Kyverno Admission Controllers

Kyverno is a powerful policy engine specifically designed for Kubernetes. The term "Kyverno" originates from the Greek word that translates to "govern."

Kyverno can be used to define and enforce policies, providing a framework for managing various aspects of the kubernetes cluster.

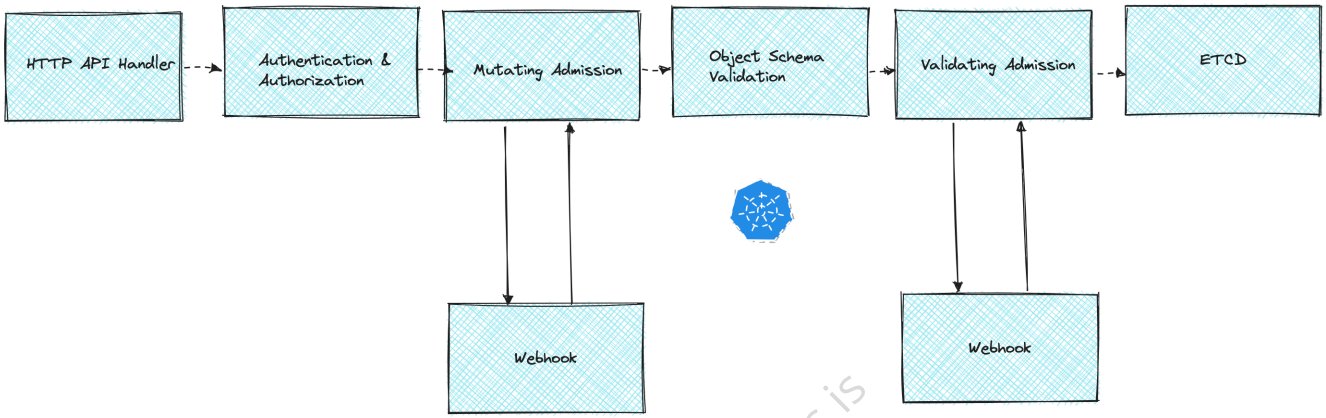
It serves three primary functions:

- Validation: Kyverno can be used to validate configurations, checking whether they comply with the defined policies before they are deployed.
- Mutation: Kyverno can be used to mutate or modify configurations based on certain policies. This can help in standardizing configurations or automatically introducing certain settings or changes as per your requirements.
- Generation: Kyverno can even generate new configurations based on the policies defined. This allows to automatically produce necessary configurations, reducing manual effort and the chance of errors.

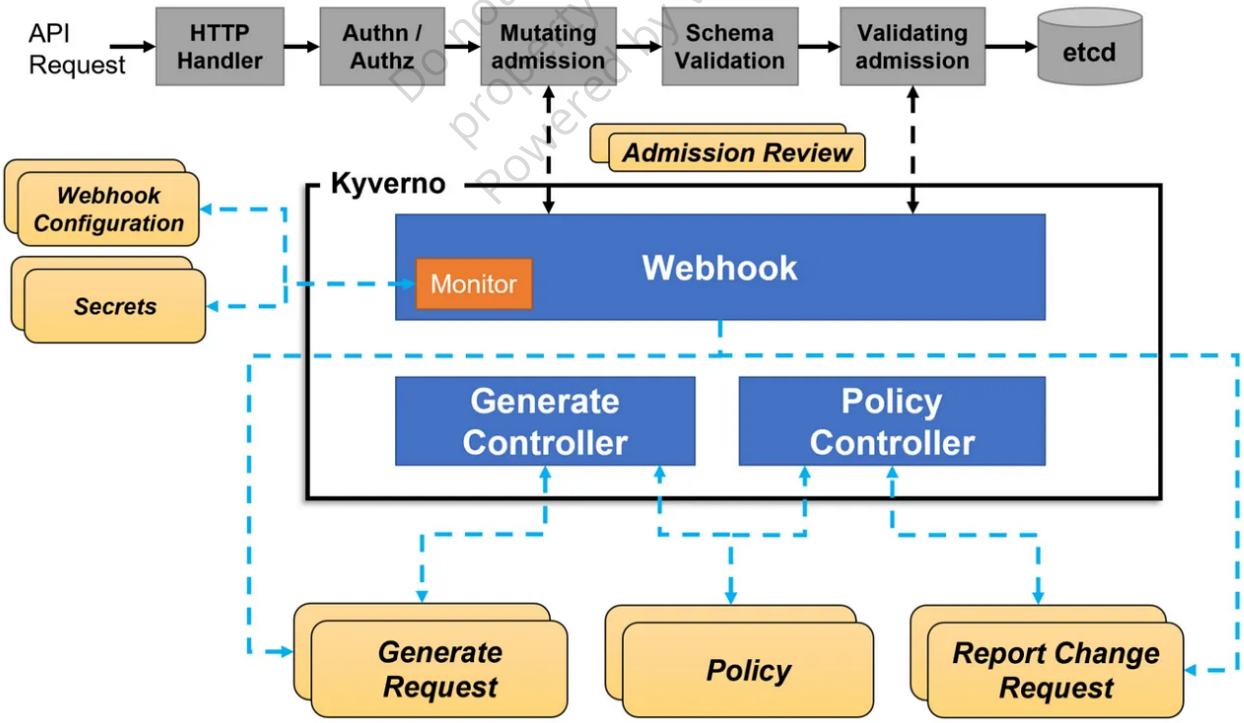
A Quick Intro on Admission Controller An admission controller is a piece of code that intercepts requests to the Kubernetes API server before the persistence(storing in ETCD) of the object, but after the request is authenticated and authorized(RBAC). It can be regarded as checkpoints for Kubernetes API requests and can completely deny/accept the requests or change the request object altogether.

Kubernetes also has some Dynamic Admission Controllers like ValidatingAdmissionWebhooks and MutatingAdmissionWebhooks, which can validate and mutate the requests sent by the user.

The following diagram below shows how the admission controller works after authorization/authentication and before persisting it to etcd.



Kyverno Working

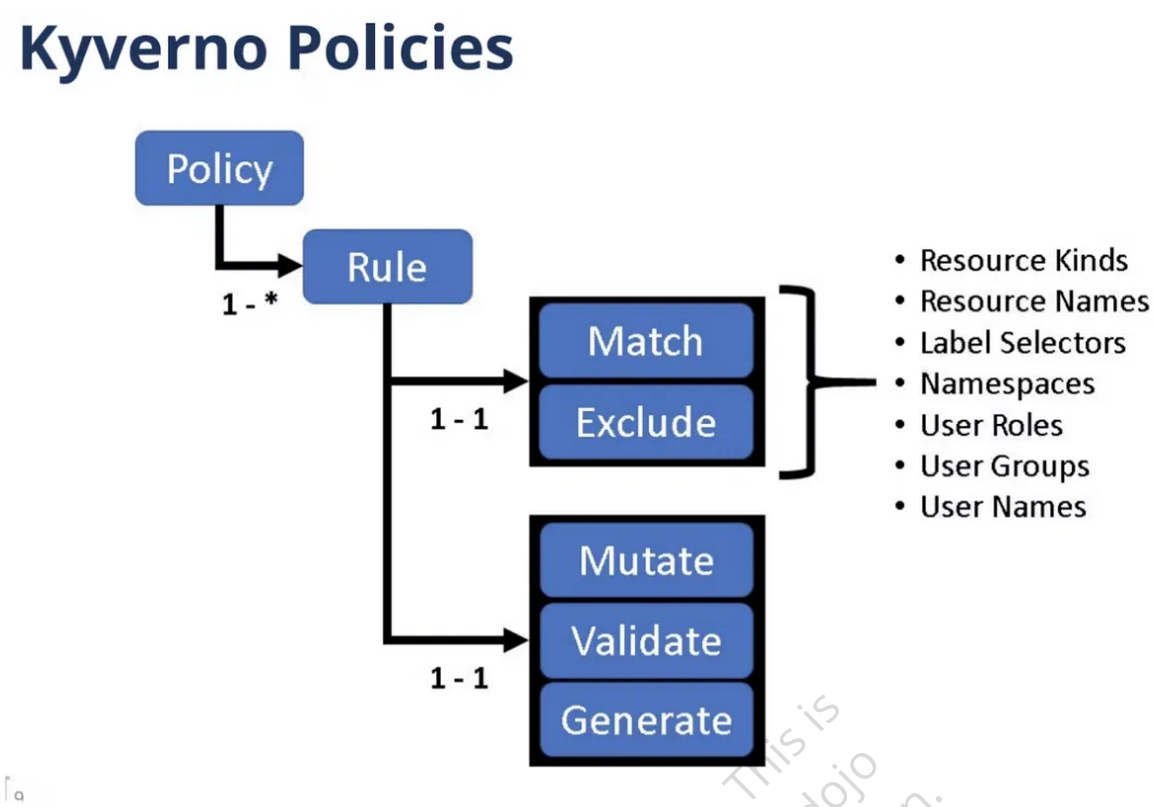


Kyverno architecture — photo from Kyverno.io

Kyverno is a tool that operates similarly to other admission webhooks utilized in Kubernetes. Its function is to manage policy enforcement within your Kubernetes environment.

- A new Kubernetes event occurs, such as the creation of a namespace, a pod, a resource, a new RBAC, etc., and the kube-api server processes the request.
- When the kube-api server is called, Kyverno then receives an AdmissionReview Request.
- Kyverno has the ability to modify the API request in accordance with the CRD policies which are created.
- After mutating the policies, Kyverno waits for the CRD schema to be validated before using the outcome to update the policies.
- Kyverno generates resources, logs events, and produces Policy Reports based on the policy.
- The AdmissionReview Response is passed on to the etcd if the AdmissionReview Request is legitimate and in accordance with all policies.
- Kyverno stops the creation of resources if the AdmissionReview Request is invalid with the mode set to Enforce and it does not respond to the etcd.
- Kyerno logs the event and sends a relevant response to the etcd if the AdmissionReview Request is invalid with the mode set to Audit.

Kyverno Policy?



A Kyverno policy is a set of rules that uses the webhook to validate the API requests we get from the Kube API server.

These rules are managed as ClusterPolicies as a CustomResourceDefinition(CRD).

- Kyverno policy is made up of several distinct rules.
- Rules: Each Kyverno policy contains multiple rules.
- Match and Exclude Clauses: Each rule has a 'match' and an 'exclude' clause. 'Match' identifies which resources the policy applies to, while 'Exclude' determines the resources the policy should ignore.

These match and exclude operations can be fine-tuned based on resource types, names, label selectors, namespaces, user names, roles, or groups.

- Admission Requests Mapping: Kyverno can automatically associate user and subject information from admission requests with specific groups configured in your cluster. This feature assists in the precise matching and exclusion of roles and groups.
- Rule Sections: Each rule includes three sections - 'Mutate', 'Validate', and 'Generate'.
    - Validate: This section checks a specified property of a resource. The resource is

only created if it meets the criteria defined in the rule.

- Mutate: This section allows Kyverno to alter the matching resources as needed.
- Generate: This section permits Kyverno to produce additional resources as required.

The below example of a Kyverno Policy could provide a better understanding of the policy structure.

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-run-as-nonroot
  annotations:
    policies.kyverno.io/title: Require runAsNonRoot
    policies.kyverno.io/category: Pod Security Standards(Restricted)
    policies.kyverno.io/severity: medium
    policies.kyverno.io/subject: Pod
    kyverno.io/kyverno-version: 1.6.0
    kyverno.io/kubernetes-version: "1.22-1.23"
    policies.kyverno.io/description: >-
      Containers must be required to run as non-root users.    Thispolicy
ensures`runAsNonRoot` is set to `true`. A known issue prevents a policy such as
this using `anyPattern` from being persisted properly in Kubernetes
1.23.0-1.23.2.
spec:
  validationFailureAction: enforce
  background: true
  rules:
    - name: run-as-non-root
      match:
        any:
        - resources:
            kinds:
             - Pod
      validate:
        message: >-
          Running as root is not allowed. Either the field
spec.securityContext.runAsNonRoot must be set to `true`, or the
fieldsspec.containers[*].securityContext.runAsNonRoot,spec.initContainers[*].se
curityContext.runAsNonRoot,
andspec.ephemeralContainers[*].securityContext.runAsNonRoot must be set to
`true`.
        anyPattern:
        - spec:
            securityContext:
              runAsNonRoot: true
            =(ephemeralContainers):
            - =(securityContext):
                =(runAsNonRoot): true
            =(initContainers):
            - =(securityContext):
                =(runAsNonRoot): true
            containers:
            - =(securityContext):
                =(runAsNonRoot): true
```


Policy Structure

- API Version and Kind: The apiVersion denotes the version of Kyverno, and kind indicates that this is a ClusterPolicy.
- Metadata: This includes details like the name of the policy and various annotations that provide additional information about the policy such as its title, category, severity, the subject it applies to, Kyverno version, Kubernetes version, and a description.
- Spec: This outlines the specifications of the policy.
- validationFailureAction: This parameter is set to 'enforce', which means any violations of this policy will result in the request being blocked.
- background: This is set to true, enabling Kyverno to perform background scans on existing resources within the cluster.
- Rules: This includes the rules that are applied to the resources.
- The match clause here is configured to apply this rule to all resources of kind "Pod".
- The validate section contains the key components of the rule:
- Message: The message that will be displayed if a policy violation occurs.
- AnyPattern: This YAML component uses a logical OR operator for pattern matching in the specs.
- Spec: This section uses AnyPattern to check if securityContext within the pod or container configuration is set to runAsNonRoot: true. If it isn't, the provided pod configuration would be flagged as a violation.

In a nutshell, this policy ensures that pods and containers in your Kubernetes cluster are not running as root, enhancing the security posture of your cluster.

Reference: tech.groww.in by Anoop Ka