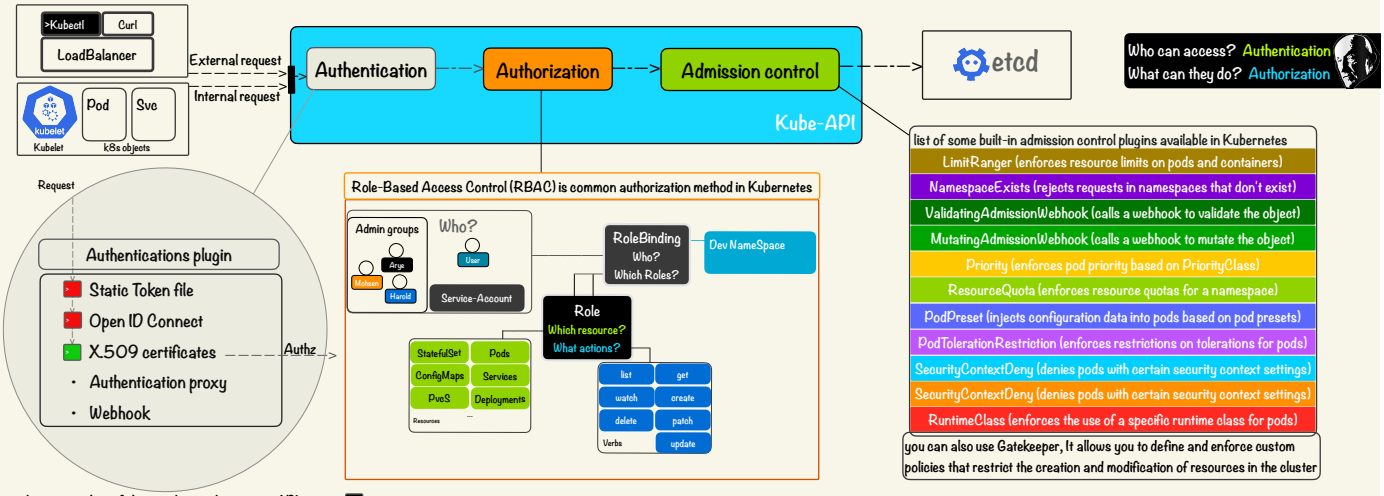
Authentication & Authorization

When a client (such as kubectl or a custom application) sends an API request to Kubernetes, the request goes through several steps before it is processed and a response is sent back to the client



if one authorization plugin fails to authorize the request, API server try another plugin until it finds one that can authorize the request

Authorization can prevent unauthorized access to resources in the cluster, it cannot prevent the creation or modification of resources that do not comply with the cluster's policies. Admission control helps ensure that only valid and compliant resources are created or updated in the cluster, which can prevent misconfigurations, security vulnerabilities, and other issues

Authentication: Kubernetes uses authentication mechanisms to verify the identity of users and components trying to access the cluster. This ensures that only authorized users and components can access the cluster.

The most significant authentication methods in k8s

Service Account authentication

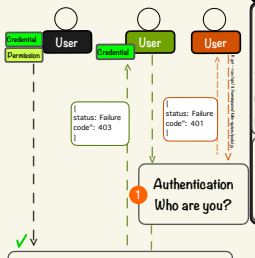
Service Account authentication is a method that uses Kubernetes Service Accounts to authenticate clients. Each Service Account in the cluster has its own token that is used to authenticate clients. This method is commonly used when a cluster has a large number of clients or when automated processes need to access the Kubernetes API

X.509 client certificates authentication

X.509 client certificates authentication is a method that uses digital certificates to authenticate clients. Each client in the cluster has its own certificate and private key that are used to authenticate and encrypt communication when communicating with other components. This method is commonly used when a cluster has a small number of clients or when strong authentication is required

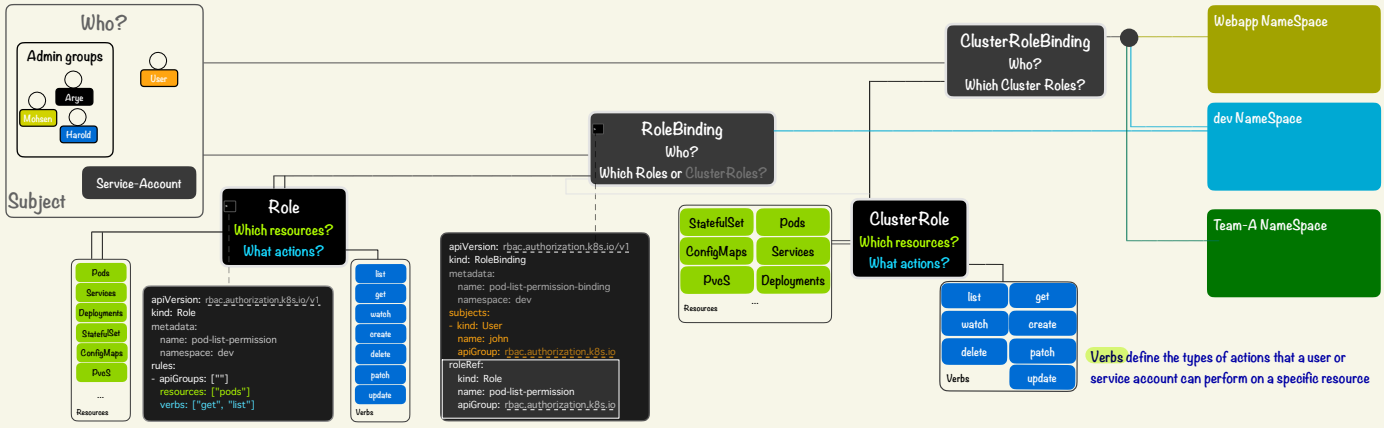
Webhook authentication

Webhook authentication is a method that uses an external HTTP service to authenticate clients. This method is commonly used when a cluster needs to integrate with acustom authentication system. The external HTTP service receives authentication requests from the Kubernetes API server and returns a response indicating whether the client is authenticated or not



Authorization: Kubernetes uses authorization mechanisms to determine what actions a user or component can perform within the cluster. This ensures that users and components have access only to the resources they are authorized to access.

Role-Based Access Control (RBAC): RBAC is a security mechanism in Kubernetes that allows you to control access to resources based on the user's role and permissions. in RBAC, you define roles and cluster roles that specify a set of permissions, such as read, write, or delete, for a particular set of resources. You then create role bindings and cluster role bindings that associate roles and cluster roles with users, groups, or service accounts.



Role	RoleBinding	ClusterRoles	ClusterRoleBinding
Role is a set of permissions that define what actions are allowed on specific resources within a namespace.	RoleBinding is a mechanism for binding a role to a ServiceAccount,a user or group of users within a namespace. Role bindings are used to grant specific permissions to users or groups of users by assigning them to a particular role	ClusterRoles: A ClusterRole is similar to a Role, but it applies to the entire cluster instead of a single namespace. ClusterRoles can be used to grant permissions for cluster-scoped resources (e.g. Nodes) or for resources in all namespaces.	ClusterRoleBinding is cluster-scoped and apply to all namespaces

Roles and ClusterRoles can be either custom or built-in

K get clusterrole

- Built-in Roles and ClusterRoles are predefined by k8s. These built-in roles are designed to provide a set of default permissions for managing Kubernetes resources
- Custom Roles and ClusterRoles are created by users to define their own set of permissions for managing Kubernetes resources

some built-in ClusterRole

- ClusterAdmin: This role is intended to be used by administrators who need full access to all resources in the cluster. It grants permissions to perform any action on any resource in any namespace.
- admin: This ClusterRole provides full access to manage resources in a specific namespace, including the ability to create, update, and delete resources
- system:controller:*, system:node:*: These ClusterRoles provide permissions for k8s controllers and nodes to manage resources in the cluster

```
kubectl describe clusterrole cluster-admin
Name:         cluster-admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  *.*        []                 []              [*]
             [*]                []              [*]
```

The 'system:node' ClusterRole is used to define the set of permissions for nodes in the cluster. This ClusterRole is typically used to grant permissions to the kubelet, to perform actions on various resources related to nodes, including nodes themselves, pods, and service accounts

Admission control: Kubernetes uses a Admission control mechanism for enforcing rules and policies on k8s resources before they are created or updated in the cluster. This can include validating the structure and content of resource manifests, applying default values, and enforcing constraints on resource usage. There are two types of admission control plugins:

Validating admission plugins: These plugins validate the request object without modifying it. They can reject the request if it doesn't meet the required criteria.

Mutating admission plugins: These plugins can modify the request object, as well as validate it. They are executed before the validating admission plugins