SIGHUP

Tu non puoi passare!

Policy compliance con OPA Gatekeeper

KCD ITALY 2021

Niccolò Raspa - *Kubernetes Engineer @ SIGHUP*

KCD Italy 🇮🇹 Nov 17-18, 2021

$whoami

👋 Ciao

My name is Niccolò and I work at SIGHUP as a Kubernetes Engineer.

linkedin.com/in/niccolo-raspa/

@niccoloraspa

Agenda

Agenda

The Balrog and policy compliance

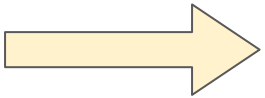
The bridge of Khazad-dûm and the road to ETCD

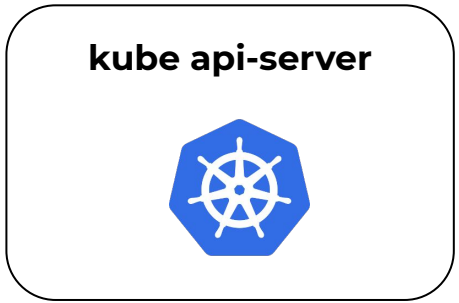
Gandalf and Opa Gatekeeper

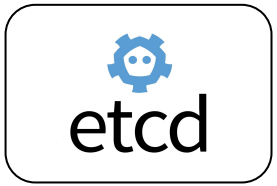
The fall and the demo

What is a Balrog?

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: balrog
spec:
  securityContext:
    runAsUser: 0
    privileged: True
    capabalities:
      add: ["CAP_SYS_BOOT"]
  containers:
  - name: busybox
    image: evil.registry.io/balrog/busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: private-volume
      mountPath: /private
  volumes:
  - name: private-volume
    hostPath:
      path: /private
      type: Directory
```

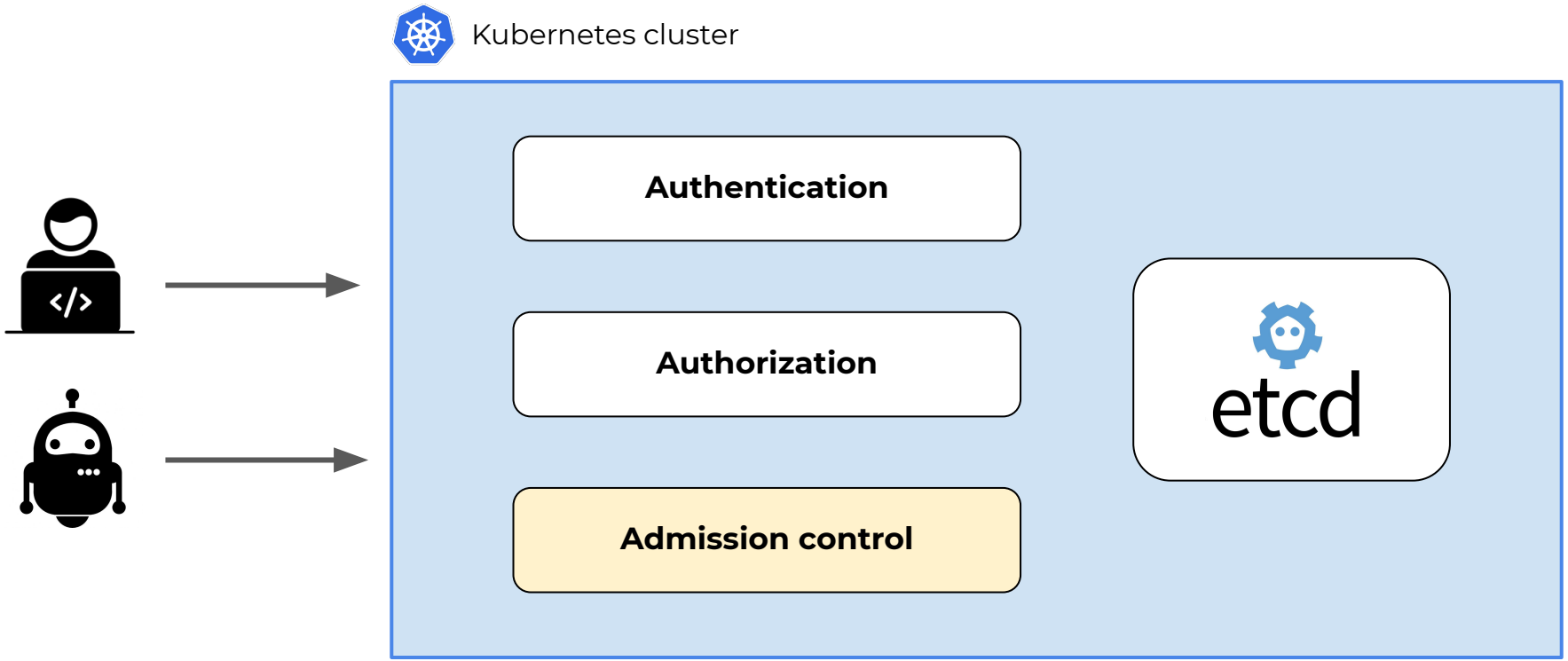






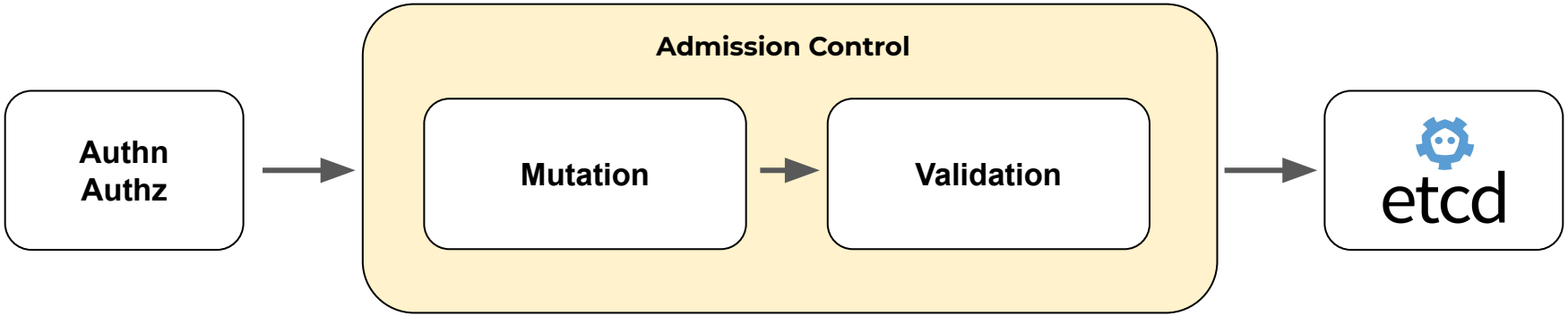
Policy and compliance

The road to etcd



Admission Control

An admission controller is a piece of code that processes requests to the Kubernetes API server prior to persistence of the object, but after the request is authenticated and authorized.



Admission Control

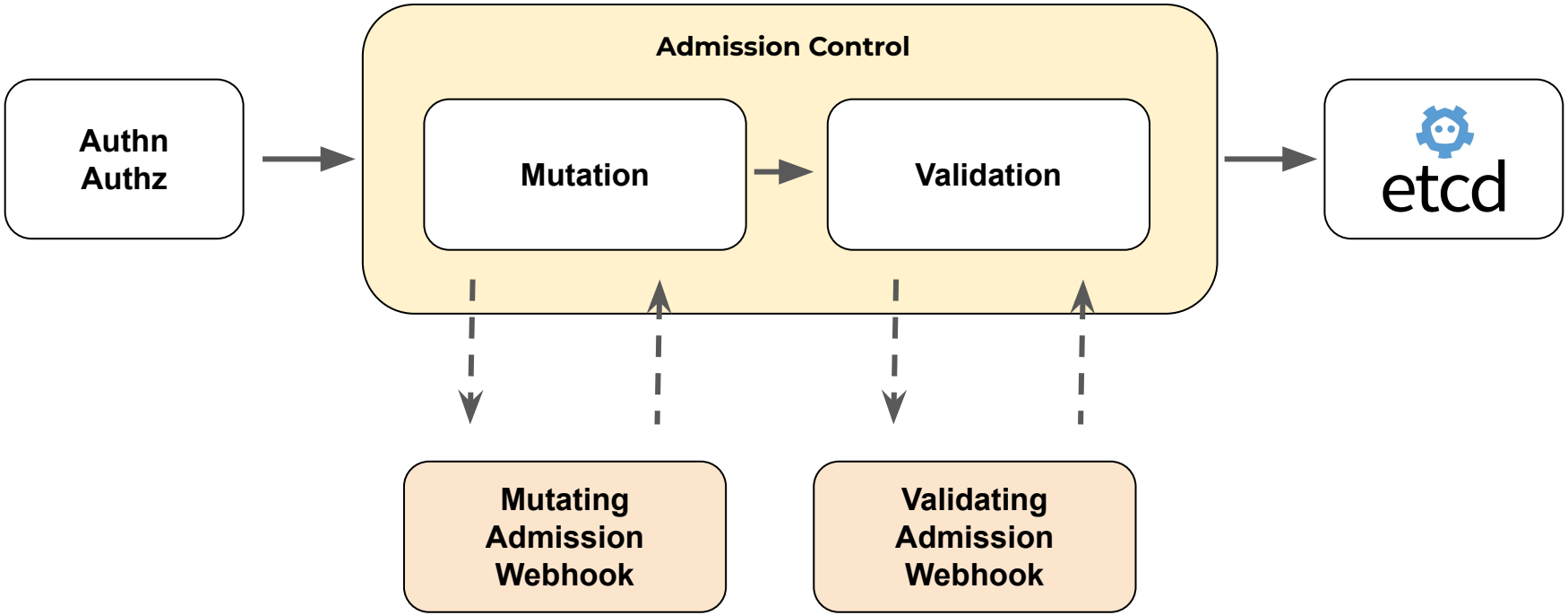
You are already using them:

```
kube-apiserver -h | grep enable-admission-plugins
```

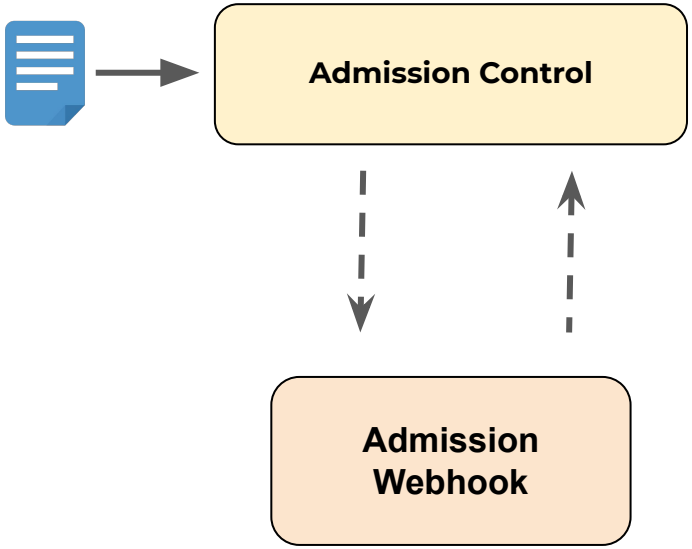
--enable-admission-plugins strings

admission plugins that should be enabled in addition to default enabled ones (NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota). Comma-delimited list of admission plugins: AlwaysAdmit, AlwaysDeny, AlwaysPullImages, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, DenyServiceExternalIPs, EventRateLimit, ExtendedResourceToleration, ImagePolicyWebhook, LimitPodHardAntiAffinityTopology, LimitRanger, MutatingAdmissionWebhook, NamespaceAutoProvision, NamespaceExists, NamespaceLifecycle, NodeRestriction, OwnerReferencesPermissionEnforcement, PersistentVolumeClaimResize, PersistentVolumeLabel, PodNodeSelector, PodSecurity, PodSecurityPolicy, PodTolerationRestriction, Priority, ResourceQuota, RuntimeClass, SecurityContextDeny, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionWebhook. The order of plugins in this flag does not matter.

Admission Webhooks

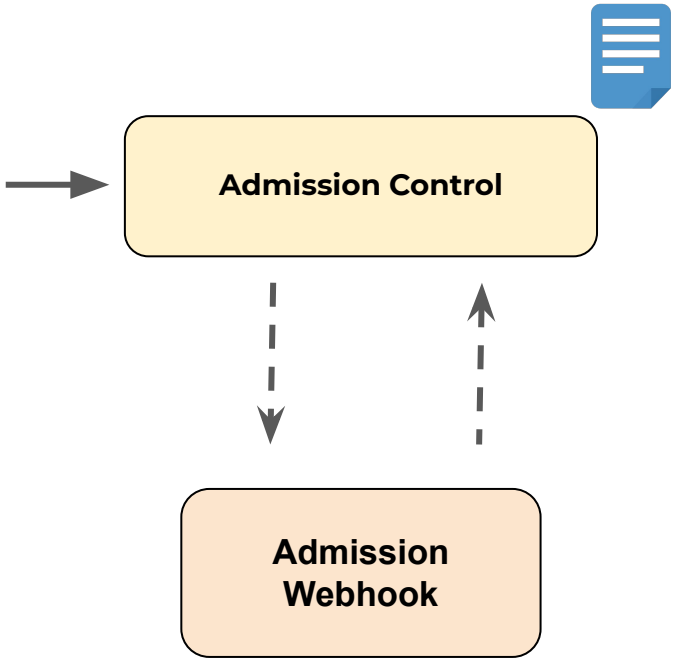


Admission Webhooks Flow



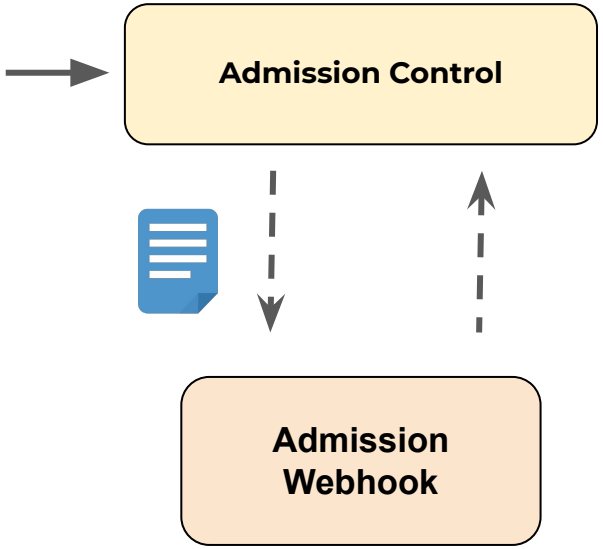

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: myapp
spec:
  containers:
  - image: nginx
    name: nginx-frontend
  - image: mysql
    name: mysql-backend
```

Admission Webhooks Flow



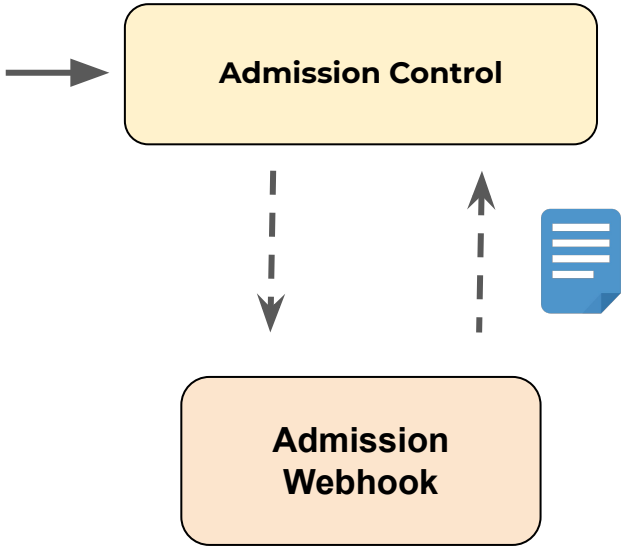

```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: "pod-policy.example.com"
webhooks:
- name: "pod-policy.example.com"
  rules:
  - apiGroups:   [""]
    apiVersions: ["v1"]
    operations:  ["CREATE"]
    resources:   ["pods"]
    scope:       "Namespaced"
  clientConfig:
    service:
      namespace: "my-namespace"
      name: "my-admission-webhook-svc"
    caBundle: "Ci0tLS0tQk.."
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 5
```

Admission Webhooks Flow




```
{
  "kind": "AdmissionReview",
  "request": {
    "kind": {
      "kind": "Pod",
      "version": "v1"
    },
    "object": {
      "metadata": {
        "name": "myapp"
      },
      "spec": {
        "containers": [
          {
            "image": "nginx",
            "name": "nginx-frontend"
          },
          {
            "image": "mysql",
            "name": "mysql-backend"
          }
        ]
      }
    }
  }
}
```

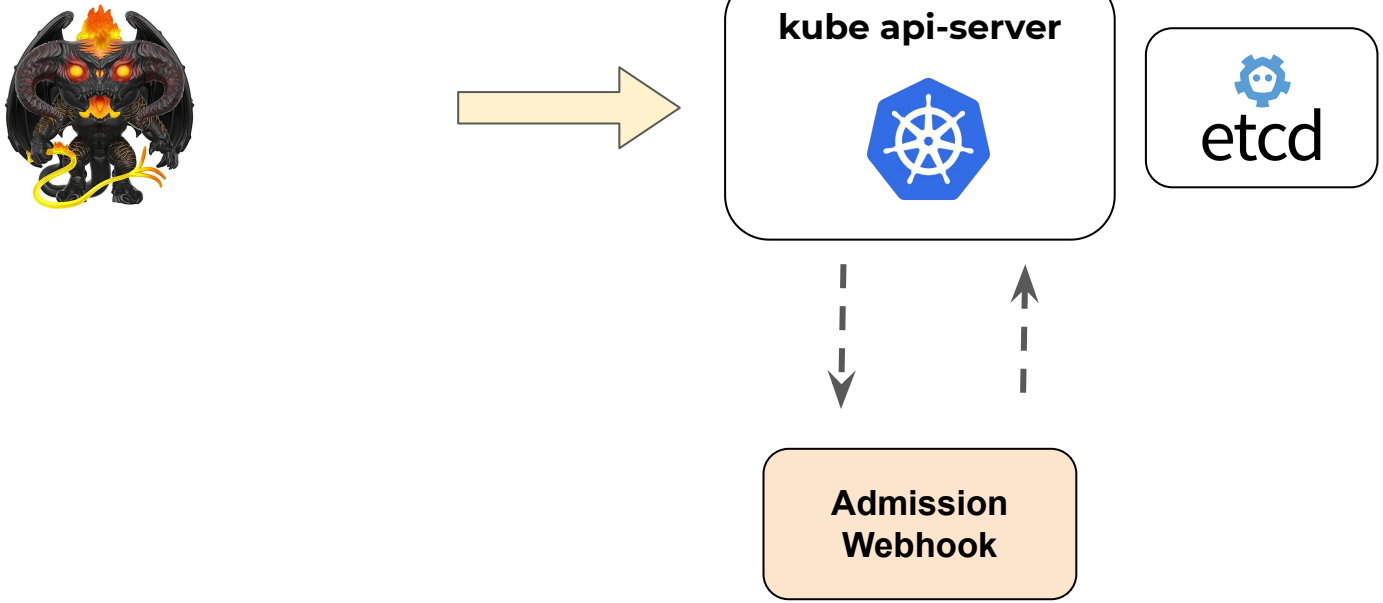
Admission Webhooks Flow




```
{
  "apiVersion": "admission.k8s.io/v1",
  "kind": "AdmissionReview",
  "response": {
    "uid": "<value from request.uid>",
    "allowed": true
  }
}
```

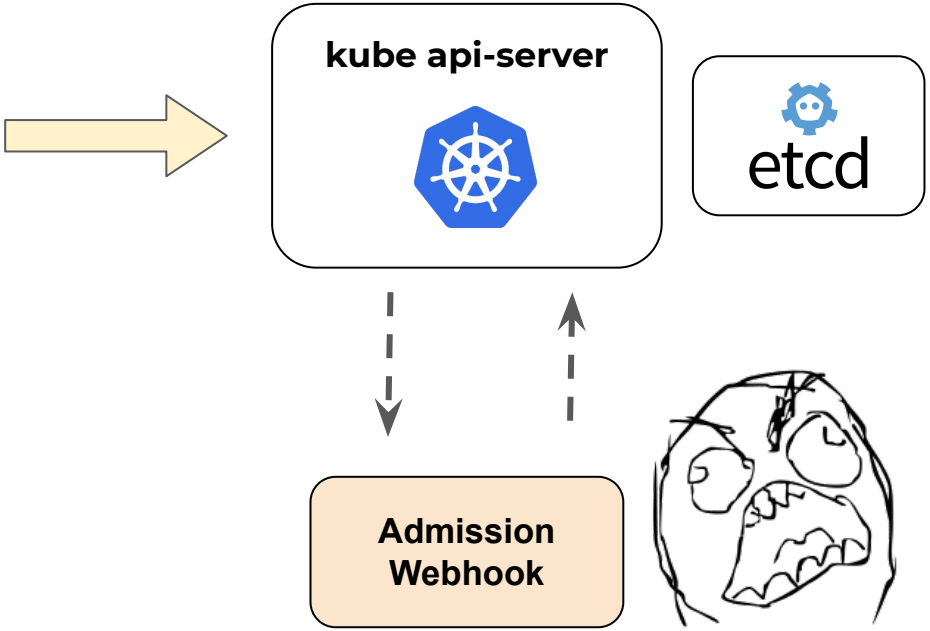
What's wrong with this approach?

Kubernetes allows **decoupling policy decisions** from the API server with admission controller webhooks intercepting admission requests before they are persisted.

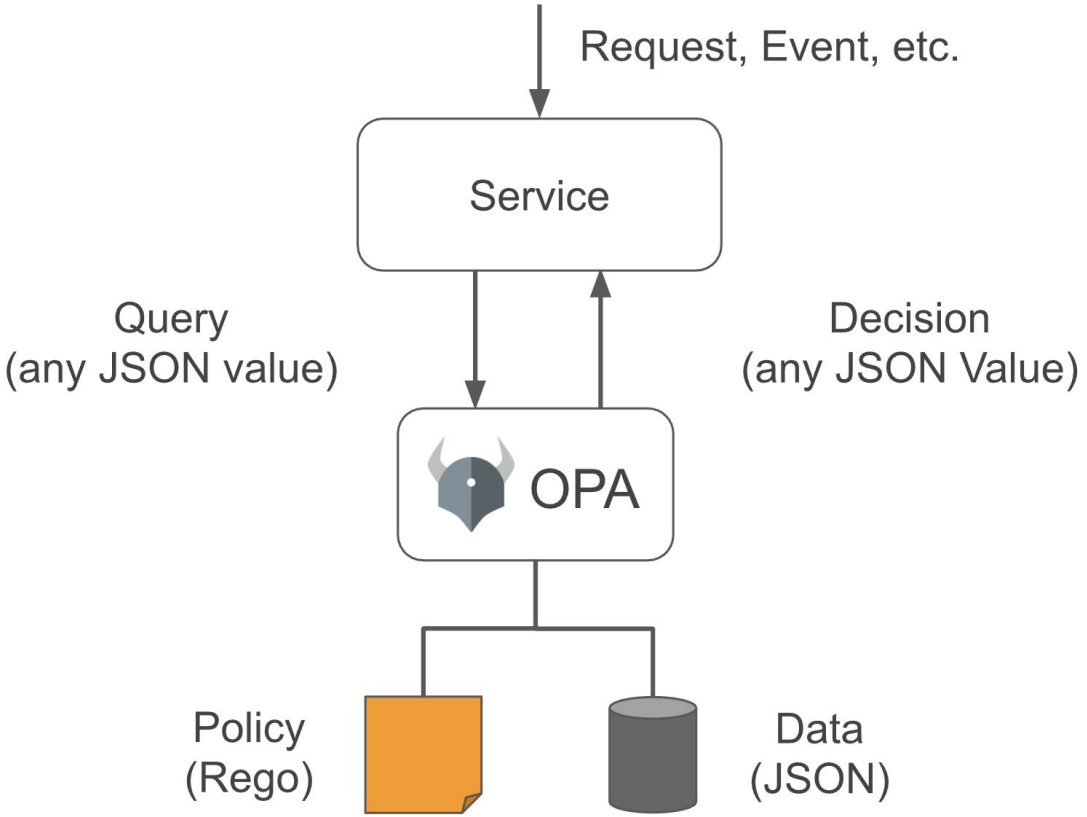


What's wrong with this approach?

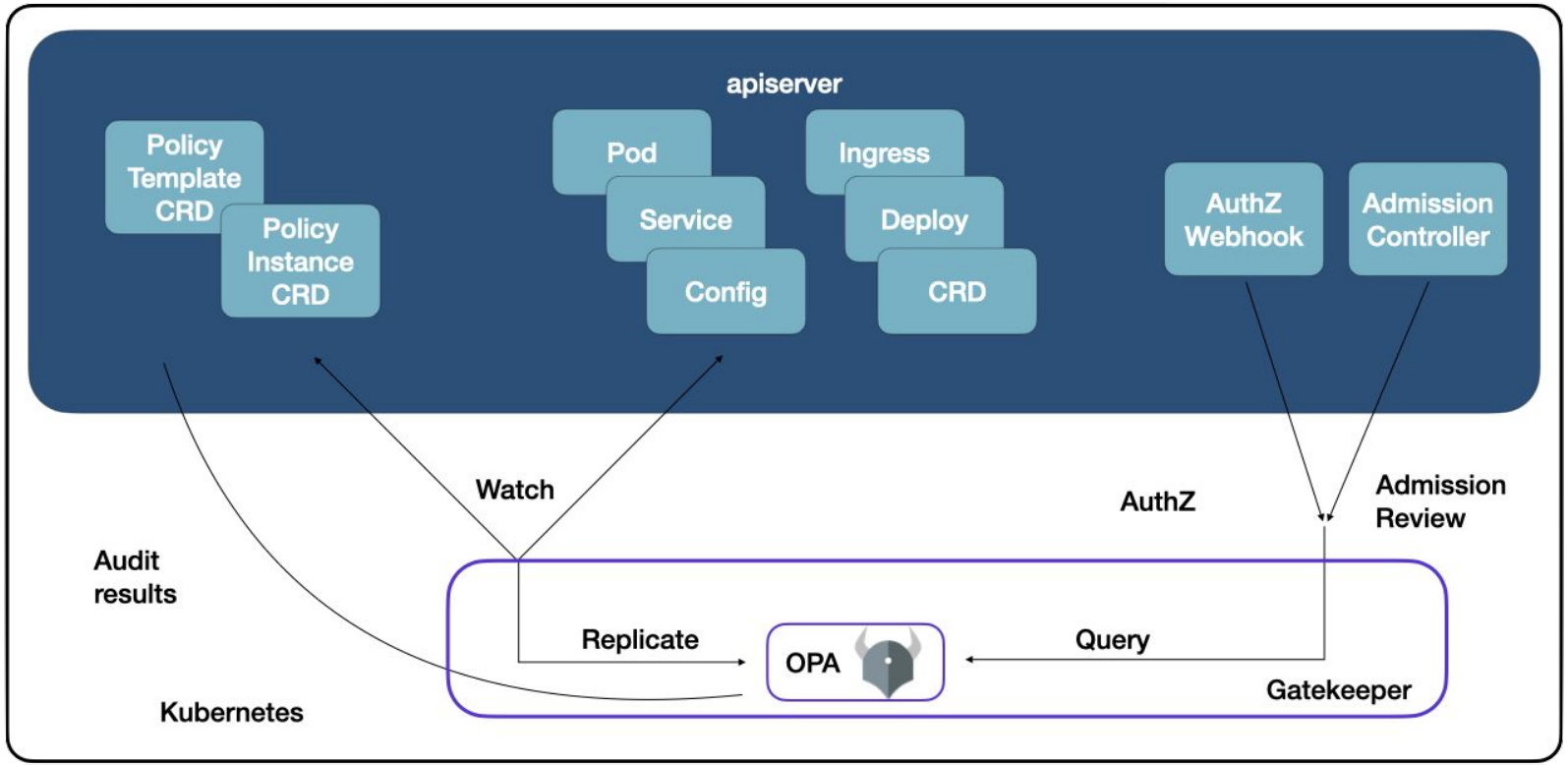
Using code to define compliance rules and security policies **is not very scalable** in the long run.



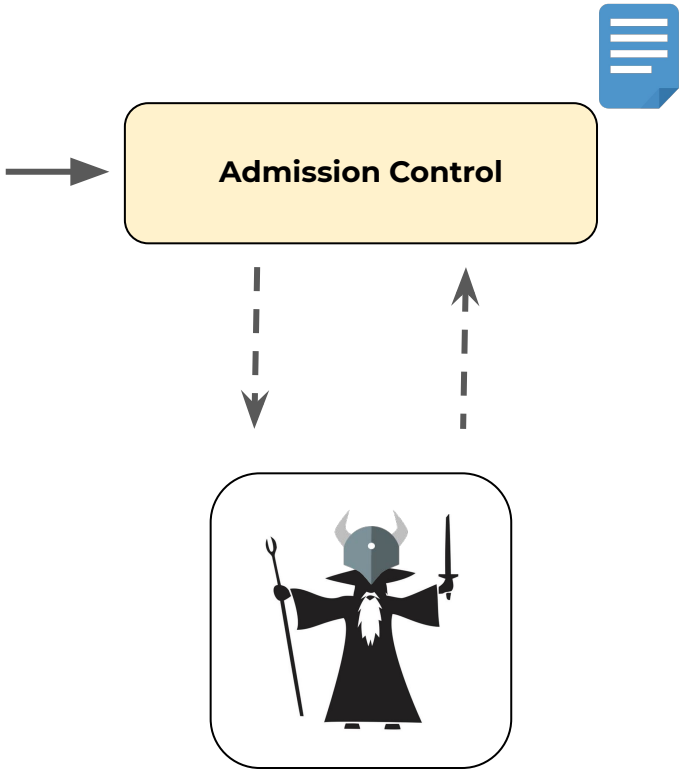
Introducing OPA



OPA Gatekeeper

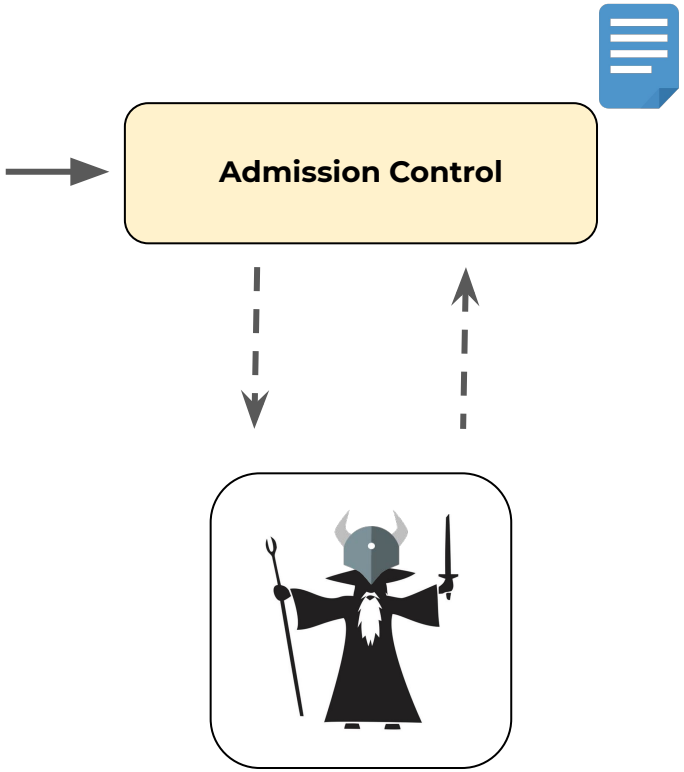


OPA Gatekeeper - WebhookConfiguration




```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  labels:
    gatekeeper.sh/system: "yes"
  name: gatekeeper-validating-webhook-configuration
webhooks:
- admissionReviewVersions:
  - v1
  - v1beta1
  clientConfig:
    service:
      name: gatekeeper-webhook-service
      namespace: gatekeeper-system
      path: /v1/admit
  failurePolicy: Ignore
  matchPolicy: Exact
  name: validation.gatekeeper.sh
  namespaceSelector:
    matchExpressions:
    - key: admission.gatekeeper.sh/ignore
      operator: DoesNotExist
  rules:
  - apiGroups:
    - '*'
    apiVersions:
    - '*'
    operations:
    - CREATE
    - UPDATE
    resources:
    - '*'
  sideEffects: None
  timeoutSeconds: 3
```

OPA Gatekeeper - WebhookConfiguration




```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  labels:
    gatekeeper.sh/system: "yes"
  name: gatekeeper-validating-webhook-configuration
webhooks:
- admissionReviewVersions:
  - v1
  - v1beta1
  clientConfig:
    service:
      name: gatekeeper-webhook-service
      namespace: gatekeeper-system
      path: /v1/admit
  failurePolicy: Ignore
  matchPolicy: Exact
  name: validation.gatekeeper.sh
  namespaceSelector:
    matchExpressions:
    - key: admission.gatekeeper.sh/ignore
      operator: DoesNotExist
  rules:
  - apiGroups:
    - '*'
    apiVersions:
    - '*'
    operations:
    - CREATE
    - UPDATE
    resources:
    - '*'
  sideEffects: None
  timeoutSeconds: 3
```

OPA Constraint Framework - Rego


```
package k8srequiredlabels

        deny[{"msg": msg, "details": {"missing_labels": missing}}] {
          provided := {label | input.review.object.metadata.labels[label]}
          required := {label | label := input.parameters.labels[_]}
          missing := required - provided
          count(missing) > 0
          msg := sprintf("you must provide labels: %v", [missing])
        }
```

https://open-policy-agent.github.io/gatekeeper/website/docs/howto/

OPA Constraint Framework - ContraintTemplate


```yaml
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
spec:
  crd:
    spec:
      names:
        kind: K8sRequiredLabels
        listKind: K8sRequiredLabelsList
        plural: k8srequiredlabels
        singular: k8srequiredlabels
      validation:
        # Schema for the `parameters` field
        openAPIV3Schema:
          properties:
            labels:
              type: array
              items: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
```

```
package k8srequiredlabels

        deny[{"msg": msg, "details": {"missing_labels": missing}}] {
          provided := {label | input.review.object.metadata.labels[label]}
          required := {label | label := input.parameters.labels[_]}
          missing := required - provided
          count(missing) > 0
          msg := sprintf("you must provide labels: %v", [missing])
        }
```

https://open-policy-agent.github.io/gatekeeper/website/docs/howto/

OPA Constraint Framework - Contraints


```yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-hr
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    labels: ["hr"]
```

https://open-policy-agent.github.io/gatekeeper/website/docs/howto/

OPA Constraint Framework - Audits


```yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-hr
spec:
  #...
status:
  auditTimestamp: "2019-08-06T01:46:13Z"
  byPod:
  - enforced: true
    id: gatekeeper-controller-manager-0
  violations:
  - enforcementAction: deny
    kind: Namespace
    message: 'you must provide labels: {"hr"}'
    name: default
  - enforcementAction: deny
    kind: Namespace
    message: 'you must provide labels: {"hr"}'
    name: gatekeeper-system
```

https://open-policy-agent.github.io/gatekeeper/website/docs/howto/

The fall (aka the demo)

1. Deploy OPA Gatekeeper
2. Deploy Gatekeeper Policy Manager
3. Deploy simple policies
4. Audit violations

Thank you

That’s it!

@niccoloraspa

/in/niccolo-raspa/

niccolo@sighup.io

KCD ITALY 2021

Slides and Demo available at **bit.ly/kcd-italy-opa**