KubeCon

North America 2017

Effective RBAC

Jordan 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

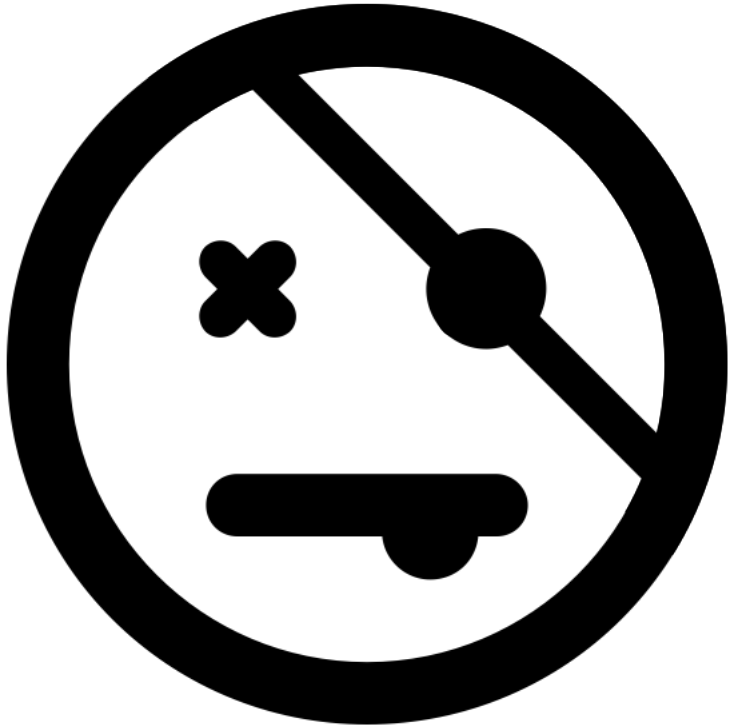
A Short Story

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

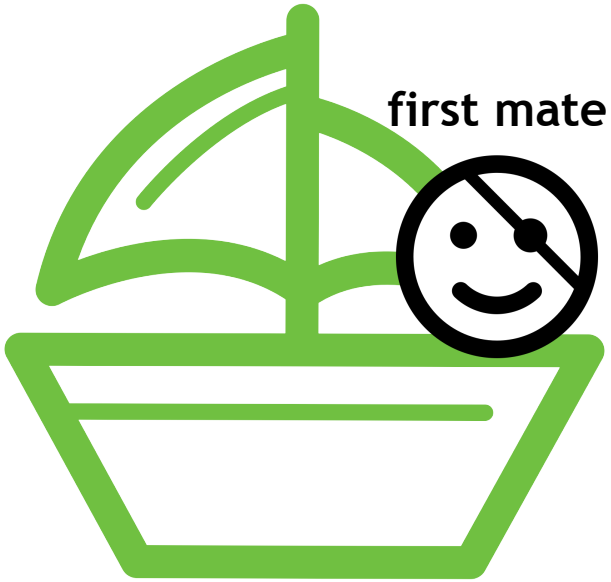
A Short Story

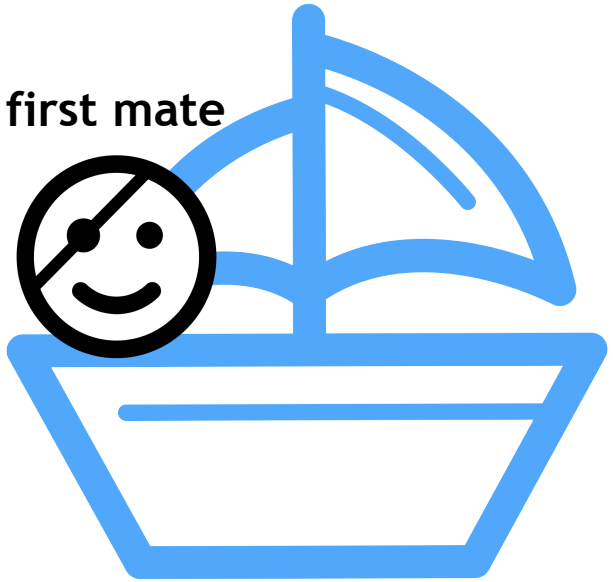
role → first mate of the green ship ← location

permissions → help captain, train crew

A Short Story

first mate: help captain, train crew

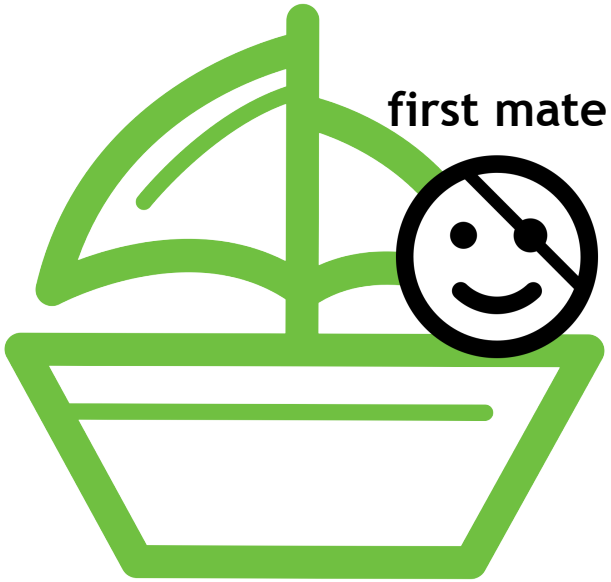


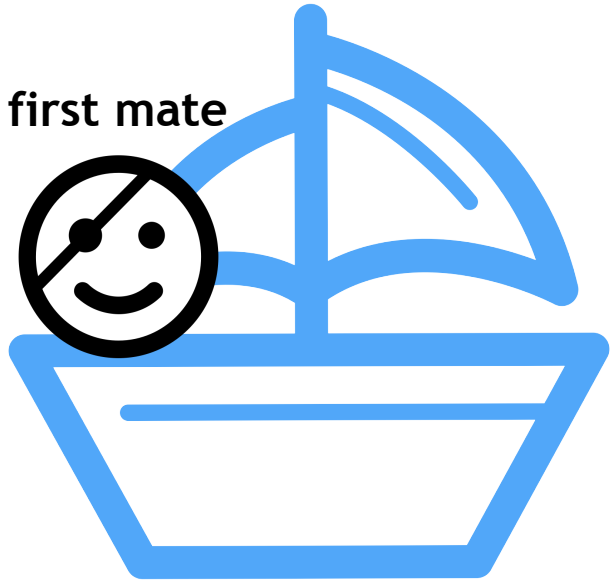


A Short Story

defined globally

first mate: help captain, train crew

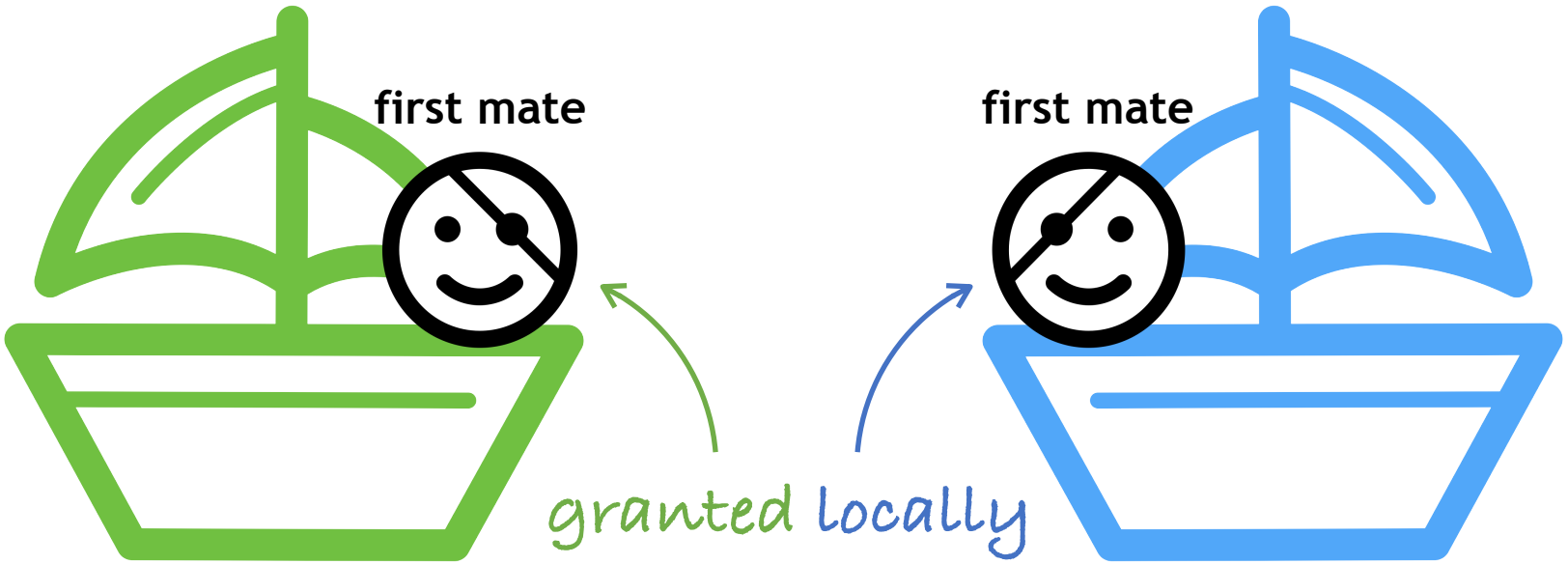




A Short Story

defined globally

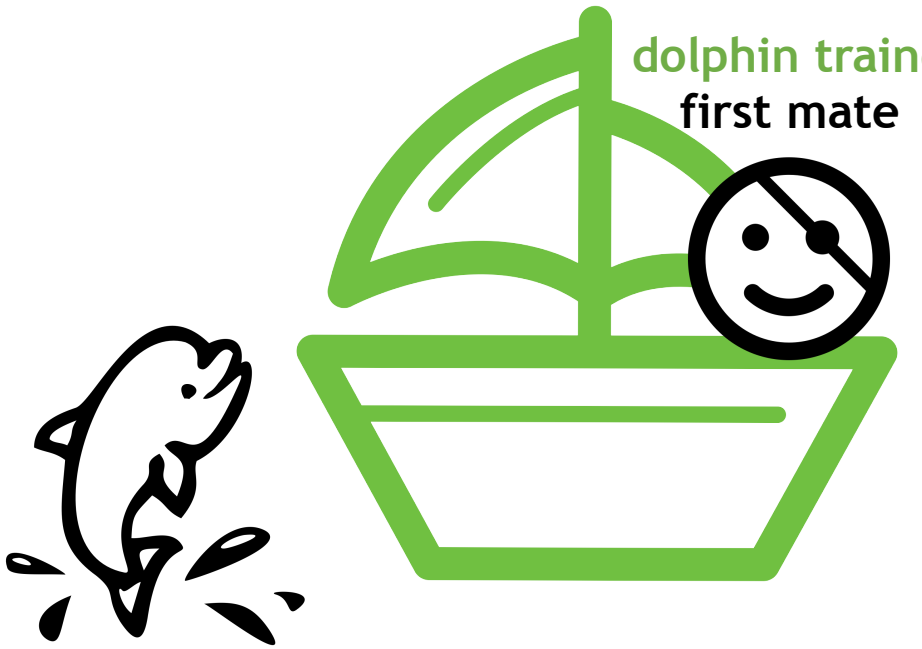
first mate: help captain, train crew



A Short Story

A Short Story

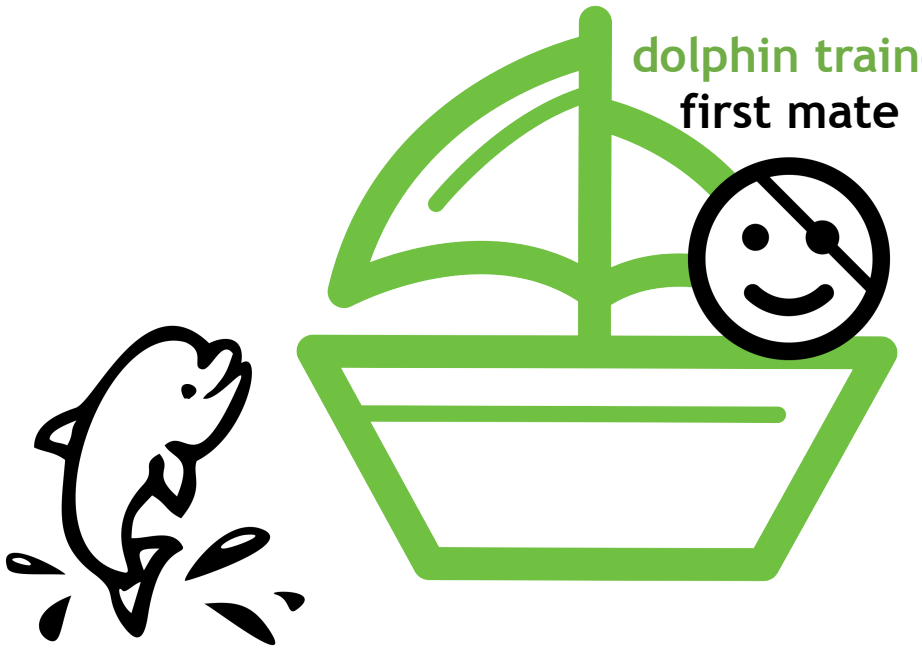
dolphin trainer: educate dolphins



A Short Story

defined locally

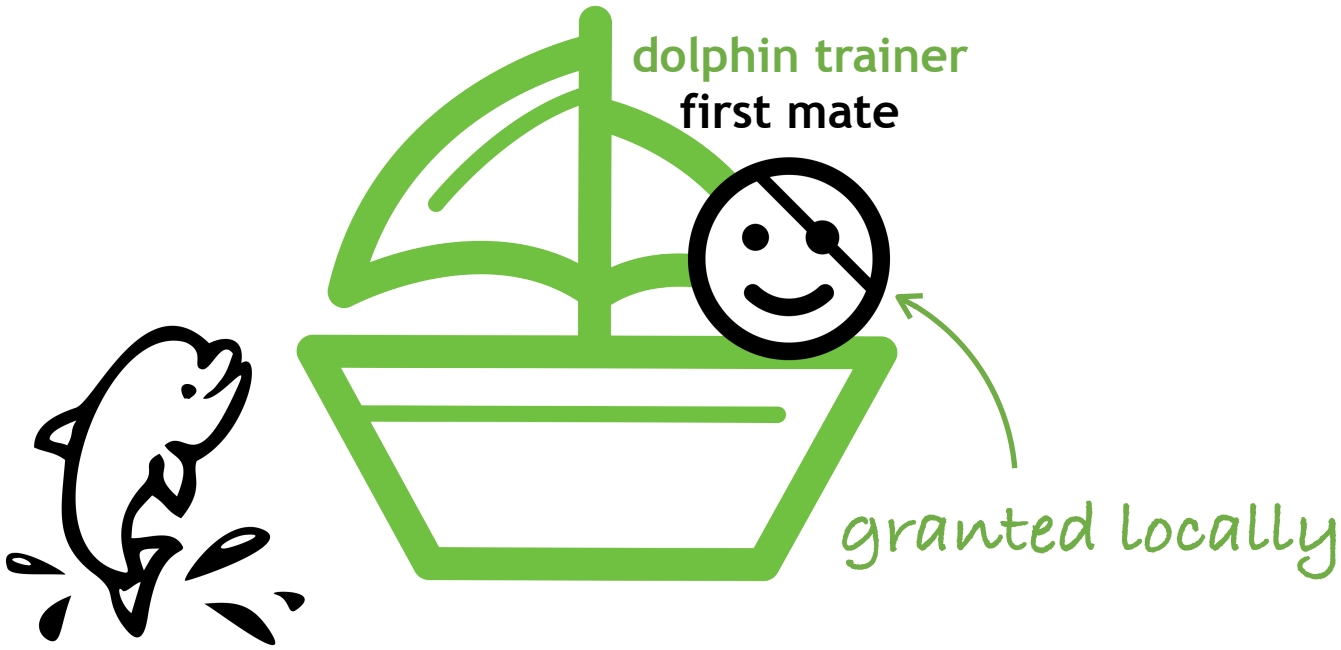
dolphin trainer: educate dolphins



A Short Story

defined locally

dolphin trainer: educate dolphins



A Short Story

“Can Bob educate dolphins ?”

subject verb object

on the green ship

A Short Story

"Can Bob educate dolphins ?"

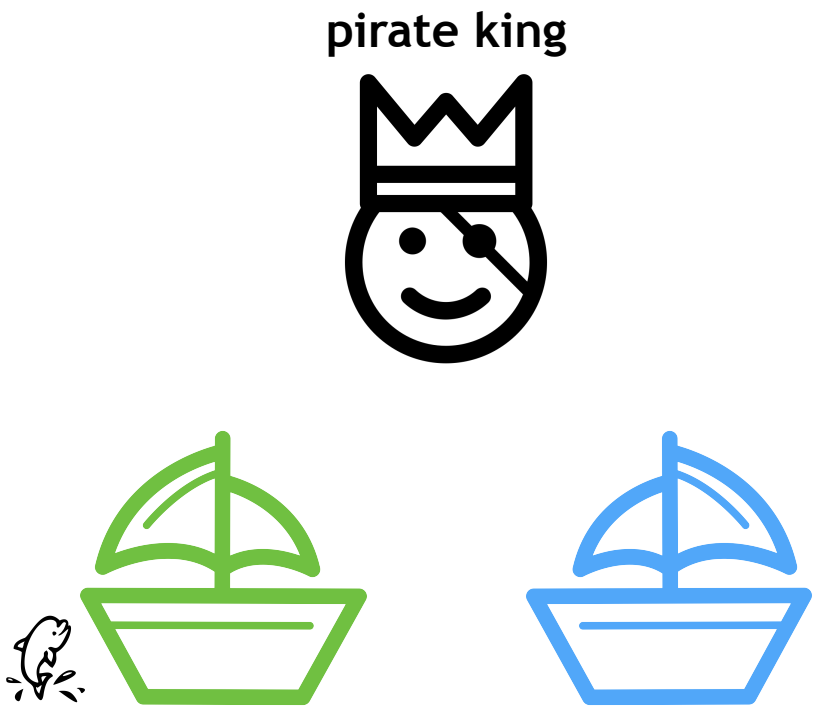
subject verb object

on the green ship

"Yes"

A Short Story

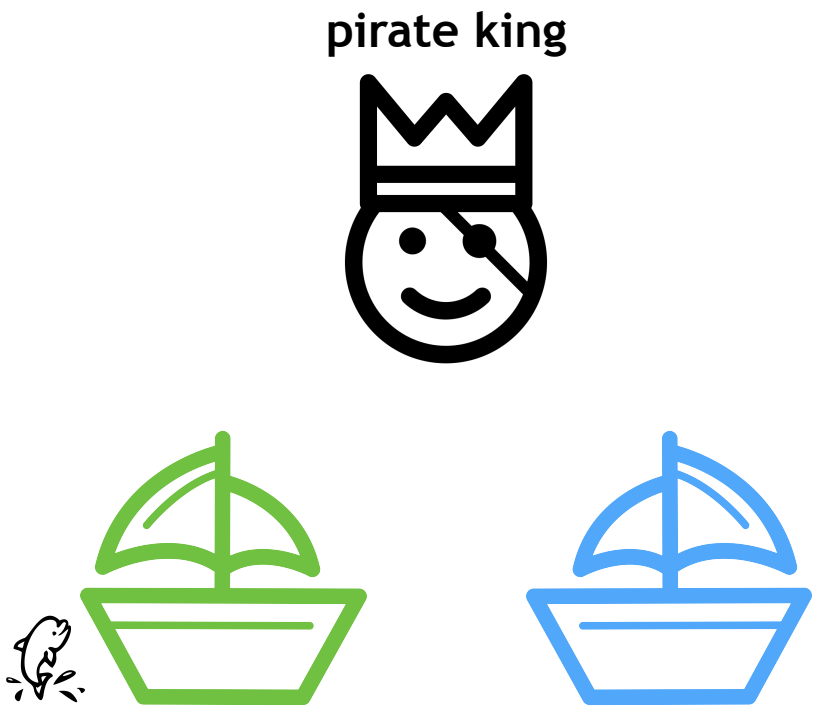
pirate king: command armada



A Short Story

defined globally

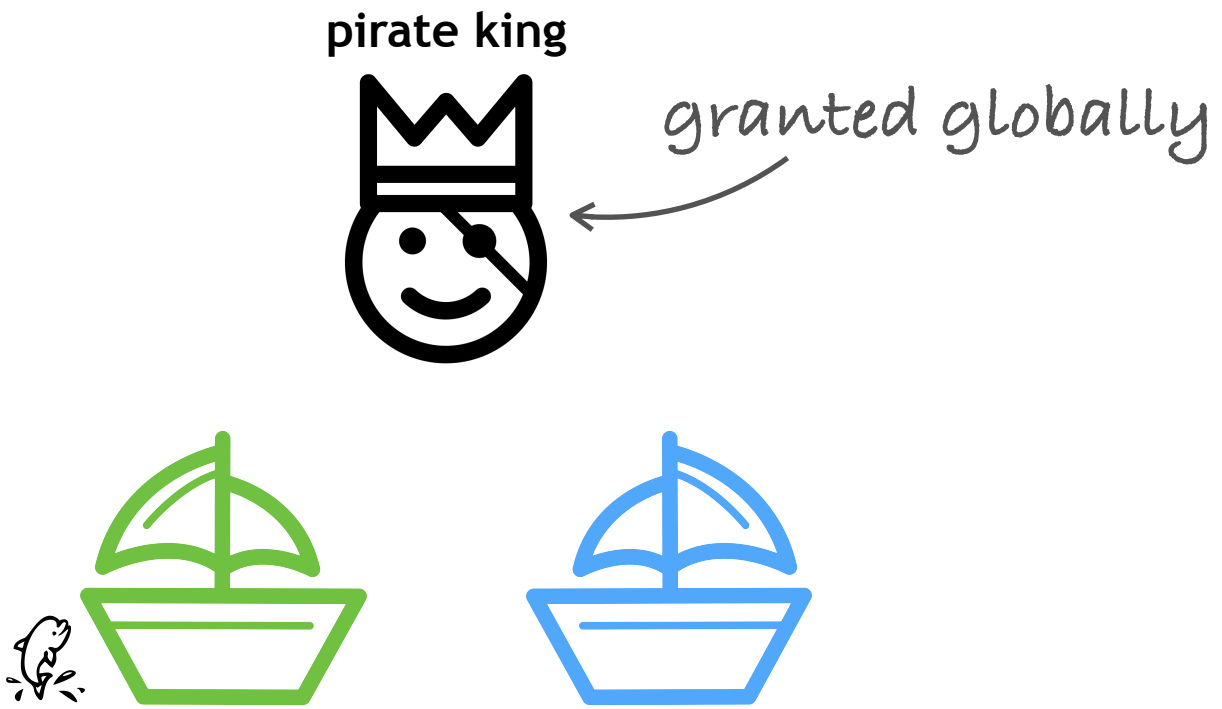
pirate king: command armada



A Short Story

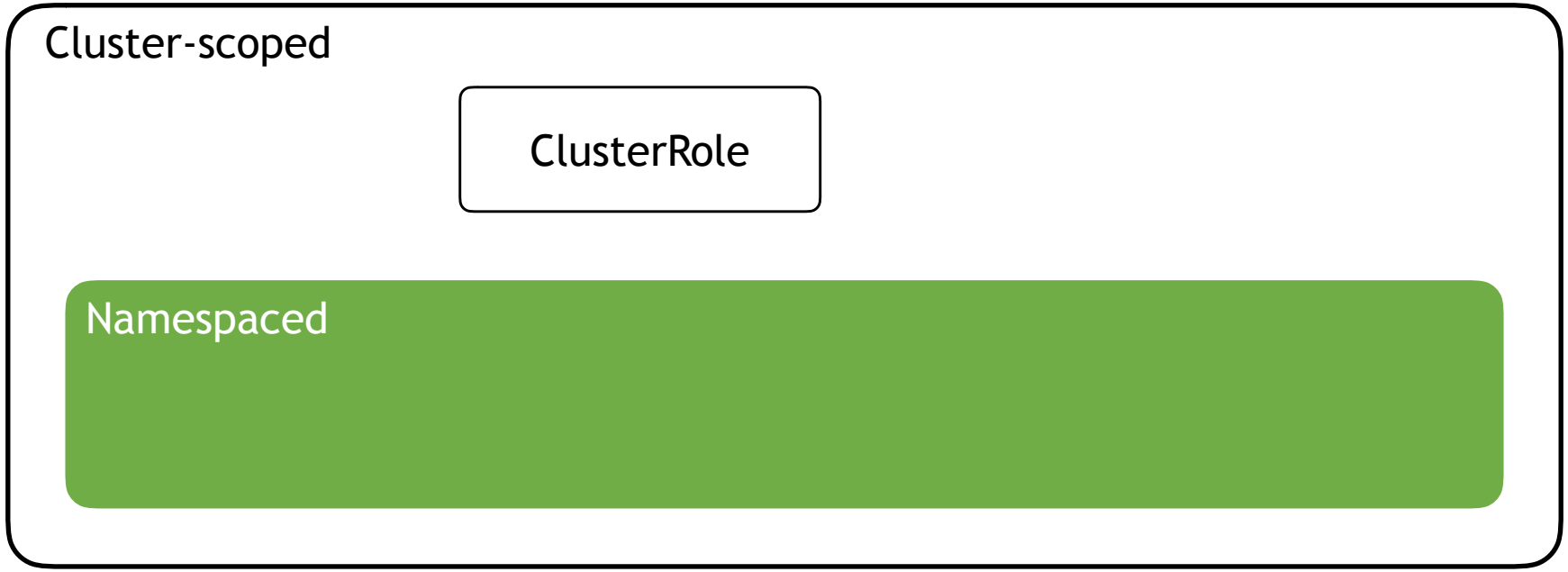
defined globally

pirate king: command armada

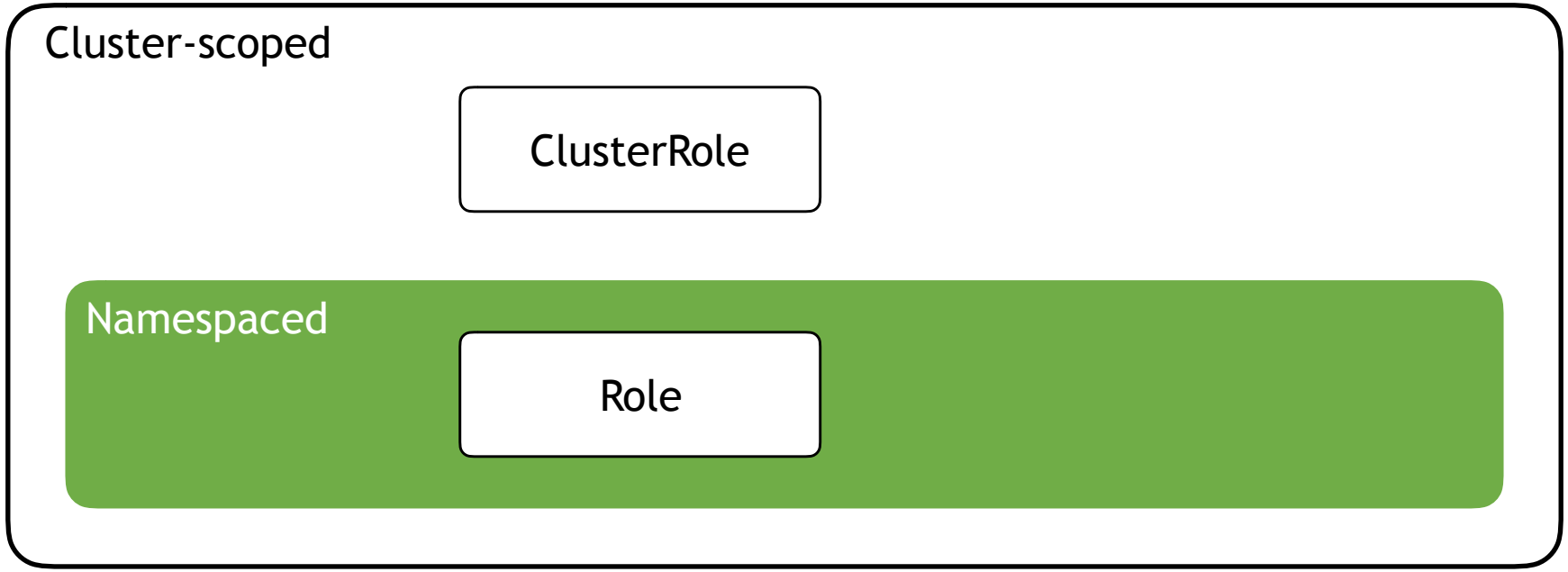


RBAC Overview

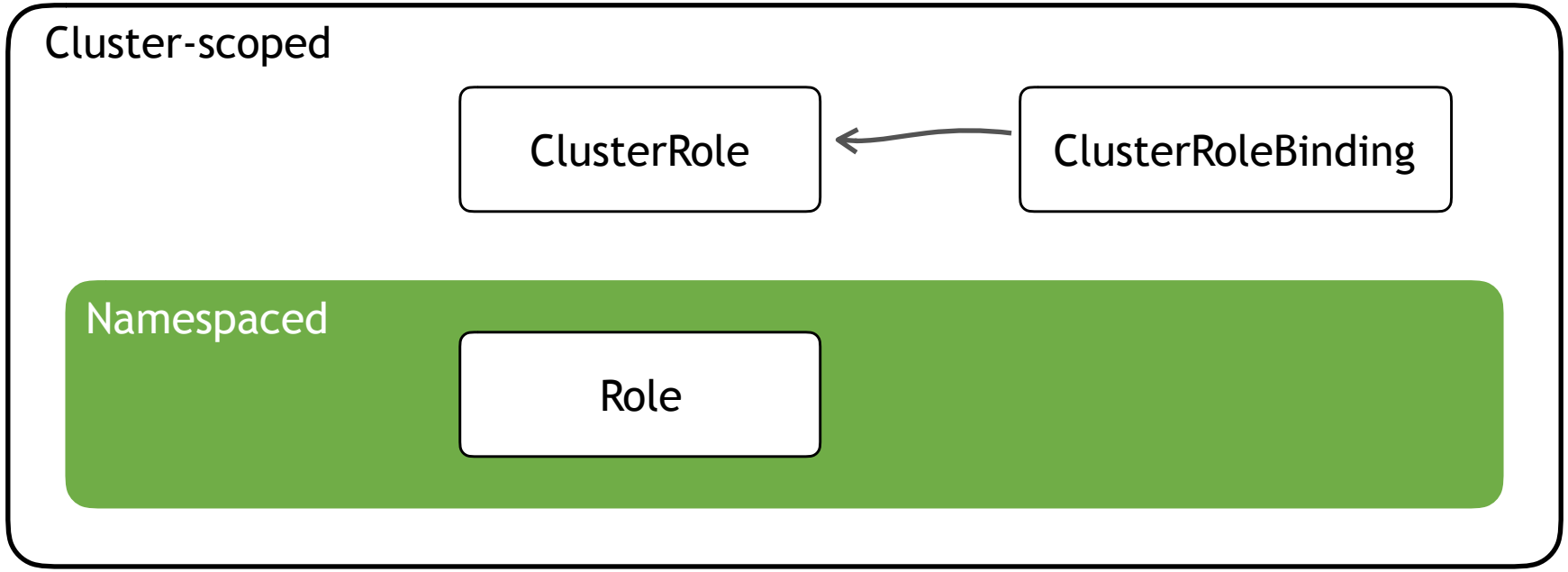
RBAC Overview



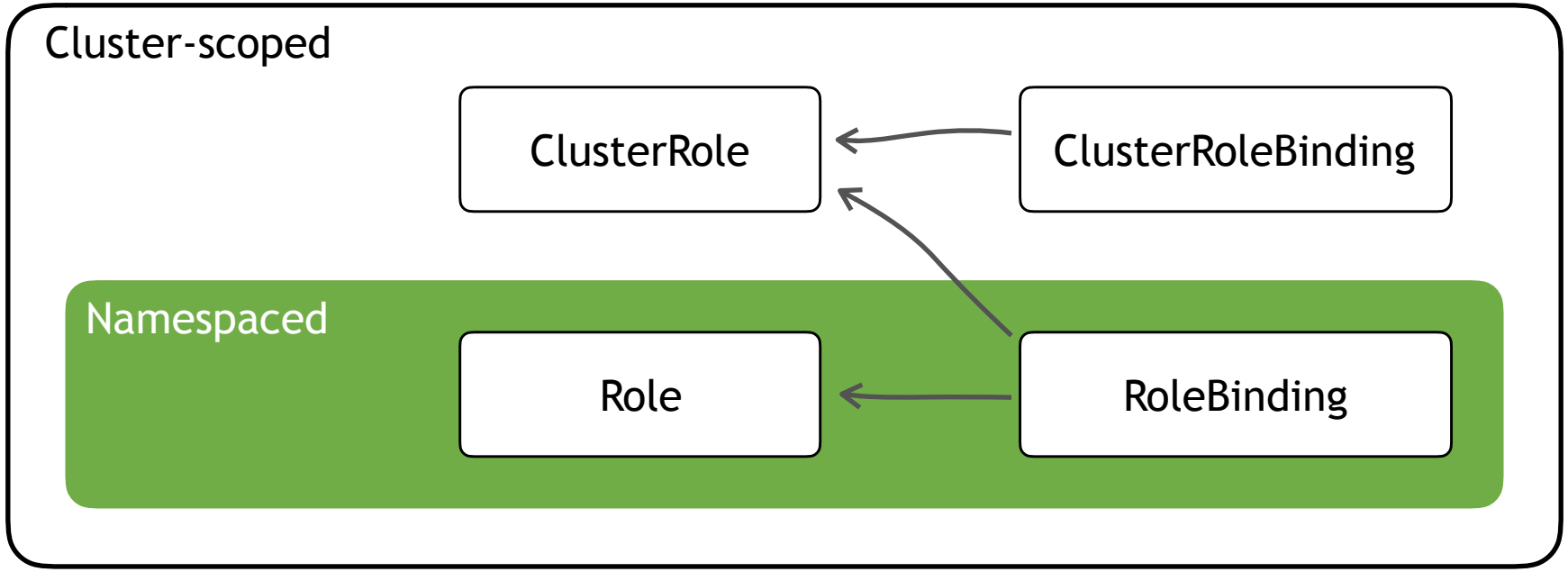
RBAC Overview



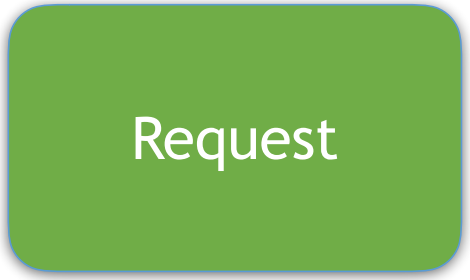
RBAC Overview



RBAC Overview



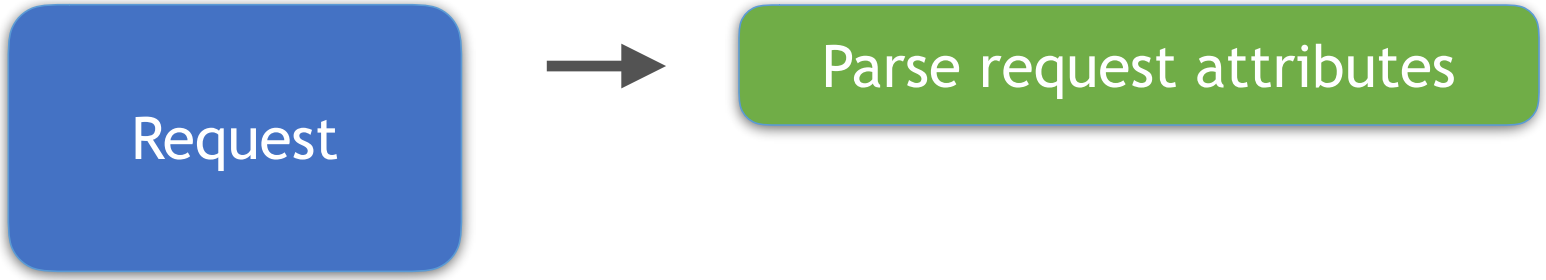
Request Handling




```
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 eyJhbGciOiJSUzI1NiI…
Content-Type: application/json
Accept: application/json

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

Verb	create
API group	apps
API version	v1
Namespace	ns1
Resource	deployments

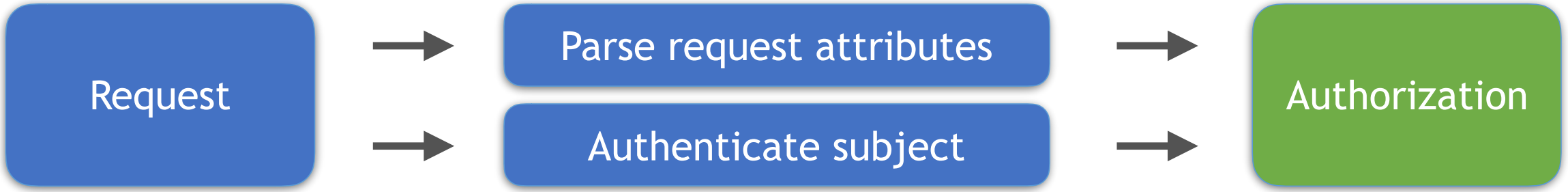
Request Handling


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

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

Username	bob
Groups	trainers system:authenticated

Request Handling



Can bob in groups trainers and system:authenticated
create
apps/v1 deployments in namespace ns1?

RBAC Overview

deployer in namespace ns1

create apps/v1 deployments

RBAC Overview

role

deployer in namespace ns1

create apps/v1 deployments

RBAC Overview

role

deployer in namespace ns1

permissions

create apps/v1 deployments

RBAC Overview

role → deployer in namespace ns1 ← location

permissions → create apps/v1 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
```

Cluster Setup

Bootstrap superuser

- Credential with `system:masters` superuser group

Alternatives:

- Use additional authorizer, bind cluster-admin ClusterRole
- Create via unsecured API server port (not recommended)

Cluster Setup

Control plane component credentials

Standard user names get roles by default:

- system:kube-scheduler
- system:kube-controller-manager
- system:kube-proxy

Cluster Setup

Kubelet authorization

- Use Node authorization mode for kubelets
- Requires standard user/group names
- Node TLS bootstrapping

Cluster Setup

kube-apiserver

- --authorization-mode=Node,RBAC,…

kube-controller-manager

- --use-service-account-credentials
- --kubeconfig

Applying Policies

Pre-defined roles:

- cluster-admin
- admin
- edit
- view
- …

Applying Policies

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

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

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

Applying Policies

Grant a role to all service accounts in a namespace

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

Building Custom Roles

1. Enable auditing
2. Exercise application using a dedicated service account
3. Capture audit logs
4. Create a role (or set of roles) that allow the requests

Building Custom Roles

Demo

https://github.com/liggitt/audit2rbac

Distributing Custom Roles

Standalone roles/bindings

- Roles/ClusterRoles
- RoleBindings/ClusterRoleBindings
- Service Account
- Deployment/Job/DaemonSet, etc

Distributing Custom Roles

- Aggregated roles (new in 1.9)
- Contribute to roles like admin/edit/view
- Aggregates labeled 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

KubeCon

North America 2017

Effective RBAC

Jordan Liggitt, *Red Hat*