


KubeCon | CloudNativeCon

Europe 2020

Virtual

A Journey Through Kubernetes Admission Controller Taxonomy

Morgan Bauer
Srini Brahmaroutu

Agenda

- History
- Built-in
- Webhook
- Advice

What are Admission Controllers?

- Abstract
  - A way to validate or modify incoming objects before they are persisted
- Technical
  - Implement ValidationInterface or MutationInterface from apiserver/pkg/admission
  - Takes in AdmissionReview and outputs AdmissionResponse

History

- v0.9 Separate object validation as pluggable interface
- v1.4 first webhooks
- v1.9 Generic Webhook with Mutation

- Everything today is v1 from k8s v1.16 onward

Webhook Types

	Validating	Mutating
Built-in	**alwayspullimages**	**podpreset**
Webhook	**imagepolicywebhook**	**None provided, but integration-tested**

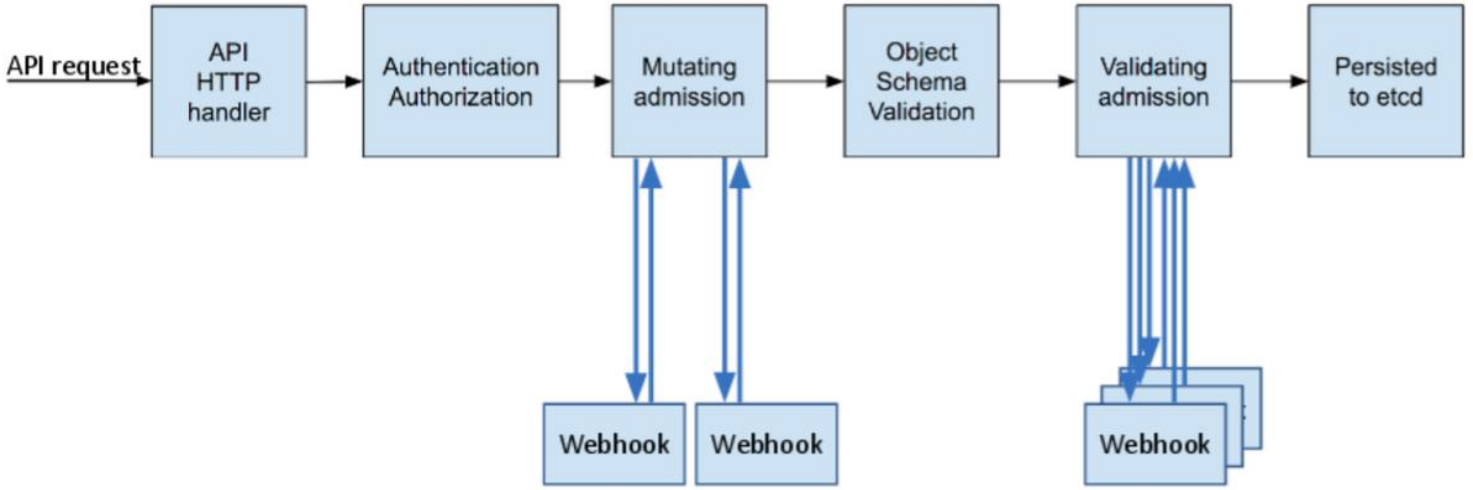
What are Admission Controllers?

- An interceptor to Kube API server calls / not a Kubernetes controller
- Admission Control Webhooks are the controllers baked into API Server
- MutatingAdmissionWebhook and ValidatingAdmissionWebhook are special controllers that are compiled into API Server
- Many, many static admission controllers compiled into API Server
- Admission controllers limit requests to create, delete, modify or connect to (proxy). They do not support read requests.

Request lifecycle

Close to 20 default admission controllers to chose

- --enable-admission-plugins and --disable-admission-plugins
- Life of a request :



Why we need Admission Controllers?

Main functions are : Security, Governance, Configuration Management

- DefaultStorageClass : allow to create storage with default storageclass
- LimitRanger : allow to set default limits to cpu/mem resources within NS
- AlwaysPullImages : protect private images in a multi-tenant env
- NamespaceLifeCycle : avoid adding new object to NS that is being deleted
- PersistentVolumeClaimResize : prevents resizing of PVC unless storageclass allows by setting allowVolumeExpansion to true
- Serviceaccount : automate service account
- TaintNodesByCondition: Set nodes to NotReady/NoSchedule until nodes come up and set their reported conditions
- DefaultTolerationSeconds : set default toleration limits to 5 min when node not ready
- StrorageObjectInUseProtection : protects PV/PVC from deletion
- RuntimeClass : set a pod overhead so that scheduler picks proper node that meets the resource requirements
- ResourceQuota : enforce quota constraints
- CertificateApproval, CertificateSigning, CertificateSubjectRestriction : Certificate management controllers

Validating Admission Webhook

- Find matching webhooks for the request
- Execute the webhooks parallelly
- If ANY fails, reject request
- No mutation can occur

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: "pod-policy.example.com"
webhooks:
- name: "pod-policy.example.com"
  rules:
  - apiGroups: [""]
    apiVersions: ["v1"]
    operations: ["CREATE"]
    resources: ["pods"]
    scope: "Namespaced"
  clientConfig:
    service:
      namespace: "example-namespace"
      name: "example-service"
    caBundle: "Ci0tLS0tQk...<`caBundle` is a PEM encoded CA bundle which will be used to validate the webhook's server certificate.>...tLS0K"
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 5
```

```
objectSelector :
  matchLabels:
      label: testSignature
```

Mutating Admission Webhook

- Matching mutating webhooks called serially
- Idempotence : a webhook can be called more than once
- Ordering : not guaranteed between all mutating webhooks
- Matching Requests: enforce exact or equivalent matching
- Timeouts : to manage webhook latency to control API latency
- Mutating-Validating webhooks should work together
- Side-effects - reconcile
  - should undo any changes in the cluster during failures or dry run
- NameSpaceSelector : use to avoid system namespaces
- Reinvocation Policy : controls whether there are multiple calls

Pod Presets

- inject information into pods at creation time
  - Ex: secrets, volumes, volume mounts, environment variables
- apply by using labels on the Pod
- upon pod creation request –
  chain of pod presets checked for label match (n-n)
- if the resource defined by pod preset fails to merge,
  pod gets started without the merge
- annotate all pod preset merges
- All containers or a pod spec can get affected
  by the merge
- Ability to disable pod presets to a pod
  - podpreset.admission.kubernetes.io/exclude: "true"

```yaml
apiVersion: settings.k8s.io/v1alpha1
kind: PodPreset
metadata:
  name: allow-database
spec:
  selector:
    matchLabels:
      role: frontend
  env:
    - name: DB_PORT
      value: "6379"
  volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
    - name: cache-volume
      emptyDir: {}
```

Built-in Admission Controllers

- At kubernetes/plugin/pkg/**admission**/
- Compiled into apiserver
- Defaults : pkg/kubeapiserver/options/plugins.go

Mutating Vs Validating – Built In

- Validating – yes or no, proceed or don't
  - Implements ValidationInterface
- Mutating – change the object
  - Implements MutationInterface
  - You get a pointer to the same object that will be persisted, thus any changes you make will get to the storage layer

Mutating Vs Validating – Webhook

- Validating
  - AdmissionResponse with Allowed field set
- Mutating
  - AdmissionResponse with a jsonpatch
  - Describes desired change to the object

Demo

Pitfalls and Idiomatic

- Webhooks with side effects
- Security aspects
- Failure scenarios
  - Ignore or fail if webhook is not contactable
- Multi-language
- Chaining does not guarantee order
- Idempotency
- Maintenance


KubeCon
CloudNativeCon
Europe 2020
Virtual
HELM
KEEP CLOUD NATIVE
CONNECTED