give access to user

rolebinding & clusterbinding:- rules, subject and roles-cluster role combine together is called RBAC

Actual Readme:-

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise.

Role and ClusterRole

1) A role contains rules that represent a set of permissions. Permissions are purely additive (there are no "deny" rules). A role can be defined within a namespace with a Role, or cluster-wide with a ClusterRole

2) A Role can only be used to grant access to resources within a single namespace. Here's an example Role in the "default" namespace that can be used to grant read access to pods:

3) A ClusterRole can be used to grant the same permissions as a Role, but because they are cluster-scoped, they can also be used to grant access to:

RoleBinding and ClusterRoleBinding

A role binding grants the permissions defined in a role to a user or set of users. It holds a list of subjects (users, groups, or service accounts), and a reference to the role being granted. Permissions can be granted within a namespace with a RoleBinding, or cluster-wide with a ClusterRoleBinding

A RoleBinding may reference a Role in the same namespace. The following RoleBinding grants the "pod-reader" role to the user "jane" within the "default" namespace. This allows "jane" to read pods in the "default" namespace.

roleRef is how you will actually create the binding. The kind will be either Role or ClusterRole, and the name will reference the name of the specific Role or ClusterRole you want. In the example below, this RoleBinding is using roleRef to bind the user "jane" to the Role created above named pod-reader

##Create user and add member in cloud##

Note:- I'm creating user (robo2) in server where kubectl tool running and depends on you that same can be done on any server.

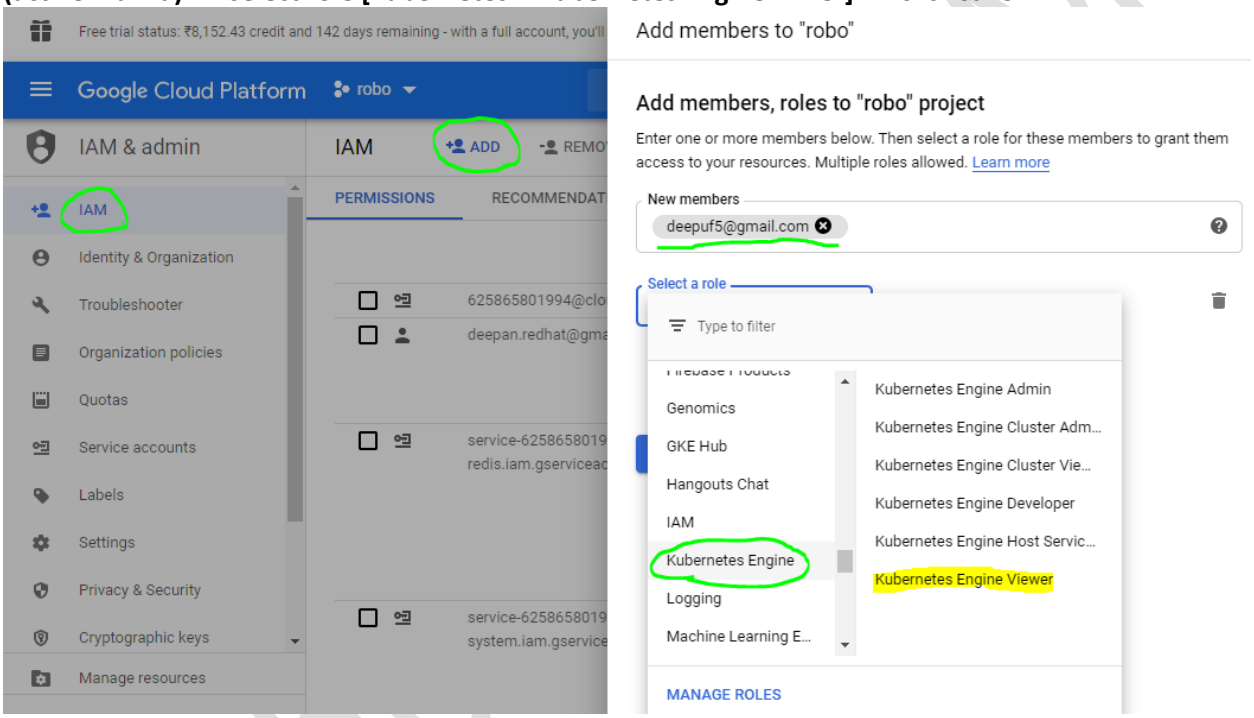
[root@ansikube ~]# su - robo2

[robo2@ansikube ~]$

@@Add member in GCP

Add your alternate email address on GCP member to your project, so that this added ID will be having access to Kubernetes cluster.

Mainmenu :- IAM & admin --> IAM --> ADD (Add members, roles to "robo" project) --> New member (active mail id) --> select role [kubernetes -- kubernetes Engine Viwer] --> click save

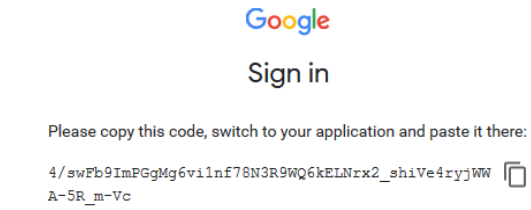


@Switch to robo2 user and run cloud command to login with above ID member of GCP.

$ gcloud auth login

```
[robo2@ansikube ~]$ gcloud auth login

You are running on a Google Compute Engine virtual machine.
It is recommended that you use service accounts for authentication.

Do you want to continue (Y/n)?  Y

Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?code_challenge=tzZqyC3YKKDPPaFBSION
S256&access_type=offline&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&respons
ttps%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googlea
uth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%
```

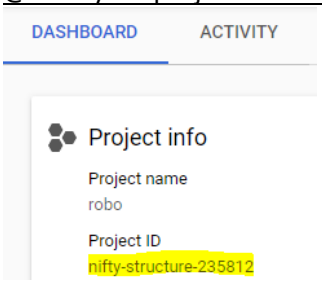
@Copy the following link and paste it on your browser and opened with email ID which you have added in GCP above and drag down, then click Allow.



@Copy the verification code from browser and paste it on terminal.

```
Enter verification code: 4/swFb9ImPGgMg6vi1nf78N3R9WQ6kELNrx2_shiVe4ryjWWA-5R_m-Vc
WARNING: `gcloud auth login` no longer writes application default credentials.
If you need to use ADC, see:
  gcloud auth application-default --help

You are now logged in as [deepuf5@gmail.com].
Your current project is [nifty-structure-235812].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
[robo2@ansikube ~]$
```

@Take your project ID from GCP console.



gcloud config set project nifty-structure-235812

```
[robo2@ansikube ~]$ gcloud config set project nifty-structure-235812
Updated property [core/project].
[robo2@ansikube ~]$
```

@Mention your cluster name and region that where cluster located.

gcloud container clusters get-credentials robo --region us-central1-a

```
[robo2@ansikube ~]$ gcloud container clusters get-credentials robo --region us-central1-a
Fetching cluster endpoint and auth data.
kubeconfig entry generated for robo.
[robo2@ansikube ~]$
```

#kubectl get nodes

```
[robo2@ansikube ~]$ kubectl get nodes
NAME                                  STATUS   ROLES    AGE    VERSION
gke-robo-default-pool-682616c4-ldd4   Ready    <none>   131m   v1.13.11-gke.9
gke-robo-default-pool-682616c4-ml55   Ready    <none>   131m   v1.13.11-gke.9
gke-robo-default-pool-682616c4-pd48   Ready    <none>   131m   v1.13.11-gke.9
[robo2@ansikube ~]$
```

Note: - Now you can able to list the kubernetes objects via system local user, which will get authenticate in GCP by using you alternate email ID (here deepuf5@gmail.com), which has added as member of GCP project.

##Create a role and roles-bindings##

Make sure of have access to kubernetes cluster (https://cloud.google.com/kubernetes-engine/docs/how-to/role-based-access-control#setting_up_role-based_access_control)

@Come back to root user and run cloud command with primary user of your GCP login.
gcloud auth login

```
You are now logged in as [deepan.redhat@gmail.com].
Your current project is [nifty-structure-235812].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
[root@ansikube ~]#
```

gcloud auth login
gcloud config set project nifty-structure-235812
gcloud container clusters get-credentials robo --region us-central1-a
kubectl get nodes

Note: - Here lab setup has done on GCP cloud, that's why gloud command is being used, incase if you have done setup on any other cloud provider, then kindly refer their command instructions.

@Create roles yml with "read,watch,list"
cat read_role.yml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

kubectl apply -f read_role.yml
kubectl get role

```
[root@ansikube manifest]# kubectl get role
NAME         AGE
pod-reader   12s
[root@ansikube manifest]#
```

kubectl describe roles pod-reader

```
[root@ansikube manifest]# kubectl describe role pod-reader
Name:         pod-reader
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"rbac.authorization.k8s.io/v1","kind":"Role"
ules"...
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  pods       []                 []              [get watch list]
[root@ansikube manifest]#
```

cat role-binding.yml

```
apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "deepuf5@gmail.com" to read pods in the "default" namespace.
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: deepuf5@gmail.com
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io
```

kubectl apply -f role-binding.yml
kubectl get roleBindings -o wide

```
[root@ansikube manifest]# kubectl get roleBindings -o wide
NAME        AGE
read-pods   24s
[root@ansikube manifest]#
```

kubectl describe roleBindings read-pods

```
[root@ansikube manifest]# kubectl describe roleBindings read-pods
Name:         read-pods
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"rbac.authorization.k8s.io/v1","kind":"RoleBinding"
t"},"...
Role:
  Kind:  Role
  Name:  pod-reader
Subjects:
  Kind  Name              Namespace
  ----  ----              ---------
  User  deepuf5@gmail.com
[root@ansikube manifest]#
```

@@Create a pods on default namespace and dev namespace.

kubectl create namespace dev
kubectl get ns

```
[root@ansikube manifest]# kubectl get ns
NAME          STATUS   AGE
default       Active   6h25m
dev           Active   20s
```

cat pods.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-web1
  labels:
    env: web
  namespace: dev
spec:
  containers:
    - name: website1
      image: nginx:1.16
      ports:
        - containerPort: 80
```

kubectl apply -f pods.yml
kubectl get pods
kubectl get pods --namespace=dev

```
[root@ansikube manifest]# kubectl get pods
No resources found in default namespace.
[root@ansikube manifest]# kubectl get pods --namespace=dev
NAME         READY   STATUS    RESTARTS   AGE
nginx-web1   1/1     Running   0          80s
[root@ansikube manifest]#
```

cat pods_defaultns.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: ngnix-web2
  labels:
    env: web
  namespace: default
spec:
  containers:
    - name: website1
      image:  nginx:1.16
      ports:
        - containerPort: 80
```

kubectl apply -f pods_defaultns.yml

kubectl get pods

```
[root@ansikube manifest]# kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
ngnix-web2   1/1     Running   0          10s
[root@ansikube manifest]#
```

@@Switch to user and try to access from ID which has added as member of this cluster.

kubectl get pods --namespace dev

kubectl get pods

```
[root@ansikube manifest]# su - robo2
Last login: Wed Nov  6 06:19:36 UTC 2019 on pts/0
[robo2@ansikube ~]$ kubectl get pods --namespace dev
NAME         READY   STATUS    RESTARTS   AGE
nginx-web1   1/1     Running   0          6m23s
[robo2@ansikube ~]$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
ngnix-web2   1/1     Running   0          2m11s
[robo2@ansikube ~]$
```

@@Try to create pod on dev and default namespace by user robo2 – gcloud user deepuf5@gmail.com.

cat pods_dev.yml

```
[robo2@ansikube ~]$ cat pods_dev.yml
apiVersion: v1
kind: Pod
metadata:
  name: ngnix-web3
  labels:
    env: web
  namespace: dev
spec:
  containers:
    - name: website1
      image:  nginx:1.16
      ports:
        - containerPort: 80
```

kubectl apply -f pods_dev.yml

```
[robo2@ansikube ~]$ kubectl apply -f pods_dev.yml
Error from server (Forbidden): error when creating "pods_dev.yml": pods is forbidden: User "deepuf5@gmail.com" cannot create resource "pods" in API
roup "" in the namespace "dev": Required "container.pods.create" permission.
[robo2@ansikube ~]$
```

Note: - since the role has created with get, watch, list only, to create pod from user then roles have to be update with create verbs.

@@add create verbs on role.yml, so that robo2 user will get write option.

cat write_role.yml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list", "create"]
```

kubectl describe roles pod-reader

```
[root@ansikube manifest]# kubectl describe roles pod-reader
Name:         pod-reader
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"rbac.authorization.k8s.io/v1","kind":"Role",
ules"...
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  pods       []                 []              [get watch list create]
[root@ansikube manifest]#
```

@Now Try to create pod on namespace by user robo2.

kubectl apply -f pods_dev.yml

kubectl get pods

```
[robo2@ansikube ~]$ kubectl apply -f pods_dev.yml
pod/ngnix-web3 created
[robo2@ansikube ~]$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
ngnix-web2   1/1     Running   0          29m
ngnix-web3   1/1     Running   0          6s
[robo2@ansikube ~]$
```

Note: - Now user able to create pod because roles. Same wait if you want to create cluster wide then you take reference from below link.

https://kubernetes.io/docs/reference/access-authn-authz/rbac/