# Creating Cluster Role Bindings

In this lesson, we will grant cluster-wide access to the user with the help of Cluster Role Bindings.

WE'LL COVER THE FOLLOWING



## View Access Across the Cluster

We'll change John's `view` permissions so that they are applied across the whole cluster.

Instead of executing yet another ad-hoc kubectl commands, we'll define `ClusterRoleBinding` resource in YAML format so that the change is documented.

### Looking into the Definition

Let's take a look at the definition in the `auth/crb-view.yml` file.

```
cat auth/crb-view.yml
```

The **output** is as follows.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: view
subjects:
- kind: User
  name: jdoe
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: view
```

```
  apiGroup: rbac.authorization.k8s.io
```

Functionally, the difference is that, this time, we're creating `ClusterRoleBinding` instead of `RoleBinding`. Also, we specified the `apiGroup` explicitly thus making it clear that the `ClusterRole` is RBAC.

## Creation of ClusterRoleBinding #

```
kubectl create -f auth/crb-view.yml \
    --record --save-config
```

We created the role defined in the YAML file, and the output confirmed that `clusterrolebinding "view"` was `created`.

We can further validate that everything looks correct by describing the newly created role.

```
kubectl describe clusterrolebinding \
    view
```

The **output** is as follows.

```
Name:         view
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"rbac.authorization.k8s.io/v1","kind":"ClusterRoleBinding","met
              kubernetes.io/change-cause: kubectl create --filename=auth/crb-view.yml --recor
Role:
  Kind:  ClusterRole
  Name:  view
Subjects:
  Kind  Name  Namespace
  ----  ----  ---------
  User  jdoe
```

Finally, we'll impersonate John and validate that he can indeed retrieve the Pods from any Namespace.

```
kubectl auth can-i get pods \
    --as jdoe --all-namespaces
```

The **output** is `yes`, thus confirming that `jdoe` can view the Pods.

We're so excited that we cannot wait to let John know that he was granted

We're so excited that we cannot wait to let John know that he was granted permissions. However, a minute into the phone call, he raises a concern.

While being able to view Pods across the cluster is a good start, he will need a place where he and other developers will have more freedom.

They will need to be able to deploy, update, delete, and access their applications. They will probably need to do more, but they can't give you more information. They are not yet very experienced with Kubernetes, so they don't know what to expect.

John is asking you to find a solution that will allow them to perform actions that will help them develop and test their software without affecting other users of the cluster.

---

In the next lesson, we will sort out how to grant more freedom to the users.