



KubeCon

— North America 2017 —

Effective RBAC

Jordan Liggitt, *@liggitt, Red Hat*

RBAC Overview

Role-Based Access Control

“Can _____ ?”
subject verb object

RBAC Overview

Role-Based Access Control

“Can Bob educate dolphins?”
subject verb object

A Short Story



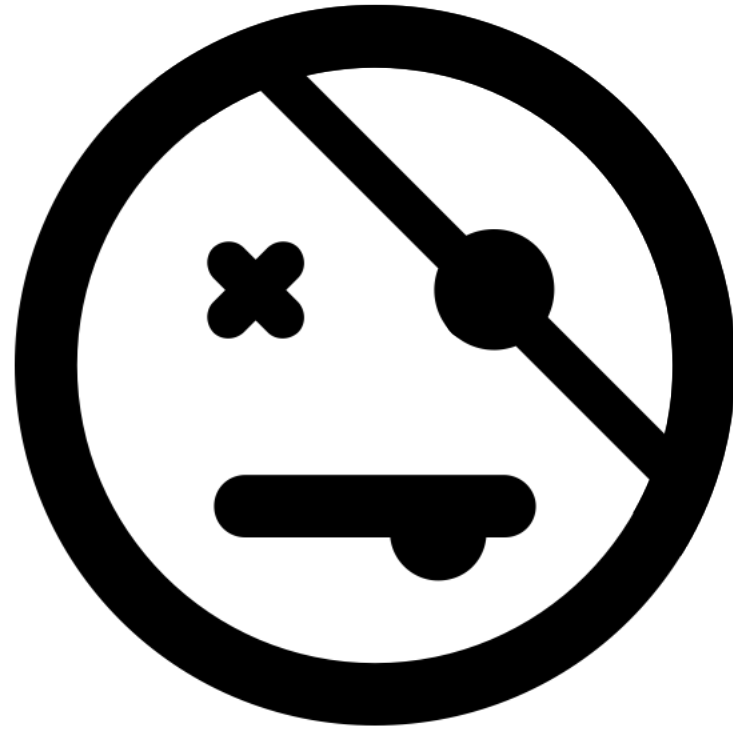
Bob

A Short Story



Bob

A Short Story



Bob

A Short Story



first mate of the **green ship**

help captain, train crew

A Short Story



role

first mate of the green ship

help captain, train crew

A Short Story



role

first mate of the green ship

permissions

help captain, train crew

A Short Story



role



first mate of the green ship

location



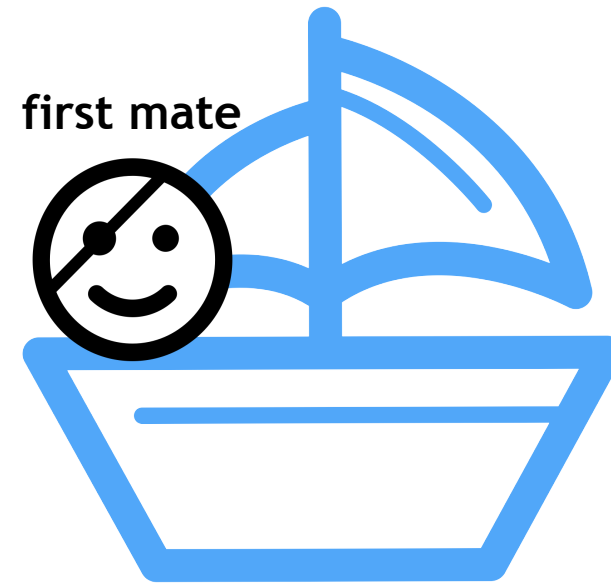
permissions



help captain, train crew

A Short Story

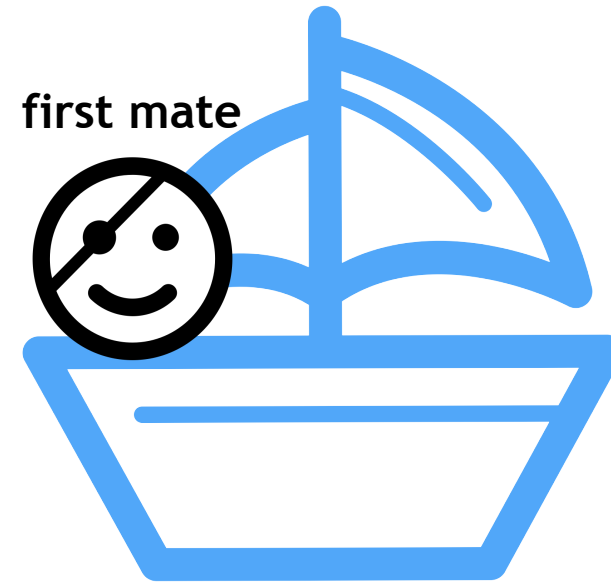
first mate: help captain, train crew



A Short Story

defined globally

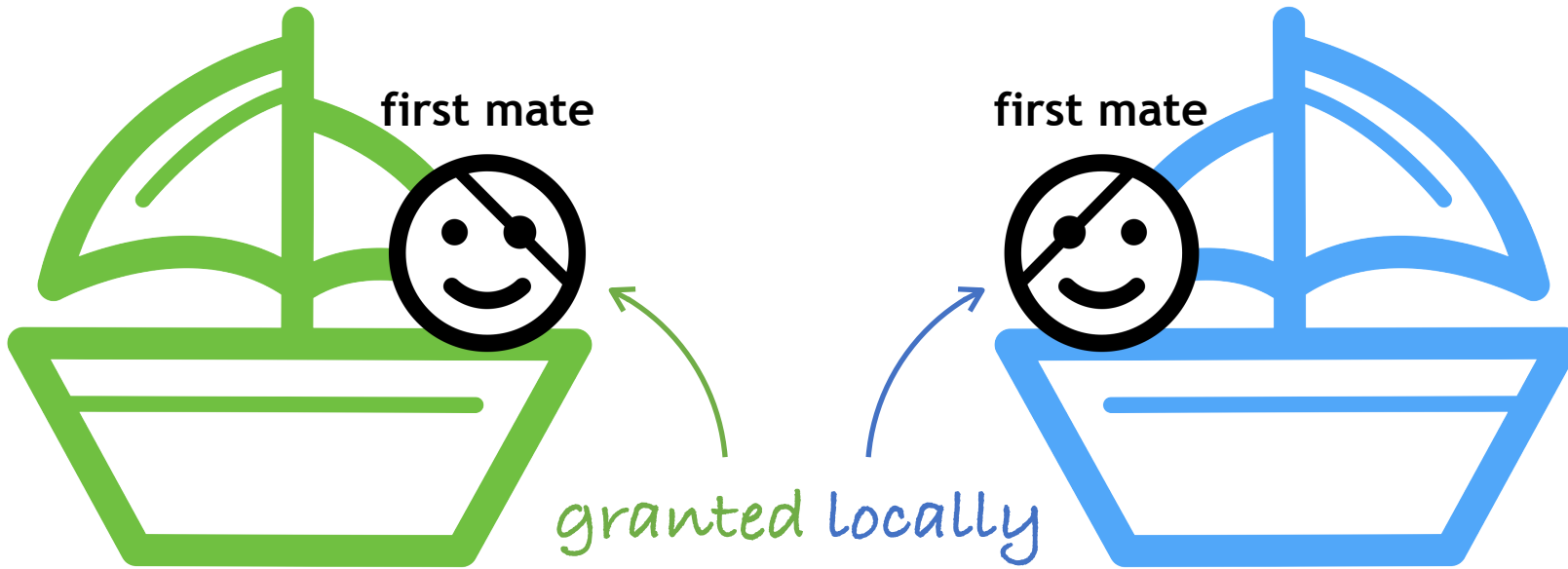
→ first mate: help captain, train crew



A Short Story

defined globally

→ first mate: help captain, train crew

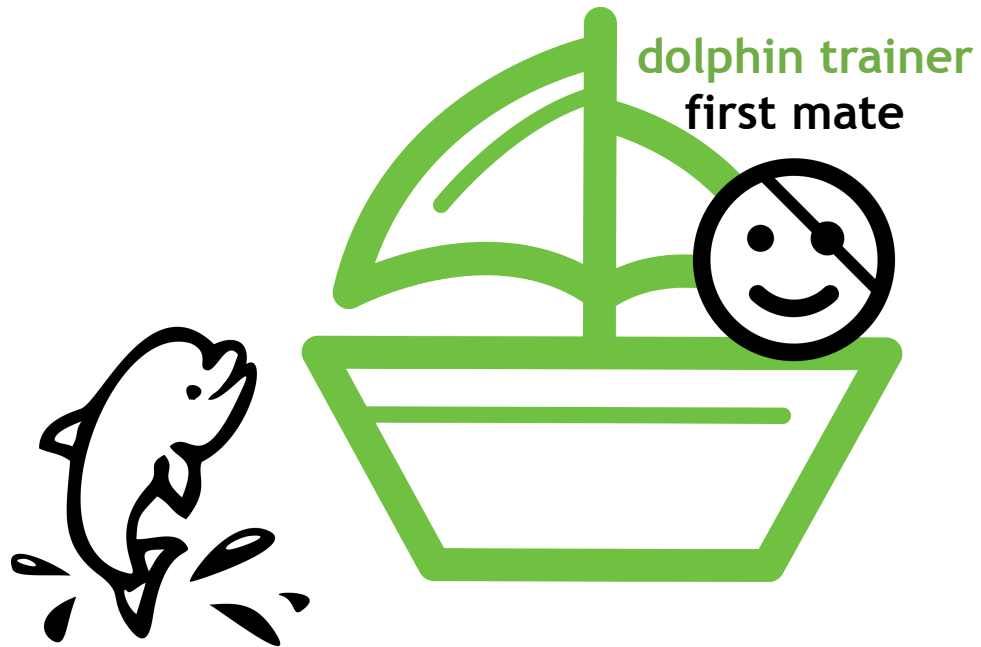


A Short Story



A Short Story

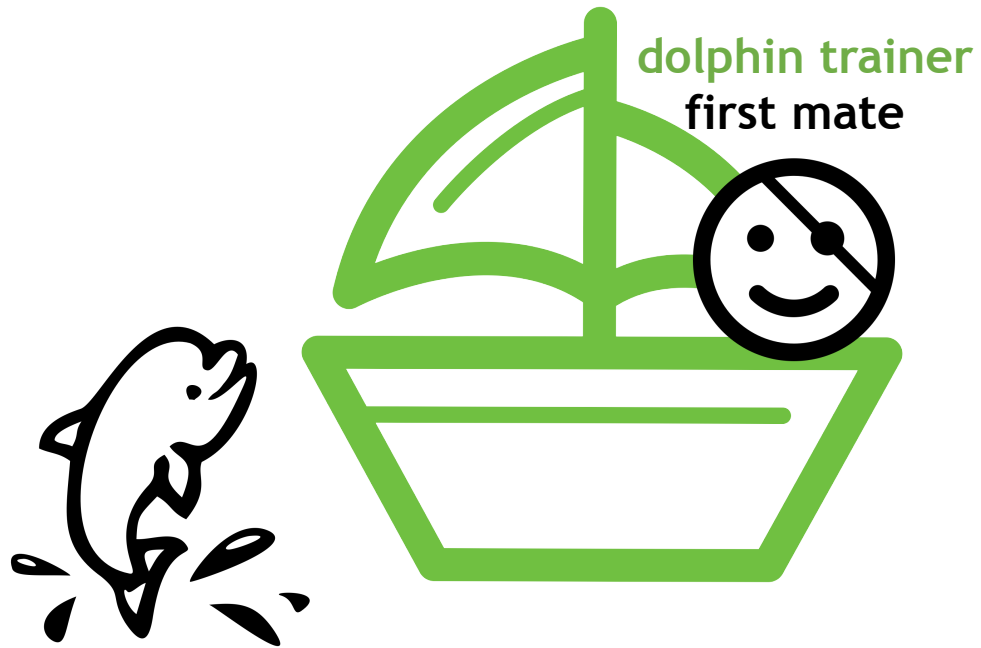
dolphin trainer: educate dolphins



A Short Story

defined locally

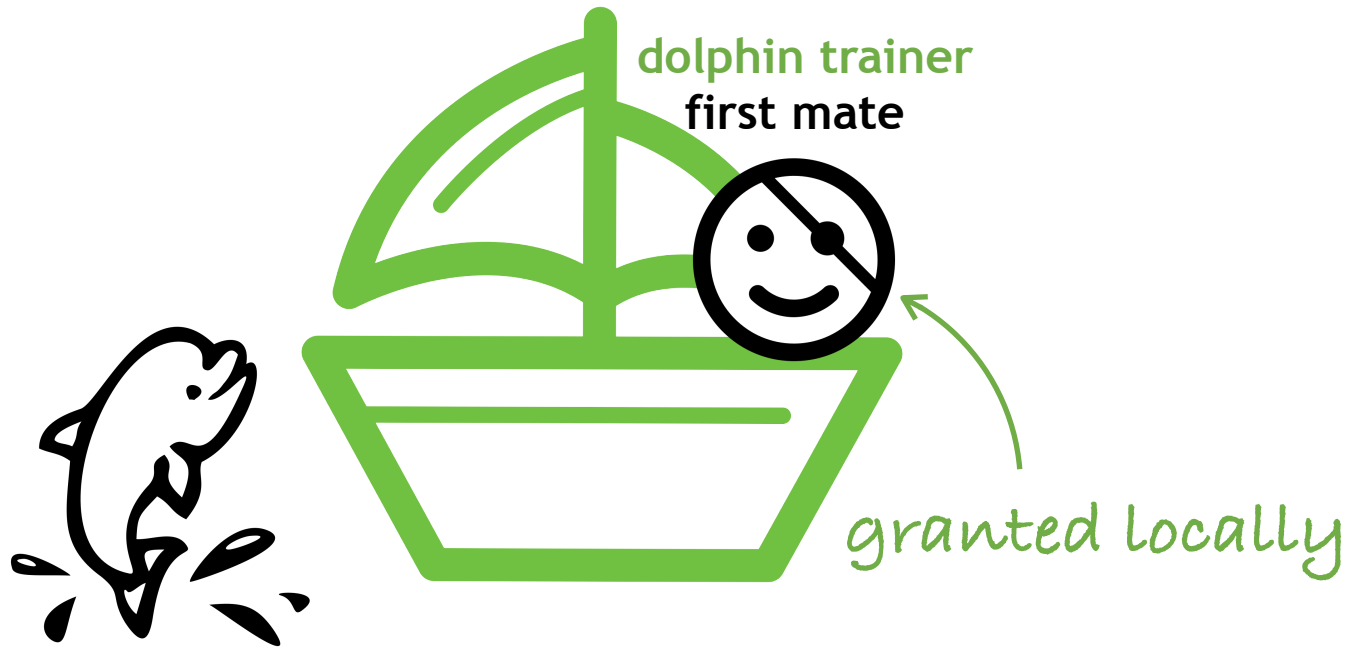
→ dolphin trainer: educate dolphins



A Short Story

defined locally

→ dolphin trainer: educate dolphins



A Short Story

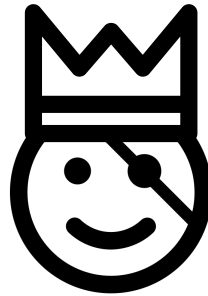
“Can Bob educate dolphins?”
subject verb object
 ^
 on the green ship

“Yes”

A Short Story

pirate king: command armada

pirate king



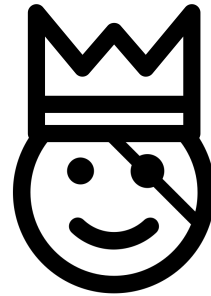
A Short Story

defined globally



pirate king: command armada

pirate king



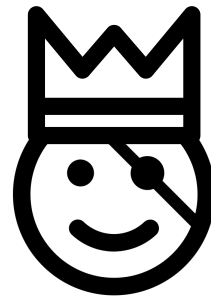
A Short Story

defined globally



pirate king: command armada

pirate king



granted globally



Request Handling

Request

```
POST /apis/apps/v1/namespaces/ns1/deployments
Authorization: Bearer eyJhbGciOiJSUzI1NiI...
Content-Type: application/json
Accept: application/json
```

```
{"apiVersion":"v1","kind":"Deployment",...
```

Request Handling



POST /apis/apps/v1/namespaces/ns1/deployments

Authorization: Bearer eyJhbGciOiJIUzI1NiI...

Content-Type: application/json

Accept: application/json

```
{"apiVersion": "v1", "kind": "Deployment", ...
```

Verb	create
API group	apps
Namespace	ns1
Resource	deployments

Request Handling



POST /apis/apps/v1/namespaces/ns1/deployments

Authorization: Bearer eyJhbGciOiJSUzI1NiI...

Content-Type: application/json

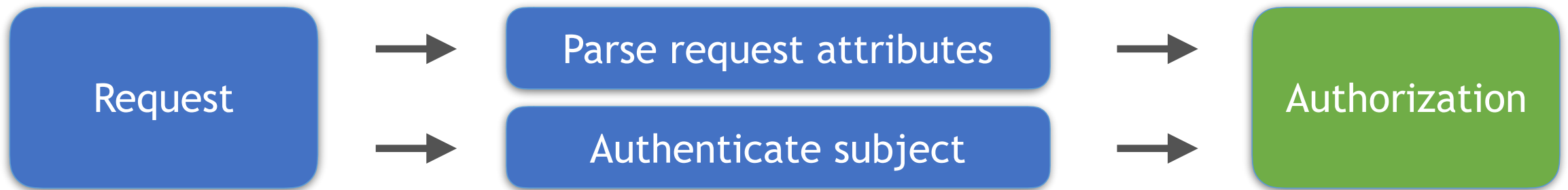
Accept: application/json

```
{"apiVersion": "v1", "kind": "Deployment", ...
```

Verb	create
API group	apps
Namespace	ns1
Resource	deployments

Username	bob
Groups	system:authenticated

Request Handling



Can **bob** in group **system:authenticated**
create
apps deployments in namespace **ns1**?

RBAC Overview



deployer in namespace ns1

create apps deployments

RBAC Overview



role

→ deployer in namespace ns1

create apps deployments

RBAC Overview



role



deployer in namespace ns1

permissions



create apps deployments

RBAC Overview



role



deployer in namespace ns1

location



permissions



create apps deployments

RBAC Overview

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
  namespace: ns1
```

RBAC Overview

kind: **Role**

apiVersion: **rbac.authorization.k8s.io/v1**

metadata:

name: **deployer**

namespace: **ns1**

rules:

– verbs: [**"create"**]

apiGroups: [**"apps"**]

resources: [**"deployments"**]

RBAC Overview

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
  namespace: ns1
```

```
rules:
- verbs: ["create"]
  apiGroups: ["apps"]
  resources: ["deployments"]
```

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
  namespace: ns1
```

RBAC Overview

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
  namespace: ns1

rules:
- verbs: ["create"]
  apiGroups: ["apps"]
  resources: ["deployments"]
```

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
  namespace: ns1

roleRef:
  kind: Role
  apiGroup: rbac.authorization.k8s.io
  name: deployer
```

RBAC Overview

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
  namespace: ns1

rules:
- verbs: ["create"]
  apiGroups: ["apps"]
  resources: ["deployments"]
```

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
  namespace: ns1

roleRef:
  kind: Role
  apiGroup: rbac.authorization.k8s.io
  name: deployer

subjects:
- kind: User
  apiGroup: rbac.authorization.k8s.io
  name: bob
```

RBAC Overview

defined locally

```
kind: Role ←
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
  namespace: ns1

rules:
- verbs: ["create"]
  apiGroups: ["apps"]
  resources: ["deployments"]
```

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
  namespace: ns1

roleRef:
  kind: Role
  apiGroup: rbac.authorization.k8s.io
  name: deployer

subjects:
- kind: User
  apiGroup: rbac.authorization.k8s.io
  name: bob
```

RBAC Overview

defined locally

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
  namespace: ns1

rules:
- verbs: ["create"]
  apiGroups: ["apps"]
  resources: ["deployments"]
```

granted locally

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
  namespace: ns1

roleRef:
  kind: Role
  apiGroup: rbac.authorization.k8s.io
  name: deployer

subjects:
- kind: User
  apiGroup: rbac.authorization.k8s.io
  name: bob
```

RBAC Overview

defined globally

```
kind: ClusterRole  
apiVersion: rbac.authorization.k8s.io/v1  
metadata:  
  name: deployer
```

```
rules:  
- verbs: ["create"]  
  apiGroups: ["apps"]  
  resources: ["deployments"]
```

granted locally

```
kind: RoleBinding  
apiVersion: rbac.authorization.k8s.io/v1  
metadata:  
  name: bob-deployer  
  namespace: ns1
```

```
roleRef:  
  kind: ClusterRole  
  apiGroup: rbac.authorization.k8s.io  
  name: deployer
```

```
subjects:  
- kind: User  
  apiGroup: rbac.authorization.k8s.io  
  name: bob
```

RBAC Overview

defined globally

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
```

```
rules:
- verbs: ["create"]
  apiGroups: ["apps"]
  resources: ["deployments"]
```

granted locally

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
  namespace: ns1
```

```
roleRef:
  kind: ClusterRole
  apiGroup: rbac.authorization.k8s.io
  name: deployer
```

```
subjects:
- kind: User
  apiGroup: rbac.authorization.k8s.io
  name: bob
```

RBAC Overview

defined globally

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
```

rules:

- verbs: ["create"]
- apiGroups: ["apps"]
- resources: ["deployments"]

granted globally

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
```

roleRef:

```
kind: ClusterRole
apiGroup: rbac.authorization.k8s.io
name: deployer
```

subjects:

- kind: User
- apiGroup: rbac.authorization.k8s.io
- name: bob

RBAC Overview

defined globally

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
```

rules:

- verbs: ["create"]
- apiGroups: ["apps"]
- resources: ["deployments"]

granted globally

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: bob-deployer
```

roleRef:

```
kind: ClusterRole
apiGroup: rbac.authorization.k8s.io
name: deployer
```

subjects:

- kind: User
- apiGroup: rbac.authorization.k8s.io
- name: bob

RBAC Overview

Define permissions in a ClusterRole object...

- ... if the resources are cluster-scoped
- ... if you want to reference the role from multiple namespaces
- ... if you want to give cluster-wide access

```
kubectl get pods --all-namespaces
```

RBAC Overview

Define permissions in a Role object...

- ... if the resources are namespaced and you only want to reference the role from one namespace

RBAC Overview

Grant a ClusterRole with a ClusterRoleBinding object...

- ... if the resources are cluster-scoped
- ... if you want to give cluster-wide access
`kubectl get pods --all-namespaces`

RBAC Overview

- Grant a ClusterRole or Role with a RoleBinding object...
- ... if the resources are namespaced and you want to limit access to a particular namespace

Cluster Setup

1 step process:

Use a distribution or installer that sets up RBAC for you

Cluster Setup

```
kube-apiserver --authorization-mode=RBAC
```

Cluster Setup

- Default roles are auto-created
- Default role bindings to `system:...` subjects are auto-created

<https://kubernetes.io/docs/admin/authorization/rbac/#default-roles-and-role-bindings>

Cluster Setup

Bootstrap superuser

- Set up a credential with the `system:masters` group
- Auto-bound to the cluster-admin superuser ClusterRole
- Use for setup, delegation, “break glass in case of emergency”

Cluster Setup

Control plane components

- kube-scheduler
 - run with a credential for `system:kube-scheduler`
- kube-controller-manager
 - run with a credential for `system:kube-controller-manager`
 - run with `--use-service-account-credentials` for control loops
- kube-proxy
 - run with a credential for `system:kube-proxy`

Cluster Setup

Kubelets

- Enable Node authorization mode and NodeRestriction admission plugin

```
kube-apiserver --authorization-mode=Node,RBAC \  
  --admission-control=...,NodeRestriction,...
```

- Run with a unique credential per node
 - username "system:node:<nodeName>"
 - group "system:nodes"
 - Node TLS bootstrapping sets up well-formed credentials

Cluster Setup

Add-ons

- Many already include RBAC role definitions
- For those that don't, grant roles to their service accounts

Applying Policies

General purpose default ClusterRoles:

- cluster-admin: superuser
- admin, edit, view: namespaced user roles

<https://kubernetes.io/docs/admin/authorization/rbac/#user-facing-roles>

Applying Policies

Best: Grant a role to an application-specific service account

```
kubectl create rolebinding my-service-account-binding \
  --clusterrole=view \
  --serviceaccount=my-namespace:my-service-account \
  --namespace=my-namespace
```

Applying Policies

OK: Grant a role to the “default” service account in a namespace

```
kubectl create rolebinding default-service-account-binding \  
  --clusterrole=view \  
  --serviceaccount=my-namespace:default \  
  --namespace=my-namespace
```

Applying Policies

OK: Grant a role to all service accounts in a namespace

```
kubectl create rolebinding all-service-accounts-binding \  
  --clusterrole=view \  
  --group=system:serviceaccounts:my-namespace \  
  --namespace=my-namespace
```


Applying Policies

Less than ideal: run as superuser

```
kubectl create clusterrolebinding my-superuser-binding \  
  --clusterrole=cluster-admin \  
  --serviceaccount=my-namespace:my-service-account
```

Building Custom Roles

Option 1:

- Know every API call an app makes
- Enjoy hand-editing RBAC YAML

Building Custom Roles

Option 2:

1. Enable audit logs

- <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>

2. Run application with a dedicated service account

3. Capture audit logs for that service account

4. Generate a role (or set of roles) that allow the requests

Building Custom Roles

[Demo](#)

\$

```
},
"sourceIPs": [
  "::1"
],
"responseStatus": {
  "metadata": {},
  "code": 304
}
}
}
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1beta1",
  "metadata": {
    "creationTimestamp": "2017-12-08T07:29:25Z"
  },
  "level": "Metadata",
  "timestamp": "2017-12-08T07:29:25Z",
  "auditID": "2b8adcc6-ead7-4f75-abe9-106150045e80",
  "stage": "ResponseComplete",
  "requestURI": "/apis/storage.k8s.io/v1beta1/storageclasses",
  "verb": "create",
  "user": {
    "username": "system:admin",
    "groups": [
      "system:masters",
      "system:authenticated"
    ]
  },
  "sourceIPs": [
    "::1"
  ],
  "objectRef": {
    "resource": "storageclasses",
    "name": "standard",
    "apiGroup": "storage.k8s.io",
    "apiVersion": "v1beta1"
  },
  "responseStatus": {
    "metadata": {},
    "code": 201
  }
}
}
```

\$

```
},
"sourceIPs": [
  "::1"
],
"responseStatus": {
  "metadata": {},
  "code": 304
}
}
}
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1beta1",
  "metadata": {
    "creationTimestamp": "2017-12-08T07:29:25Z"
  },
  "level": "Metadata",
  "timestamp": "2017-12-08T07:29:25Z",
  "auditID": "2b8adcc6-ead7-4f75-abe9-106150045e80",
  "stage": "ResponseComplete",
  "requestURI": "/apis/storage.k8s.io/v1beta1/storageclasses",
  "verb": "create",
  "user": {
    "username": "system:admin",
    "groups": [
      "system:masters",
      "system:authenticated"
    ]
  },
  "sourceIPs": [
    "::1"
  ],
  "objectRef": {
    "resource": "storageclasses",
    "name": "standard",
    "apiGroup": "storage.k8s.io",
    "apiVersion": "v1beta1"
  },
  "responseStatus": {
    "metadata": {},
    "code": 201
  }
}
}
```

Building Custom Roles

[Demo](#)

Building Custom Roles

audit2rbac - <https://github.com/liggitt/audit2rbac>

- verb expansion
 - list → get+list+watch, update → patch+update
- multi-name inference
 - multiple names → any name
- multi-namespace inference
 - multiple namespaces → any namespace

Building Custom Roles

audit2rbac - <https://github.com/liggitt/audit2rbac>

- Workflows
 - Deny → Audit → Apply → Allow
 - Allow all in CI/dev → Audit → Apply in production
 - ...

Aggregated Roles

- Aggregated roles (new in 1.9)
- Easily contribute to default admin/edit/view cluster roles
- Label your ClusterRoles:
 - `rbac.authorization.k8s.io/aggregate-to-admin="true"`
 - `rbac.authorization.k8s.io/aggregate-to-edit="true"`
 - `rbac.authorization.k8s.io/aggregate-to-view="true"`

Aggregated Roles

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: myco.acme.com:catset-admin
  labels:
    rbac.authorization.k8s.io/aggregate-to-admin="true"
    rbac.authorization.k8s.io/aggregate-to-edit="true"

rules:
- verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  apiGroups: ["myco.acme.com"]
  resources: ["catsets"]
```



KubeCon

— North America 2017 —

Effective RBAC

Jordan Liggitt, *@liggitt, Red Hat*

<http://bit.ly/effective-rbac>