Kubernetes Security

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

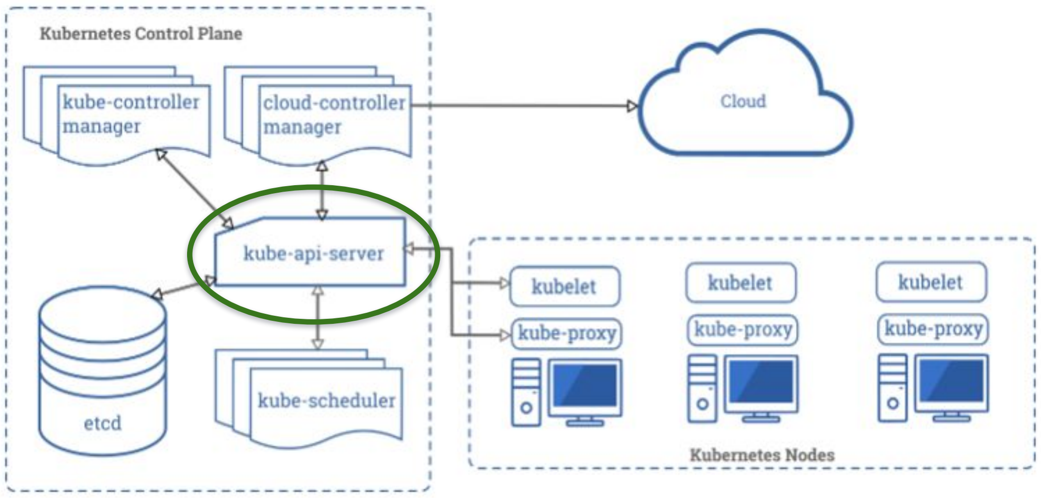
CLARUSWAY©
WAY TO REINVENT YOURSELF

1

Core Concepts

CLARUSWAY©
WAY TO REINVENT YOURSELF

Core Concepts



CLARUSWAY©
WAY TO REINVENT YOURSELF

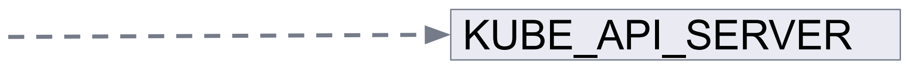
Core Concepts

kube-apiserver:

- Provides a forward facing REST interface into the kubernetes control plane and datastore.
- All clients and other applications interact with kubernetes strictly through the API Server.
- Acts as the gatekeeper to the cluster by handling **authentication** and **authorization**, request validation, mutation, and admission control in addition to being the front-end to the backing datastore.

Core Concepts

Who can Access?



What can they do?

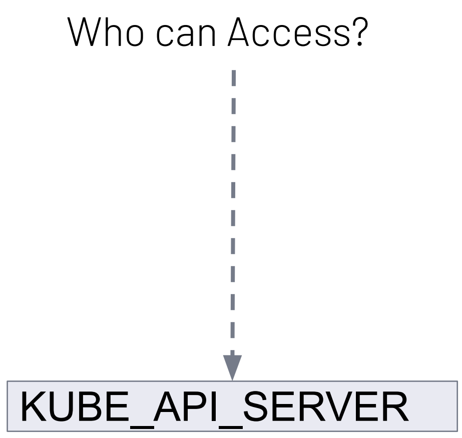
Core Concepts

Who can Access? $\dashrightarrow$ **Authentication**

What can they do? $\dashrightarrow$ **Authorization**

2 Authentication

Authentication



Authentication

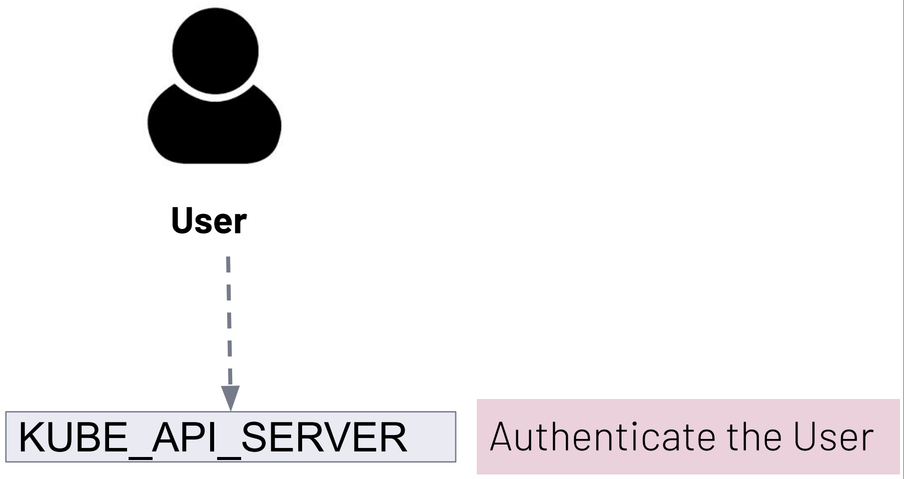
Who can Access?

User

Service Accounts

- **User accounts** are for humans. **Service accounts** are for **processes**, which run in pods.
- **User accounts** are intended to be **global**. Names must be unique across all namespaces of a cluster.
- Service accounts are namespaced.

Authentication

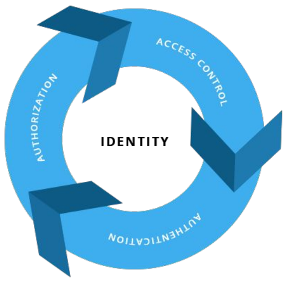


Authentication Strategies

client certificates

Static Token File

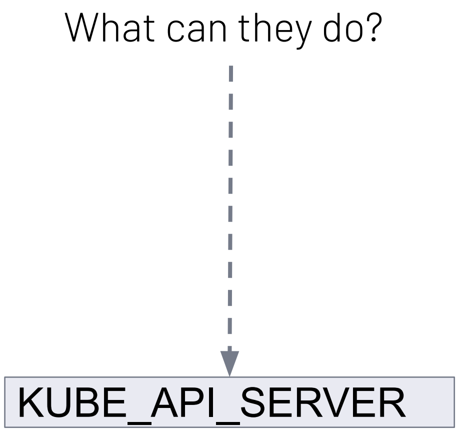
Identity Services



3 Authorization

CLARUSWAY©
WAY TO REINVENT YOURSELF

Authorization



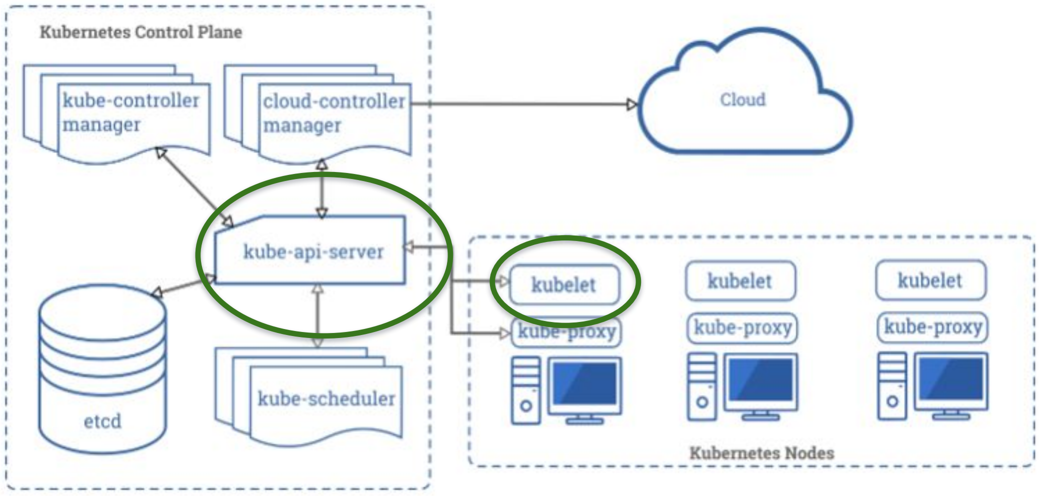
CLARUSWAY©
WAY TO REINVENT YOURSELF

Authorization Modes

AlwaysAllow | Node | ABAC | RBAC | Webhook | AlwaysDeny

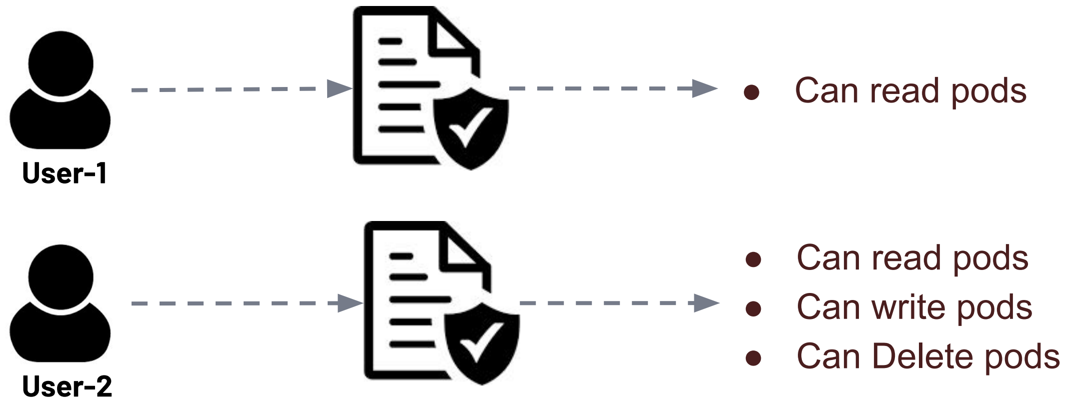
Node

Node authorization is a special-purpose authorization mode that specifically authorizes API requests made by kubelets.



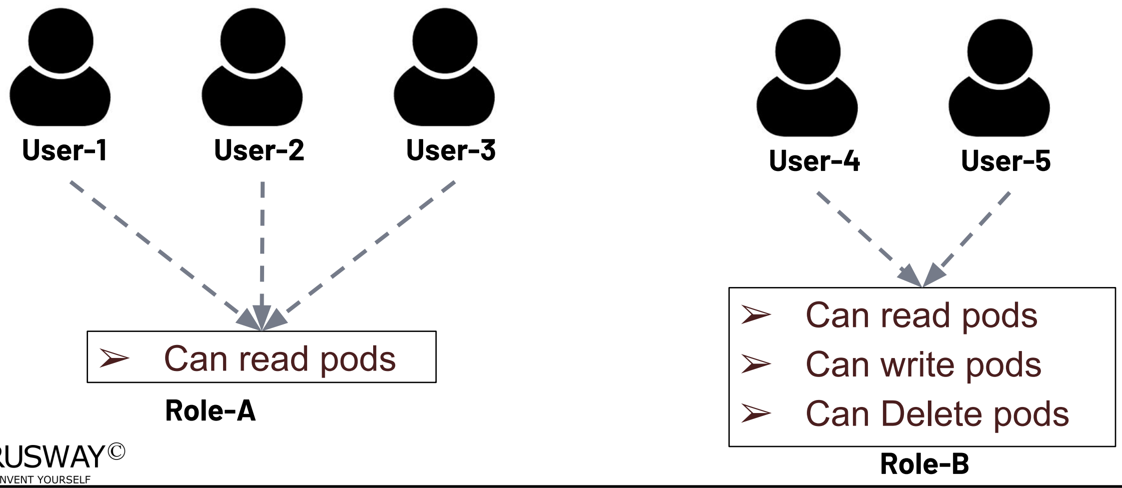
ABAC

Attribute-based access control (ABAC) defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together.



RBAC

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within your organization.



Role and ClusterRole

RBAC Role or ClusterRole contains rules that represent a set of permissions.

- A **Role** always sets permissions within a **particular namespace**; when you create a Role, you have to specify the namespace it belongs in.

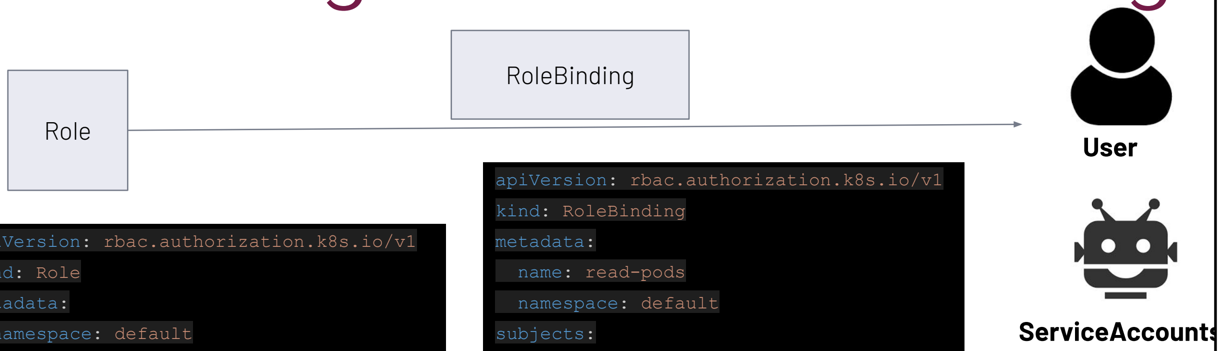
- **ClusterRole**, by contrast, is a **non-namespaced** resource.

RoleBinding and ClusterRoleBinding

- A **role binding** grants the permissions defined in a role to a user or set of users.

- A **RoleBinding** grants permissions within a specific namespace whereas a **ClusterRoleBinding** grants that access cluster-wide.

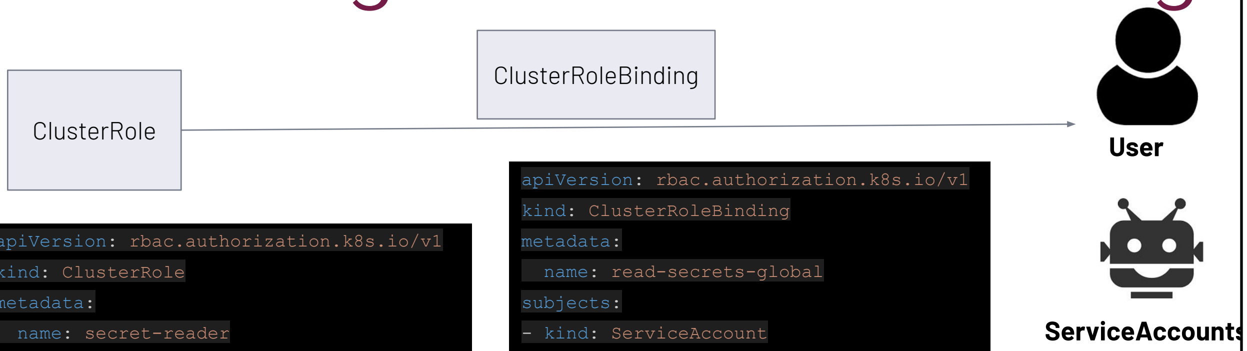
RoleBinding and ClusterRoleBinding



```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: Jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

RoleBinding and ClusterRoleBinding



```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: ServiceAccount
  name: mysa
  namespace: default
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

4

API Groups

CLARUSWAY©
WAY TO REINVENT YOURSELF

API Groups

- API groups make it easier to extend the Kubernetes API. The API group is specified in a REST path and in the apiVersion field of a serialized object.
- There are several API groups in Kubernetes such as **apis**, **healthz**, **metrics** etc.

/api	/apis	/healthz	/metrics	/logs	/version

CLARUSWAY©
WAY TO REINVENT YOURSELF

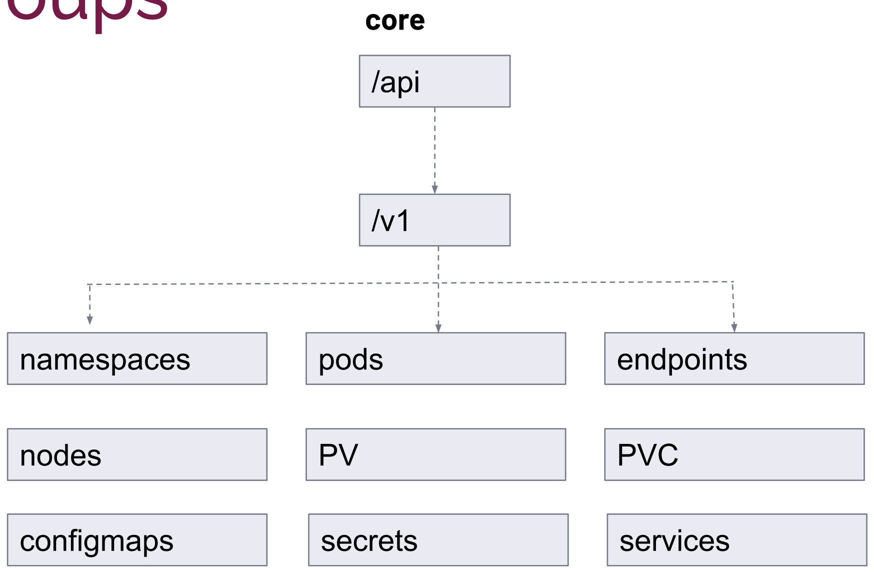
API Groups

- **api** and **apis** are responsible for the cluster of functionality.
- These APIs are categorized into two groups. The **core** group and the **named** group.
- **The core** (also called legacy) group is found at REST path **/api/v1.**
- **The named groups** are at REST path **/apis/$GROUP_NAME/$VERSION** and use apiVersion: $GROUP_NAME/$VERSION (for example, apiVersion: batch/v1).

API Groups

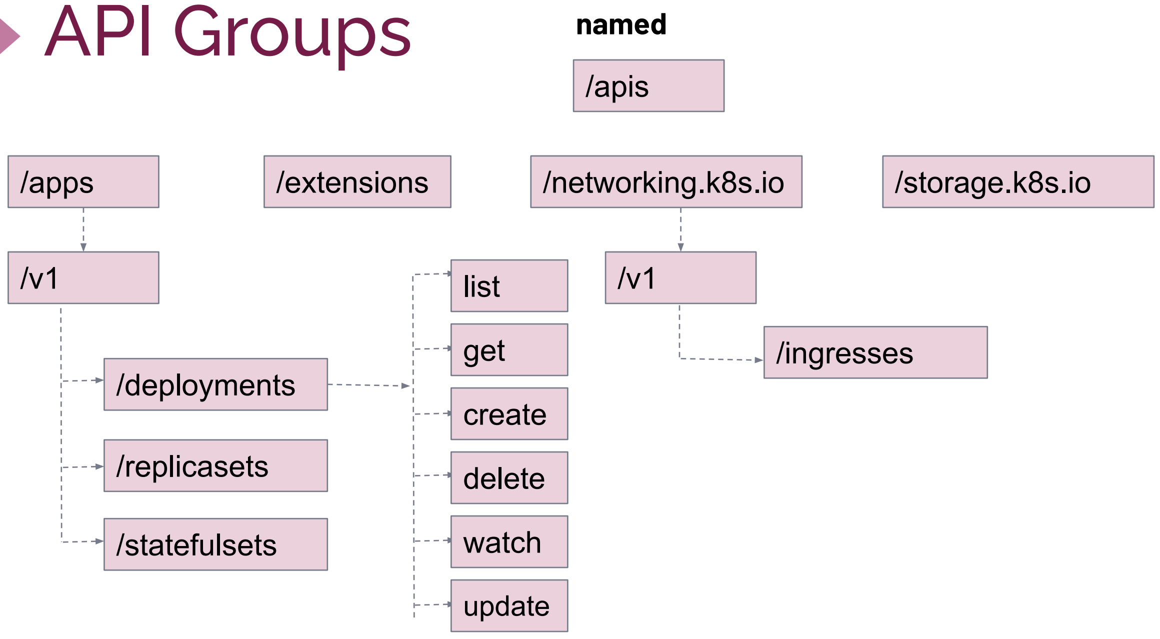
core
/api
named
/apis

API Groups



CLARUSWAY©
WAY TO REINVENT YOURSELF

API Groups



CLARUSWAY©
WAY TO REINVENT YOURSELF

API Groups

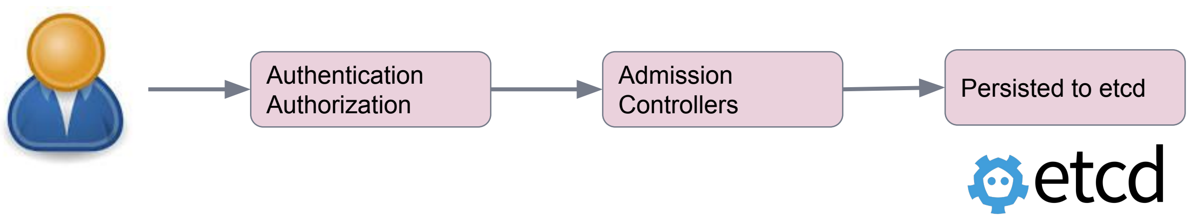
Commands For check api groups:

- kubectl proxy --port=8080 &
- curl localhost:8080
- curl localhost:8080/version → kubectl version
- curl localhost:8080/api/v1/pods
- curl http://localhost:8080/api/v1/namespaces/default/pods

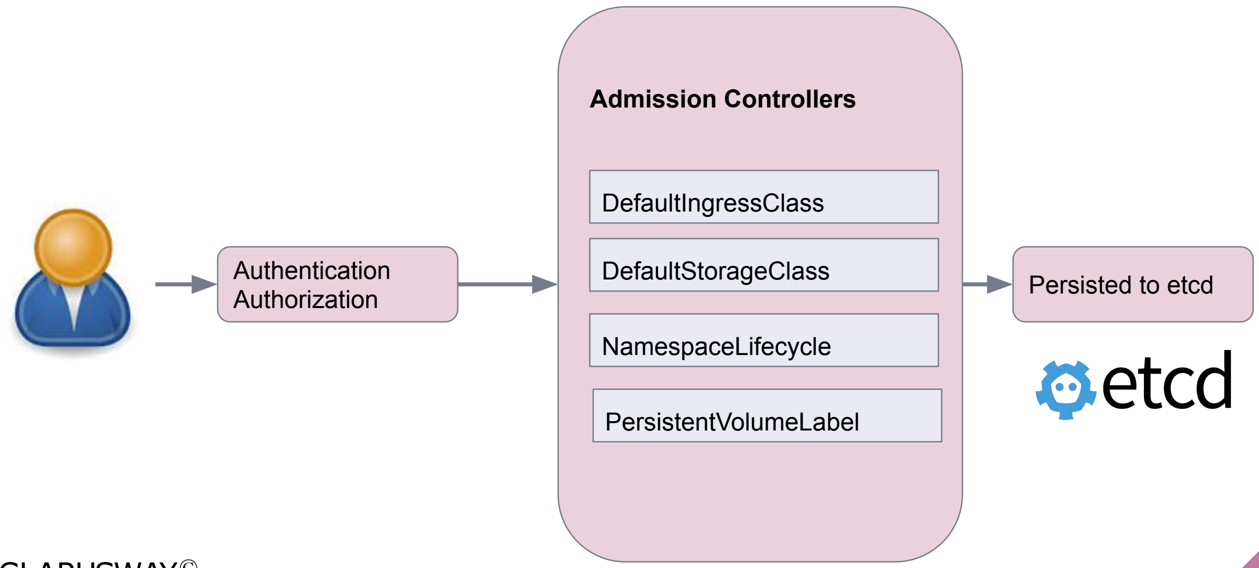
5 Admission Controllers

Admission Controllers

An admission controller is a piece of code that intercepts requests to the Kubernetes API server prior to persistence of the object, but after the request is authenticated and authorized.

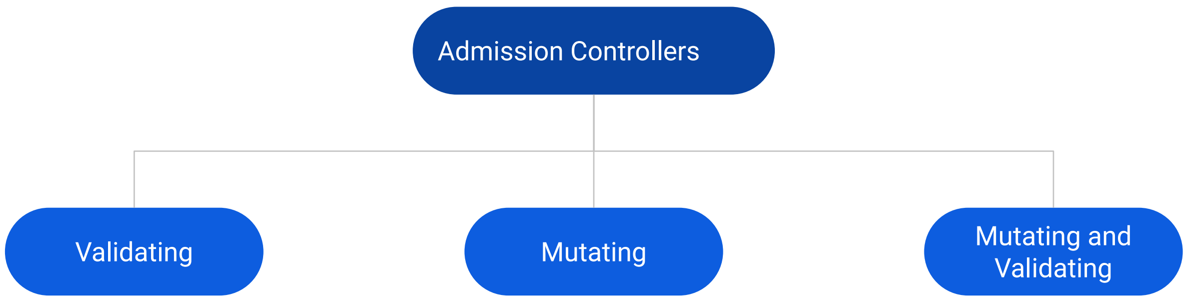


Admission Controllers



Admission Controllers

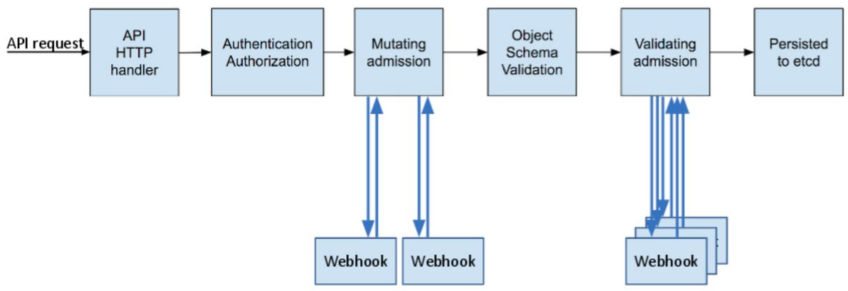
Admission controllers may be validating, mutating, or both. Mutating controllers may modify related objects to the requests they admit; validating controllers may not.



Admission Controllers

Admission control phases

The admission control process proceeds in two phases. In the first phase, mutating admission controllers are run. In the second phase, validating admission controllers are run. Note again that some of the controllers are both.



THANKS!

Any questions?