Exploring RBAC Authorization

This lesson introduces us to different components of the RBAC authorization.

WE'LL COVER THE FOLLOWING

The RBAC Components

Managing Kubernetes RBAC requires knowledge of a few elements. Specifically, we should learn about Rules, Roles, Subjects, and RoleBindings.

Rules

> A *Rule* is a set of operations (verbs), resources, and API groups. Verbs describe activities that can be performed on resources which belong to different API Groups.

Currently supported verbs are as follows.

Verb	Description
`get`	Retrieves information about a specific object
`list`	Retrieves information about a

	collection of objects
`create`	Creates a specific object
`update`	Updates a specific object
`patch`	Patches a specific object
`watch`	Watches for changes to an object
`proxy`	Proxies requests
`redirect`	Redirects requests
`delete`	Deletes a specific object
`deletecollection`	Deletes a collection of objects

Permissions defined through Rules are additive. We cannot deny access to some resources.

If, for example, we'd like to allow a user only to create objects and retrieve their information, we'd use the verbs `get`, `list` and `create`. A verb can be an asterisk (`*`), thus allowing all verbs (operations).

Verbs are combined with Kubernetes resources. For example, if we'd like to allow a user only to create Pods and retrieve their information, we'd mix `get`, `list` and `create` verbs with the `pods` resource.

The last element of a Rule is the API Group. RBAC uses the `rbac.authorization.k8s.io` group. If we'd switch to a different authorization method, we'd need to change the group as well.

Roles

> A *Role* is a collection of Rules. It defines one or more Rules that can be bound to a user or a group of users.

The vital aspect of Roles is that they are applied to a Namespace. If we'd like to create a role that refers to a whole cluster, we'd use *ClusterRole* instead. Both are defined in the same way, and the only difference is in the scope (Namespace or an entire cluster).

Subjects

The next piece of the authorization mechanism is *Subjects*.

> Subjects define entities that are executing operations. A Subject can be a *User*, a *Group*, or a *Service Account*.

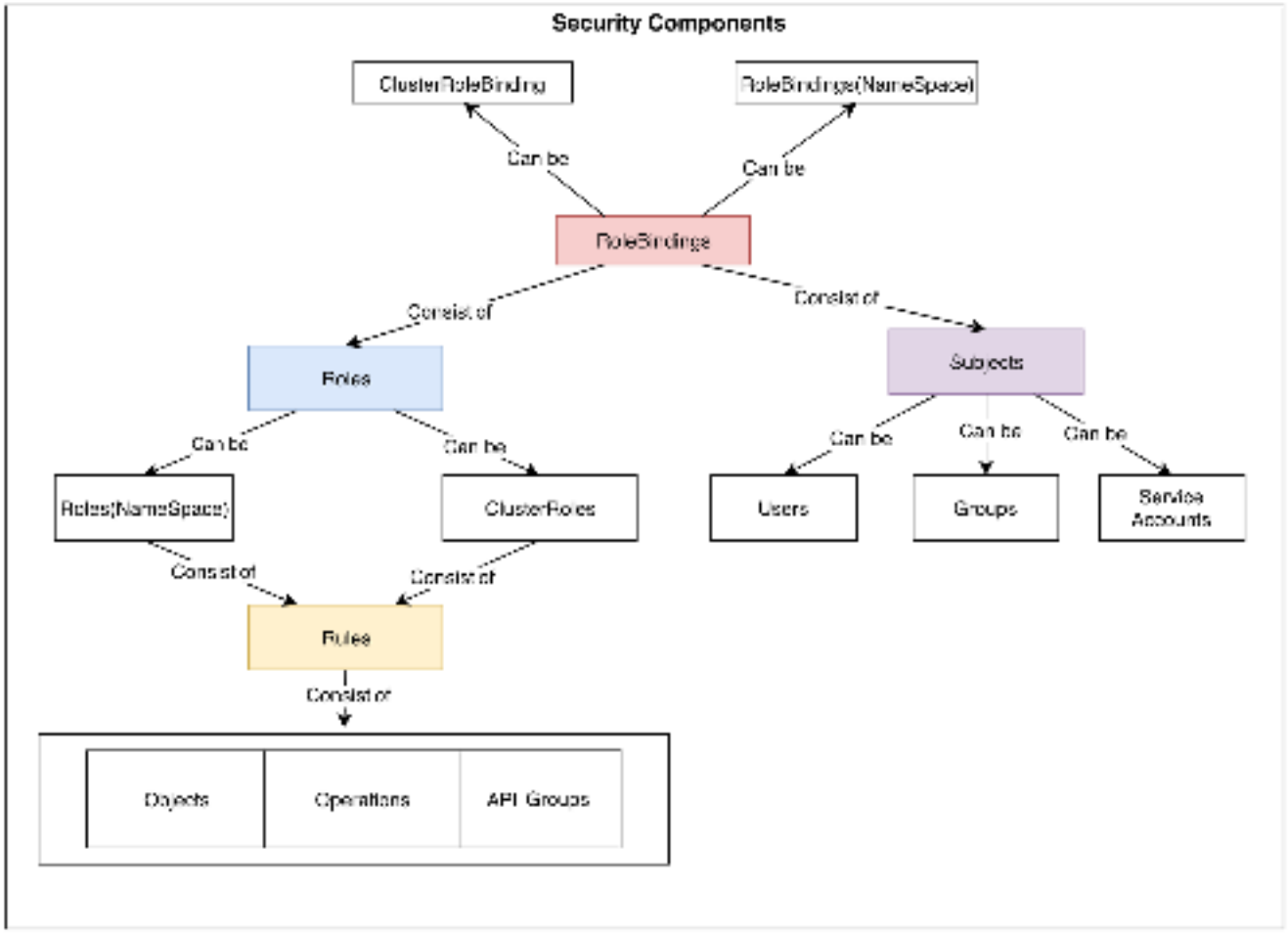
A User is a person or a process residing outside a cluster. A Service Account is used for processes running inside Pods that want to use the API. Since this chapter focuses on human authentication, we won't explore them right now. Finally, Groups are collections of Users or Service Accounts. Some Groups are created by default (e.g., `cluster-admin`).

RoleBindings

Finally, we need *RoleBindings*.

> As the name suggests, RoleBindings bind Subjects to Roles.

Since Subjects define users, RoleBindings effectively bind users (or Groups or Service Accounts) to Roles, thus giving them permissions to perform certain operations on specific objects within a Namespace. Just like roles, RoleBindings have a cluster-wide alternative called *ClusterRoleBindings*. The only difference is that their scope is not limited to a Namespace, but applied to a whole cluster.



The security components explored so far

All that might seem confusing and overwhelming. You might even say that you did not understand anything. Don't worry, we'll explore each of the RBAC components in more details through practical examples.

The examples that follow will clarify everything.

In the next lesson, let's go back to John's issue and try to solve it.