RBAC

12 January 2021 11:42

Role-based access control (RBAC)

- Method of regulating access to computer or network resources based on the roles of individual users within organization
- Uses API rbac.authorization.k8s.io

RBAC API declares four kind of kubernetes objects

Role = RoleBinding
ClusterRole = ClusterRoleBinding

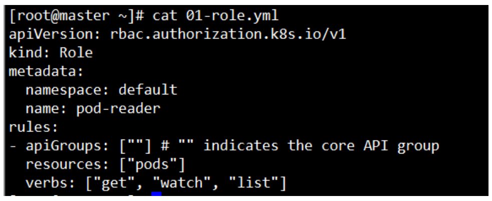
1. **Role** : Always set permissions within a particular namespace.
2. **ClusterRole** :
   - Set permissions on non-namespaced resource.
   - Define permissions on namespaced resources and be granted within individual namespace(s)
   - Define permissions on namespaced resources and be granted across all namespaces
   - Define permissions on cluster-scoped resources

3) **RoleBinding** :
   - Grants permissions within a specific namespace.
   - RoleBinding may reference any Role in the same namespace.
   - RoleBinding can reference a ClusterRole and bind that ClusterRole to the namespace of the RoleBinding.

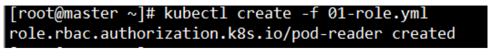
4) **ClusterRoleBinding**
   - Grants access cluster-wide.
   - If you want to bind a ClusterRole to all the namespaces in your cluster, you use a ClusterRoleBinding

Role and Role Bindings

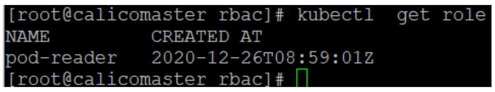
1) **Create Role**



```
[root@master ~]# cat 01-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"]
```

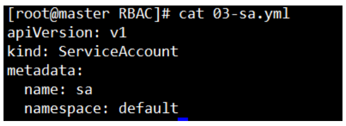


```
[root@master ~]# kubectl create -f 01-role.yml
role.rbac.authorization.k8s.io/pod-reader created
```

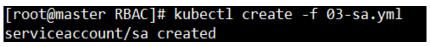


```
[root@calicomaster rbac]# kubectl  get role
NAME         CREATED AT
pod-reader   2020-12-26T08:59:01Z
[root@calicomaster rbac]#
```

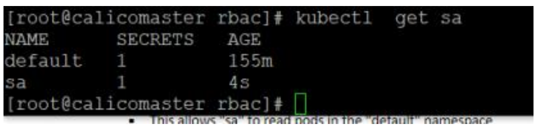
1) **Create Service Account**



```
[root@master RBAC]# cat 03-sa.yml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: sa
  namespace: default
```



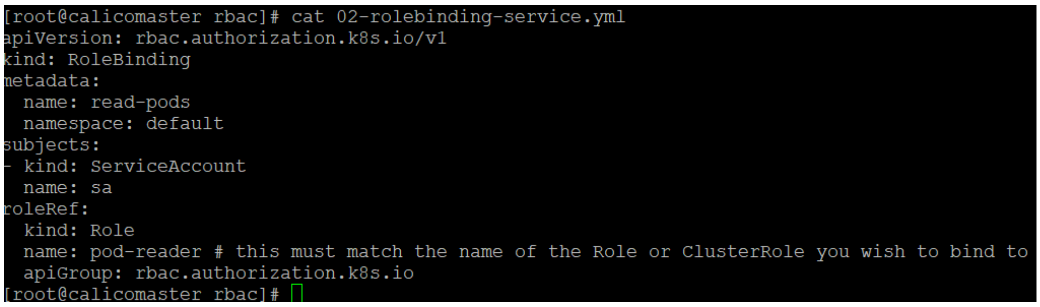
```
[root@master RBAC]# kubectl create -f 03-sa.yml
serviceaccount/sa created
```



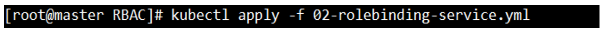
```
[root@calicomaster rbac]# kubectl  get sa
NAME      SECRETS   AGE
default   1         155m
sa        1         4s
[root@calicomaster rbac]#
```

1) **Create RoleBinding for Service Account**

- Grants the "pod-reader" Role to the service account "sa" within the "default" namespace.
- This allows "sa" to read pods in the "default" namespace



```
[root@calicomaster rbac]# cat 02-rolebinding-service.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: ServiceAccount
  name: sa
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
[root@calicomaster rbac]#
```

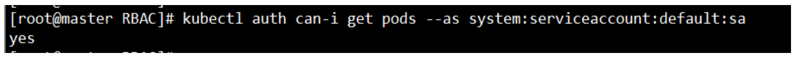


```
[root@master RBAC]# kubectl apply -f 02-rolebinding-service.yml
```

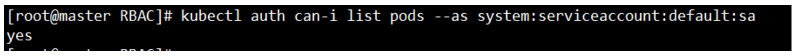
Verification :

Verification can be done through auth can-I .

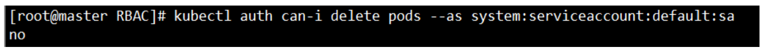
# kubectl auth can-i get pods --as system:serviceaccount:default:sa
# kubectl auth can-i list pods --as system:serviceaccount:default:sa
# kubectl auth can-i delete pods --as system:serviceaccount:default:sa



```
[root@master RBAC]# kubectl auth can-i get pods --as system:serviceaccount:default:sa
yes
```



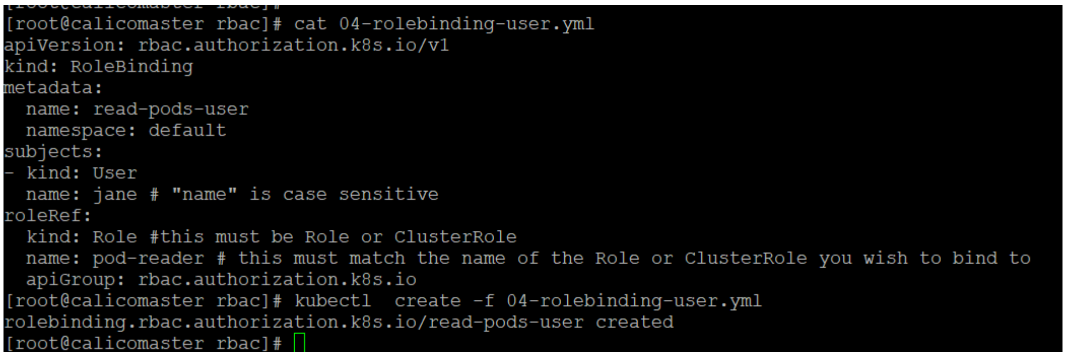
```
[root@master RBAC]# kubectl auth can-i list pods --as system:serviceaccount:default:sa
yes
```



```
[root@master RBAC]# kubectl auth can-i delete pods --as system:serviceaccount:default:sa
no
```

4) **Create Role binding for User**

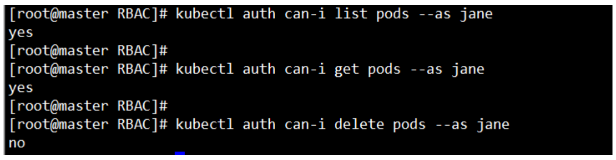
- Grants the "pod-reader" Role to the user "jane" within the "default" namespace.
- This allows "jane" to read pods in the "default" namespace



```
[root@calicomaster rbac]# cat 04-rolebinding-user.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods-user
  namespace: default
subjects:
- kind: User
  name: jane # "name" is case sensitive
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io
[root@calicomaster rbac]# kubectl  create -f 04-rolebinding-user.yml
rolebinding.rbac.authorization.k8s.io/read-pods-user created
[root@calicomaster rbac]#
```

Verification

# kubectl auth can-i list pods --as jane
# kubectl auth can-i get pods --as jane
# kubectl auth can-i delete pods --as jane

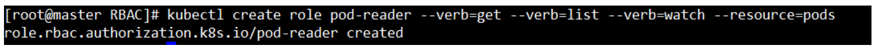


```
[root@master RBAC]# kubectl auth can-i list pods --as jane
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i get pods --as jane
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i delete pods --as jane
no
```

Imperative commands for Role and RoleBindings

Create Role

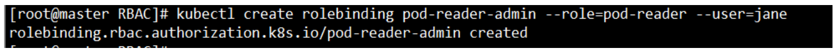
# kubectl create role pod-reader --verb=get --verb=list --verb=watch --resource=pods



```
[root@master RBAC]# kubectl create role pod-reader --verb=get --verb=list --verb=watch --resource=pods
role.rbac.authorization.k8s.io/pod-reader created
```

Create Rolebinding

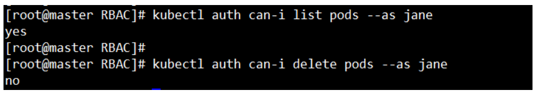
# kubectl create rolebinding pod-reader-admin --role=pod-reader --user=jane



```
[root@master RBAC]# kubectl create rolebinding pod-reader-admin --role=pod-reader --user=jane
rolebinding.rbac.authorization.k8s.io/pod-reader-admin created
```

Verification

# kubectl auth can-i list pods --as jane
# kubectl auth can-i delete pods --as jane

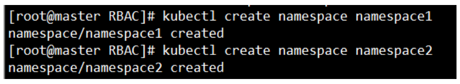


```
[root@master RBAC]# kubectl auth can-i list pods --as jane
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i delete pods --as jane
no
```

ClusterRole and ClusterRoleBinding

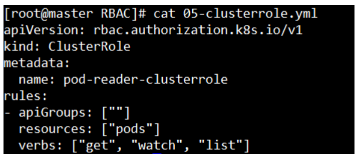
1) **Create NameSpaces**

- Firstly create 2 Namespace i.e. namespace1 and namespace2

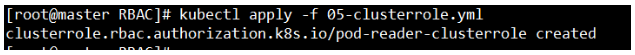


```
[root@master RBAC]# kubectl create namespace namespace1
namespace/namespace1 created
[root@master RBAC]# kubectl create namespace namespace2
namespace/namespace2 created
```

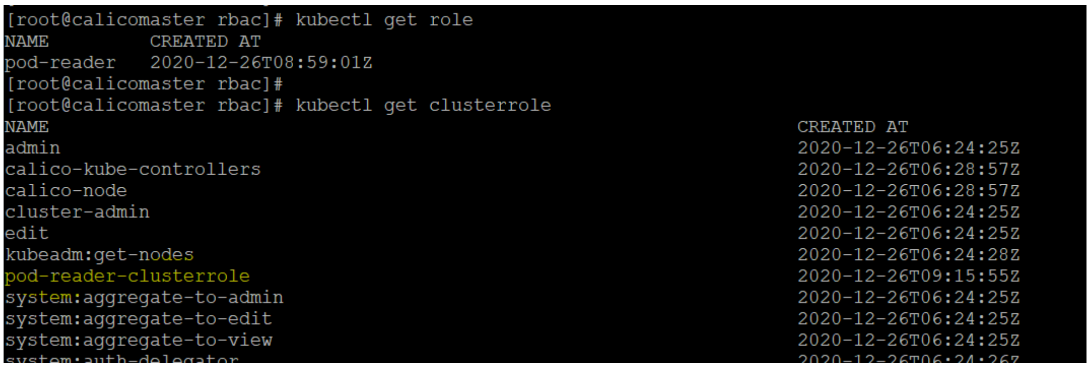
2) **Create ClusterRole**



```
[root@master RBAC]# cat 05-clusterrole.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: pod-reader-clusterrole
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```



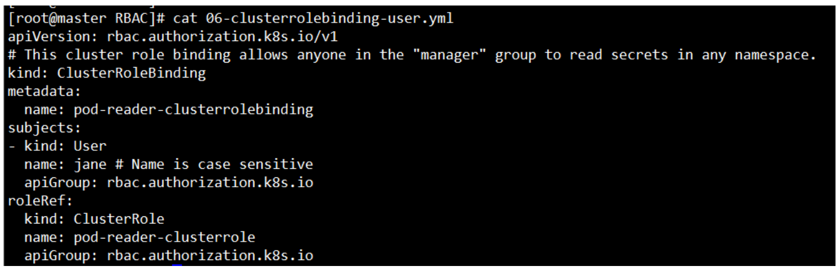
```
[root@master RBAC]# kubectl apply -f 05-clusterrole.yml
clusterrole.rbac.authorization.k8s.io/pod-reader-clusterrole created
```



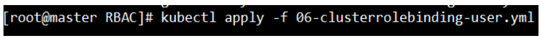
```
[root@calicomaster rbac]# kubectl get role
NAME         CREATED AT
pod-reader   2020-12-26T08:59:01Z
[root@calicomaster rbac]#
[root@calicomaster rbac]# kubectl get clusterrole
NAME                                                                   CREATED AT
admin                                                                  2020-12-26T06:24:25Z
calico-kube-controllers                                                2020-12-26T06:28:57Z
calico-node                                                            2020-12-26T06:28:57Z
cluster-admin                                                          2020-12-26T06:24:25Z
edit                                                                   2020-12-26T06:24:25Z
kubeadm:get-nodes                                                      2020-12-26T06:24:28Z
pod-reader-clusterrole                                                 2020-12-26T09:15:55Z
system:aggregate-to-admin                                              2020-12-26T06:24:25Z
system:aggregate-to-edit                                               2020-12-26T06:24:25Z
system:aggregate-to-view                                               2020-12-26T06:24:25Z
system:auth-delegator                                                  2020-12-26T06:24:26Z
```

3) **Create clusterrolebinding for User**

- Associate with clusterrole "pod-reader-clusterrole"



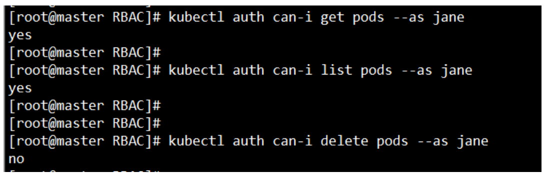
```
[root@master RBAC]# cat 06-clusterrolebinding-user.yml
apiVersion: rbac.authorization.k8s.io/v1
# This cluster role binding allows anyone in the "manager" group to read secrets in any namespace.
kind: ClusterRoleBinding
metadata:
  name: pod-reader-clusterrolebinding
subjects:
- kind: User
  name: jane # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: pod-reader-clusterrole
  apiGroup: rbac.authorization.k8s.io
```



```
[root@master RBAC]# kubectl apply -f 06-clusterrolebinding-user.yml
```

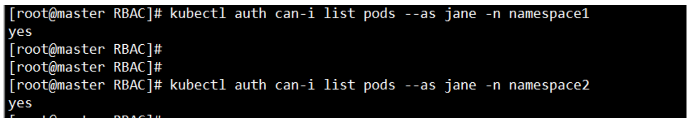
Verification:

# kubectl auth can-i get pods --as jane
# kubectl auth can-i list pods --as jane
# kubectl auth can-i delete pods --as jane



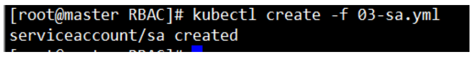
```
[root@master RBAC]# kubectl auth can-i get pods --as jane
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i list pods --as jane
yes
[root@master RBAC]#
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i delete pods --as jane
no
```

Since it is bind to cluster role , so it has permission in all namespaces



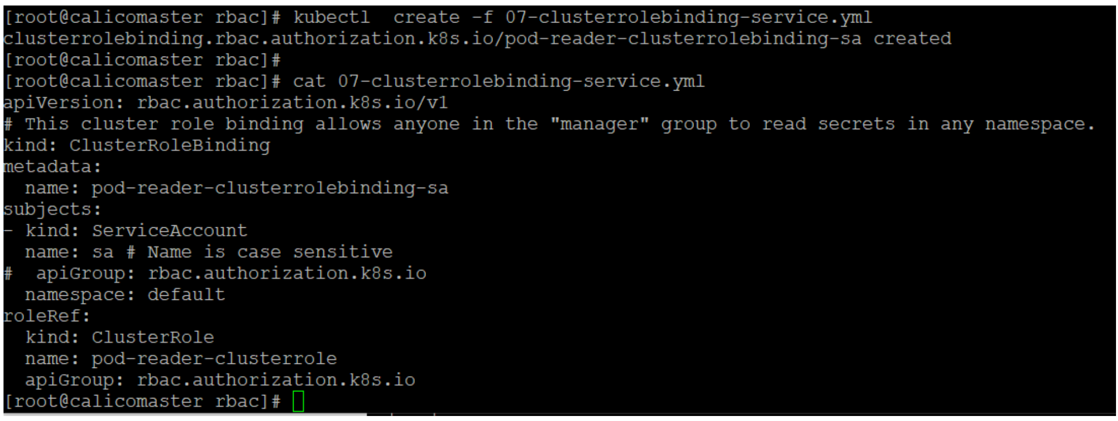
```
[root@master RBAC]# kubectl auth can-i list pods --as jane -n namespace1
yes
[root@master RBAC]#
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i list pods --as jane -n namespace2
yes
```

4) **Create service account**



```
[root@master RBAC]# kubectl create -f 03-sa.yml
serviceaccount/sa created
```

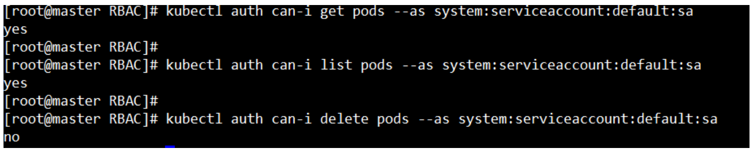
5) ClusterRoleBinding for Service account



```
[root@calicomaster rbac]# kubectl  create -f 07-clusterrolebinding-service.yml
clusterrolebinding.rbac.authorization.k8s.io/pod-reader-clusterrolebinding-sa created
[root@calicomaster rbac]#
[root@calicomaster rbac]# cat 07-clusterrolebinding-service.yml
apiVersion: rbac.authorization.k8s.io/v1
# This cluster role binding allows anyone in the "manager" group to read secrets in any namespace.
kind: ClusterRoleBinding
metadata:
  name: pod-reader-clusterrolebinding-sa
subjects:
- kind: ServiceAccount
  name: sa # Name is case sensitive
#   apiGroup: rbac.authorization.k8s.io
  namespace: default
roleRef:
  kind: ClusterRole
  name: pod-reader-clusterrole
  apiGroup: rbac.authorization.k8s.io
[root@calicomaster rbac]#
```

Verification

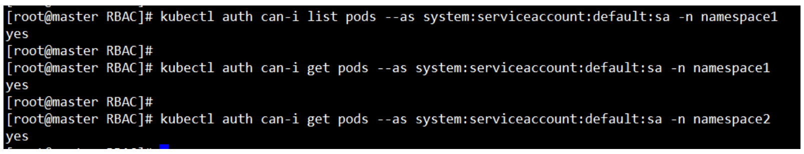
# kubectl auth can-i get pods --as system:serviceaccount:default:sa
# kubectl auth can-i list pods --as system:serviceaccount:default:sa
# kubectl auth can-i delete pods --as system:serviceaccount:default:sa



```
[root@master RBAC]# kubectl auth can-i get pods --as system:serviceaccount:default:sa
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i list pods --as system:serviceaccount:default:sa
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i delete pods --as system:serviceaccount:default:sa
no
```

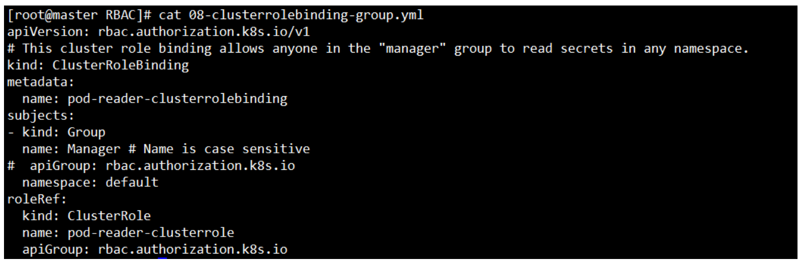
It is able to list and get in difference namespace because it is binding to clusterrole

# kubectl auth can-i list pods --as system:serviceaccount:default:sa -n namespace1
# kubectl auth can-i get pods --as system:serviceaccount:default:sa -n namespace1
# kubectl auth can-i get pods --as system:serviceaccount:default:sa -n namespace2

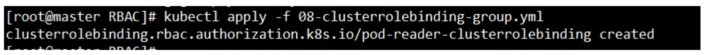


```
[root@master RBAC]# kubectl auth can-i list pods --as system:serviceaccount:default:sa -n namespace1
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i get pods --as system:serviceaccount:default:sa -n namespace1
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i get pods --as system:serviceaccount:default:sa -n namespace2
yes
```

6) ClusterRoleBinding for Group



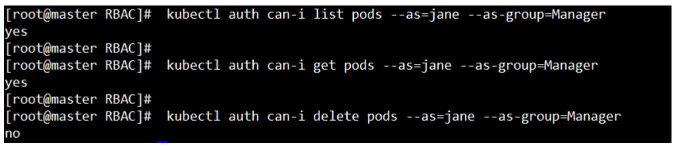
```
[root@master RBAC]# cat 08-clusterrolebinding-group.yml
apiVersion: rbac.authorization.k8s.io/v1
# This cluster role binding allows anyone in the "manager" group to read secrets in any namespace.
kind: ClusterRoleBinding
metadata:
  name: pod-reader-clusterrolebinding
subjects:
- kind: Group
  name: Manager # Name is case sensitive
#  apiGroup: rbac.authorization.k8s.io
  namespace: default
roleRef:
  kind: ClusterRole
  name: pod-reader-clusterrole
  apiGroup: rbac.authorization.k8s.io
```



```
[root@master RBAC]# kubectl apply -f 08-clusterrolebinding-group.yml
clusterrolebinding.rbac.authorization.k8s.io/pod-reader-clusterrolebinding created
```

Verification:

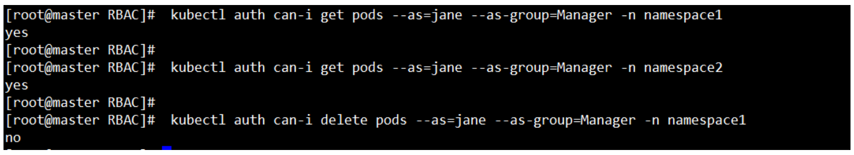
# kubectl auth can-i list pods --as=jane --as-group=Manager
# kubectl auth can-i get pods --as=jane --as-group=Manager
# kubectl auth can-i delete pods --as=jane --as-group=Manager



```
[root@master RBAC]#  kubectl auth can-i list pods --as=jane --as-group=Manager
yes
[root@master RBAC]#
[root@master RBAC]#  kubectl auth can-i get pods --as=jane --as-group=Manager
yes
[root@master RBAC]#
[root@master RBAC]#  kubectl auth can-i delete pods --as=jane --as-group=Manager
no
```

It works in other namespace too

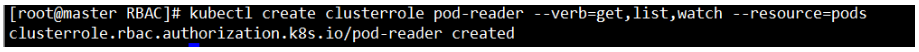
# kubectl auth can-i get pods --as=jane --as-group=Manager -n namespace1
# kubectl auth can-i get pods --as=jane --as-group=Manager -n namespace2
# kubectl auth can-i delete pods --as=jane --as-group=Manager -n namespace1


```
[root@master RBAC]#  kubectl auth can-i get pods --as=jane --as-group=Manager -n namespace1
yes
[root@master RBAC]#
[root@master RBAC]#  kubectl auth can-i get pods --as=jane --as-group=Manager -n namespace2
yes
[root@master RBAC]#
[root@master RBAC]#  kubectl auth can-i delete pods --as=jane --as-group=Manager -n namespace1
no
```

Imperative Command for ClusterRole and ClusterRolebindings

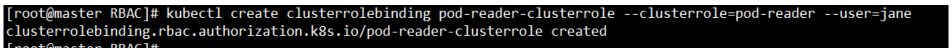
Create ClusterRole

# kubectl create clusterrole pod-reader --verb=get,list,watch --resource=pods


```
[root@master RBAC]# kubectl create clusterrole pod-reader --verb=get,list,watch --resource=pods
clusterrole.rbac.authorization.k8s.io/pod-reader created
```

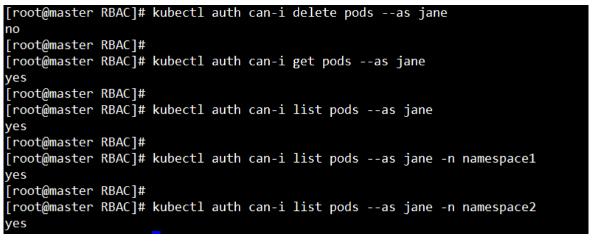
Create ClusterRoleBindings

# kubectl create clusterrolebinding pod-reader-clusterrole --clusterrole=pod-reader --user=jane


```
[root@master RBAC]# kubectl create clusterrolebinding pod-reader-clusterrole --clusterrole=pod-reader --user=jane
clusterrolebinding.rbac.authorization.k8s.io/pod-reader-clusterrole created
```

Verification

# kubectl auth can-i delete pods --as jane
# kubectl auth can-i get pods --as jane
# kubectl auth can-i list pods --as jane
# kubectl auth can-i list pods --as jane -n namespace1
# kubectl auth can-i list pods --as jane -n namespace2

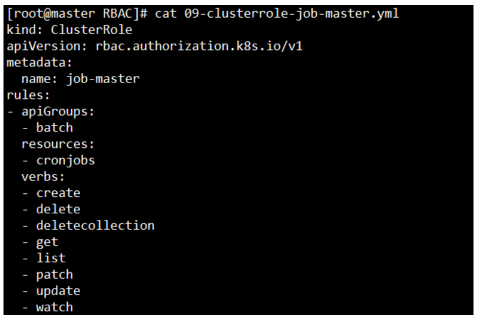

```
[root@master RBAC]# kubectl auth can-i delete pods --as jane
no
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i get pods --as jane
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i list pods --as jane
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i list pods --as jane -n namespace1
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i list pods --as jane -n namespace2
yes
```

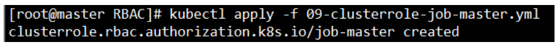
Resource Accessibility in different Namespaces

Objective: Access resources in different namespace

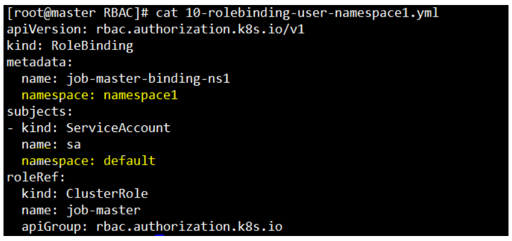
Scenario:
- Create **ClusterRole** and bind it with **RoleBinding**
- Need Service account.
  - We used as earlier crated "sa"
- Need 2 namespace :
  - Used as created earlier : namesapce1 and namespace2

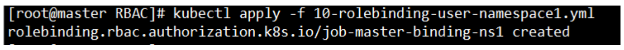
1) **Create cluster role named Job-master**


```
[root@master RBAC]# cat 09-clusterrole-job-master.yml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: job-master
rules:
- apiGroups:
  - batch
  resources:
  - cronjobs
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
```


```
[root@master RBAC]# kubectl apply -f 09-clusterrole-job-master.yml
clusterrole.rbac.authorization.k8s.io/job-master created
```

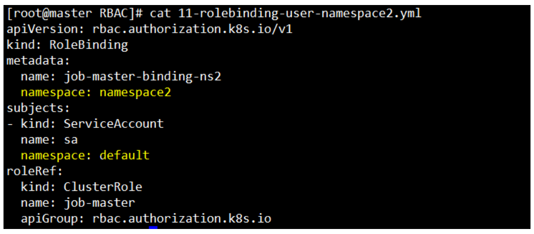
2) **Create Rolebinding in namespace1**


```
[root@master RBAC]# cat 10-rolebinding-user-namespace1.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: job-master-binding-ns1
  namespace: namespace1
subjects:
- kind: ServiceAccount
  name: sa
  namespace: default
roleRef:
  kind: ClusterRole
  name: job-master
  apiGroup: rbac.authorization.k8s.io
```

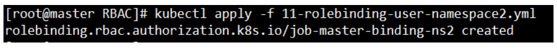


```
[root@master RBAC]# kubectl apply -f 10-rolebinding-user-namespace1.yml
rolebinding.rbac.authorization.k8s.io/job-master-binding-ns1 created
```

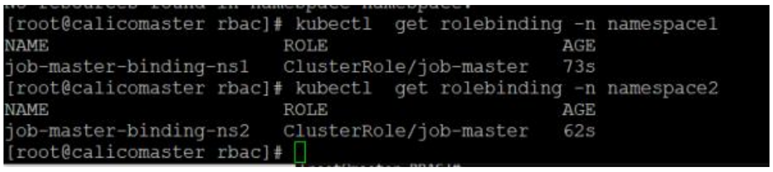
3) Create Rolebinding in namespace2



```
[root@master RBAC]# cat 11-rolebinding-user-namespace2.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: job-master-binding-ns2
  namespace: namespace2
subjects:
- kind: ServiceAccount
  name: sa
  namespace: default
roleRef:
  kind: ClusterRole
  name: job-master
  apiGroup: rbac.authorization.k8s.io
```



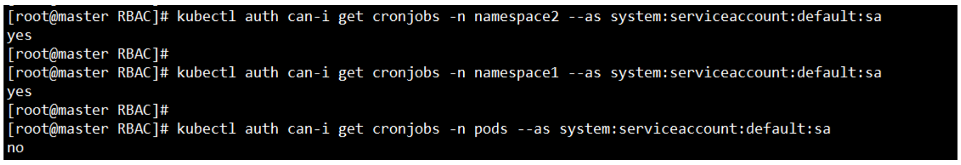
```
[root@master RBAC]# kubectl apply -f 11-rolebinding-user-namespace2.yml
rolebinding.rbac.authorization.k8s.io/job-master-binding-ns2 created
```



```
[root@calicomaster rbac]# kubectl  get rolebinding -n namespace1
NAME                      ROLE                     AGE
job-master-binding-ns1    ClusterRole/job-master   73s
[root@calicomaster rbac]# kubectl  get rolebinding -n namespace2
NAME                      ROLE                     AGE
job-master-binding-ns2    ClusterRole/job-master   62s
[root@calicomaster rbac]#
```

Verification

# kubectl auth can-i get cronjobs -n namespace2 --as system:serviceaccount:default:sa
# kubectl auth can-i get cronjobs -n namespace1 --as system:serviceaccount:default:sa
# kubectl auth can-i get cronjobs -n pods --as system:serviceaccount:default:sa



```
[root@master RBAC]# kubectl auth can-i get cronjobs -n namespace2 --as system:serviceaccount:default:sa
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i get cronjobs -n namespace1 --as system:serviceaccount:default:sa
yes
[root@master RBAC]#
[root@master RBAC]# kubectl auth can-i get cronjobs -n pods --as system:serviceaccount:default:sa
no
```