KUBERNETES – A & A

Github repo..

https://github.com/vsaini44/KubernetesRepo.git

Understanding Authentication

The API server can be configured with one or more authentication plugins

Several authentication plugins are available. They obtain the identity of the client using the following methods:

- **From the client certificate**
- **From an authentication token passed in an HTTP header**
- **Basic HTTP authentication**
- **Others**

Users

An authentication plugin returns the username and group(s) of the authenticated user. Kubernetes doesn’t store that information anywhere; it uses it to verify whether the user is authorized to perform an action or not.

Kubernetes distinguishes between two kinds of clients connecting to the API server:

- **Actual humans (users)**
- **Pods (more specifically, applications running inside them)**

Service Account

A service account provides an identity for processes that run in a Pod.

When you (a human) access the cluster (for example, using kubectl), you are authenticated by the apiserver as a particular User Account.

Processes in containers inside pods can also contact the apiserver. When they do, they are authenticated as a particular Service Account

Authorization

The Kubernetes API server can be configured to use an authorization plugin to check whether an action is allowed to be performed by the user

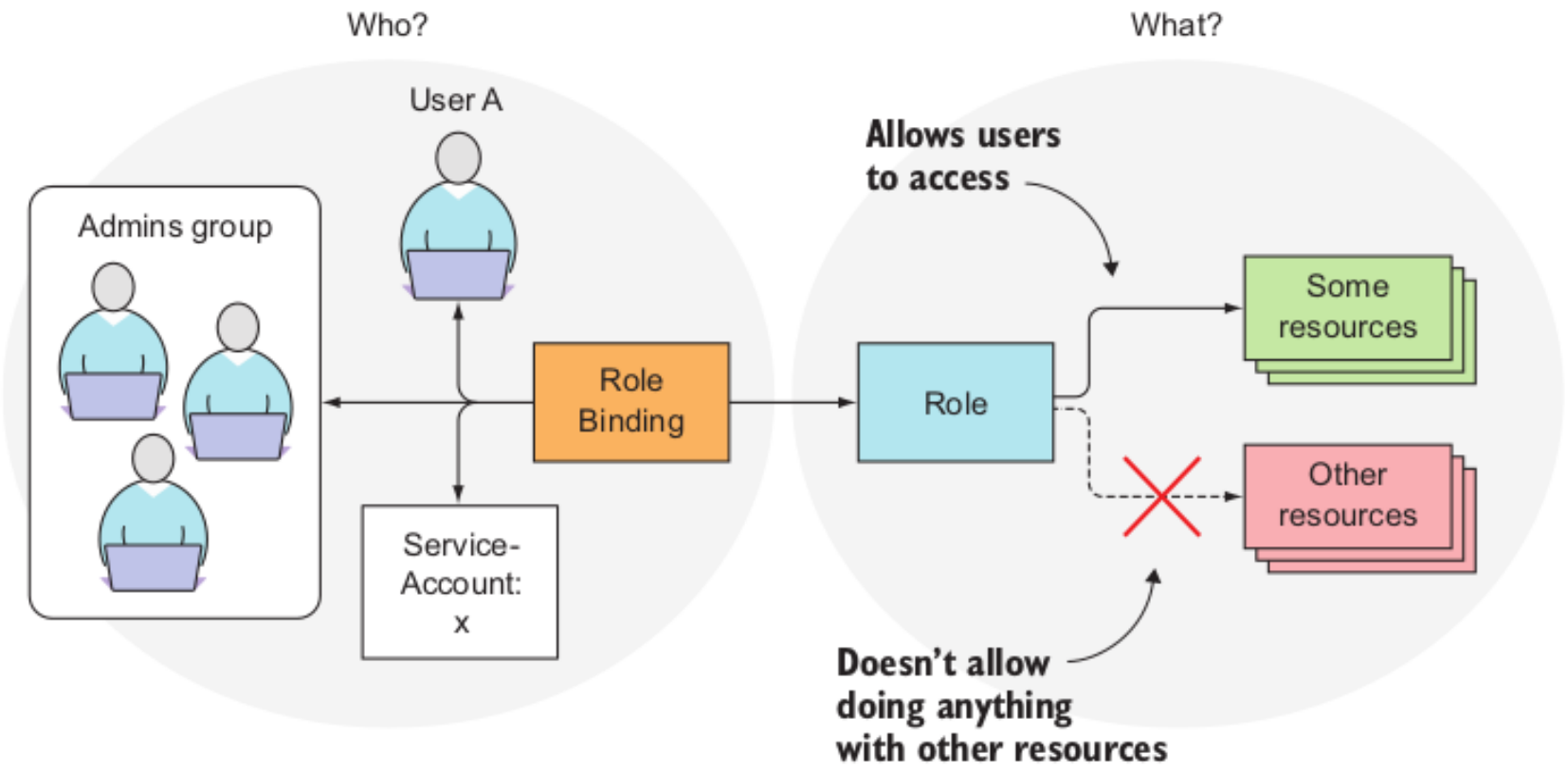
Actions like:

Get Pods

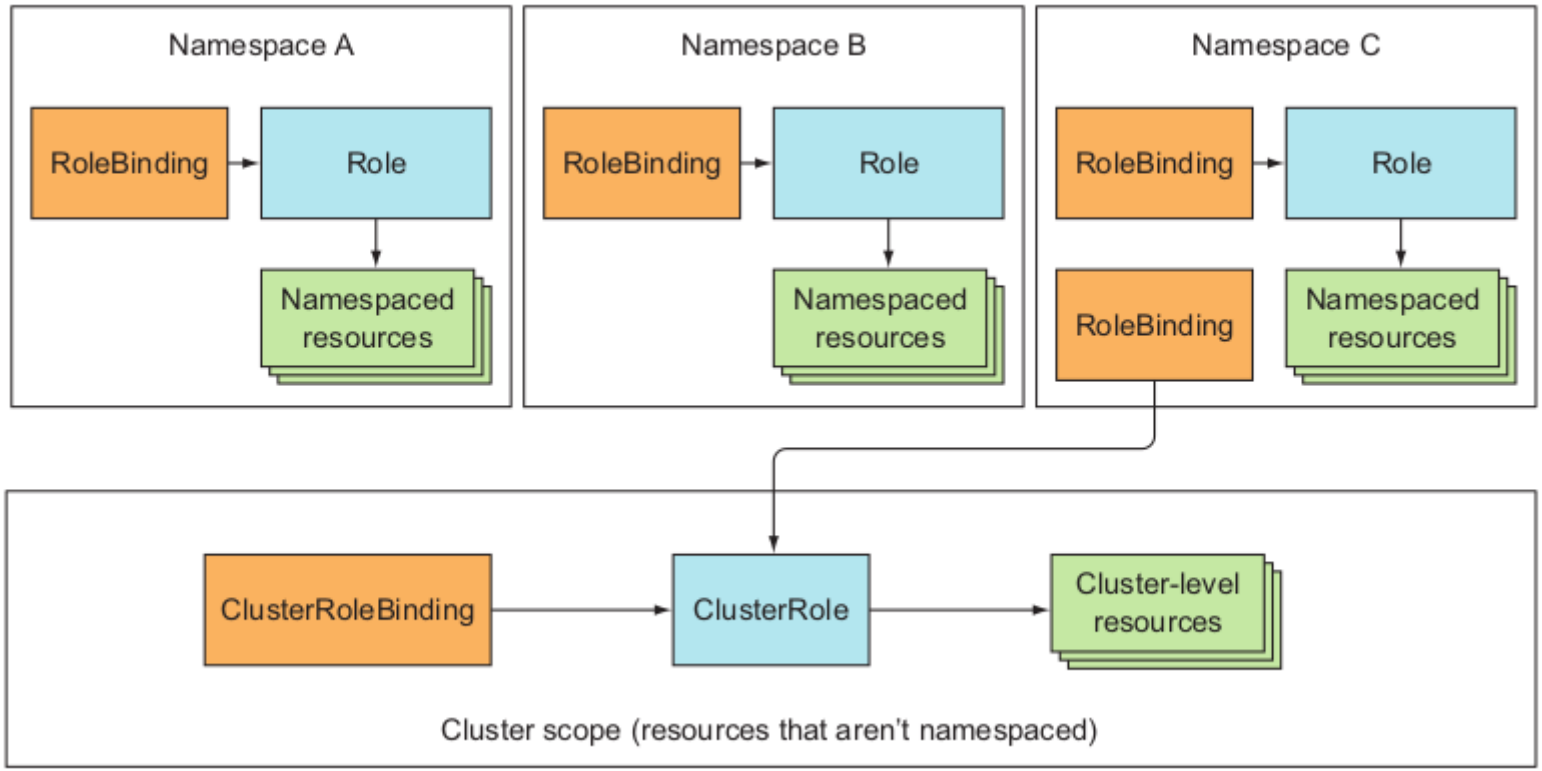
Create Services

Update secrets and so on

Authorization



Role, Clusterrole and Rolebinding



The distinction between a Role and a ClusterRole, or between a RoleBinding and a ClusterRoleBinding, is that the Role and RoleBinding are namespaced resources, whereas the ClusterRole and ClusterRoleBinding are cluster-level resources