



Kubernetes Security



CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents



- Core Concepts
- Authentication
- Authorization

CLARUSWAY©
WAY TO REINVENT YOURSELF

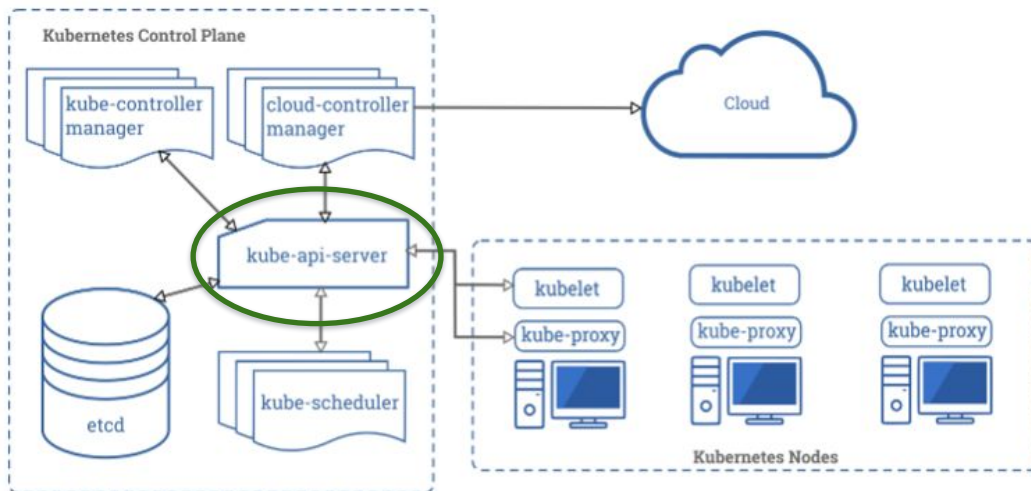


1

Core Concepts



Core Concepts





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?

-----> KUBE_API_SERVER

What can they do?



Core Concepts




Who can Access? -----> **Authentication**

What can they do? -----> **Authorization**

CLARUSWAY[©]
WAY TO REINVENT YOURSELF

7





2 Authentication

CLARUSWAY[©]
WAY TO REINVENT YOURSELF



► Authentication

Who can Access?



KUBE_API_SERVER



► Authentication

Who can Access?



User

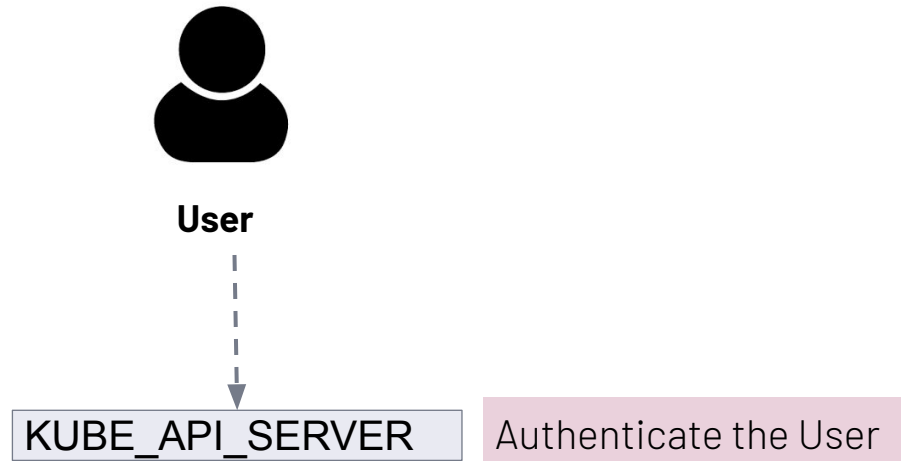


Service Accounts

- **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

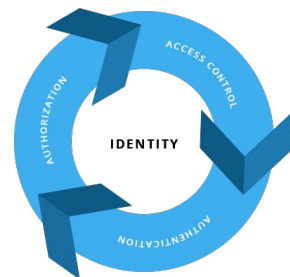
client certificates



Static Token File



Identity Services





3

Authorization



Authorization

What can they do?



KUBE_API_SERVER



► Authorization Modes

AlwaysAllow

Node

ABAC

RBAC

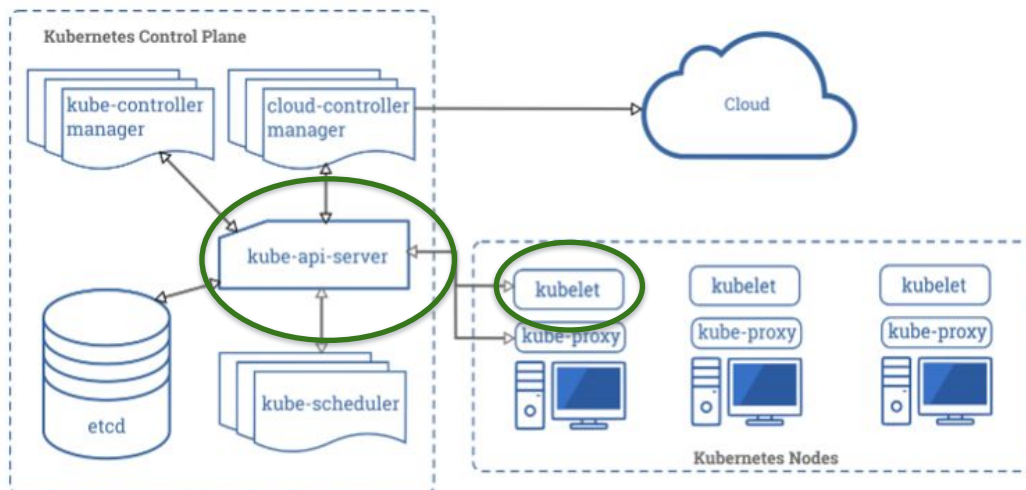
Webhook

AlwaysDeny



► Node

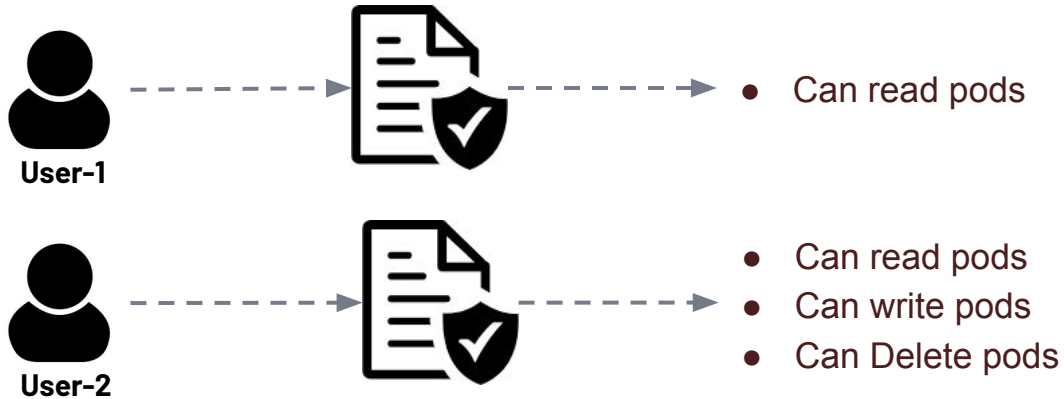
Node authorization is a special-purpose authorization mode that specifically authorizes API requests made by kubelets.





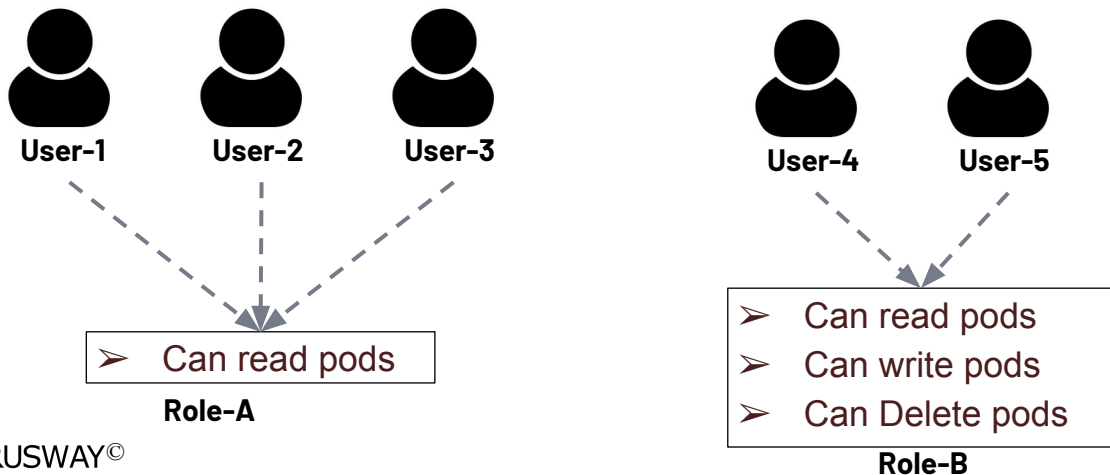
ABAC

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.



Webhook

A **WebHook** is an HTTP callback: an HTTP POST that occurs when something happens; a simple event-notification via HTTP POST. A web application implementing WebHooks will POST a message to a URL when certain things happen.

When specified, mode Webhook causes Kubernetes to query an outside REST service when determining user privileges.



THANKS!

Any questions?





▶ 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:

- The core (also called legacy) group is found at REST path `/api/v1`. The core group is not specified as part of the `apiVersion` field, for example, `apiVersion: 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

- Kubernetes API is grouped into multiple such groups based on their purpose. Such as one for `apis`, one for `healthz`, `metrics` and `logs` etc.
- The version API is for viewing the version of the cluster.
- `metrics` and `healthz` api are used to monitor the health of the cluster.

`/api`

`/apis`

`/logs`

`/healthz`

`/metrics`

`/version`



▶ API Groups

/api

/apis

/healthz

/metrics

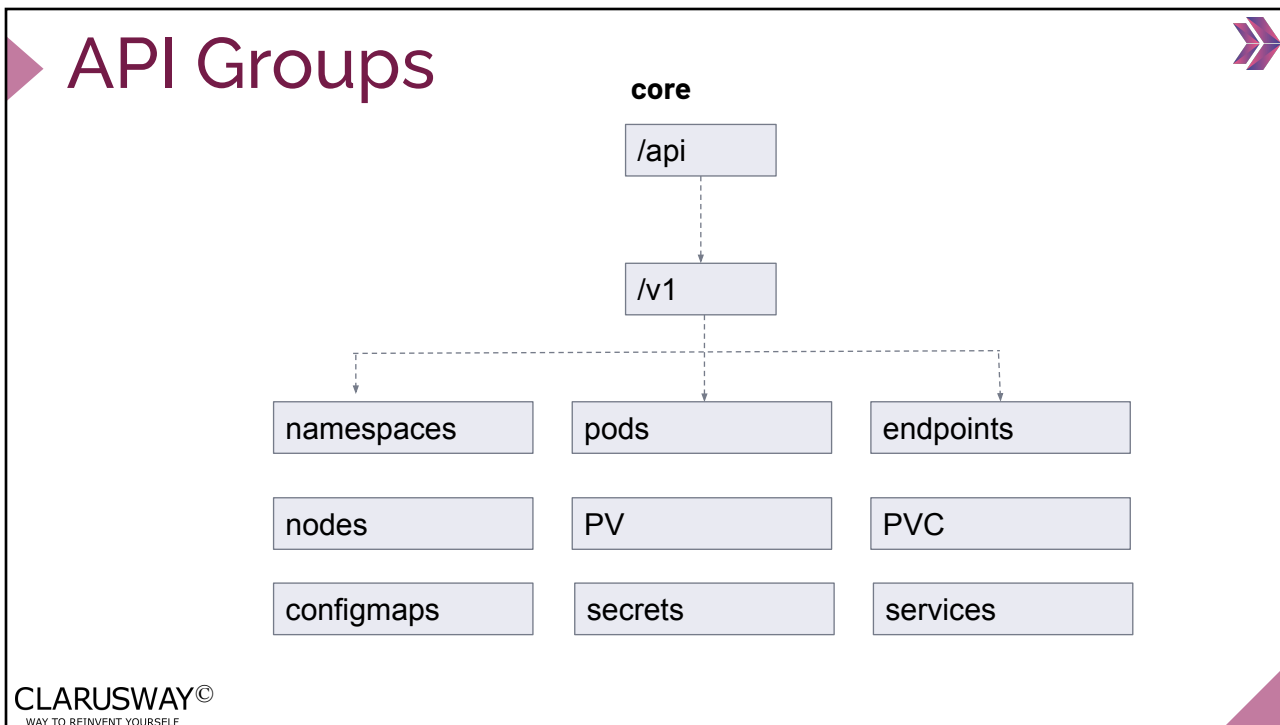
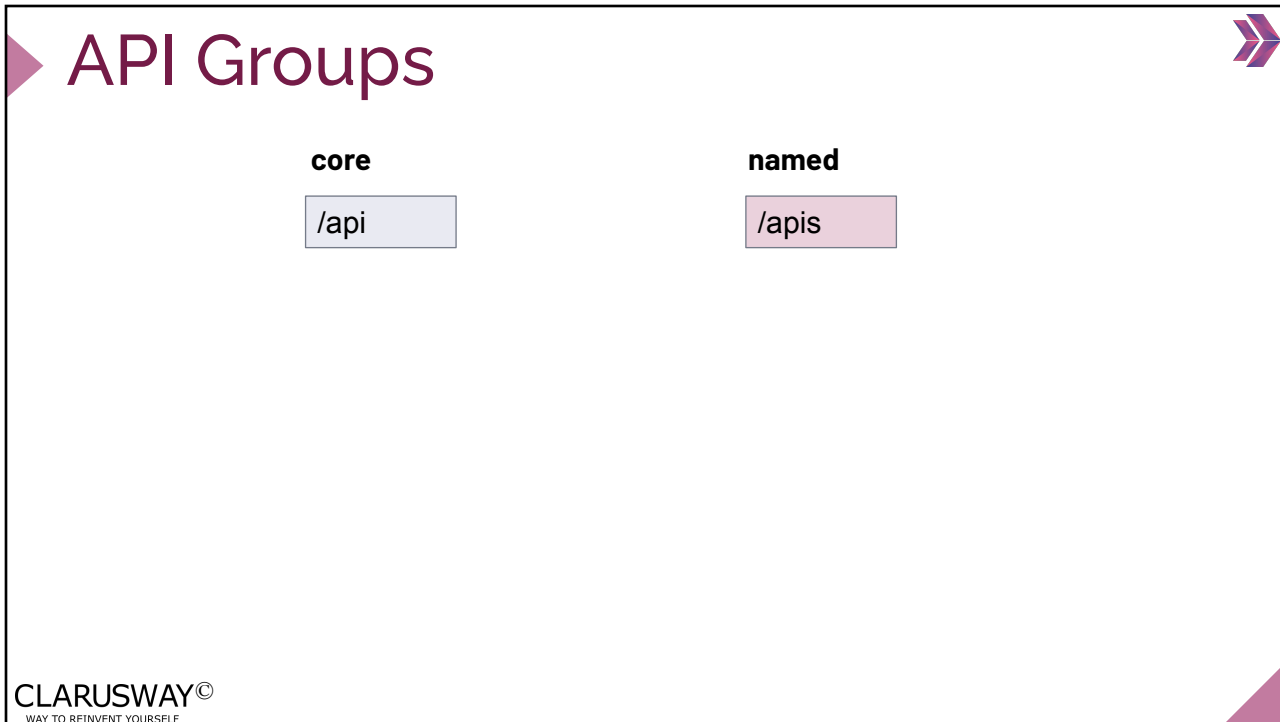
/logs

/version



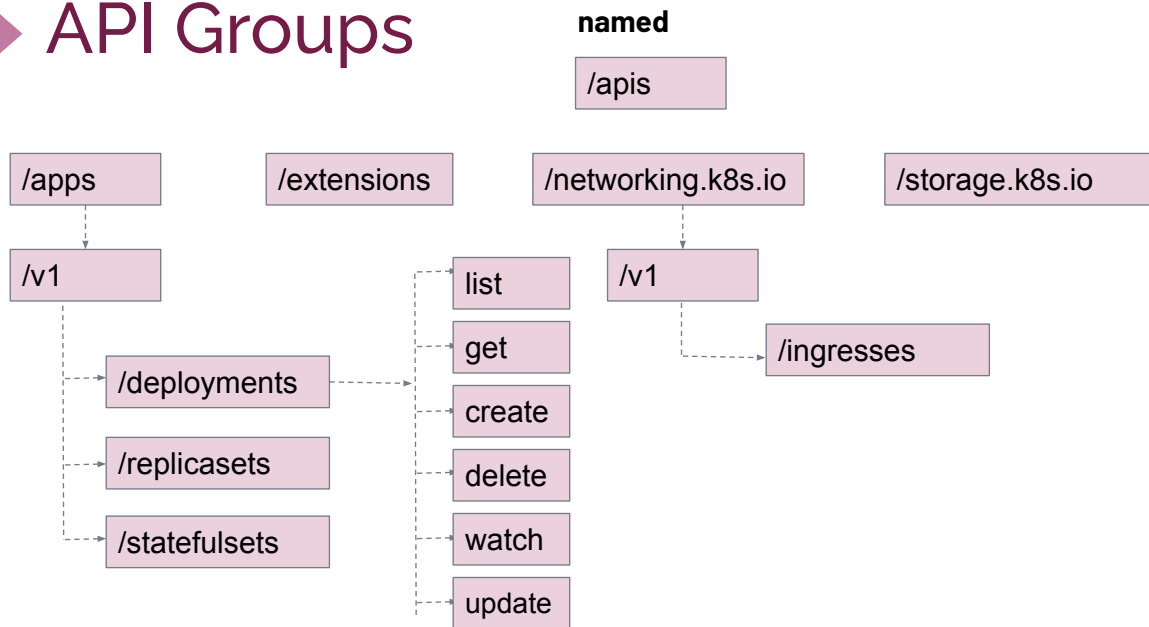
▶ API Groups

- **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 group is where all core functionality exists. Such as namespaces, pods, replication controllers, events, endpoints, nodes, bindings, Persistent volumes, persistent volume claims, configmaps, secrets, services etc.





API Groups



API Groups

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