Managing users in multi-tenant kubernetes cluster

Jessica Andersson, Meltwater

@solidtubez

Authorization

Authentication

Authorization

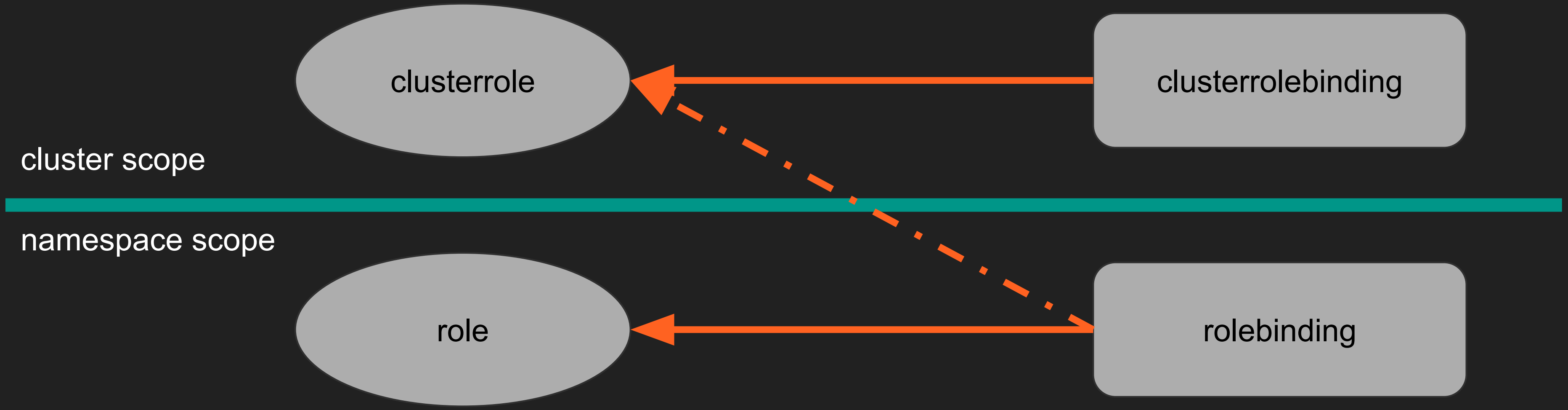
Role Based Access Control (rbac)

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: deployment-manager
rules:
 - apiGroups:
     - ''
   resources:
     - pods
     - pods/portforward
     - pods/proxy
   verbs:
     - create
     - delete
     - ...
```

Additive

- every get, list, watch, create, update, patch, delete
- on every resource (jobs, cronjobs, daemonsets, deployments, ingresses, replicasets, secrets, statefulsets, bindings, events, resourcequotas, pods……)

Clusterroles vs Roles



One clusterrole and many rolebindings!

Admin roles vs User roles

- Admin role has access to *everything*
- User roles only have access to defined resources in an add-when-needed style

Test the user experience before release!

kubectx/kubens

kubens

didn't work for the users 😞

Namespaces are cluster scoped resources

Users only have access to namespace scoped resources

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: kubens
rules:
 - apiGroups:
     - ''
   resources:
     - namespaces
   verbs:
     - get
     - list
     - watch
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: kubens-crb
subjects:
 - kind: Group
   name: 'system:authenticated'
   apiGroup: rbac.authorization.k8s.io
roleRef:
 kind: ClusterRole
 name: kubens
 apiGroup: rbac.authorization.k8s.io
```

Authentication

X509 Client Certs

```
openssl req -new -key jbeda.pem -out jbeda-csr.pem -subj "/CN=jbeda/O=app1/O=app2"
```

- once per user
- if groups change, new certificate needs to be created
- sharing files

OpenID Connect Tokens

- Many identity providers; Google, Salesforce etc
- access_token, id_token, refresh_token
- kubectl --token or kubeconfig

dex

Github Organisation

Managing repository administrators and access

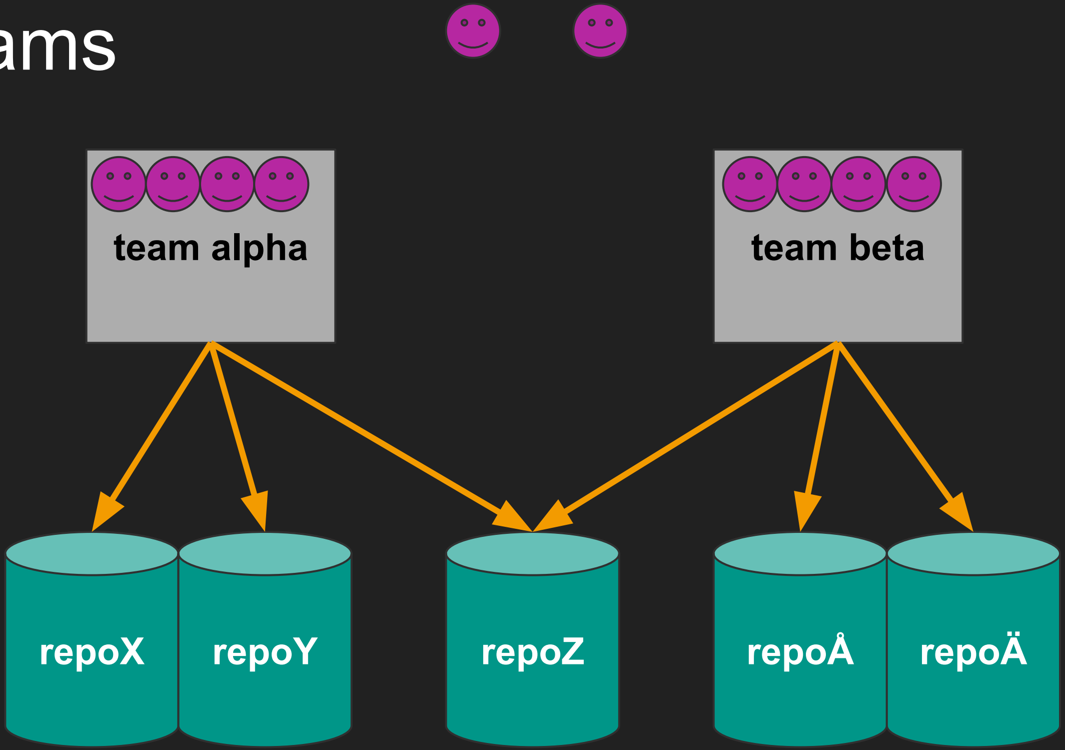
Onboarding and offboarding employee git access

pros: already in place, teams can take responsibility for who should have access, logical separation

```yaml
kind: Namespace
apiVersion: v1
metadata:
 name: bravo-demo
 labels:
   name: bravo-demo
   team: bravo
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
 name: bravo-demo-manager-binding
 namespace: bravo-demo
subjects:
- kind: User1
 name: user1@internet.com
 apiGroup: rbac.authorization.k8s.io
- kind: User2
 name: user2@internet.com
 apiGroup: rbac.authorization.k8s.io
roleRef:
 kind: ClusterRole
 name: deployment-manager
 apiGroup: rbac.authorization.k8s.io
```

Github teams



```yaml
subjects:
- kind: User1
 name: user1@internet.com
 apiGroup: rbac.authorization.k8s.io
- kind: User2
 name: user2@internet.com
 apiGroup: rbac.authorization.k8s.io
- kind: User3
 name: user2@internet.com
 apiGroup: rbac.authorization.k8s.io
- kind: User4
 name: user2@internet.com
 apiGroup: rbac.authorization.k8s.io
- kind: User5
 name: user2@internet.com
 apiGroup: rbac.authorization.k8s.io
```

```yaml
subjects:
- kind: Group
 name: "meltwater:Bravo"
 apiGroup: rbac.authorization.k8s.io
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
 name: bravo-demo-manager-binding
 namespace: bravo-demo
subjects:
- kind: Group
 name: "meltwater:Bravo"
 apiGroup: rbac.authorization.k8s.io
roleRef:
 kind: ClusterRole
 name: deployment-manager
 apiGroup: rbac.authorization.k8s.io
```

Lessons learned

Authorization

- rbac is additive
- 1 clusterrole ← * rolebindings
- test as your users!

Authentication

- There's many ways to do it
- Find what fits your organization
- Groups vs individual

Thanks

Managing users in multi-tenant kubernetes cluster

Jessica Andersson, Meltwater

@solidtubez