Understand Service Accounts

Elle Krout
Principal Course Author, Pluralsight

Kubernetes Service Accounts

Service Account Use Cases

- Automation
- Kubernetes API Access
- External Authentication
- Admission Controllers

View Service Accounts

```
> kubectl get serviceaccounts

NAME                  SECRETS   AGE
default               0         62d
prometheus-operator   0         30d
```

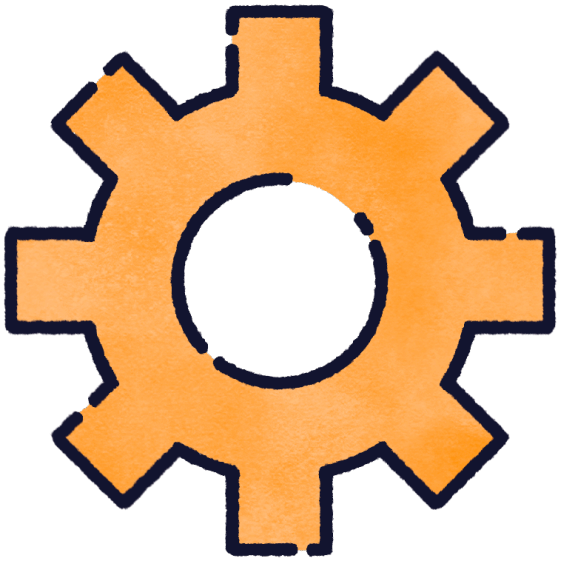
View Service Accounts

```
> kubectl get sa

NAME                  SECRETS   AGE
default               0         62d
prometheus-operator   0         30d
```

View Service Account Information

```
> kubectl describe sa default

Name:                default
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   <none>
Tokens:              <none>
Events:              <none>
```

Service Accounts



Default service account

- Assigned to all pods in default namespace unless overridden
- No tokens or permissions

All pods need to be mapped to only one service account

Create a Kubernetes Service Account

Create a Service Account

```
> kubectl create serviceaccount <service_account_name>

serviceaccount/<service_account_name> created
```

Service Account Manifest

sa.yaml

```yaml
apiVersion: v1
kind: ServiceAccount
automountServiceAccountToken: true
metadata:
  name: service-account
```

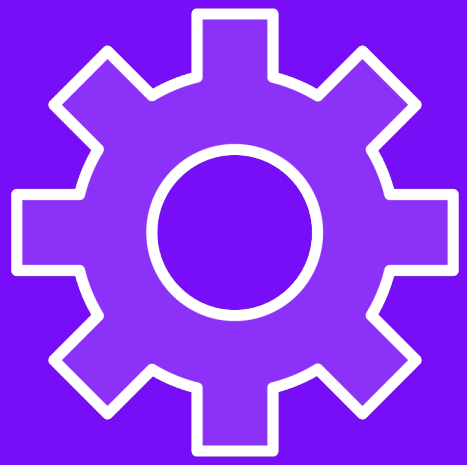
Assign Service Account to Pod

test-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  serviceAccountName: test-service-account
  containers:
  - name: app
    image: busybox
    command: ["sh", "-c", "sleep 3600"]
```

View Service Account Mapped to Pod

```
> kubectl get pod nginx -o jsonpath='{.spec.serviceAccountName}'

Default


> kubectl describe pod nginx | grep "Service Account"

Service Account:  default
```



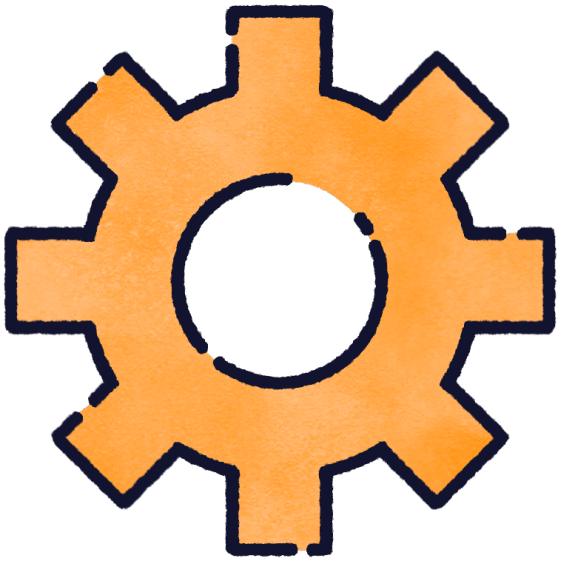
Service Accounts

Provide pods a secure identity for interacting with the Kubernetes API.

Kubernetes Service Accounts and RBAC

Service accounts must be provided permissions through role-based access control

Service Account Access



Create a `Role` or `ClusterRole`

Done the same way as for human users and groups

Create a Role

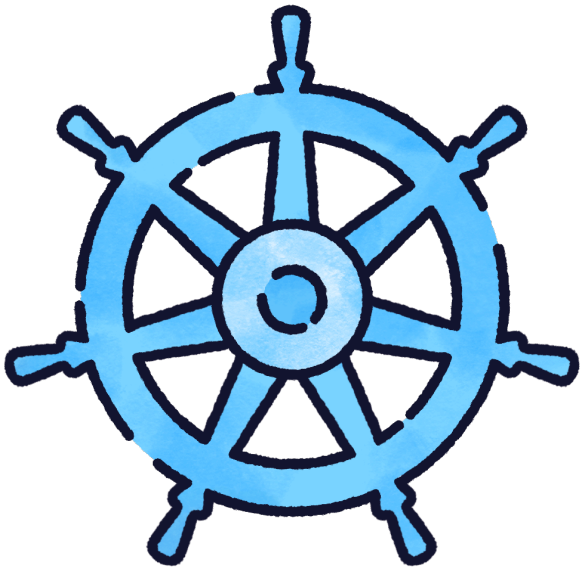
pod-reader-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-reader
  namespace: backend
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list"]
```

Create a RoleBinding

pod-reader-rolebinding.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods-binding
  namespace: backend
subjects:
  - kind: ServiceAccount
    name: backend-sa
    namespace: backend
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```



Service Accounts...

Have no access on their own; RBAC is what provides them with the ability to perform actions.

Demo: Assigning Access with Kubernetes Service Accounts

Exam Scenario

The `pod-viewer.yaml` pod is a diagnostics tool that needs to be able to list pods in its own namespace, `ops`. Create a service account with the minimal amount of RBAC permissions required so it can accomplish this task