Kubernetes Admission Controller

kubernetes

Ali Akil

"The future belongs to those who believe in ~~the beauty of their dreams~~ enforcing policies for Kubernetes workloads." - Eleanor Kubevelt


ADMISSION CONTROLLER
POLICY ENGINE
imgflip.com

Policies? Why?

- Enhanced Security
  - Ensuring compliance with security best practices and preventing misconfigurations
- Resource Management
  - Efficiently allocate and regulate resources
- Consistency
  - Maintaining standardization and reduces the likelihood of misconfigurations
- Compliance and Governance
  - Ensuring alignment with both external regulations and internal organizational policies
- Automation
  - Automatically modifying resource configurations or generate complementary resources to meet compliance requirements

Kubernetes Policy Types

Validate

- Disallow Privileged Containers
- Restrict Image Registries
- Require limits and requests
- Require network policies

Mutate

- Add Default Resources
- Add emptyDir sizeLimit
- Add Labels
- Add Default securityContext

Generate

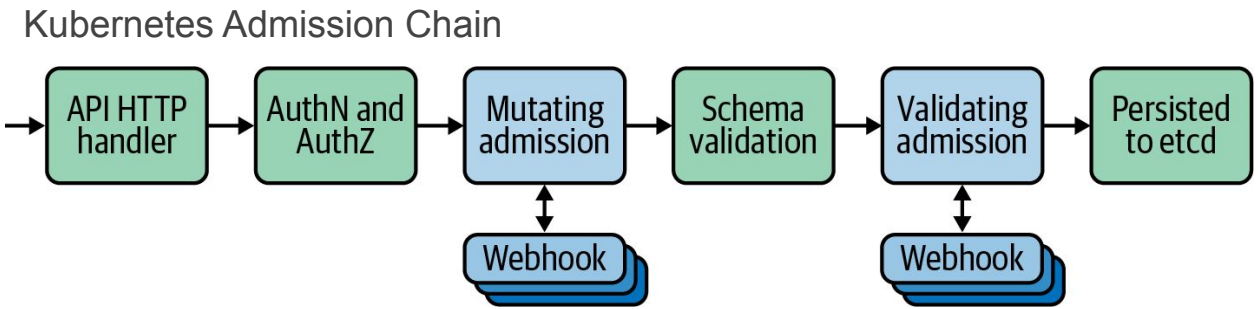
- Add Network Policy
- Add Pod Disruption Budget
- Add Namespace Quota

Kubernetes Admission Chain

Ali Akil

Admission Controllers

Validating and mutating requests to the Kubernetes API server before they are persisted in etcd




Mutating admission
Validating admission
imgflip.com
JAKE-CLARK.TUMBLR

Mutating Admission Controller

- These controllers are able to generate and modify the resource attributes before they pass onto future phases
- Ordering is important
    - Setting fields in mutating admission controllers before the object is validated
- Service Account controller (Built in and enabled by default)
    - Generation example
        - When a namespace is created the service account controller and the token controller the default service account the corresponding secret
    - Mutation example:
        - When a Pod is submitted, the Service Account controller inspects the Pod's spec to ensure that it has the serviceAccount (SA) field set.
        - If not, then it adds the field and sets it to the default SA for the Namespace.
        - It also adds ImagePullSecrets and a Volume

kubernetes

Validating Admission Controller

- Final stage before the object is persisted to etcd is for it to pass through validating admission controllers
- These can be built-in controllers or calls to external (out-of-tree) validating webhooks
- Only able to admit or reject the request, not modify the payload
- Differ from the prior schema validation step in that they are concerned with validating against operational logic, not a standardized schema

In-tree Admission controllers

In-tree Admission controllers

Default admission controllers

- DefaultIngressClass
- DefaultStorageClass
- LimitRanger
- NamespaceLifecycle
- ResourceQuota
- RuntimeClass
- ServiceAccount
- …

Non-Default admission controllers

- PodNodeSelector
- AlwaysPullImages
- NamespaceExists
- …

kubernetes

Webhooks

Webhooks

- Allows configuring the Kubernetes API server to send API requests to external webhook endpoints and receive decisions.
- Two types of admission webhooks, validating admission webhook and mutating admission webhook.
- The receiving web server can be written in any language that can expose an HTTPS listener
- Can be run in or out of cluster
    - Utilizing in-cluster execution by leveraging the discovery and operator primitives available.
    - Implementing reusable functionality in serverless functions for out-of-cluster execution

kubernetes

Writing Webhooks

Ali Akil

Plain HTTPS Handler

Implement webhooks from scratch

+ Provides the most flexibility in terms of language choices and deployment options to integrate with the current stacks
- Requires more domain knowledge
- Limited functionality and features out of the box
- Harder to maintain and troubleshoot over time

Controller Runtime

Controller-runtime is a subproject of Kubebuilder

- Set of Go libraries/SDK for building controllers
- \+ Simplifies building admission controllers by minimizing boilerplate and allowing developers to focus on implementing the desired logic
- \- Learning curve
- \- Limited language support

Example

```yaml
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: <CA_BUNDLE>
  admissionReviewVersions: ["v1"]
  sideEffects: None
  timeoutSeconds: 5
```

Centralized Policy Systems

- Policy as a Code
- Centralizing policy logic into one place and using standardized languages to express the allow/deny rules
- Components:
  - A policy language that can express conditions on whether an object should be admitted or rejected.
  - A controller that sits in the cluster serving as an admission controller

\+ Programming knowledge is not required to create admission controllers

  - Changes to logic do not require rebuilding and redeploying the controller each time

\+ Policies and rules are stored in a single location for viewing, editing, and auditing

\+ Policy catalog

\- Learning configuration language

kubernetes

Gatekeeper

- Kubernetes-specific implementation of a lower-level tool called Open Policy Agent (general-purpose policy engine)
- Uses a specialized programming language (Rego) in order to implement the logic necessary for policy decisions

\+ Capable of expressing very complex policy

\+ Multiple replicas for scaling and availability are supported

\+ Extensible ConstraintTemplate model

\- learning curve of new programming language (Rego)

\- No generation ability

Example: Disallow Privileged Containers

Constraints template

```
spec:
  crd:
    spec:
      names:
        kind: K8sPSPPrivilegedContainer
      validation:
        openAPIV3Schema:
          type: object
          description: >-
            Controls the ability of any container to enable privileged mode.
            Corresponds to the `privileged` field in a PodSecurityPolicy. For more
            information, see
            https://kubernetes.io/docs/concepts/policy/pod-security-policy/#privileged
          properties:
            exemptImages:
              description: >-
                Any container that uses an image that matches an entry in this list will be excluded
                from enforcement. Prefix-matching can be signified with `*`. For example: `my-image-*`.

                It is recommended that users use the fully-qualified Docker image name (e.g. start with a domain name)
                in order to avoid unexpectedly exempting images from an untrusted repository.
              type: array
              items:
                type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8spspprivileged

        import data.lib.exempt_container.is_exempt

        violation[{"msg": msg, "details": {}}] {
            c := input_containers[_]
            not is_exempt(c)
            c.securityContext.privileged
            msg := sprintf("Privileged container is not allowed: %v, securityContext: %v", [c.name, c.securityContext])
        }
```

Constraint

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPPrivilegedContainer
metadata:
  name: psp-privileged-container
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
    excludedNamespaces: ["kube-system"]
```

Kyverno 

- Open-source Kubernetes policy-engine originally from Nirmata and later donated to the CNCF
- Utilizes yaml to implement the logic necessary for policy decisions

\+ Validation, mutation, image verification and resource generation abilities

\+ Does not require knowledge of a specialized programming language

\- Highly complex policy may not be possible since no programming language is exposed

\- Less mature than OPA but still has a fast-growing community

Example: Disallow Privileged Containers

Cluster Policy

```yaml
spec:
  validationFailureAction: audit
  background: true
  rules:
    - name: privileged-containers
      match:
        any:
        - resources:
            kinds:
              - Pod
      validate:
        message: >-
          Privileged mode is disallowed. The fields spec.containers[*].securityContext.privileged
          and spec.initContainers[*].securityContext.privileged must be unset or set to `false`.
        pattern:
          spec:
            =(ephemeralContainers):
              - =(securityContext):
                  =(privileged): "false"
            =(initContainers):
              - =(securityContext):
                  =(privileged): "false"
            containers:
              - =(securityContext):
                  =(privileged): "false"
```

Other tools

-  Kubewarden https://github.com/kubewarden
-  Datree https://github.com/datreeio

Thanks!

kubernetes

"Two things are infinite: the Kubernetes ecosystem and the number of times you'll say, 'Who let that pod through?' " - Albert Kubestein