Introduction to Admission Control

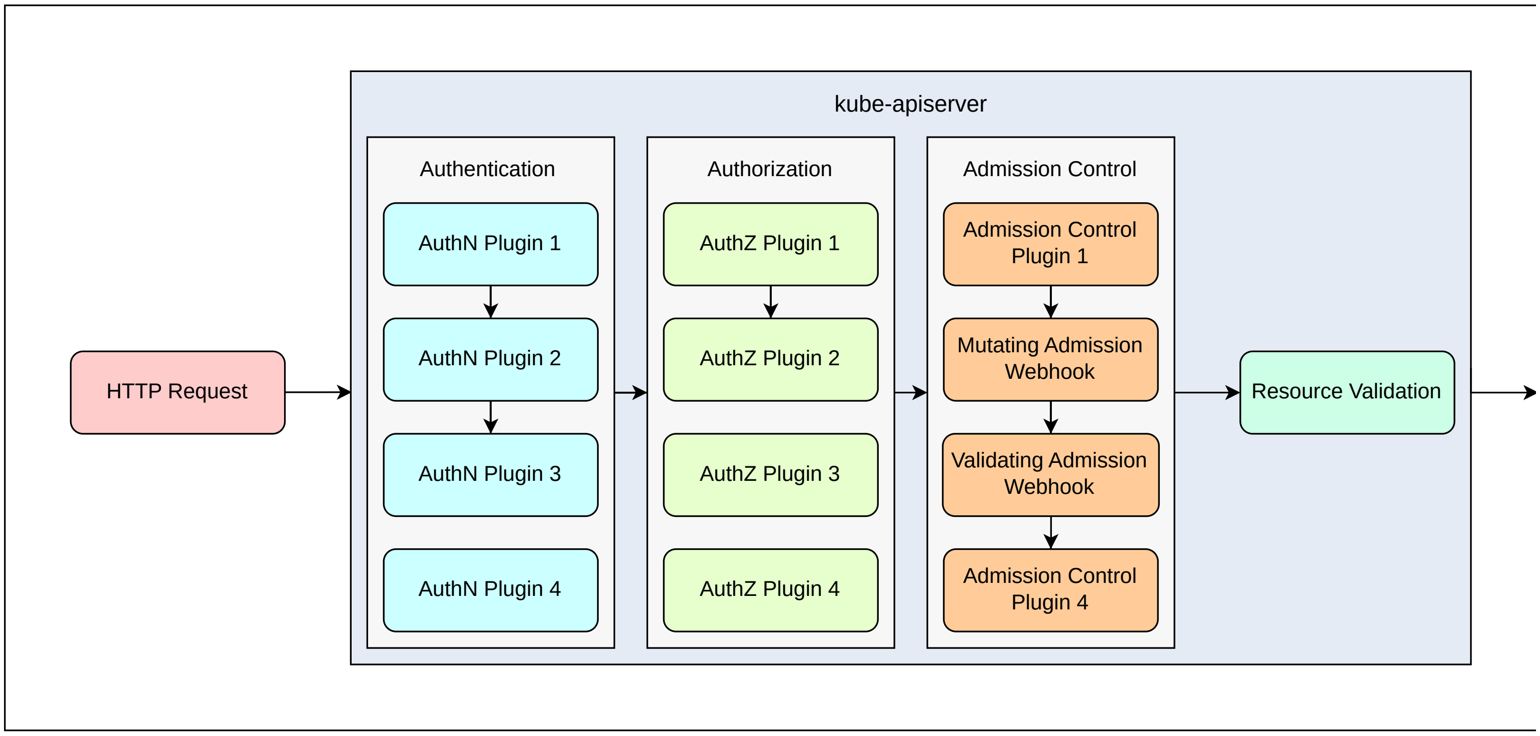
Get introduced to admission control in Kubernetes.

We'll cover the following

- Authentication and authorization
- Why do we need admission control?
  - Advantages of admission control
- Built-in admission controller plugins
- How to configure admission plugins

Authentication and authorization

All the requests that are being sent to the `kube-apiserver` need to pass through the authentication, authorization, and admission control stages, and then come to the final resource validation and persistent storage stages.



Inside the kube-apiserver

It's quite straightforward that all the requests need to get authenticated and authorized, because we need to know exactly who the request senders are and make sure they have the privileges to do these operations.

Why do we need admission control?

The `kube-apiserver` serves all the CRUD requests. However, sometimes we need more granularity on the resource operations, not only RBAC rules. For example, creating a resource in a terminating namespace shouldn't be allowed. Pods running with insecure or malicious Docker images may put the whole cluster in danger. Sometimes, we want to add our own rules and strategies when objects are being created, updated, and deleted.

As a result, we need to have a way to intercept the requests and make some decisions to allow or deny the requests, apply our default configurations, inject sidecar containers, etc. These admission controllers work as gatekeepers that enforce a series of configurable rules to check the validity of the requests and allow customizations on what is allowed to run in our cluster.

Advantages of admission control

Admission controllers bring in multiple advantages, such as:

- **Security:** Admission controllers help tighten the security of the Kubernetes cluster. Several built-in admission plugins on security are provided, such as `PodSecurity`, `StorageObjectInUseProtection`, `ImagePolicyWebhook`, `NodeRestriction`, etc. These plugins keep the cluster safe and help keep the Pod running in a deterministic state.
- **Governance:** The `kube-apiserver` provides services not only for Kubernetes components, but also for other users/clients. A group of plugins like `EventRateLimit`, `ResourceQuota`, `NamespaceLifecycle`, `AlwaysPullImages`, and `CertificateSigning` can help us better govern the cluster. For example, too many events may flood the `kube-apiserver`. Setting limits on resources can help keep our cluster from resource exhaustion. The admission plugin `NamespaceLifecycle` can't help prevent the deletion of Kubernetes reserved namespaces (`kube-system`, `kube-public`, `default`) and creating resources in terminating namespaces.
- **Defaults:** Sometimes, we might want to set a default value on the objects being created and updated. For example, we may want to apply the default resource limits and requests. Plugins like `LimitRanger`, `DefaultStorageClass`, `ServiceAccount`, `RuntimeClass`, etc., can inject default values and configurations when creating Pods.
- **Third-party integration:** In addition to the built-in logic, `ImagePolicyWebhook`, `ValidatingAdmissionWebhook`, and `MutatingAdmissionWebhook` also provide us with ways to implement our own strategies. By running as an external service, the `kube-apiserver` will call the webhook to do the validating or mutating operations.

Built-in admission controller plugins

Now, let's see all the built-in admission controller plugins.

```
// Codes from https://github.com/kubernetes/kubernetes/blob/master/pkg/kubeapiserver/options/plugins.go#
// AllOrderedPlugins is the list of all the plugins in order.
var AllOrderedPlugins = []string{
   admit.PluginName,                        // AlwaysAdmit
   autoprovision.PluginName,                // NamespaceAutoProvision
   lifecycle.PluginName,                    // NamespaceLifecycle
   exists.PluginName,                       // NamespaceExists
   scdeny.PluginName,                       // SecurityContextDeny
   antiaffinity.PluginName,                 // LimitPodHardAntiAffinityTopology
   limitranger.PluginName,                  // LimitRanger
   serviceaccount.PluginName,               // ServiceAccount
   noderestriction.PluginName,              // NodeRestriction
   nodetaint.PluginName,                    // TaintNodesByCondition
   alwayspullimages.PluginName,             // AlwaysPullImages
   imagepolicy.PluginName,                  // ImagePolicyWebhook
   podsecurity.PluginName,                  // PodSecurity - comes before PodSecurityPolicy so the audi
   podsecuritypolicy.PluginName,            // PodSecurityPolicy
   podnodeselector.PluginName,              // PodNodeSelector
   podpriority.PluginName,                  // Priority
   defaulttolerationseconds.PluginName,     // DefaultTolerationSeconds
   podtolerationrestriction.PluginName,     // PodTolerationRestriction
   eventratelimit.PluginName,               // EventRateLimit
   extendedresourcetoleration.PluginName,   // ExtendedResourceToleration
   label.PluginName,                        // PersistentVolumeLabel
   setdefault.PluginName,                   // DefaultStorageClass
   storageobjectinuseprotection.PluginName, // StorageObjectInUseProtection
   gc.PluginName,                           // OwnerReferencesPermissionEnforcement
   resize.PluginName,                       // PersistentVolumeClaimResize
   runtimeclass.PluginName,                 // RuntimeClass
   certapproval.PluginName,                 // CertificateApproval
```

All the ordered admission plugins in the kube-apiserver

All the admission controller plugins need to implement either the `MutationInterface` or `ValidationInterface` or both. This means a plugin could be mutating, validating, or both. The mutating phase runs before the validating phase. During the **mutating phase**, the plugin may modify the objects, such as apply defaults, inject sidecar containers, etc. While during the **validating phase**, the plugin can't perform any kind of modifications and only validation checks are allowed.

All these enabled plugins compose an admission chain, which will be called one by one with the order defined in `AllOrderedPlugins` above. If any of the plugins reject the request, the entire HTTP request is marked as rejected with a returned error to the end user.

How to configure admission plugins

Even though all the plugins are built in and compiled in, they are configurable at runtime. By default, the plugins given below are enabled.

```
// Codes from https://github.com/kubernetes/kubernetes/blob/master/pkg/kubeapiserver/options/plugins.go#
defaultOnPlugins := sets.NewString(
   lifecycle.PluginName,                    // NamespaceLifecycle
   limitranger.PluginName,                  // LimitRanger
   serviceaccount.PluginName,               // ServiceAccount
   setdefault.PluginName,                   // DefaultStorageClass
   resize.PluginName,                       // PersistentVolumeClaimResize
   defaulttolerationseconds.PluginName,     // DefaultTolerationSeconds
   mutatingwebhook.PluginName,              // MutatingAdmissionWebhook
   validatingwebhook.PluginName,            // ValidatingAdmissionWebhook
   resourcequota.PluginName,                // ResourceQuota
   storageobjectinuseprotection.PluginName, // StorageObjectInUseProtection
   podpriority.PluginName,                  // PodPriority
   nodetaint.PluginName,                    // TaintNodesByCondition
   runtimeclass.PluginName,                 // RuntimeClass
```

```
    certapproval.PluginName,                // CertificateApproval
    certsigning.PluginName,                 // CertificateSigning
    certsubjectrestriction.PluginName,      // CertificateSubjectRestriction
    defaultingressclass.PluginName,         // DefaultIngressClass
    podsecurity.PluginName,                 // PodSecurity
)
```

Default enabled admission plugins in the kube-apiserver

All the other plugins not mentioned in the `defaultOnPlugins` list are regarded as plugins that are disabled by default.

We can enable the desired admission plugins with the flag `--enable-admission-plugins` and disable those with the flag `--disable-admission-plugins` when starting the `kube-apiserver`. The value takes a comma-delimited list.

Back

Quiz on Customizing AuthX

Next

Validating Admission Webhooks