Policy Management with Kyverno



Marcel Müller
Platform Engineer @ Giant Swarm
Kyverno Maintainer
@muemarcel

A simple example: Requests and Limits

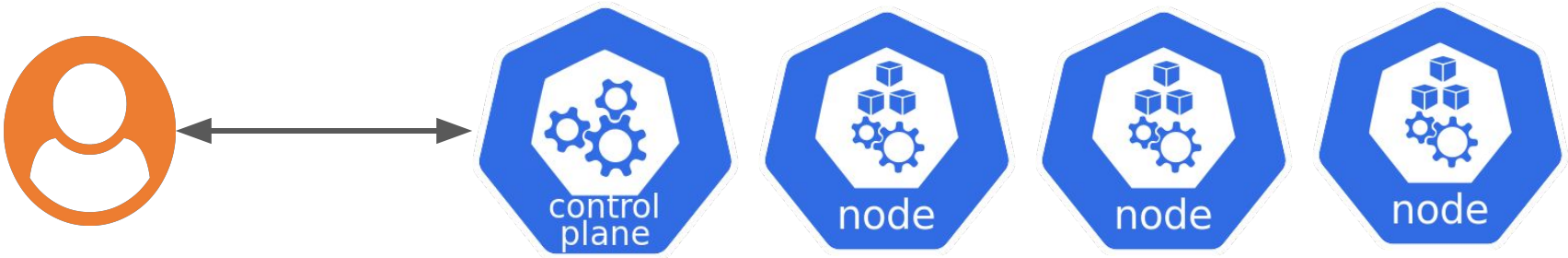
Requests and Limits should be set on all pods!

- Creating a Pod without a requests and limits shouldn’t be allowed!

- Kubernetes should be responsible for enforcement!

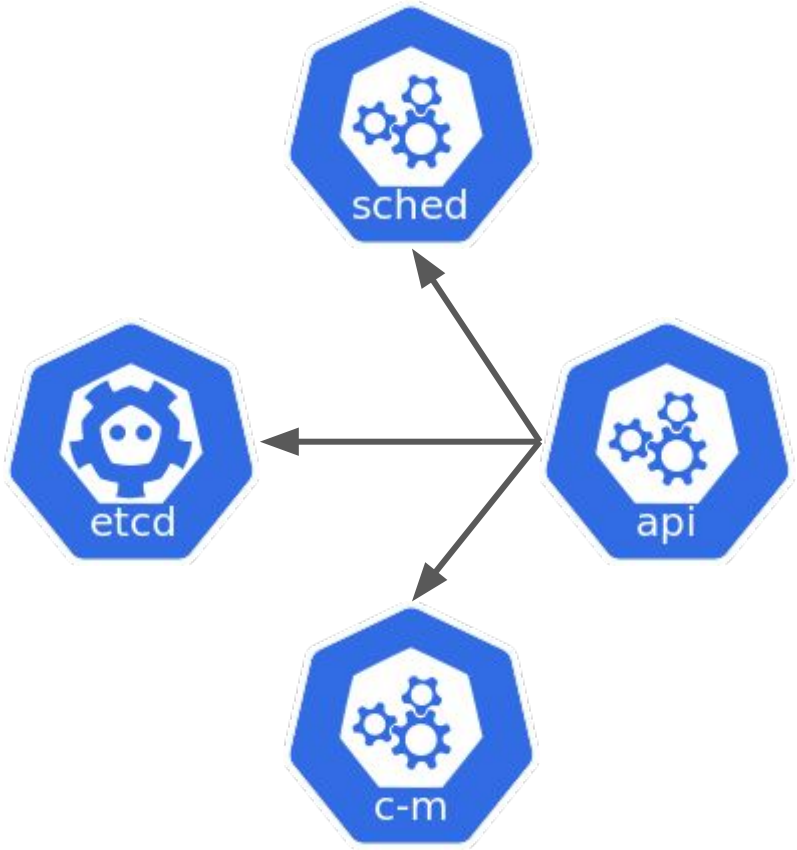
What is the Kubernetes native solution for this problem?

Understanding the basics



Every request to Kubernetes reaches the control plane first!

The Kubernetes Control Plane

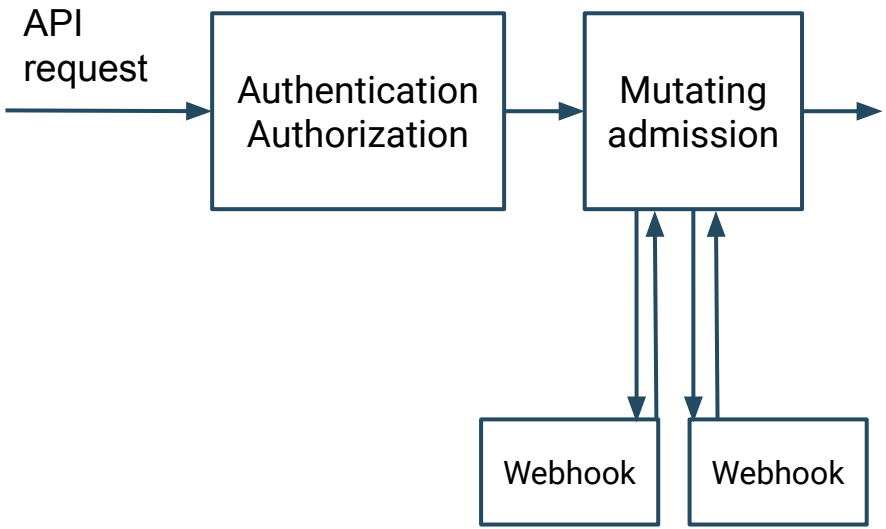


- Every resource requests has to go “through” the API
- Persisted in ETCD

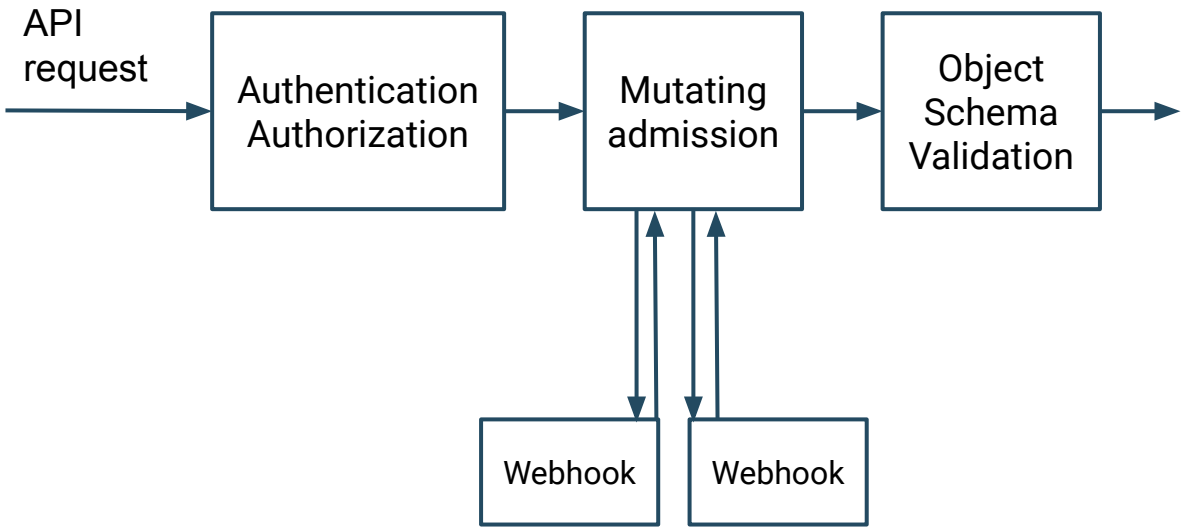
-> The API is responsible for also denying requests!

Giant Swarm

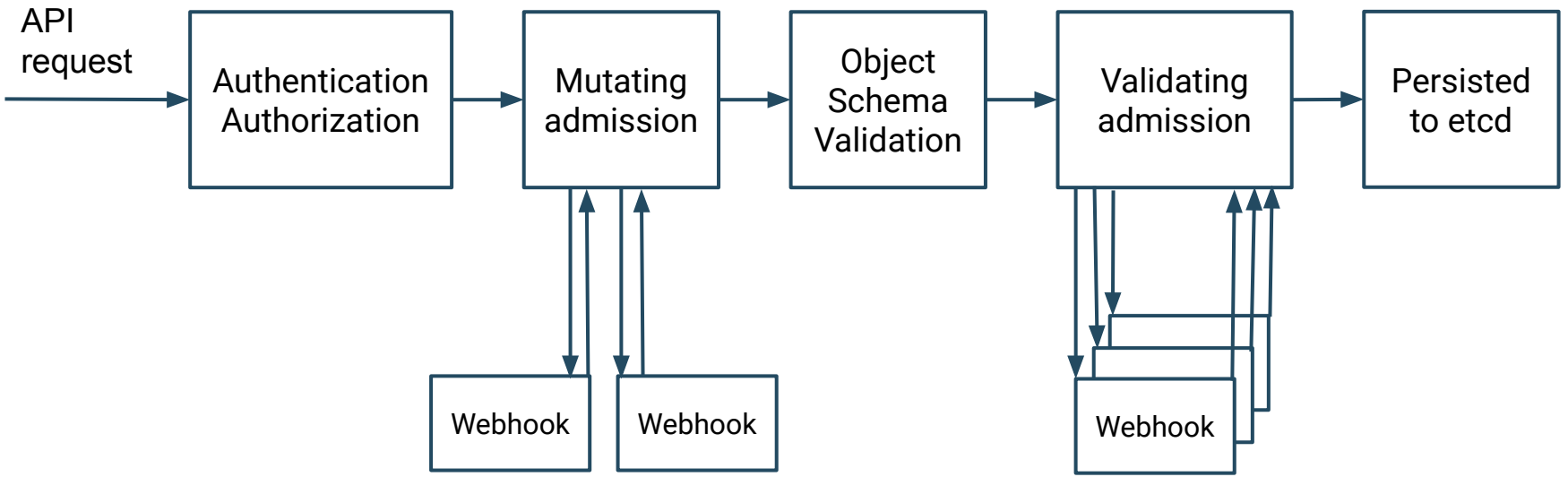
Request flow in the Kubernetes API



Request flow in the Kubernetes API



Request flow in the Kubernetes API



Registration of Admission Webhooks

Webhooks have to be registered **declaratively** in the Kubernetes API:

- ValidatingWebhookConfiguration
- MutatingWebhookConfiguration

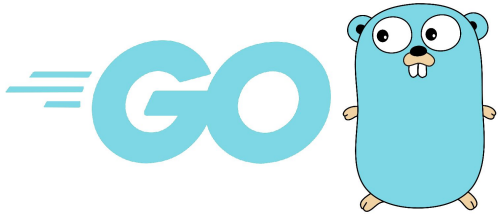
The configuration contains a reference to a service, which has to implement the actual logic.

Example admission webhook config

```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: "..."
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 5
```

Implementing admission webhooks yourself?





- Can be implemented in any programming language!

- Generally good library support for popular languages.

-> Not simple to implement and maintain!

Policy Management tools in Kubernetes





- Open Policy Agent Gatekeeper
  - Uses rego for policies
  - Mainly used for validation policies

- Kyverno
  - Kubernetes native Policy Management
  - Validation, Mutation and Generation use-cases

Obviously we focus on **Kyverno** here :)

What are the goals for Kyverno?

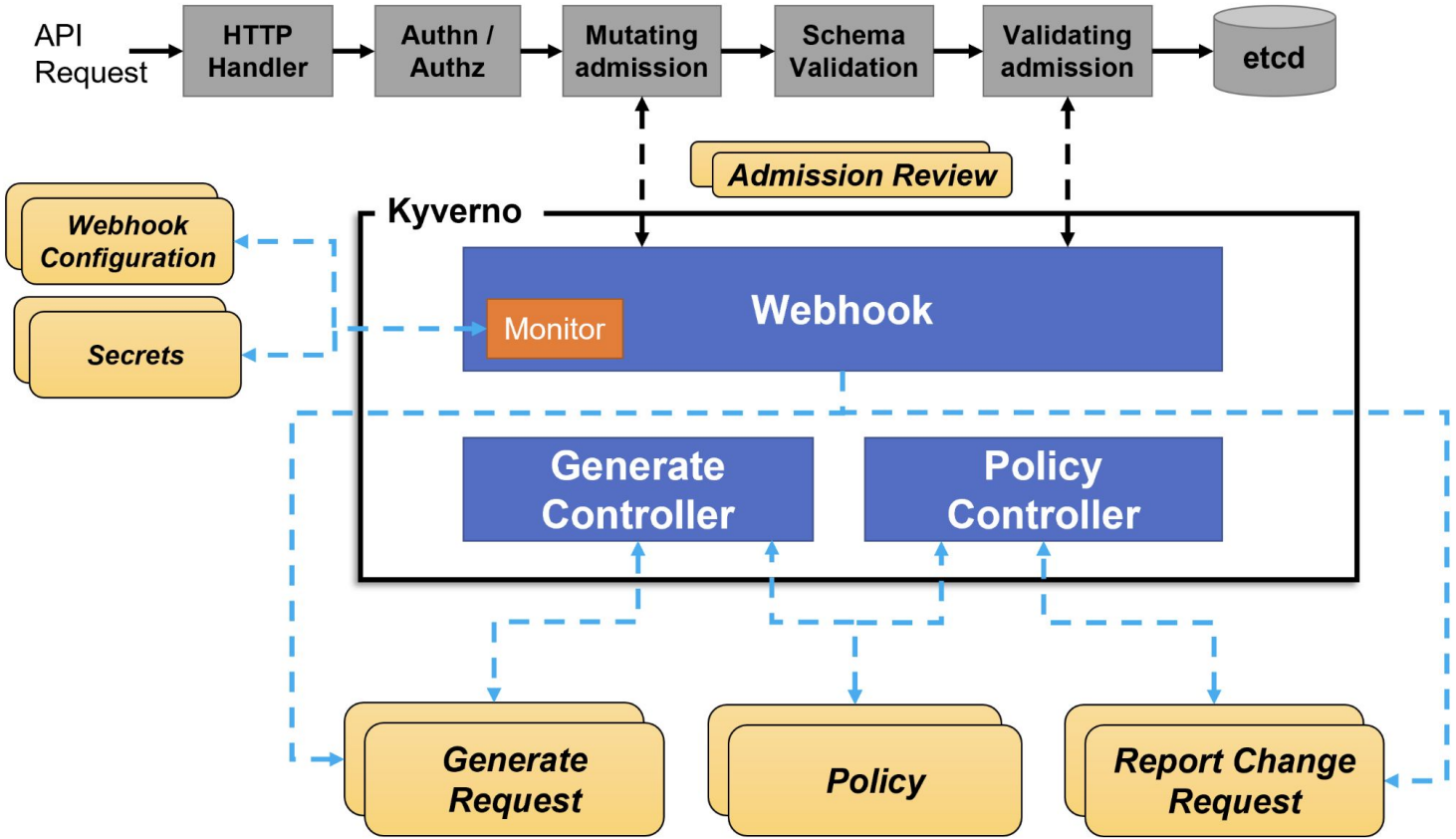
- Represent policies as Kubernetes resources

- Simple mechanisms for validation and mutation
  - Not custom programming language
  - Representation in YAML

- Coverage of Kubernetes specific use-cases
  - Not applicable outside of Kubernetes



Kyverno Architecture



Requests & Limits - Policy Example

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-requests-limits
spec:
```

Requests & Limits - Policy Example

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-requests-limits
spec:
  rules:
  - name: validate-resources
    match:
      resources:
        kinds:
        - Pod
```

Requests & Limits - Policy Example

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-requests-limits
spec:
  rules:
  - name: validate-resources
    match:
      resources:
        kinds:
        - Pod
    validate:
      message: "CPU and memory resource requests and limits are required."
      pattern:
        spec:
          containers:
          - resources:
              requests:
                memory: "?*"
                cpu: "?*"
              limits:
                memory: "?*"
```

But what does this look like in real life?!



Demo time!

Existing Kyverno Policies

https://github.com/kyverno/policies

Kyverno policies are easily shareable as they are custom resources.

Main collection areas:

- Best practices
- Security

Conclusion: Kyverno

+ Reading and creating Policies is simple
  - Complex Policies use JMESPath

+ Validation, mutation and generation policies can be implemented

+ Metrics and audit reports are built in

- Complex policies are hard / impossible to implement

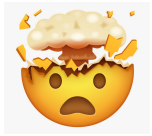
We are hiring!

Questions?

Twitter: @MueMarcel

E-Mail: marcel@giantswarm.io

Giant Swarm

GitOps + Webhooks = 

- Besonders bei Mutation + Generation aufpassen!

- GitOps tooling wird **wahrscheinlich** versuchen Änderung rückgängig zu machen!

- GitOps tools haben oft verschiedene Wege Felder auszuschließen
  - Dennoch ein Antipattern (meiner Meinung nach)

- Kubernetes unterstützt **server side dry runs**!
  - In kubectl *--server-dry-run*
  - Webhooks können angewendet werden bevor files in Git kommen