# Enforcing Bespoke Policies
## in Cloud Native Systems

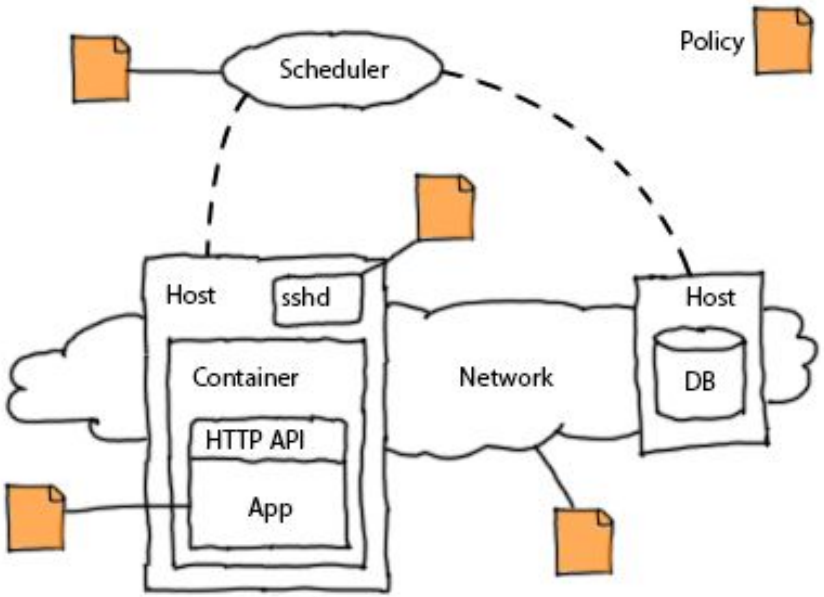
Torin Sandall
@sometorin
openpolicyagent.org

# Overview

- What Is Policy?
- Example Scenario
- Admission Control (Before & After 1.7)
- Custom Resource Definitions
- Open Policy Agent

# What Is Policy?

- Every organization has unique requirements that affect the entire stack and change over time

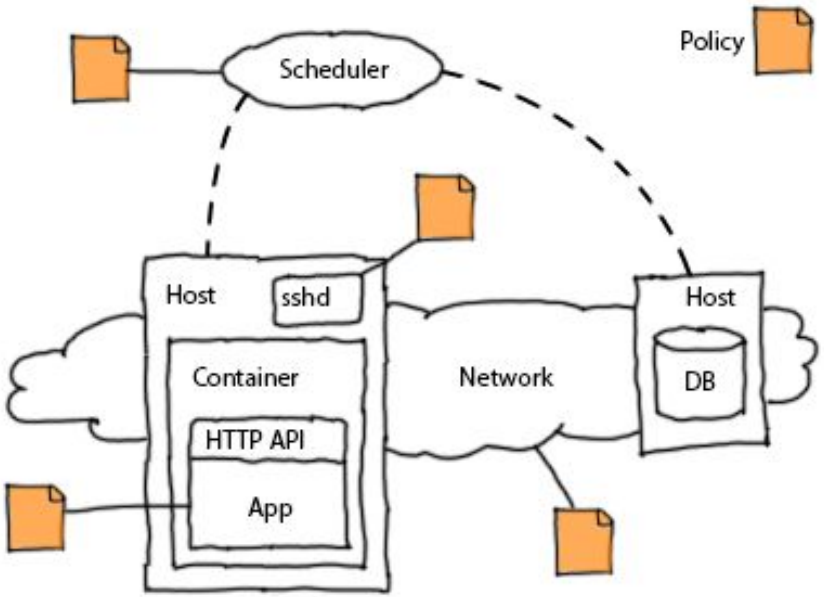



openpolicyagent.org

# What Is Policy?

- Every organization has unique requirements that affect the entire stack and change over time

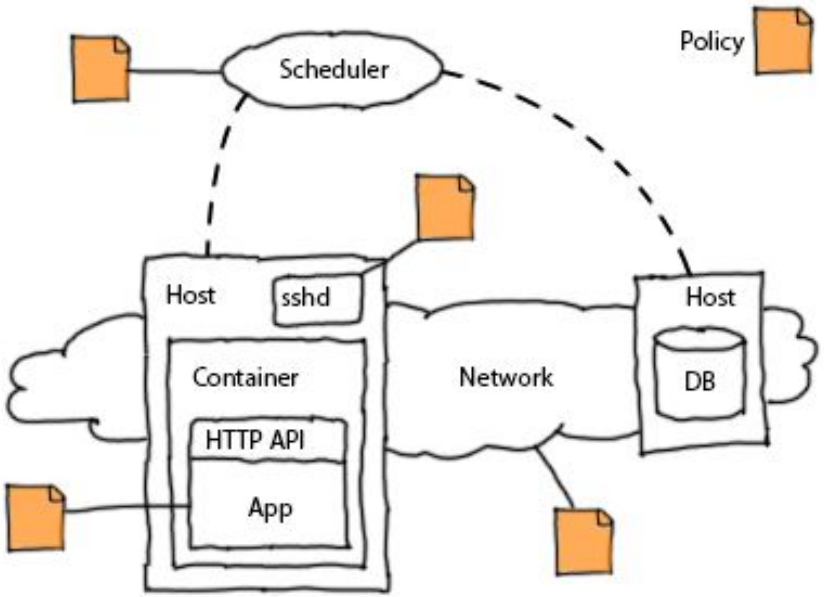
- Policies are sets of rules that govern how the system *should* behave




openpolicyagent.org

# What Is Policy?

- Every organization has unique requirements that affect the entire stack and change over time
- Policies are sets of rules that govern how the system *should* behave
- Policies are vital to the long-term success of organizations




openpolicyagent.org

# Example Scenario

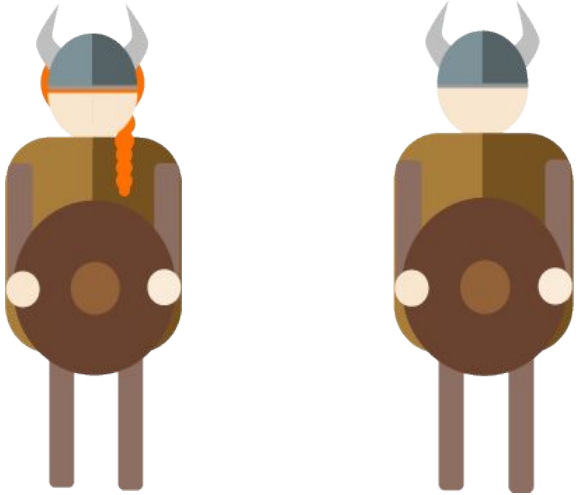
- **Alice** and **Bob** work for AcmeCorp

**Alice**
Platform Engineer

**Bob**
App Engineer

# Example Scenario

- **Alice** and **Bob** work for AcmeCorp
- Bob needs shell access to containers running on Kubernetes

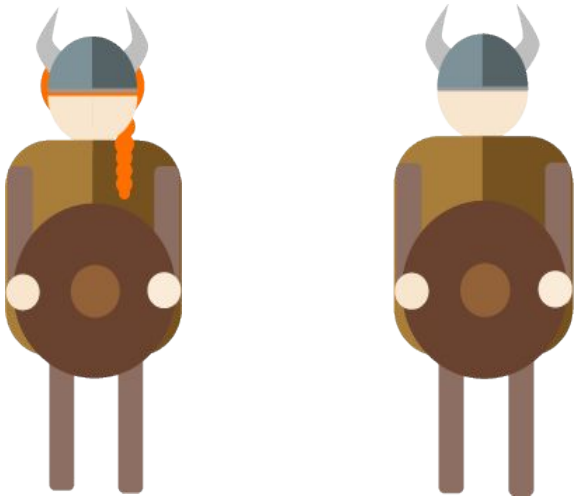


**Alice**
Platform Engineer

**Bob**
App Engineer

# Example Scenario

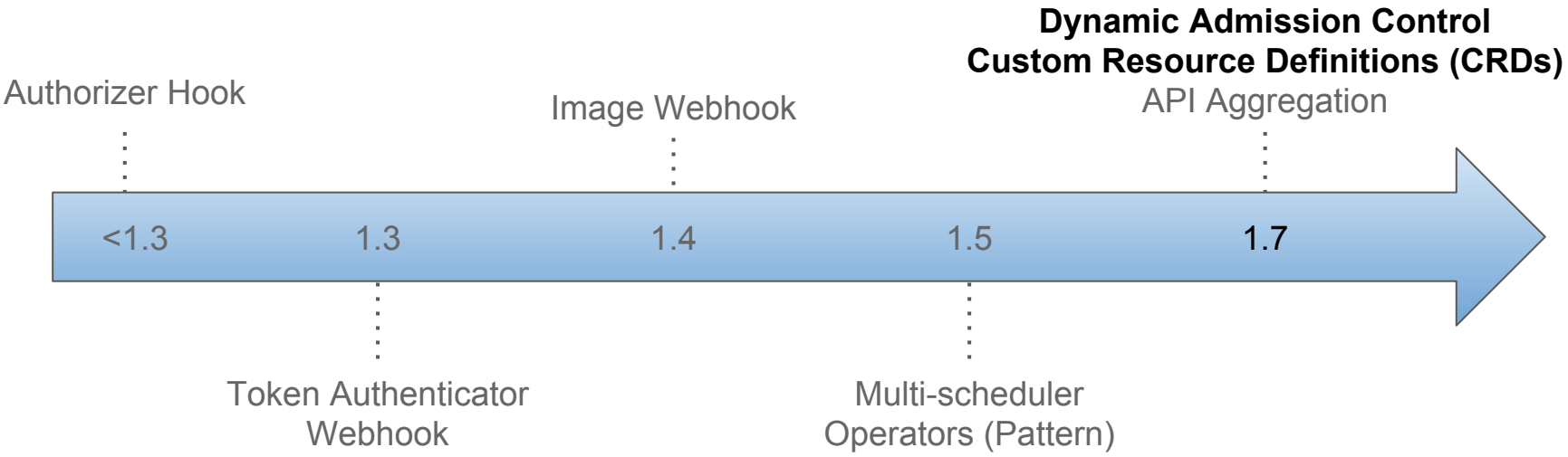
- **Alice** and **Bob** work for AcmeCorp
- Bob needs shell access to containers running on Kubernetes
- Bob cannot be trusted with access to **privileged containers** running in the **production namespace**



**Alice**
Platform Engineer

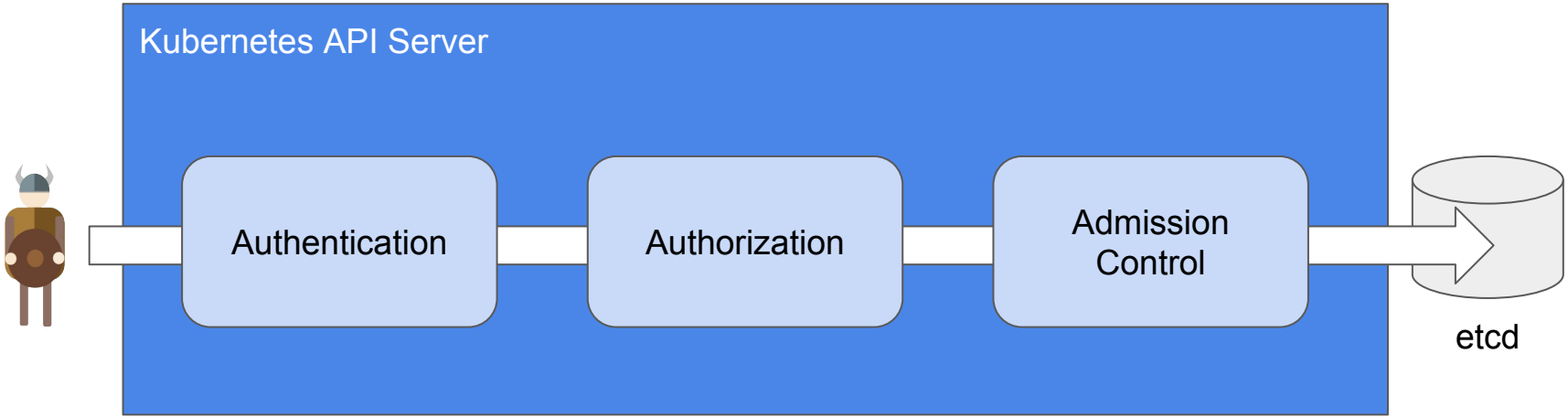
**Bob**
App Engineer

# Kubernetes Extensibility



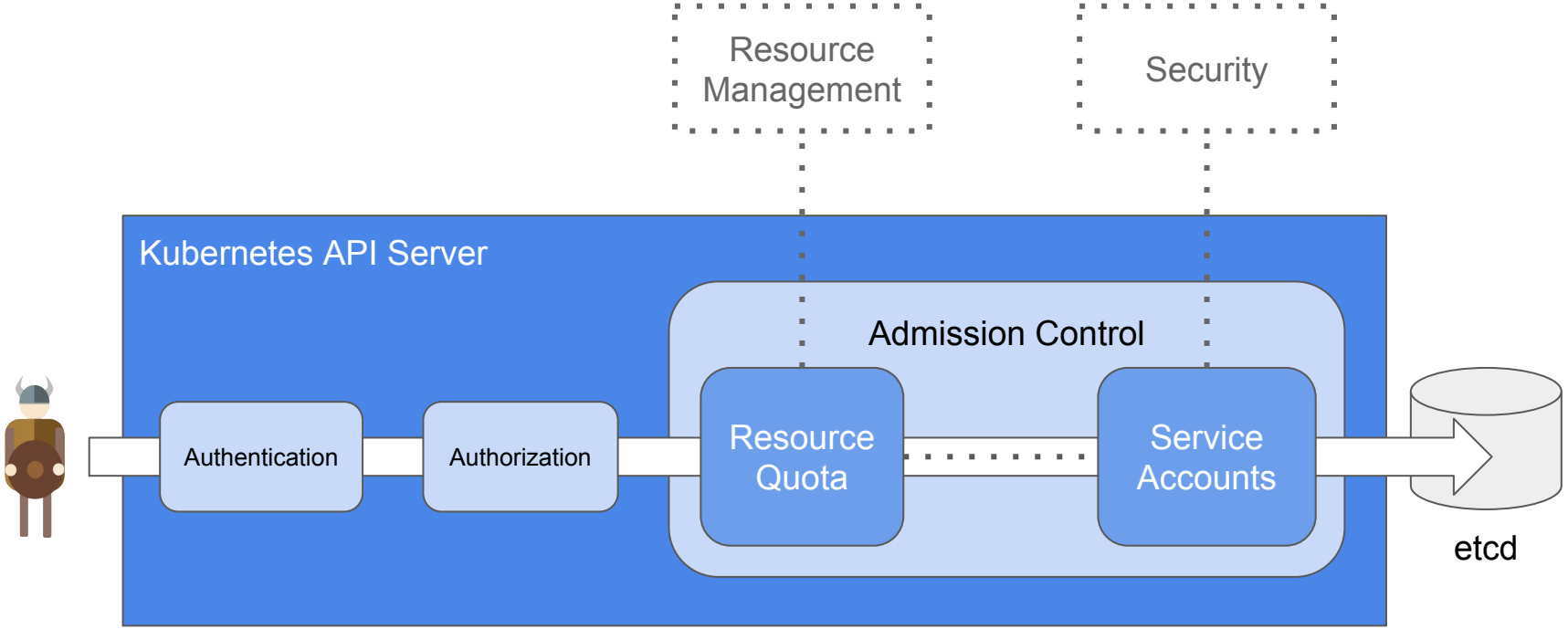

openpolicyagent.org

# Admission Control



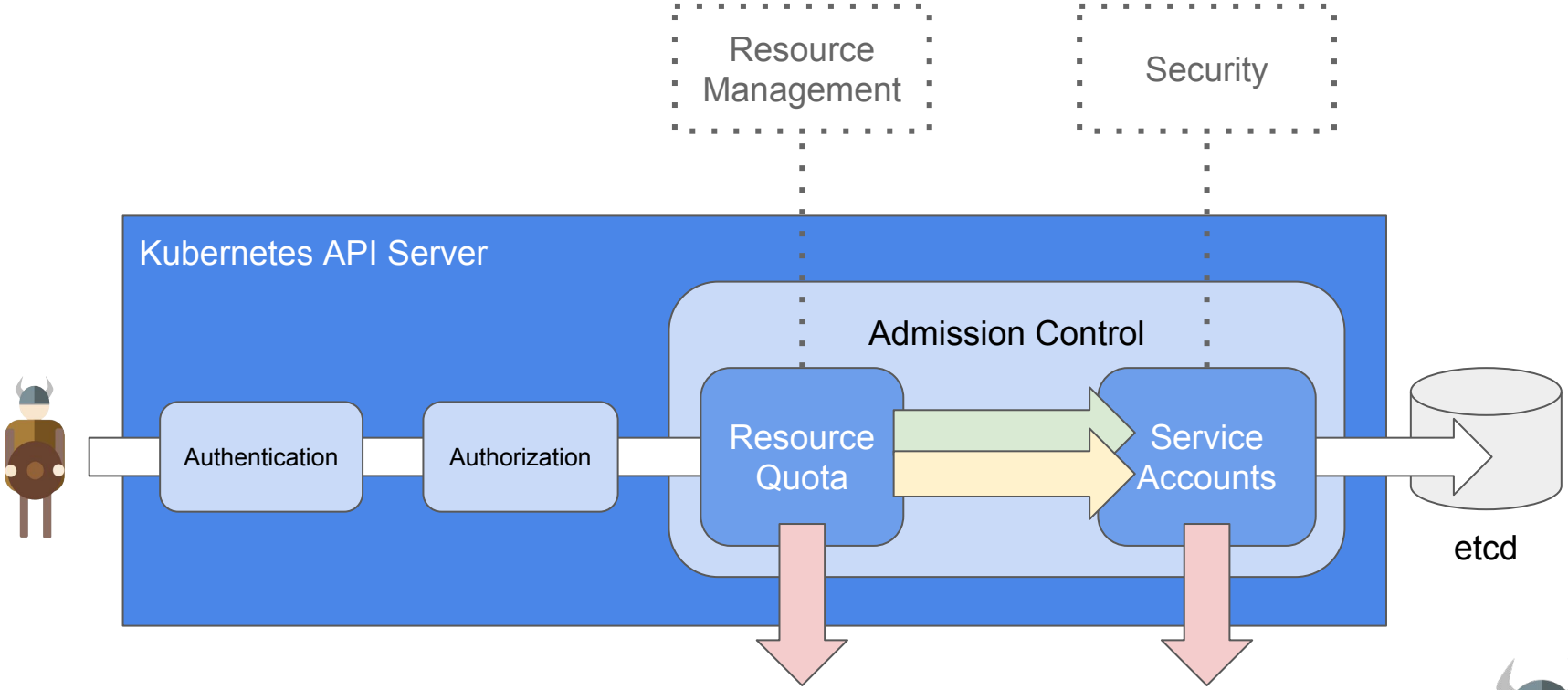

openpolicyagent.org

# Admission Control




openpolicyagent.org

# Admission Control




openpolicyagent.org

# Admission Control: Before 1.7

- Static compilation & configuration
  - 30+ admission controllers
  - 1-4 added per release
  - Command line arguments
  - Static configuration files

admit
deny
exec
limitranger
namespace
resourcequota
securitycontext
serviceaccount
initialresources
alwayspullimages
antiaffinity
persistentvolume
security
imagepolicy
storageclass
gc
podnodeselector
defaulttolerationseconds
podpreset
initialization
noderestriction
podtolerationrestriction
schedulingpolicy
image/imagelimitrangerplugin
image/imagepolicyplugin
ingress/ingress
project/lifecycle
project/podnodeenvironment
project/projectrequestlimit
quota/quotaclusterresourceoverride
quota/clusterquota
quota/runonceduration
scheduler/podnodeconstraints
security/constraint

# Admission Control: Before 1.7

- Static compilation & configuration
  - 30+ admission controllers
  - 1-4 added per release
  - Command line arguments
  - Static configuration files

- Example Scenario
  - Alice forks Kubernetes into a private repository
  - Alice implements the policy inside the plugin framework
  - Alice now has to build, push, and upgrade Kubernetes itself

admit
deny
exec
limitranger
namespace
resourcequota
securitycontext
serviceaccount
initialresources
alwayspullimages
antiaffinity
persistentvolume
security
imagepolicy
storageclass
gc
podnodeselector
defaulttolerationseconds
podpreset
initialization
noderestriction
podtolerationrestriction
schedulingpolicy
image/imagelimitrangerplugin
image/imagepolicyplugin
ingress/ingress
project/lifecycle

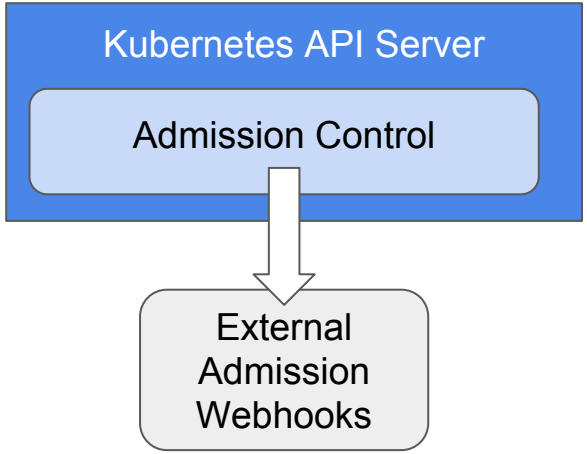
**bobprotectionpolicy**

project/podnodeenvironment
project/projectrequestlimit
quota/quotaclusterresourceoverride
quota/clusterquota
quota/runonceduration
scheduler/podnodeconstraints
security/constraint

openpolicyagent.org

# Admission Control: Webhooks

- In 1.7, admission controllers can be implemented as webhooks that run on top of Kubernetes
- Webhooks can **allow** or **deny** incoming requests
  - Before etcd is updated
  - Before clients are notified
- Webhooks are configured **dynamically** via Kubernetes APIs



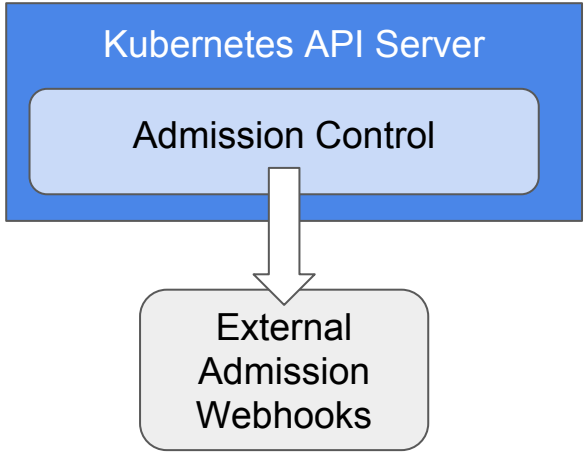
# Admission Control: Webhooks

- The API Server calls webhooks whose configuration rules match the incoming request:

```
match [
  {operations: ["create"], kinds: ["pods"]},
  {operations: ["delete"], kinds: ["services"]}
]
```

- Rules can include wildcards:

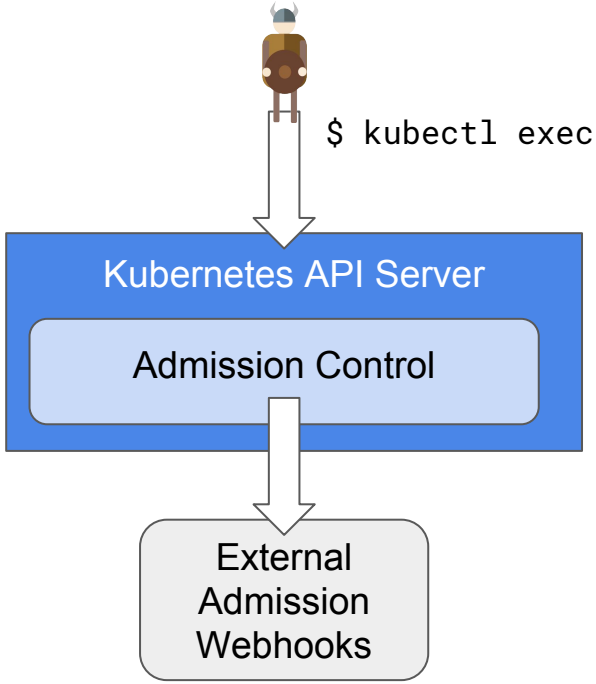
```
match [
  {operations: ["*"], kinds: ["*"]}
]
```



# Admission Control: Webhooks

- The API Server provides the **operation, entire object, and user info** in the webhook call

```
kind: AdmissionReview
spec:
  kind: {kind: Pod, version: v1}
  name: admission-webhook-demo-373699553-8srx8
  namespace: default
  object:
    Options:
      Command: [sh]
      Container: admission-webhook-demo
      ...
    ResourcePath: pods/exec
  operation: CONNECT
  userInfo:
    groups: ['system:masters', 'system:authenticated']
    username: minikube
```

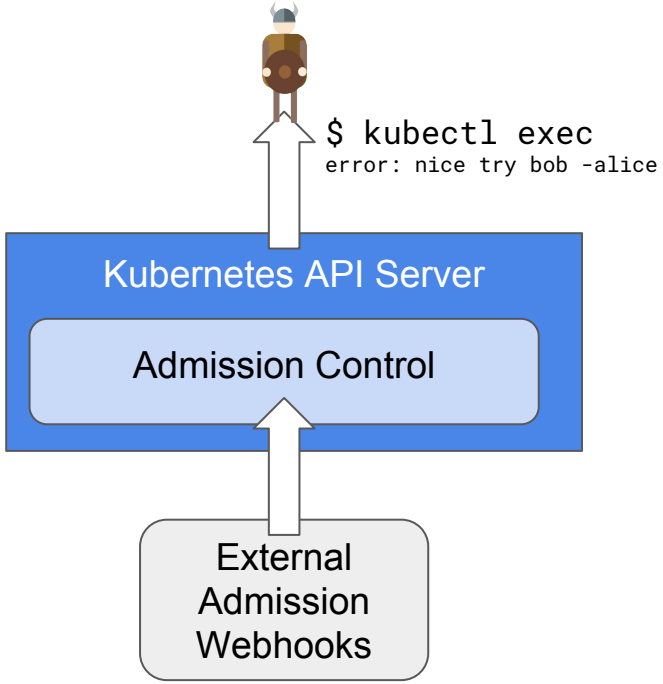


# Admission Control: Webhooks

- Webhooks respond with an **AdmissionReview** that indicates whether to **allow** or **deny** the request

```
kind: AdmissionReview
status:
  allowed: false
  reason:
    message: “nice try bob -alice”
```

- The API Server rejects the request **IF ANY** of the webhooks return a denial




openpolicyagent.org

# Demo

# Webhooks: Lessons Learned

# Webhooks: Lessons Learned

- API server connects by IP address => subjectAltName hardcoded

# Webhooks: Lessons Learned

- API server connects by IP address => subjectAltName hardcoded
- API server “fails open” if webhook fails (not configurable in alpha)



# Webhooks: Lessons Learned

- API server connects by IP address => subjectAltName hardcoded
- API server “fails open” if webhook fails (not configurable in alpha)
- API server sends “internal representation” of Kubernetes objects over the wire

# Webhooks: Lessons Learned

- API server connects by IP address => subjectAltName hardcoded
- API server “fails open” if webhook fails (not configurable in alpha)
- API server sends “internal representation” of Kubernetes objects over the wire
- Webhooks must serve POST requests at https://<ip>:<port>/

# Webhooks: Lessons Learned

- API server connects by IP address => subjectAltName hardcoded
- API server “fails open” if webhook fails (not configurable in alpha)
- API server sends “internal representation” of Kubernetes objects over the wire
- Webhooks must serve POST requests at https://<ip>:<port>/
- Dependency hell with client-go (v4.0.0.beta0) and Kubernetes 1.7

# Webhooks...all the way down?

- Webhooks & Initializers lay the groundwork for **extensible** policy enforcement

# Webhooks...all the way down?

- Webhooks & Initializers lay the groundwork for **extensible** policy enforcement

- Policy **decisions** have been decoupled from **enforcement**

# Webhooks...all the way down?

- Webhooks & Initializers lay the groundwork for **extensible** policy enforcement
- Policy **decisions** have been decoupled from **enforcement**
- Is there a better way to **author** policies to control system **behaviour**?

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
  name: nginx-1493591563-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    securityContext:
      privileged: true
  dnsPolicy: ClusterFirst
  nodeName: minikube
  restartPolicy: Always
status:
  containerStatuses:
  - name: nginx
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: 2017-08-01T06:34:22Z
  hostIP: 192.168.99.100
  phase: Running
  podIP: 172.17.0.4
  startTime: 2017-08-01T06:34:13Z
```

openpolicyagent.org

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
  name: nginx-1493591563-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    securityContext:
      privileged: true
  dnsPolicy: ClusterFirst
  nodeName: minikube
  restartPolicy: Always
status:
  containerStatuses:
  - name: nginx
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: 2017-08-01T06:34:22Z
  hostIP: 192.168.99.100
  phase: Running
  podIP: 172.17.0.4
  startTime: 2017-08-01T06:34:13Z
```

# references
spec.containers

openpolicyagent.org

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
  name: nginx-1493591563-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    securityContext:
      privileged: true
  dnsPolicy: ClusterFirst
  nodeName: minikube
  restartPolicy: Always
status:
  containerStatuses:
  - name: nginx
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: 2017-08-01T06:34:22Z
  hostIP: 192.168.99.100
  phase: Running
  podIP: 172.17.0.4
  startTime: 2017-08-01T06:34:13Z
```

```
# references
spec.containers

# variables
container = spec.containers[_]
```

openpolicyagent.org

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
  name: nginx-1493591563-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    securityContext:
      privileged: true
  dnsPolicy: ClusterFirst
  nodeName: minikube
  restartPolicy: Always
status:
  containerStatuses:
  - name: nginx
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: 2017-08-01T06:34:22Z
  hostIP: 192.168.99.100
  phase: Running
  podIP: 172.17.0.4
  startTime: 2017-08-01T06:34:13Z
```

```
# references
spec.containers

# variables
container = spec.containers[_]

# expressions/assertions
container.securityContext.privileged = true
```

openpolicyagent.org

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
  name: nginx-1493591563-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    securityContext:
      privileged: true
  dnsPolicy: ClusterFirst
  nodeName: minikube
  restartPolicy: Always
status:
  containerStatuses:
  - name: nginx
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: 2017-08-01T06:34:22Z
  hostIP: 192.168.99.100
  phase: Running
  podIP: 172.17.0.4
  startTime: 2017-08-01T06:34:13Z
```

```
# references
spec.containers

# variables
container = spec.containers[_]

# expressions/assertions
container.securityContext.privileged = true

# functions
is_privileged {
  container = spec.containers[_]
  container.securityContext.privileged = true
}
```

openpolicyagent.org

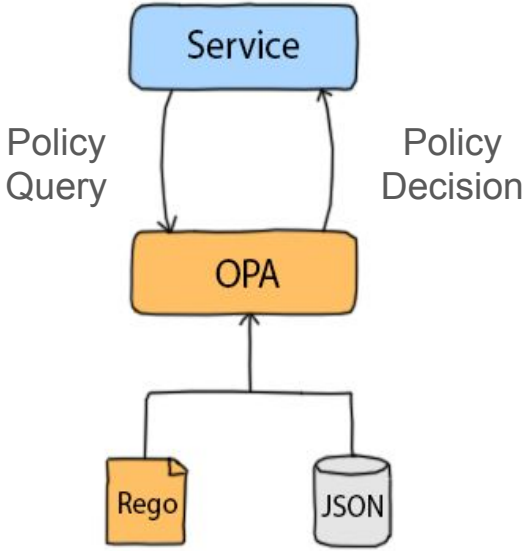
```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
  name: nginx-1493591563-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    securityContext:
      privileged: true
  dnsPolicy: ClusterFirst
  nodeName: minikube
  restartPolicy: Always
status:
  containerStatuses:
  - name: nginx
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: 2017-08-01T06:34:22Z
  hostIP: 192.168.99.100
  phase: Running
  podIP: 172.17.0.4
  startTime: 2017-08-01T06:34:13Z
```

```
# references
spec.containers

# variables
container = spec.containers[_]

# expressions/assertions
container.securityContext.privileged = true

# functions
is_privileged {
  container = spec.containers[_]
  container.securityContext.privileged = true
}

# policies
deny {
  review.user      = “bob”
  review.operation = “CONNECT”
  review.namespace = “production”
  is_privileged
  ...
}
```

openpolicyagent.org

# OPA is an open source, general-purpose policy engine

- Declarative Language (Rego)
- Document-oriented (JSON, YAML)
- Daemon, Library, REPL (Go)
- Policy, Data & Query APIs (HTTP)
- Integration with Kubernetes
  - Federation 1.7 placement policies powered by OPA
- Apache License 2.0




openpolicyagent.org

# Custom Resource Definitions (CRDs)

- Add custom resource types to the Kubernetes API

```
v1/pods
apps/v1beta1/deployments
...
acmecorp.com/v1/policies
```

- Deprecates `ThirdPartyResources`
  - Same wire format
  - Configurable pluralization
  - Cluster versus namespace scope

# Custom Resource Definitions (CRDs)

- Write definitions in YAML and register them via the Kubernetes API

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: policies.acmecorp.com
spec:
  group: acmecorp.com
  version: v1
  scope: Cluster
  names:
    plural: policies
    singular: policy
    kind: Policy
```

```
alice:~$ kubectl create -f policies-crd.yaml
customresourcedefinition "policies.acmecorp.com" created

alice:~$ kubectl get crd
NAME                   KIND
policies.acmecorp.com CustomResourceDefinition.v1beta1.apiex
```

# Custom Resource Definitions (CRDs)

- Interact with custom resources **the same way** as Deployments, Services, etc.

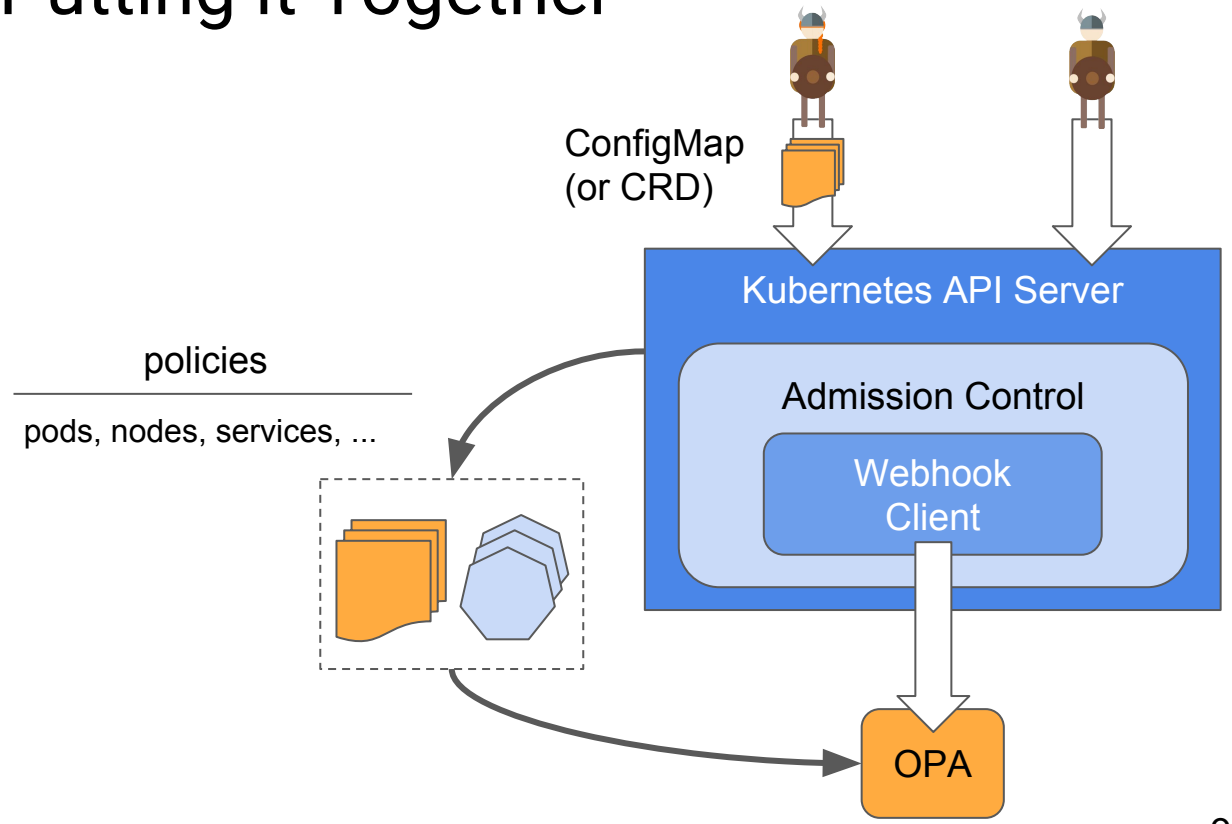
```yaml
apiVersion: acmecorp.com/v1
kind: Policy
metadata:
  name: internal-image-registry
spec:
  expressions:
    - kind: Pod
      path: "spec.containers[].image"
      match: "acmecorp.io/.*$"
---
apiVersion: acmecorp.com/v1
kind: Policy
metadata:
  name: team-label-exists
spec:
  expressions:
    - kind: "*"
      path: "metadata.labels.team"
      exists: true
```

```
alice:~$ kubectl create -f custom-policies.yaml
policy "internal-image-registry" created
policy "team-label-exists" created

alice:~$ kubectl get policies.acmecorp.com
NAME                        KIND
internal-image-registry     Policy.v1.acmecorp.com
team-label-exists           Policy.v1.acmecorp.com

alice:~$ kubectl delete policies.acmecorp.com team-label-exists
policy "team-label-exists" deleted
```

openpolicyagent.org

# Demo

# Putting it Together




openpolicyagent.org

# Conclusions

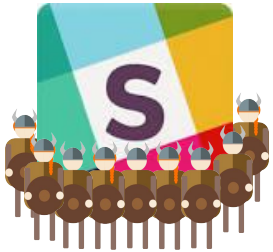
- Think about leveraging Kubernetes 1.7 extensibility features today
  - Initializers & Webhooks in alpha => beta in 1.8
  - Custom Resource Definitions in beta

- Check out the Open Policy Agent (OPA) project
  - Unified, context-aware policies across the stack
  - High-level declarative policy language
  - Rich support for complex/nested data

- Accept uncertainty
  - Requirements evolve => design for change
  - Decouple policy decisions from enforcement

openpolicyagent.org

# Thank you!




slack.openpolicyagent.org

github.com/open-policy-agent/opa