Understanding RBAC

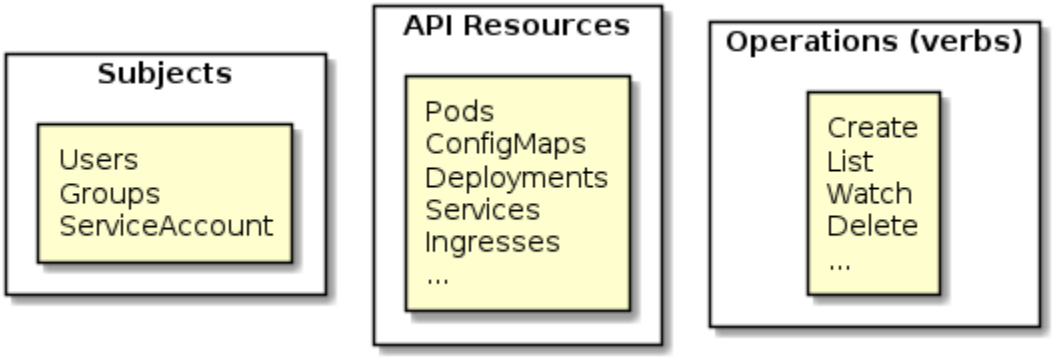
Background

- Establish a system of users with different roles to access a set of Kubernetes resources.
- Control processes running in a Pod and operations they can perform via the Kubernetes API.
- Limit the visibility of certain resources per namespaces.

Overview

RBAC has three main building blocks.

Subject	The user or process that wants to access a resource.
Resource	The Kubernetes API resource type.
Verb	The operation that can be executed on the resource.



Calls to the API server needs to be authenticated and Kubernetes offers a variety of authentication methods for those API requests.

Authentication Strategy	Description
X.509 client certificate	Use an OpenSSL client certificate to authenticate.
Basic authentication	Username and password.
Bearer tokens	Use an OpenID or web-hooks as ways to authenticate.

ServiceAccount

- Kubernetes does not store or represent users or groups with an API resources, it does not exist in its etcd database, however ServiceAccounts exists as objects in Kubernetes and are used by processes running inside the cluster.
- Kubernetes clusters come with a ServiceAccount called `default` that lives in the `default` namespace. Any pod that doesn't explicitly assign a ServiceAccount uses the `default` ServiceAccount.

Creating a ServiceAccount

Method	Example

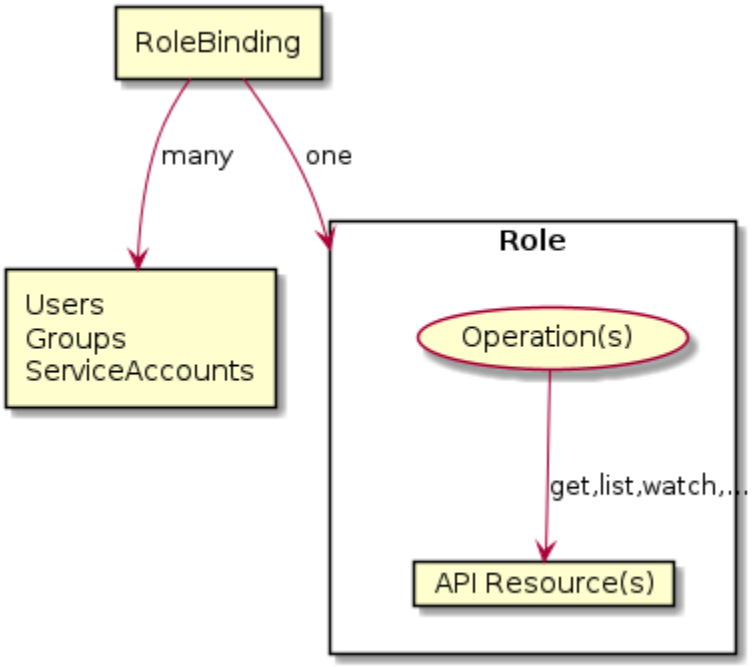
kubectl CLI	`kubectl create serviceaccount test-bot`

YAML manifest	`apiVersion: v1`
	`kind: ServiceAccount`
	`metadata:`
	`  name: test-bot`

RBAC API Primitives

Kubernetes has two API resource primitives used to implement the RBAC functionality.

Resource	Description
Role	The Role API primitive declares the API resources and the set of allowed operations on those resources.
RoleBinding	The RoleBinding API primitive binds the Role object to the subject(s).



Default Roles in Kubernetes

Default ClusterRole	Description
cluster-admin	Allows read/write access to any resources across all namespaces.
admin	Allows read/write to any resources in a namespace.
edit	Allows read/write access to resources in a namespace except Roles and RoleBindings. Does provide access to Secrets.
view	Allows read-only access to resource in a namespace except Roles, RoleBindings, and Secrets.

Creating Roles

Method	Example

kubectl CLI	`kubectl create role my-ro --verb=get,list,watch --resource=pods,deployments,services`

YAML manifest	```apiVersion: rbac.authroization.k8s.io/v1
	kind: Role
	metadata:
	  name: my-ro
	rules:
	  - apiGroups:
	    - ""
	    resources:
	    - pods
	    - services
	    verbs:
	    - list
	    - get
	    - watch
	  - apiGroups:
	    - apps
	    resources:
	    - deployments
	    verbs:
	    - list
	    - get
	    - watch```
	The API group name for a resource can be identified by performing `kubectl explain <resource>` .