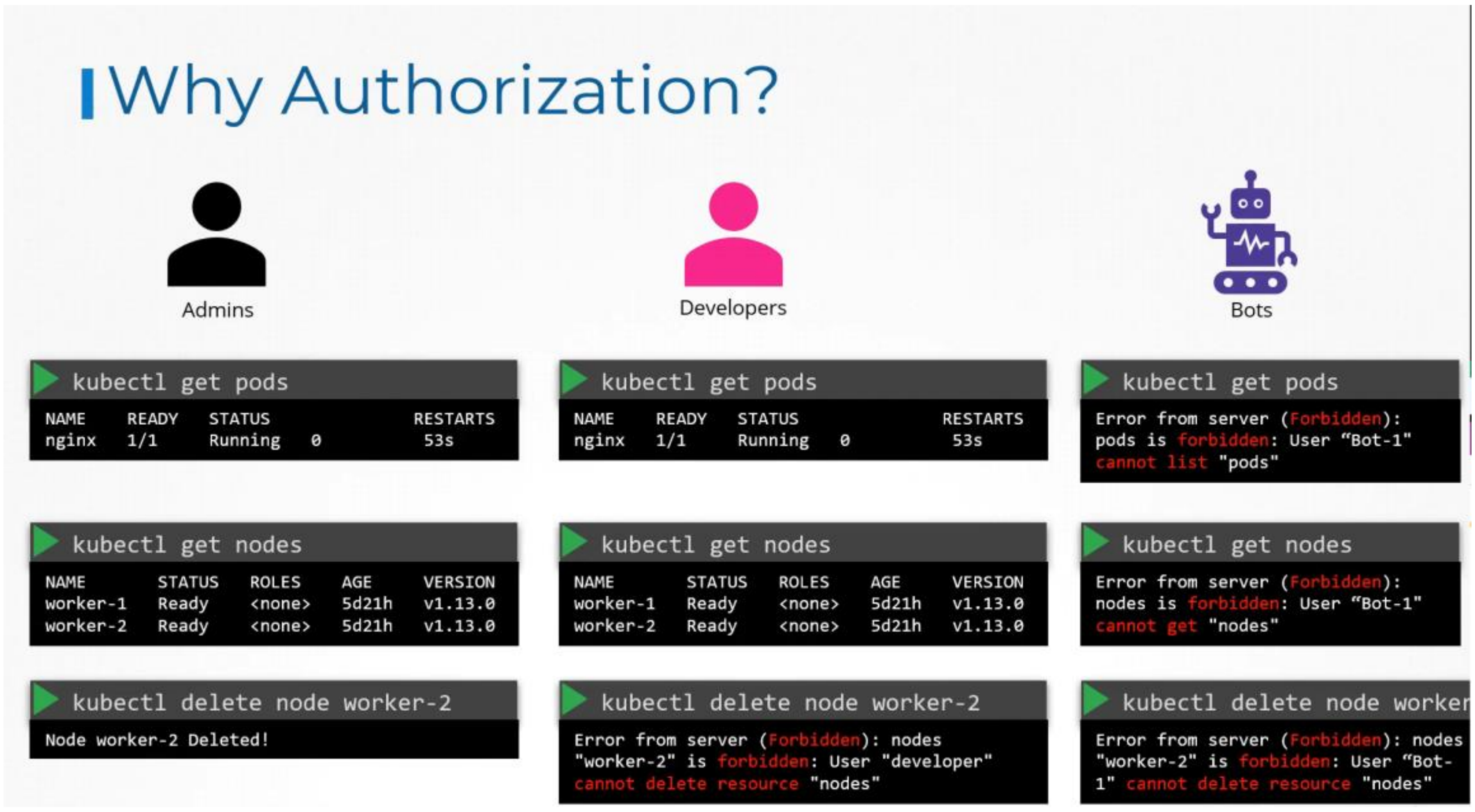
# AUTHORIZATION



## Why Authorization?

Admins:

```
kubectl get pods
NAME   READY  STATUS          RESTARTS
nginx  1/1    Running   0       53s
```

```
kubectl get nodes
NAME      STATUS  ROLES   AGE    VERSION
worker-1  Ready   <none>  5d21h  v1.13.0
worker-2  Ready   <none>  5d21h  v1.13.0
```

```
kubectl delete node worker-2
Node worker-2 Deleted!
```

Developers:

```
kubectl get pods
NAME   READY  STATUS          RESTARTS
nginx  1/1    Running   0       53s
```

```
kubectl get nodes
NAME      STATUS  ROLES   AGE    VERSION
worker-1  Ready   <none>  5d21h  v1.13.0
worker-2  Ready   <none>  5d21h  v1.13.0
```

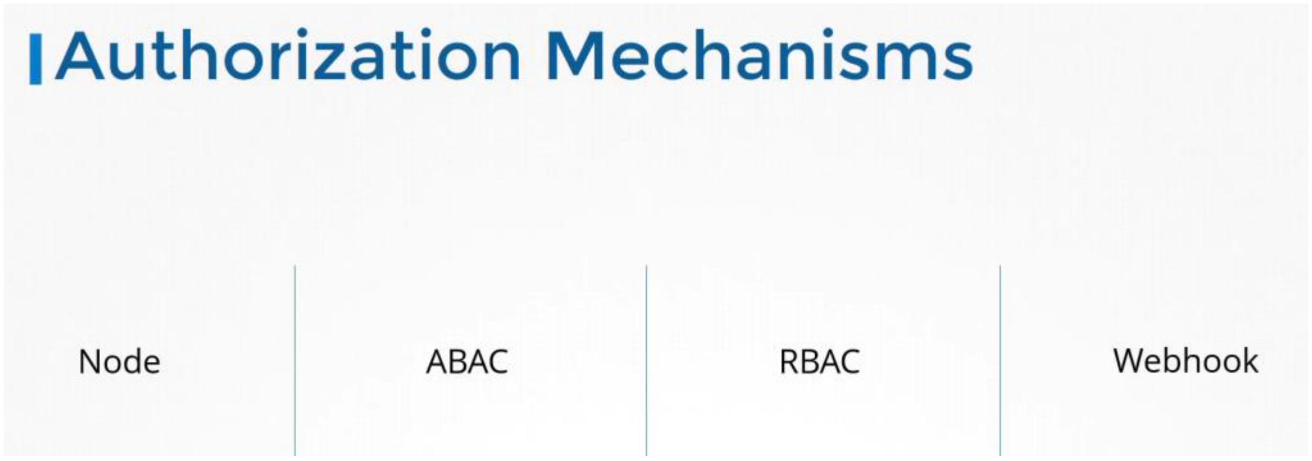
```
kubectl delete node worker-2
Error from server (Forbidden): nodes
"worker-2" is forbidden: User "developer"
cannot delete resource "nodes"
```

Bots:

```
kubectl get pods
Error from server (Forbidden):
pods is forbidden: User "Bot-1"
cannot list "pods"
```

```
kubectl get nodes
Error from server (Forbidden):
nodes is forbidden: User "Bot-1"
cannot get "nodes"
```

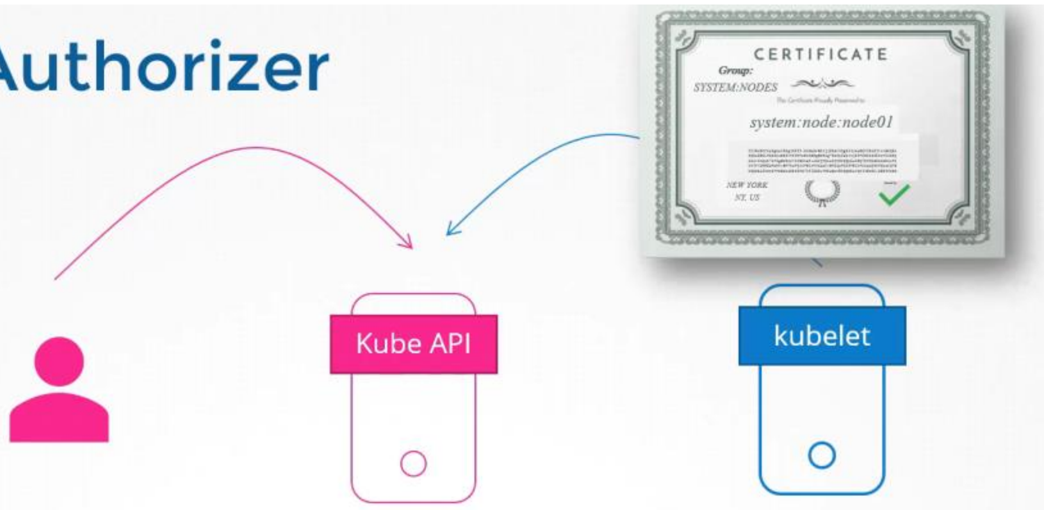
```
kubectl delete node worker
Error from server (Forbidden): nodes
"worker-2" is forbidden: User "Bot-
1" cannot delete resource "nodes"
```



## Authorization Mechanisms

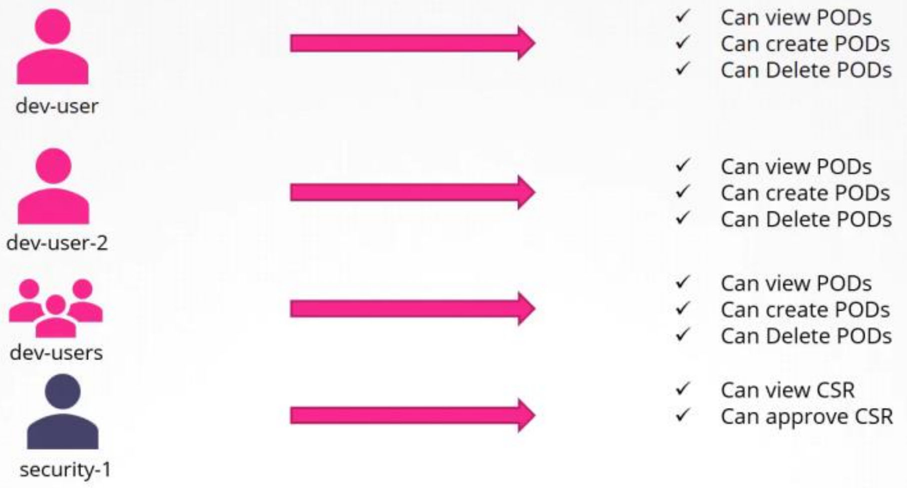
Node | ABAC | RBAC | Webhook

## Node Authorizer

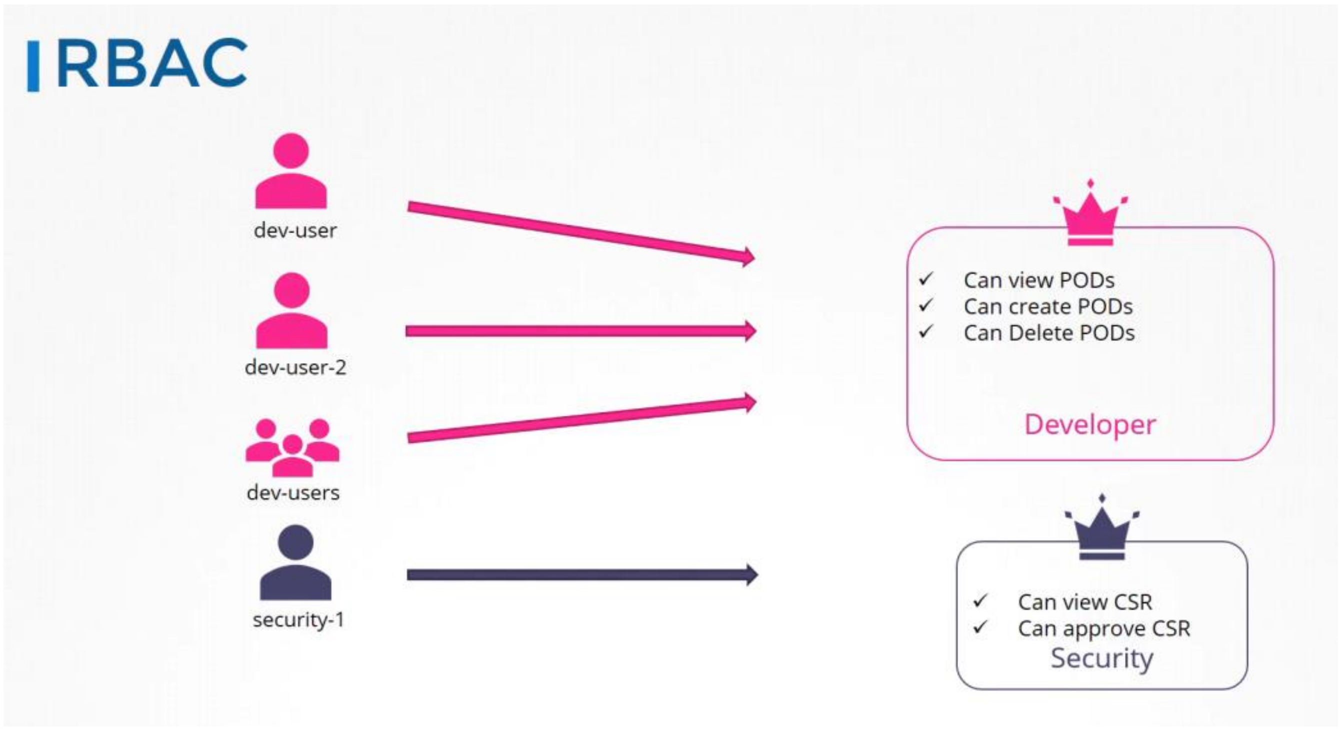


- Read
  - Services
  - Endpoints
  - Nodes
  - Pods
- Write
  - Node status
  - Pod status
  - events

## ABAC

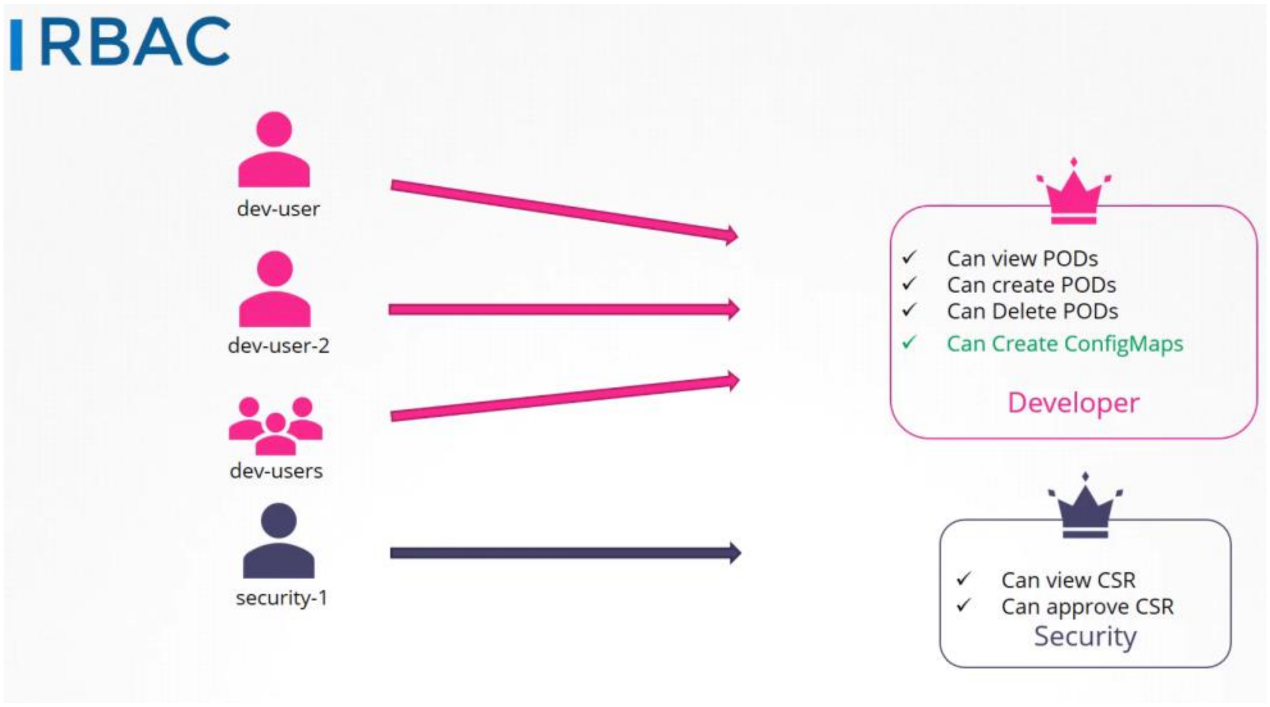


```
{"kind": "Policy", "spec": {"user": “dev-user", "namespace": "*", "resource": “pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {"user": “dev-user-2", "namespace": "*", "resource": “pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {“group": "dev-users", "namespace": "*", "resource": “pods", "apiGroup": "*"}}
{"kind": "Policy", "spec": {"user": “security-1", "namespace": "*", "resource": “csr", "apiGroup": "*"}}
```



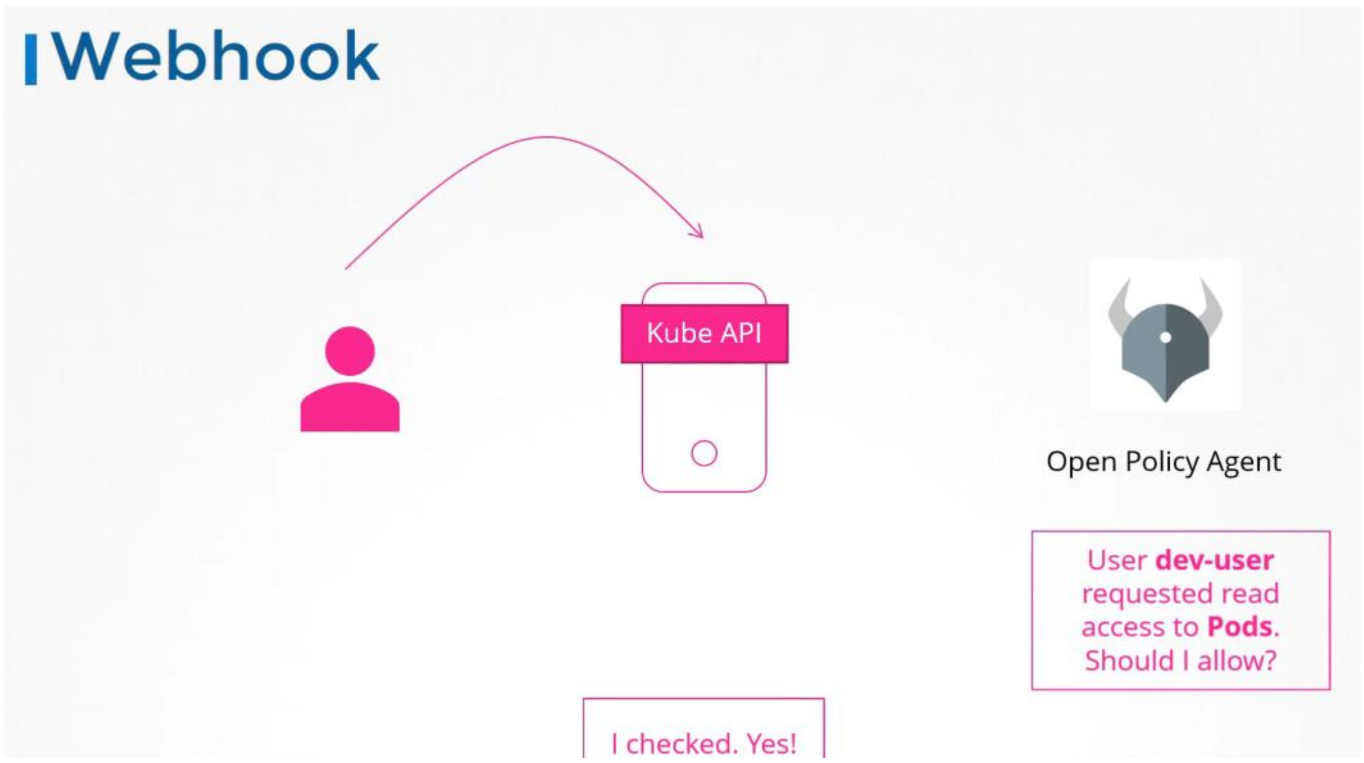
## RBAC

dev-user
dev-user-2
dev-users
security-1
✓ Can view PODs
✓ Can create PODs
✓ Can Delete PODs
Developer
✓ Can view CSR
✓ Can approve CSR
Security



## RBAC

dev-user
dev-user-2
dev-users
security-1
✓ Can view PODs
✓ Can create PODs
✓ Can Delete PODs
✓ Can Create ConfigMaps
Developer
✓ Can view CSR
✓ Can approve CSR
Security

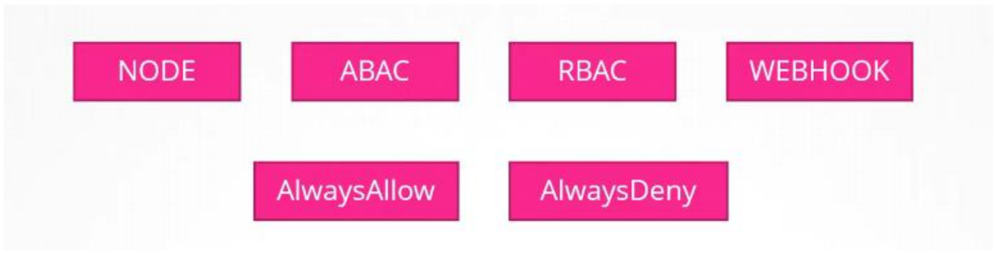


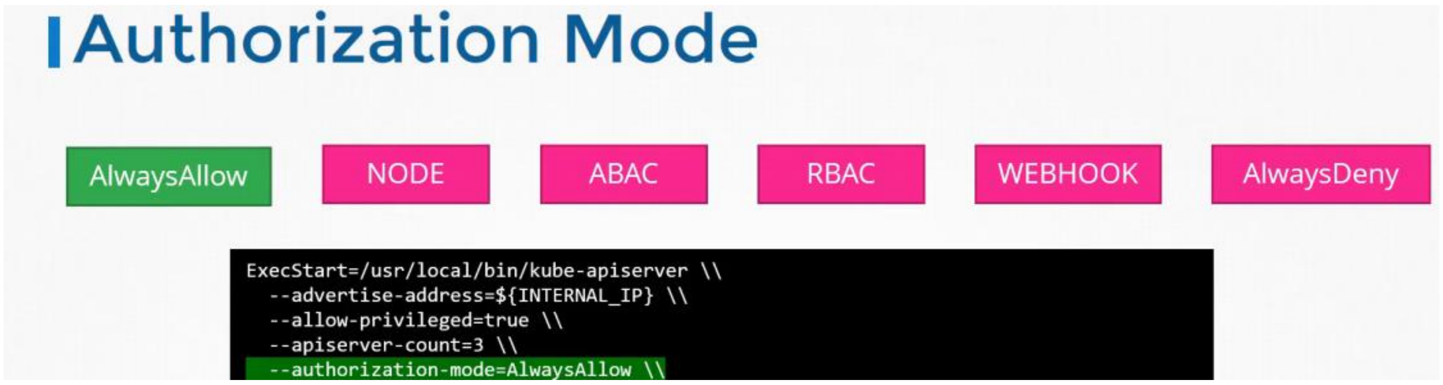
## Webhook

User **dev-user** requested read access to **Pods**. Should I allow?

I checked. Yes!

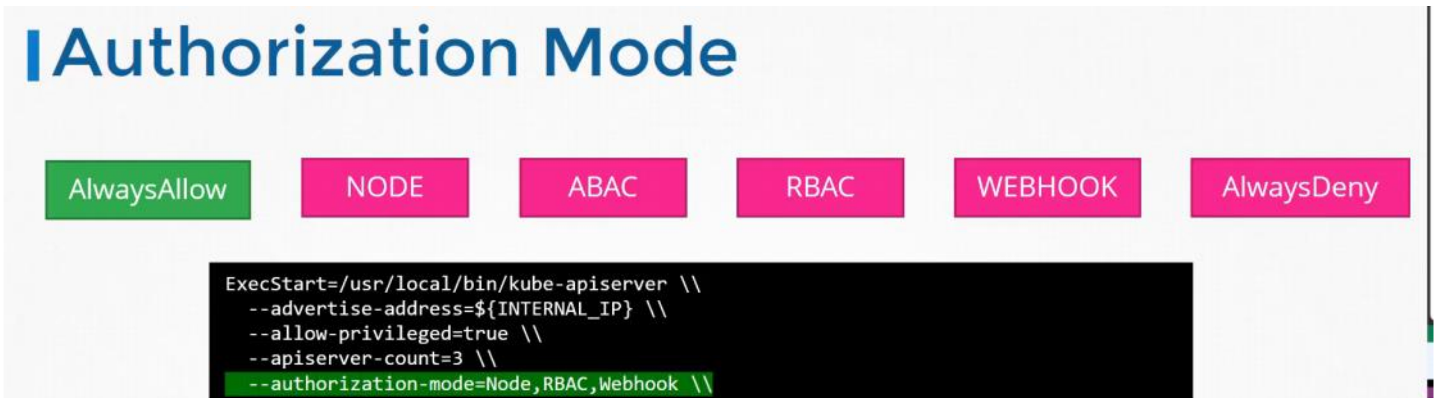
## Authorization Mode





## Authorization Mode

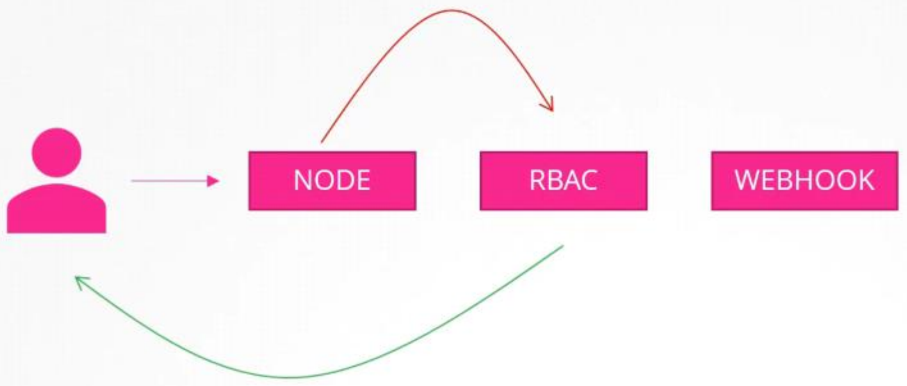
```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=AlwaysAllow \\
```



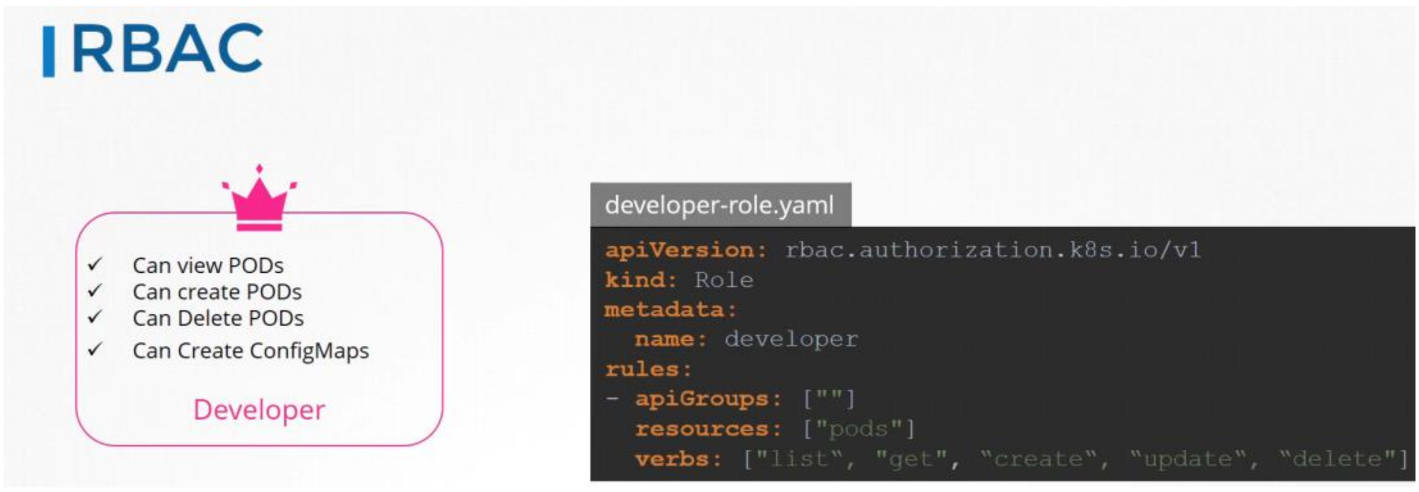
## Authorization Mode

```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=Node,RBAC,Webhook \\
```

## Authorization Mode



RBAC



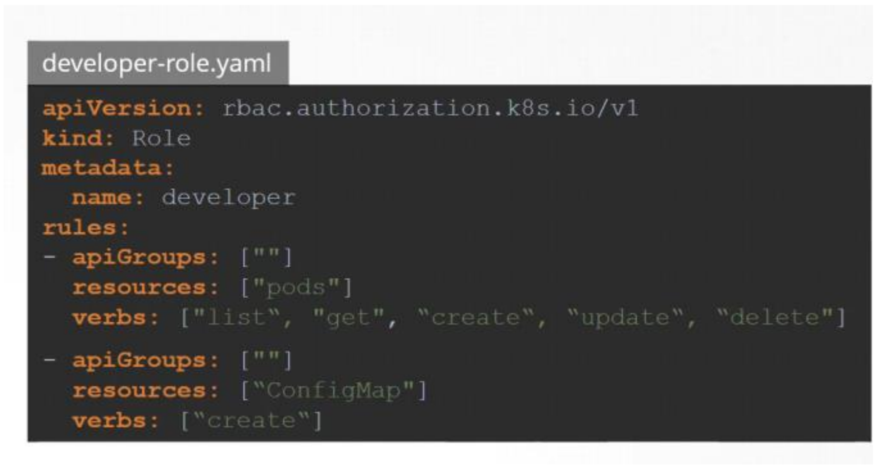
## RBAC

- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs
- ✓ Can Create ConfigMaps

Developer

developer-role.yaml

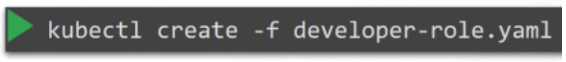
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "delete"]
```



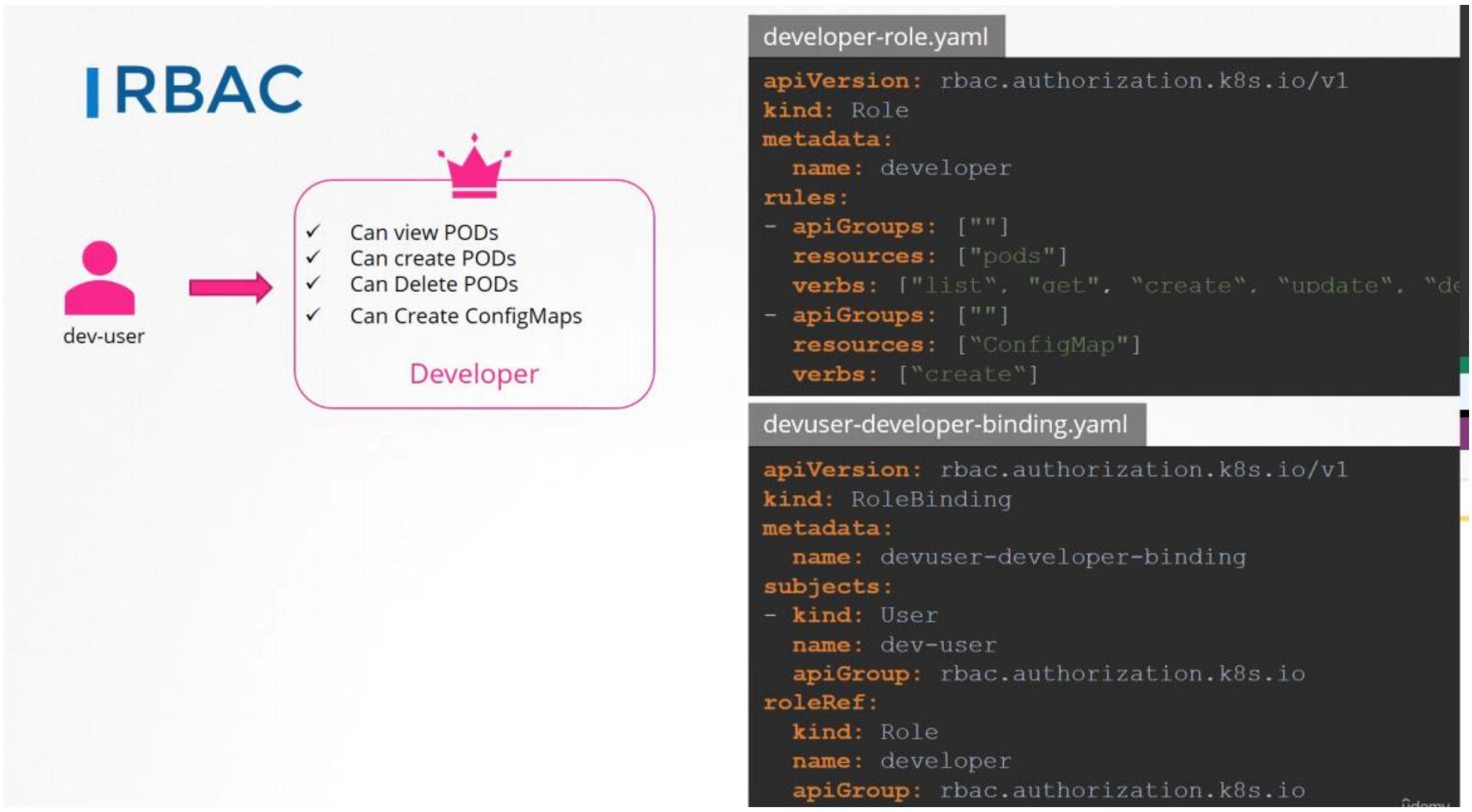
developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "delete"]

- apiGroups: [""]
  resources: ["ConfigMap"]
  verbs: ["create"]
```



```
kubectl create -f developer-role.yaml
```



## RBAC

dev-user

- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs
- ✓ Can Create ConfigMaps

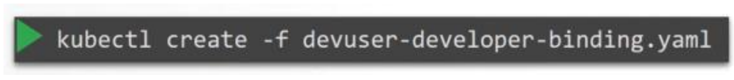
Developer

developer-role.yaml

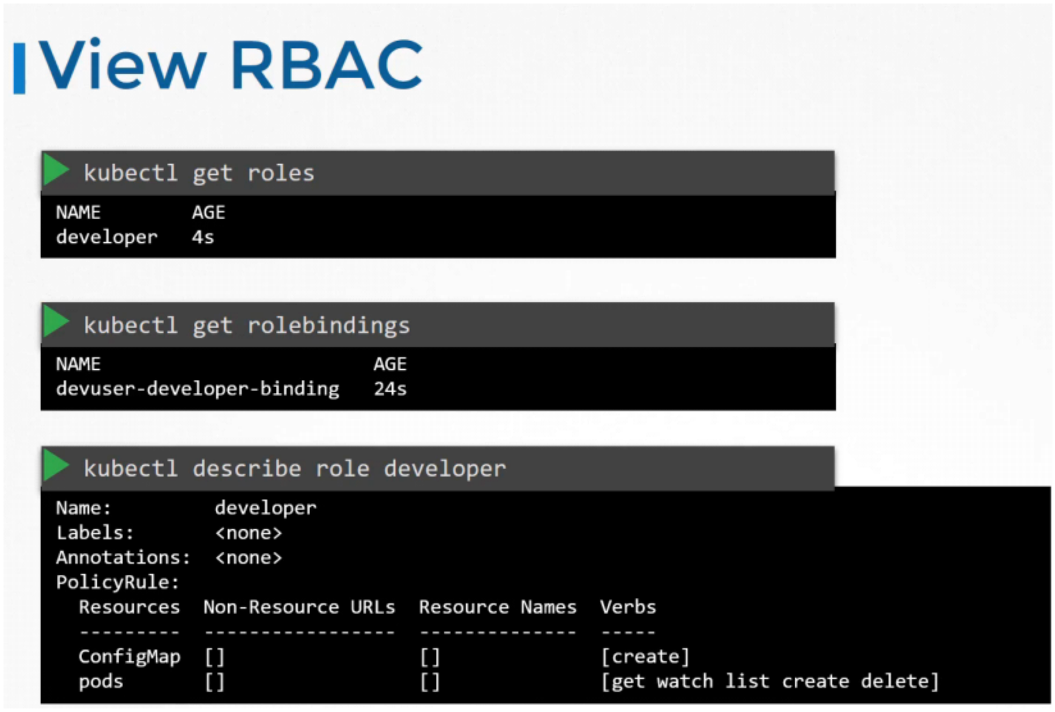
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "de
- apiGroups: [""]
  resources: ["ConfigMap"]
  verbs: ["create"]
```

devuser-developer-binding.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: devuser-developer-binding
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: developer
  apiGroup: rbac.authorization.k8s.io
```



```
kubectl create -f devuser-developer-binding.yaml
```

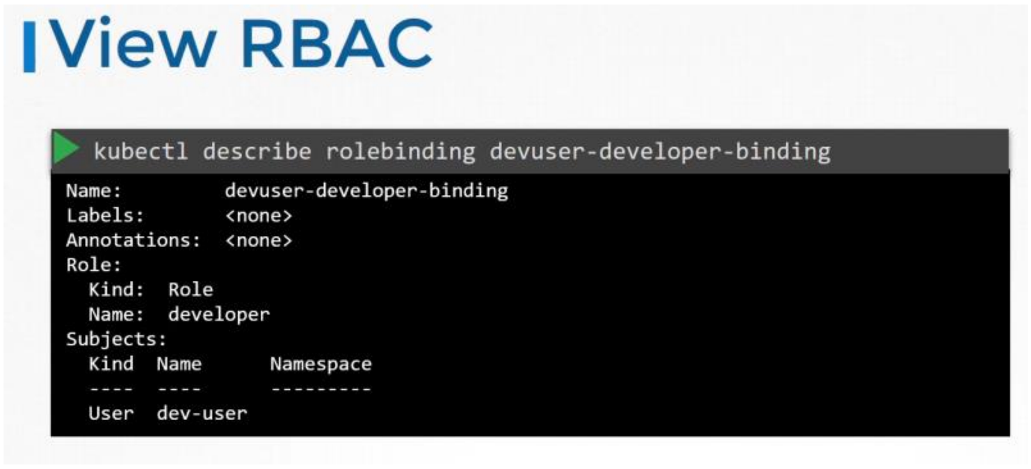


## View RBAC

```
kubectl get roles
NAME        AGE
developer   4s
```

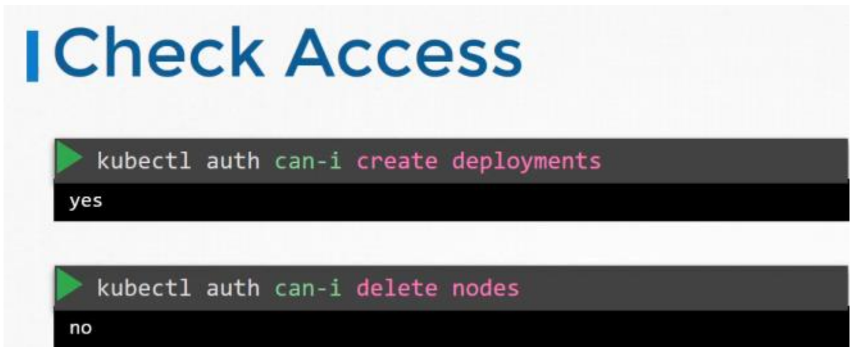
```
kubectl get rolebindings
NAME                        AGE
devuser-developer-binding   24s
```

```
kubectl describe role developer
Name:         developer
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  ConfigMap  []                 []              [create]
  pods       []                 []              [get watch list create delete]
```



## View RBAC

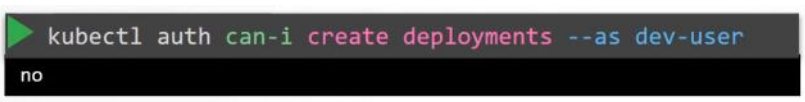
```
kubectl describe rolebinding devuser-developer-binding
Name:         devuser-developer-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  developer
Subjects:
  Kind  Name      Namespace
  ----  ----      ---------
  User  dev-user
```



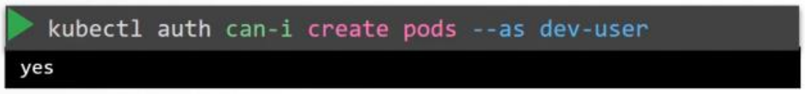
## Check Access

```
kubectl auth can-i create deployments
yes
```

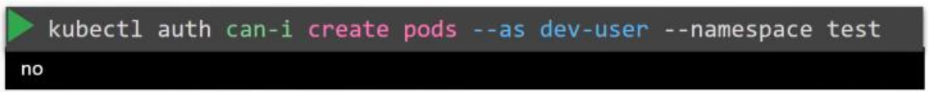
```
kubectl auth can-i delete nodes
no
```



```
kubectl auth can-i create deployments --as dev-user
no
```

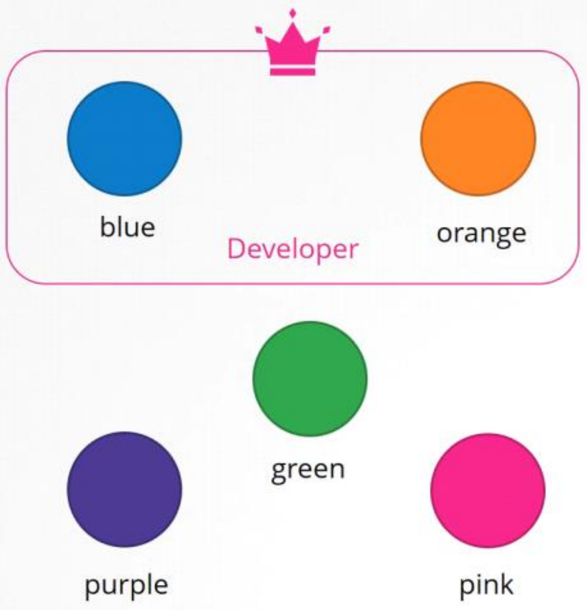


```
kubectl auth can-i create pods --as dev-user
yes
```



```
kubectl auth can-i create pods --as dev-user --namespace test
no
```

## Resource Names



developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "create", "update"]
  resourceNames: ["blue", "orange"]
```