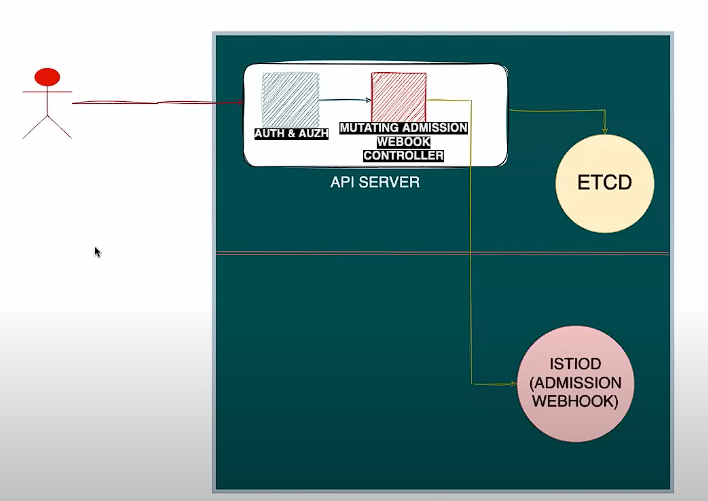
Dynamic admission control in Kubernetes is like having a security checkpoint for requests to the cluster. When someone (or something) tries to create, update, or delete resources in the cluster, dynamic admission control steps in to check if the request follows certain rules.

It uses **webhooks**, which are like custom inspectors. There are two types:

1. **Mutating Webhooks**: These can modify the request before it's processed. For example, they might add default values to a resource if they're missing.
2. **Validating Webhooks**: These check if the request is valid and can reject it if it doesn't meet the rules.

Think of it as a two-step process:

* First, the mutating webhook makes any necessary changes to the request.
* Then, the validating webhook ensures everything is correct before the request is allowed.



Dynamic admission controllers in Kubernetes act like gatekeepers for requests to the cluster. Projects like Istio use them to enforce rules and automate tasks when resources (like pods) are created or updated. Here's how it works in simple terms:

1. **Webhook Setup**:
   * Istio sets up **webhooks**, which are like checkpoints. These webhooks are small programs that run when something happens in the cluster, such as creating a pod.
2. **Mutating Webhooks**:
   * These webhooks can modify requests before they are processed. For example, Istio uses a mutating webhook to automatically add its **sidecar proxy** (Envoy) to your pods. This happens without you needing to do anything manually.
3. **Validating Webhooks**:
   * These webhooks check if a request follows the rules. For instance, Istio uses validating webhooks to ensure that the configurations you apply (like Virtual Services or Destination Rules) are correct and won't break anything.
4. **API Server Interaction**:
   * When you make a request (e.g., to create a pod), it goes to the Kubernetes **API server**. The API server sends the request to Istio's webhooks for validation or modification. Based on the webhook's response, the API server either allows, denies, or modifies the request.

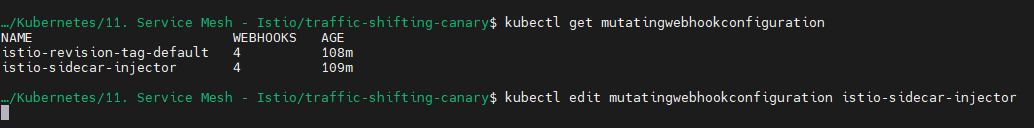
In short, Istio uses dynamic admission controllers to:

* Automatically inject its sidecar proxy into pods.
* Validate configurations to ensure they are correct.
* Enforce security and traffic management policies.

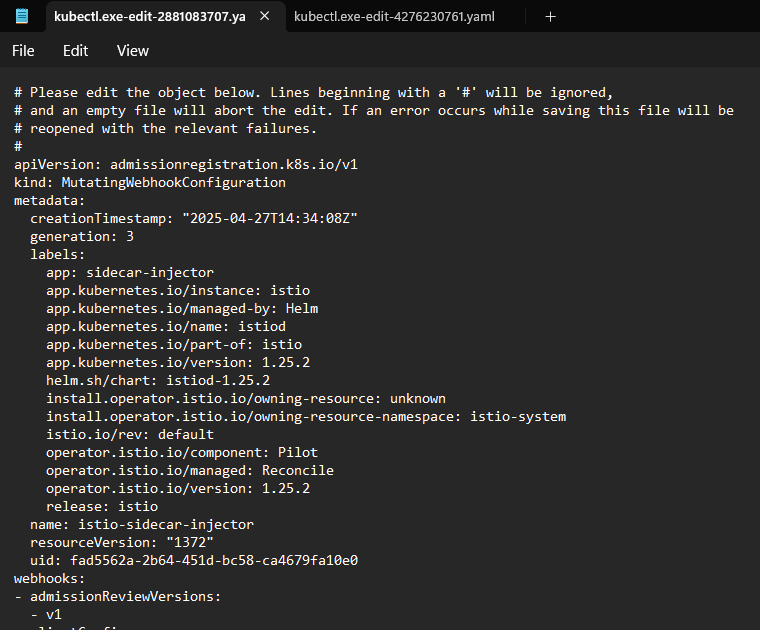
This makes managing and securing your applications easier without requiring manual intervention.

You can review the Admission controllers provided by istio:

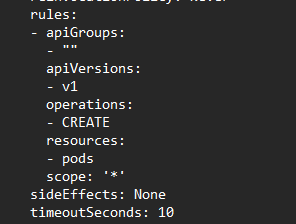
* kubectl get mutatingwebhookconfiguration
* kubectl edit mutatingwebhookconfiguration istio-sidecar-injector



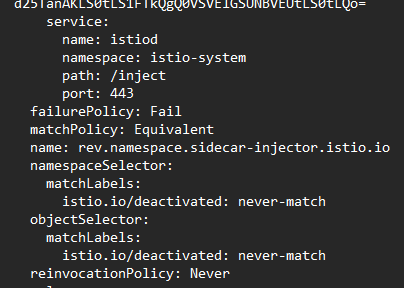
If you review:



Whenever the pods are created:



Forward the request to istiod (admission webhook):



The istiod mutating the request send the object back to api server and from API server the request is persisted in ETCD