Managing RBAC in Source Control

- Like all resources in k8s, RBAC resources are modelled using JSON or YAML
- Since this is text based expression it makes sense to store these resources in version control. This will help for auditing, accountability and rollback changes for RBAC
- In kubectl command-line has a reconcile command that operates much like kubectl apply

kubectl auth reconcile -f some-rbac-config.yaml

Aggregating ClusterRoles

- Sometimes we want to be able to define roles that are combination of other roles. K8s RBAC supports the usage of aggregation rule to combine multiple roles together in a new role.
- Like all aggregations or grouping in K8s the ClusterRoles to be aggregated are specified using label selectors

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: learningrole
aggregationRule:
  clusterRoleSelectors:
    - matchLabels:
        rbac.authorization.k8s.io/aggregate-to-learn: "true"
```

Using Groups for Bindings

- To bind a ClusterRole we can use a Group kind for subject in Bindings

```
....
subjects:
- apiGroup: "rbac.authorization.k8s.io"
  kind: Group
  name: dev-group
```