



Enabling Autonomous Teams

WITH POLICY ENFORCEMENT AT YUBICO

JAMES ALSETH and JOHN REESE



James Alseth is a Security Engineer at Yubico, currently focused on cloud infrastructure security. He works on building self-service security solutions that enable engineers to be more confident in their design, implementation, and deployment decisions and strategies.



John Reese is a Software Engineer at Yubico, who specializes in Kubernetes and Go. He is an active open source contributor and is a core maintainer for Conftest, a tool that helps you write tests against structured configuration data.

In his free time, he enjoys playing hockey and video games, both of which he takes way too seriously.



Agenda

- Brief history of Kubernetes at Yubico
- How policy enables autonomous teams
- Tooling used to enforce the policy
- Yubico's K8s policy journey
- Q&A



Kubernetes at Yubico



How can we ensure K8s workloads are
configured **securely**?





Strong Authentication



Role-based Authorization



Peer Review



About peer review...

- Needs to be performed consistently to be effective
- Consistent, accurate review requires a significant time investment
- Bottlenecks on the team or individual with the most experience
- Slows the release cycle



✨ Policy ✨



Policy enforces what we **actually** care about



Rego

<https://play.openpolicyagent.org>

Rego queries are assertions on input. The result of the query tells the asker if the input violates a policy or not.

```
package main

deny[msg] {
    input.kind == "Namespace"
    not input.metadata.labels["owner"]

    msg := "Namespaces must have an owner"
}
```





Open Policy Agent



KubeCon // Cloud Native Security Day 2020

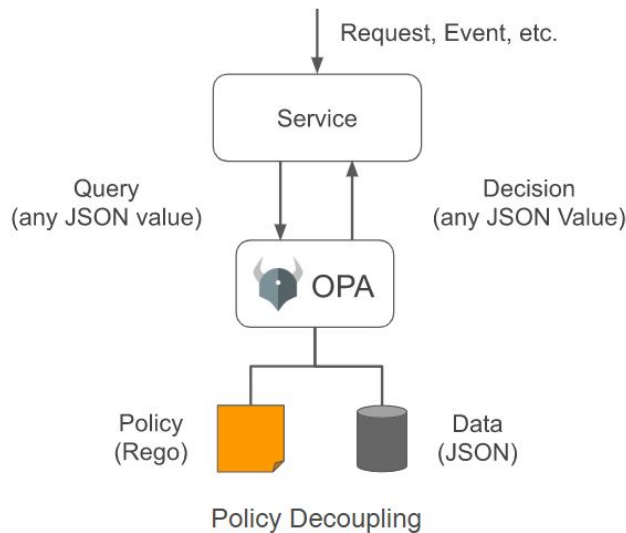
 @allset_

 @johnpreese

Open Policy Agent

<https://openpolicyagent.org>

Open Policy Agent is an open source, general-purpose policy engine that unifies policy enforcement across the stack.



The Open Policy Agent engine as a **library**
makes it even easier to enforce policy



Shifting policy enforcement to the **left**



Conftest

<https://github.com/open-policy-agent/conftest>

Conftest allows you to write tests against structured configuration data using the **Rego** query language.

Supports a number of configuration formats:

- ✓ YAML
- ✓ JSON
- ✓ INI
- ✓ TOML
- ✓ HCL
- ✓ CUE
- ✓ DOCKERFILE
- ✓ ... and more!



policy.rego

```
package main

deny[msg] {
  input.kind == "Deployment"
  not input.spec.template.spec.securityContext.runAsNonRoot

  msg := "Containers must not run as root"
}

deny[msg] {
  input.kind == "Deployment"
  not input.spec.selector.matchLabels.app

  msg := "Containers must provide app label for pod selectors"
}
```

deploy.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busy
spec:
  template:
    metadata:
      labels:
        app: busy
    spec:
      containers:
        - name: busy
          image: busybox:latest
```

```
$ conftest test -p policy.rego deploy.yaml
```

```
FAIL - deploy.yaml - Containers must not run as root
```

```
FAIL - deploy.yaml - Containers must provide app label for pod selectors
```

```
2 tests, 0 passed, 0 warnings, 2 failures, 0 exceptions
```



Share policies across the organization

```
$ conftest pull internal.azurecr.io/policies/cluster
```

```
$ conftest test -p cluster deploy.yaml
```

```
FAIL - deploy.yaml - Containers must not run as root
```

```
FAIL - deploy.yaml - Containers must provide app label for pod selectors
```

```
2 tests, 0 passed, 0 warnings, 2 failures, 0 exceptions
```





Artifact **HUB**

Artifact Hub is a web-based application that enables finding, installing, and publishing packages and configurations for CNCF projects

<https://artifacthub.io>



Continuous enforcement in **Production**



Gatekeeper

<https://github.com/open-policy-agent/gatekeeper>

Gatekeeper is an admission controller on Kubernetes that enforces policies.

- ✓ Conftest for shifting policy concerns to the **left**
- ✓ Gatekeeper for enforcing those policies in **production**
- ✓ Using the **same policies!**



... **almost** the same policies



```
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: namespaceowner
spec:
  crd:
    spec:
      names:
        kind: NamespaceOwner
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package namespaceowner

        violation[{"msg": msg, "details": {}}] {
          input.review.object.kind == "Namespace"
          not input.review.object.metadata.labels["owner"]
          msg := "Owner label is required on Namespaces"
        }
```



Rego is the **source of truth** for your policies



Konstraint

<https://github.com/plexsystems/konstraint>

Konstraint is a tool to assist with the management of templates and constraints when using Gatekeeper.

- ✓ Provides a library to write policies that work with Conftest and Gatekeeper
- ✓ Template and constraint creation and management
- ✓ Generates documentation for your policies



```
# @title Images must not use the latest tag
# Using the latest tag on images can cause unexpected problems in production. By specifying a pinned version
# we can have higher confidence that our applications are immutable and do not change unexpectedly.
# @enforcement deny
# @kinds apps/DaemonSet apps/Deployment apps/StatefulSet core/Pod
package container_deny_latest_tag

import data.lib.core
import data.lib.pods

violation[msg] {
    pods.containers[container]
    endswith(container.image, ":latest")
    msg := core.format(sprintf("%s/%s: Images must not use the latest tag", [core.kind, core.name]))
}
```

```
# Generate the Templates and ConstraintTemplates
$ konstraint create policies/

# Generate a Markdown file describing the policies
$ konstraint doc policies/
```





P1001: Containers must drop all capabilities

Severity: Violation

Resources: apps/DaemonSet apps/Deployment apps/StatefulSet core/Pod

Granting containers privileged capabilities on the node makes it easier for containers to escalate their privileges. As such, this is not allowed outside of Kubernetes controller namespaces.

Rego

```
package container_deny_added_caps

import data.lib.core
import data.lib.pods
import data.lib.security

policyID := "P1001"

violation[msg] {
  pods.containers[container]
  not container_dropped_all_capabilities(container)

  msg := core.format_with_id(sprintf("%s/%s/%s: Does not drop all capabilities", [core.kind, core.name, container.name]), policyID)
}

container_dropped_all_capabilities(container) {
  security.dropped_capability(container, "all")
}
```

source: [container-deny-added-caps](#)



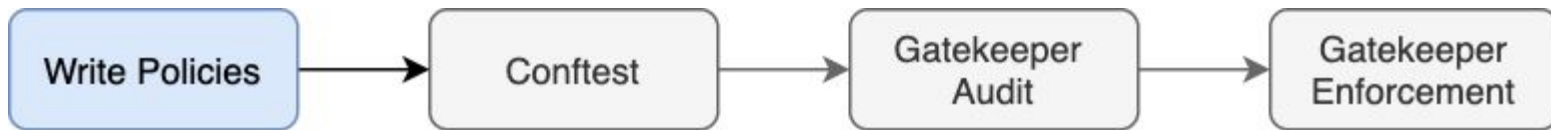
Yubico's K8s policy **journey**...



Plan v0.0.1a



Plan v0.0.1a



What went wrong?

- There were more policy violations than expected
- Policies were always “prod” with no way to safely test
- Adding Conftest to CI wasn’t as easy as it could be
- Conftest results weren’t surfaced to the teams
- Policy admins had no visibility into the Conftest results



We needed a **policy pipeline**
and to make policy adoption **easy**



Conftest Action

github.com/YubicoLabs/action-conftest

A GitHub Action that wraps Conftest, bringing everything into one **easy** step

- ✓ Pulls the latest policies from a remote source
- ✓ Surfaces policy violations and warnings in pull request comments
- ✓ Submits test results to a remote server



Template HelmRelease CRDs

github.com/YubicoLabs/action-template-helmrelease

A GitHub Action to template out one or more Flux CD HelmRelease resources

- ✓ Parses the Helm chart information from the HelmRelease CRD
- ✓ Sets up the Helm repository
- ✓ Templates out the resources



Policy pipeline rule #1:
Must not break production



Policy Promotion Strategy

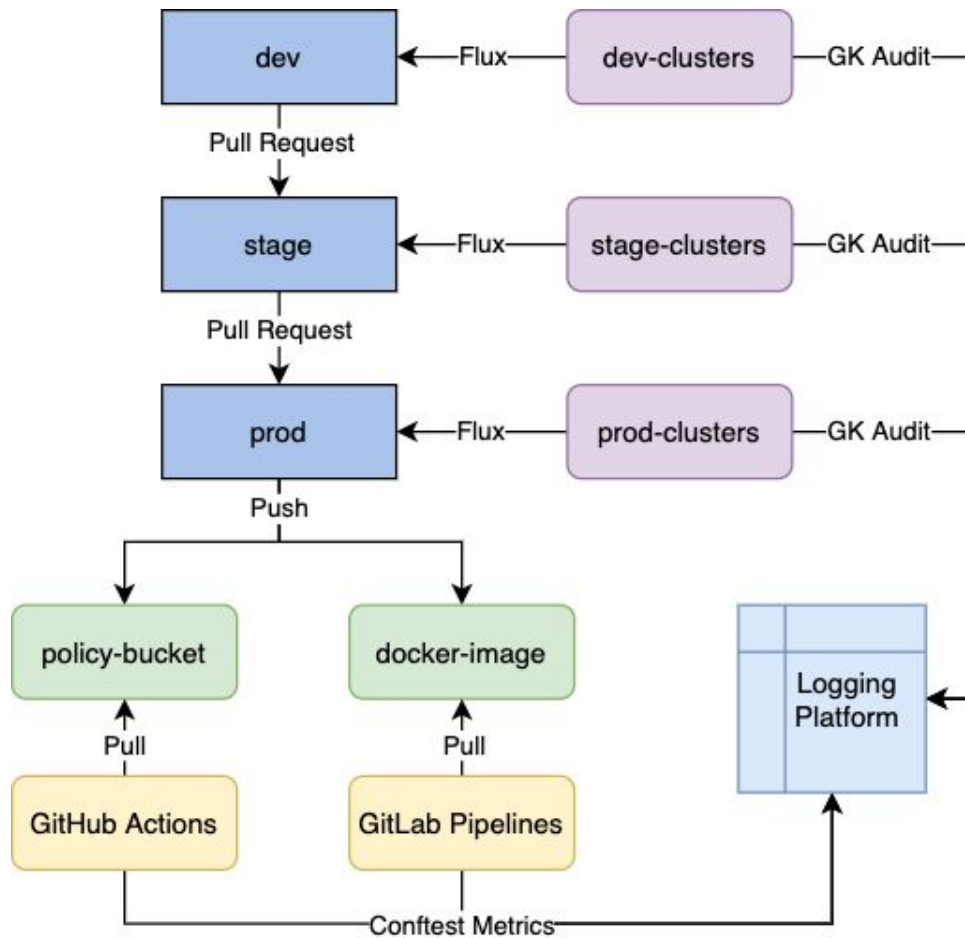
- Work on remediating one policy at a time
- Use Gatekeeper's `enforcementAction` property to introduce policies in dry run mode
- Use Gatekeeper audit data to determine when to promote
- Only enforce after all offending resources have been remediated
- Promotion to production must be scheduled



GitOps Approach

- Pull requests move policies through the pipeline
- Branch protection rules ensure peer review and unit tests pass
- **CODEOWNERS** ensures the peer review comes from policy admins
- Automation creates the Gatekeeper and Conftest resources





Where are we **now**?
How can we **improve** in the future?



THANKS!

yubico

Open Policy Agent

<https://www.openpolicyagent.org>

Conftest

<https://github.com/open-policy-agent/conftest>

Gatekeeper

<https://github.com/open-policy-agent/gatekeeper>

Konstraint

<https://github.com/plexsystems/konstraint>

Conftest Action

<https://github.com/YubicoLabs/action-conftest>

Artifact Hub

<https://artifacthub.io>

