Open Policy Agent

Introduction @ KubeCon Seattle 2018

Agenda

- Background & 5-minute crash course (presented by Torin Sandall)
- Use Case: Capital One (presented by Zach Abrahamson)
- Use Case: Intuit (presented by Todd Ekenstam)
- Q&A

OPA: General-purpose policy engine

Inception

Project started in 2016 at Styra.

Goal

Unify policy enforcement across the stack.

Users

Netflix
Chef
Medallia
Cloudflare
State Street
Pinterest
Intuit
Capital One
...and many more.

Use Cases

Admission control
Authorization
- ACLs
- RBAC
- IAM
- ABAC

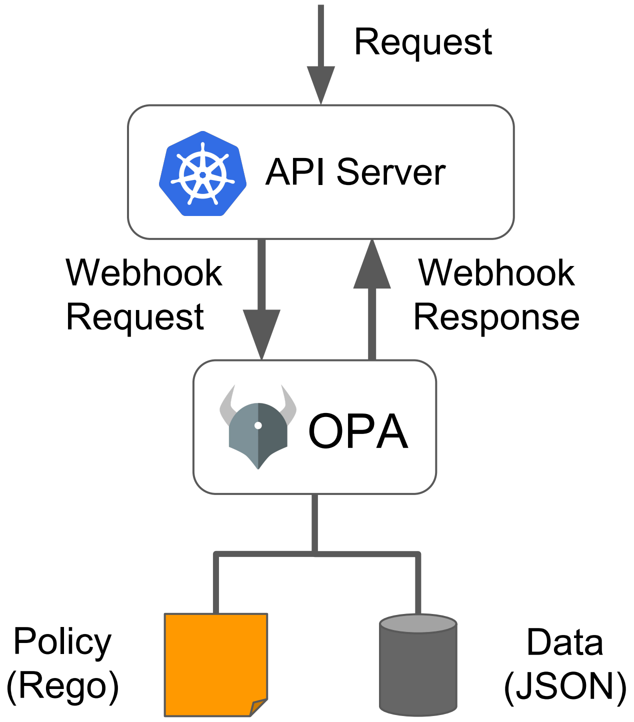
Risk management
Data Protection
Data Filtering

Today

CNCF project (Sandbox)

36 contributors
400 slack members
25K image pulls/week
20+ integrations

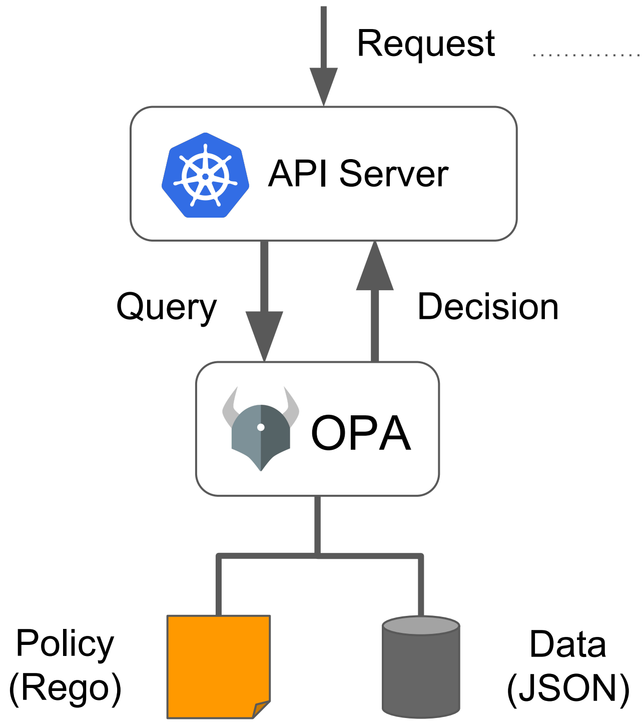
Why OPA for Admission Control with Kubernetes



- "Least privilege"
  - RBAC not enough, e.g., kubectl cordon
- Problem space
  - Complex inputs (YAML)
  - Organization-specific
  - Context-dependant
  - Enforce, audit, and dry-run
- Example policies
  - *"Require specific labels on deployments..."*
  - *"Restrict ingress hostnames…"*
  - *"Prevent vulnerable images…"*

openpolicyagent.org

OPA: Admission Control with Kubernetes



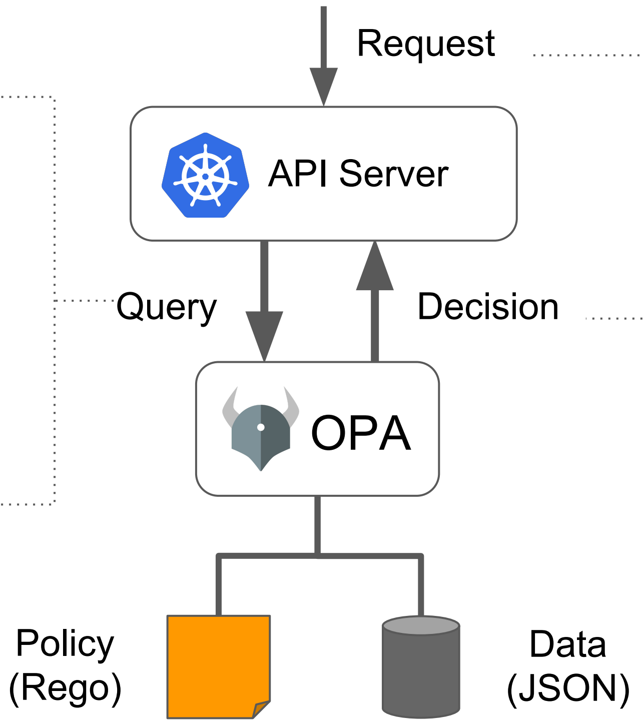
```
POST /namespaces/default/pods
Authorization: alice

<POD spec>
```

openpolicyagent.org

OPA: Admission Control with Kubernetes

```
metadata:
  name: web
  labels:
    app: web
spec:
  name: web
  containers:
    - name: nginx
      image: nginx
```



```
POST /namespaces/default/pods
Authorization: alice

<POD spec>
```

```
true or false and...
reason and...
JSON Patch (optional).
```

openpolicyagent.org

Want to learn more about OPA?

Join us at the Deep Dive session tomorrow! Topics:

- How OPA works
- New feature: Data filtering w/ SQL & Elasticsearch
- New feature: Rego → WebAssembly compiler

When: Thursday @ 11:40AM
Where: 3 A/B

Kubernetes Admission Control with Open Policy Agent

Zach Abrahamson
Capital One Cloud Engineering

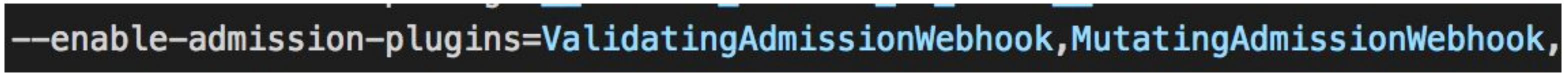
A little bit about me and what we’re building…

- Part of a systems engineering and software development team building a Kubernetes-based container platform for the enterprise
- We’re building tooling for k8s cluster provisioning as well as cluster lifecycle management
- A suite of services and integrations that sit on top of our clusters and play nice with other enterprise services (logging, metrics, etc.)
- Multi-cloud and multi-tenant

…all with the regulatory constraints of a financial institution

Kubernetes Admission Controllers

- Simply put, admission controllers are code that police the application of changes to a k8s cluster, in order to control how the cluster is used
  - Governance
  - Force desired behavior
- Admission controllers are configured to be loaded when the k8s API server boots:



```
--enable-admission-plugins=ValidatingAdmissionWebhook,MutatingAdmissionWebhook,
```

Why do we need admission control?

- Security
- Governance
- Configuration Management
- Scoping

As a financial institution, risk *is* our business. Admission control and policy management help make sure our apps are able to stay in compliance with our ever changing landscape of controls.

What we like about OPA Policies

- “A policy is a set of rules that governs the behavior of a service.”
- Written in Rego
- Multiple policies can be written for the same k8s objects
- Simply put, we define the data to be evaluated, and the policy by which the perform the evaluation
- Define policies with minimal LOC

Use Case – Image Registry Whitelist

```
package kubernetes.admission
import data.kubernetes.namespaces

deny[msg] {
     input.request.kind.kind = "Deployment"
     input.request.operation = "CREATE"
     registry = input.request.object.spec.template.spec.containers[_].image
     name = input.request.object.metadata.name
     namespace = input.request.object.metadata.namespace
     not reg_matches_any(registry,valid_deployment_registries)
     msg = sprintf("invalid deployment: whitelisted registry not found.,
          namespace=%q, name=%q, registry=%q",
          [namespace,name,registry])
}

valid_deployment_registries = {registry |
     whitelist = ”<internal registry url>"
     registries = split(whitelist, ",")
     registry = registries[_]
}
```

Use Case – Image Registry Whitelist

```
package kubernetes.admission
import data.kubernetes.namespaces

deny[msg] {
     input.request.kind.kind = "Deployment"
     input.request.operation = "CREATE"
     registry = input.request.object.spec.template.spec.containers[_].image
     name = input.request.object.metadata.name
     namespace = input.request.object.metadata.namespace
     not reg_matches_any(registry,valid_deployment_registries)
     msg = sprintf("invalid deployment: whitelisted registry not found.,
          namespace=%q, name=%q, registry=%q",
          [namespace,name,registry])
}

valid_deployment_registries = {registry |
     whitelist = ”<internal registry url>"
     registries = split(whitelist, ",")
     registry = registries[_]
}
```

Use Case – Image Registry Whitelist

```
package kubernetes.admission
import data.kubernetes.namespaces

deny[msg] {
     input.request.kind.kind = "Deployment"
     input.request.operation = "CREATE"
     registry = input.request.object.spec.template.spec.containers[_].image
     name = input.request.object.metadata.name
     namespace = input.request.object.metadata.namespace
     not reg_matches_any(registry,valid_deployment_registries)
     msg = sprintf("invalid deployment: whitelisted registry not found.,
          namespace=%q, name=%q, registry=%q",
          [namespace,name,registry])
}

valid_deployment_registries = {registry |
     whitelist = ”<internal registry url>"
     registries = split(whitelist, ",")
     registry = registries[_]
}
```

Use Case – Image Registry Whitelist

```
package kubernetes.admission
import data.kubernetes.namespaces

deny[msg] {
     input.request.kind.kind = "Deployment"
     input.request.operation = "CREATE"
     registry = input.request.object.spec.template.spec.containers[_].image
     name = input.request.object.metadata.name
     namespace = input.request.object.metadata.namespace
     not reg_matches_any(registry,valid_deployment_registries)
     msg = sprintf("invalid deployment: whitelisted registry not found.,
          namespace=%q, name=%q, registry=%q",
          [namespace,name,registry])
}

valid_deployment_registries = {registry |
     whitelist = ”<internal registry url>"
     registries = split(whitelist, ",")
     registry = registries[_]
}
```

With the OPA k8s admission controller, we can…

…control where container images are sourced

…enforce metadata on k8s resources (labels, annotations, etc.)

…prevent unwanted changes to k8s resources

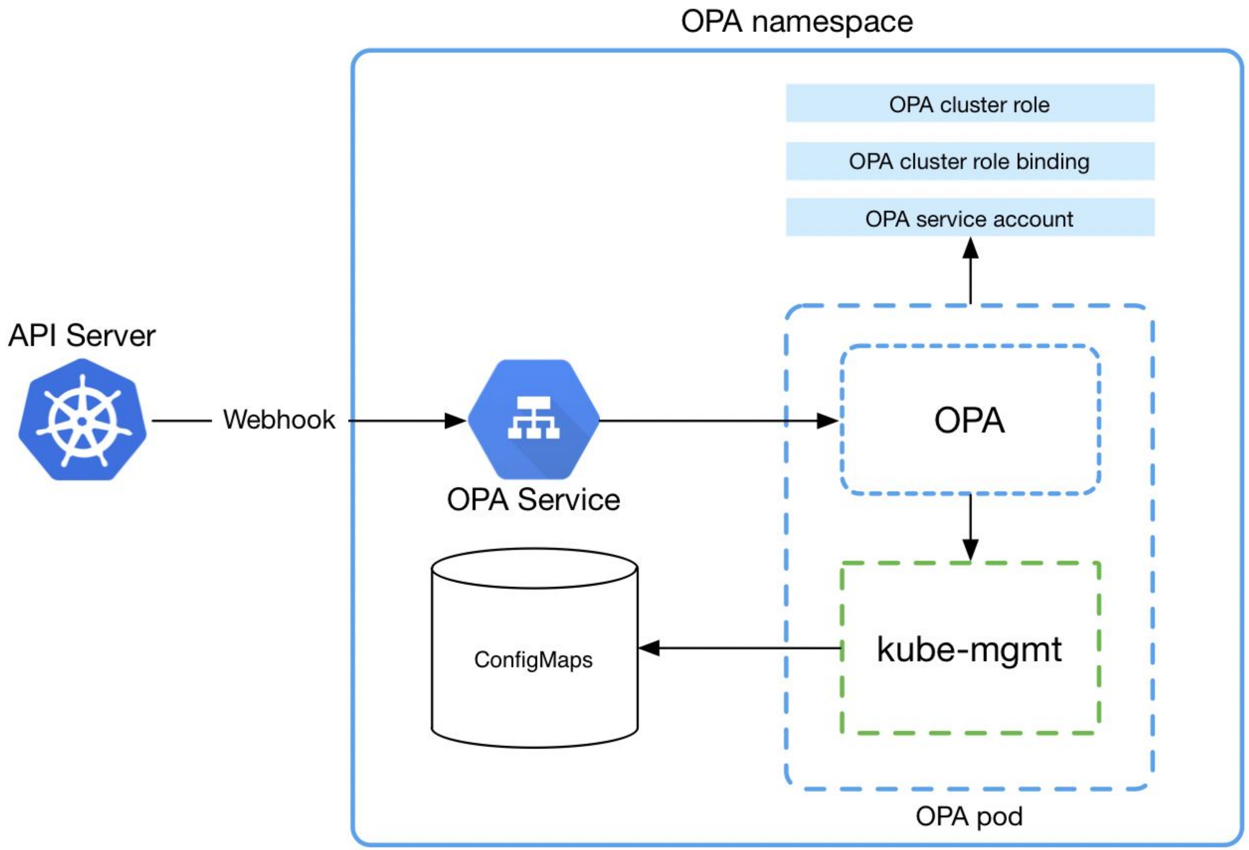
…utilize domain agnostic approach of OPA to define and implement additional governance, at a later date, as needed

Each of these are real use cases at Capital One for which we are leveraging OPA

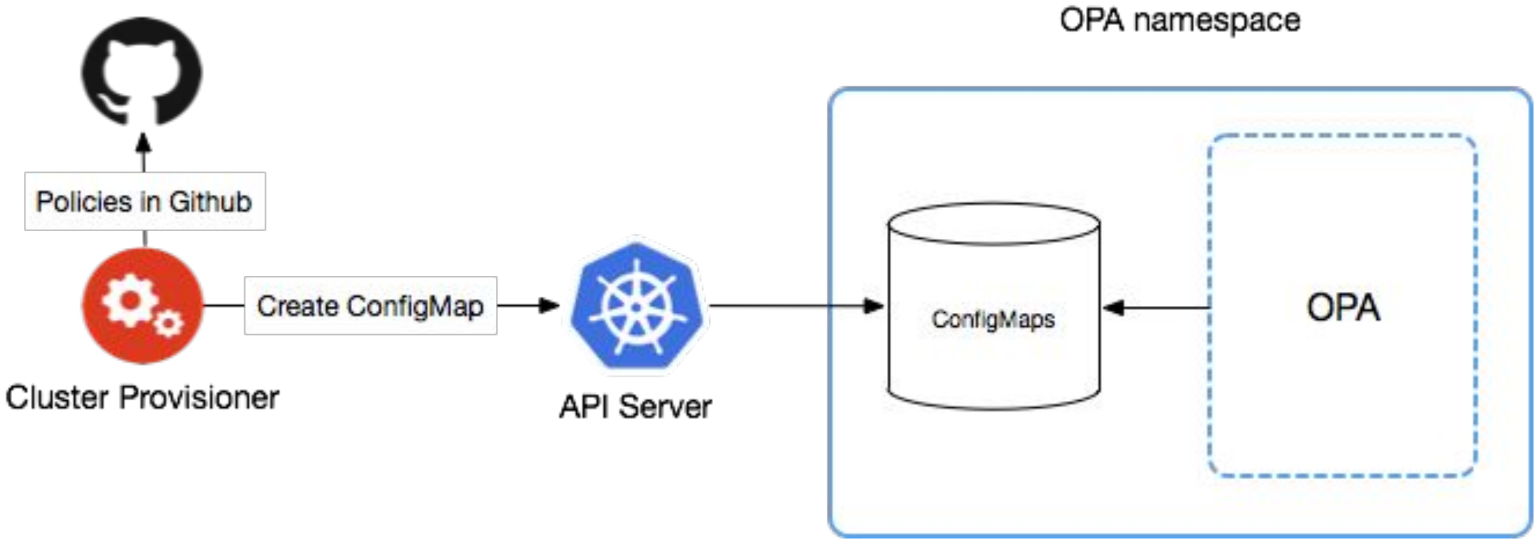
How we deploy OPA with Kubernetes

- Deployed as REST API in a container, fronted by a k8s service
- OPA pod uses service account, that uses cluster roles and cluster role bindings, to provide access to k8s resources
- Policies are uploaded to OPA via k8s config maps
- kube-mgmt side-car container compiles policies to verify they are correct, loads them into OPA, and replicates k8s resource data for OPA usage (deployment, namespace, etc.)

OPA with Kubernetes - Deployment Architecture



OPA with Kubernetes - Provisioning



Lessons Learned and Future Direction

- Make sure cluster NO_PROXY rules include “.svc”
  - This could lead to errors in connecting the webhook to the OPA server
  - “x509: certificate signed by unknown authority” – red-herring errors
- Pair with Cloud Custodian Kubernetes integration – currently WIP
- Would be nice to decouple policy management from cluster provisioning
  - Create a service that scans cluster fleet and applies/updates policies from version control as needed

We’re hiring!

Booth P11

https://www.capitalonecareers.com/

Thanks!

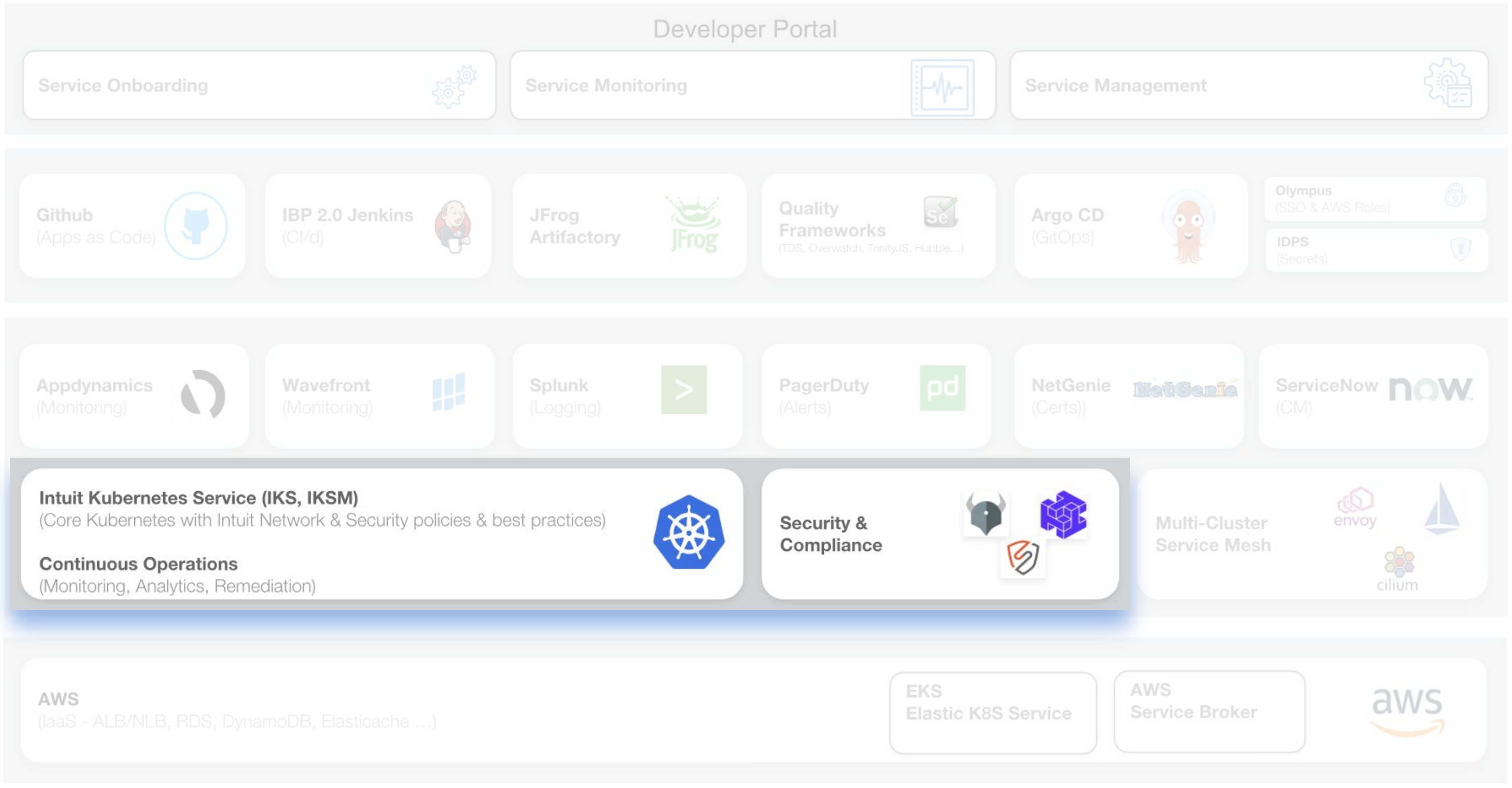
Medium article for this talk:
https://bit.ly/2BbEAZw

Using OPA to build secure multi-tenant K8s clusters at Intuit

Todd Ekenstam, Intuit, Modern SaaS Infrastructure

KubeCon 2018

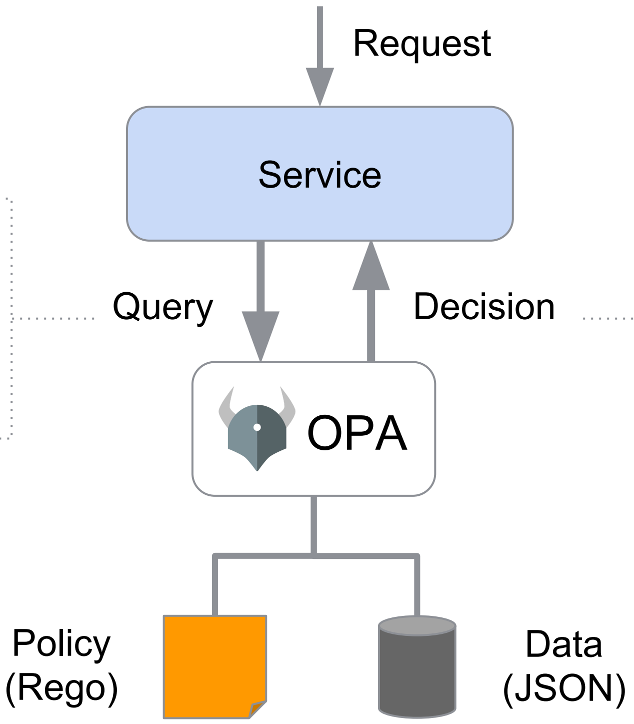
One Platform - Intuit Modern SaaS



Why OPA?

Validating &
Mutating
Webhooks...

Mutating Webhooks



Input can be ***any*** JSON value:

```
{"kind": "Pod", "spec": …}
```

Output can be ***any*** JSON value:

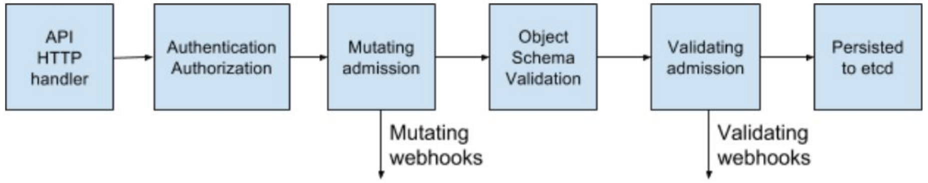
```
{"kind": "Pod", "spec": …}
```

Mutating Webhooks

Special type of `AdmissionController`

- `MutatingAdmissionWebhook`
    - Calls mutating webhooks which match request
    - Matching webhooks called in serial
    - Each one may modify the object if it desires
    - Only runs in the mutating phase

Lots of use-cases where this is useful!



Multi-tenant - Transparent NodeSelector and Tolerations

Enable Creation of Multi-tenant K8s Clusters

- Each tenant assigned one or more namespaces
- Pods in a namespace may only run on associated Nodes
- Patch node selector and tolerations of Pod specs with MutatingAdmissionWebhook

```
nodeSelector:
  kops.k8s.io/instancegroup: my-ig
tolerations:
- key: ig.iks.intuit.com/dedicated
  value: my-ig
```

“Repetition doesn’t spoil the prayer.”

- Brad Smith, CEO, Intuit

Security - Enforce Docker Image Repository Policy

- Validating webhook for pod creation to validate spec with OPA
- By default, all images must be pulled from private repository
- Allowed registries can be specified as namespace annotation
- Reject containers referencing image from invalid registry

```
spec:
  containers:
  - image: docker.io/library/nginx
```

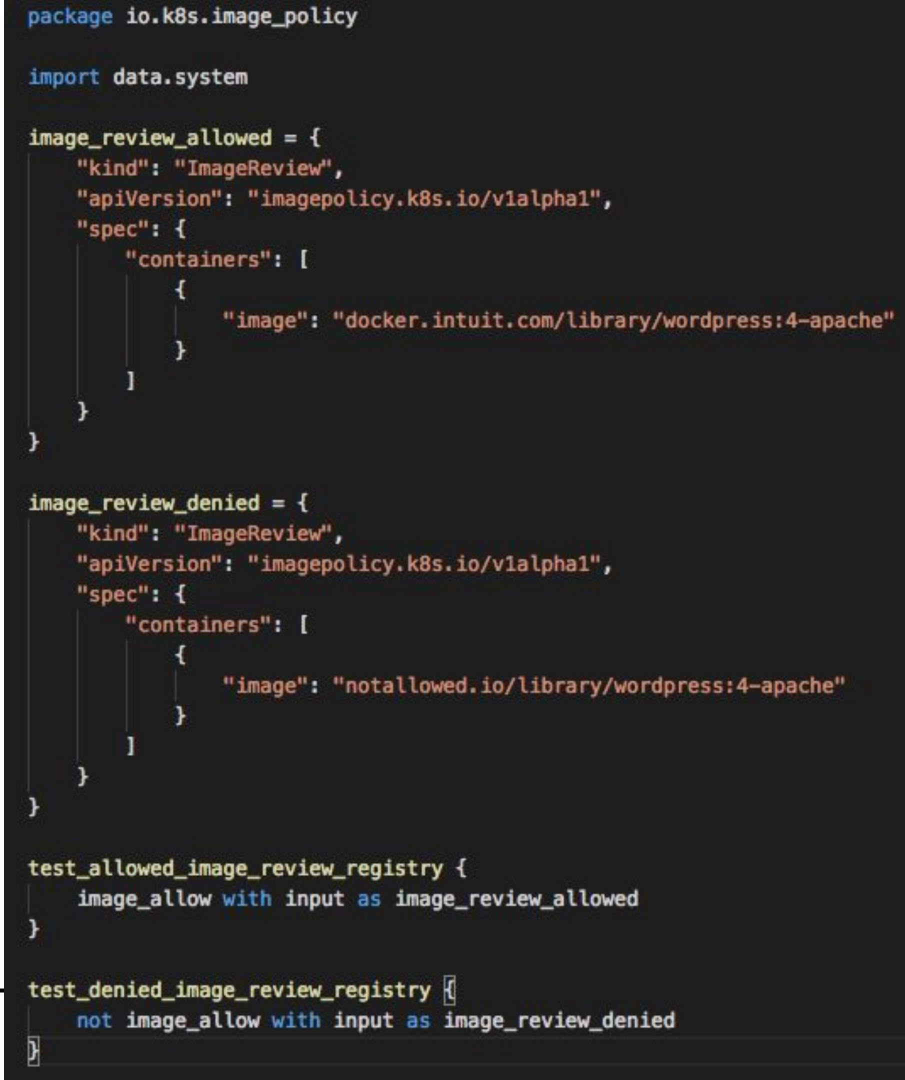
```
spec:
  containers:
  - image: docker.intuit.com/library/nginx
```

Other Use-cases

- Deny request to create LoadBalancer without specifying loadBalancerSourceRanges (to avoid default 0.0.0.0/0 access)
- Prevent Ingress objects without a security group ("Fail Fast")

Unit Testing

- OPA can run standalone, allowing unit test of policies
- OPA critical to K8s infrastructure; testing policies is important
- OPA policies can be complex, increasing need for unit testing



```
package io.k8s.image_policy

import data.system

image_review_allowed = {
    "kind": "ImageReview",
    "apiVersion": "imagepolicy.k8s.io/v1alpha1",
    "spec": {
        "containers": [
            {
                "image": "docker.intuit.com/library/wordpress:4-apache"
            }
        ]
    }
}

image_review_denied = {
    "kind": "ImageReview",
    "apiVersion": "imagepolicy.k8s.io/v1alpha1",
    "spec": {
        "containers": [
            {
                "image": "notallowed.io/library/wordpress:4-apache"
            }
        ]
    }
}

test_allowed_image_review_registry {
    image_allow with input as image_review_allowed
}

test_denied_image_review_registry {
    not image_allow with input as image_review_denied
}
```

Challenges/Future

DENY by default (`failurePolicy: Fail`) when OPA not available

- OPA is running inside the cluster it is managing
- But if OPA crashes, how to make API Server calls to fix it?
- Need HA OPA

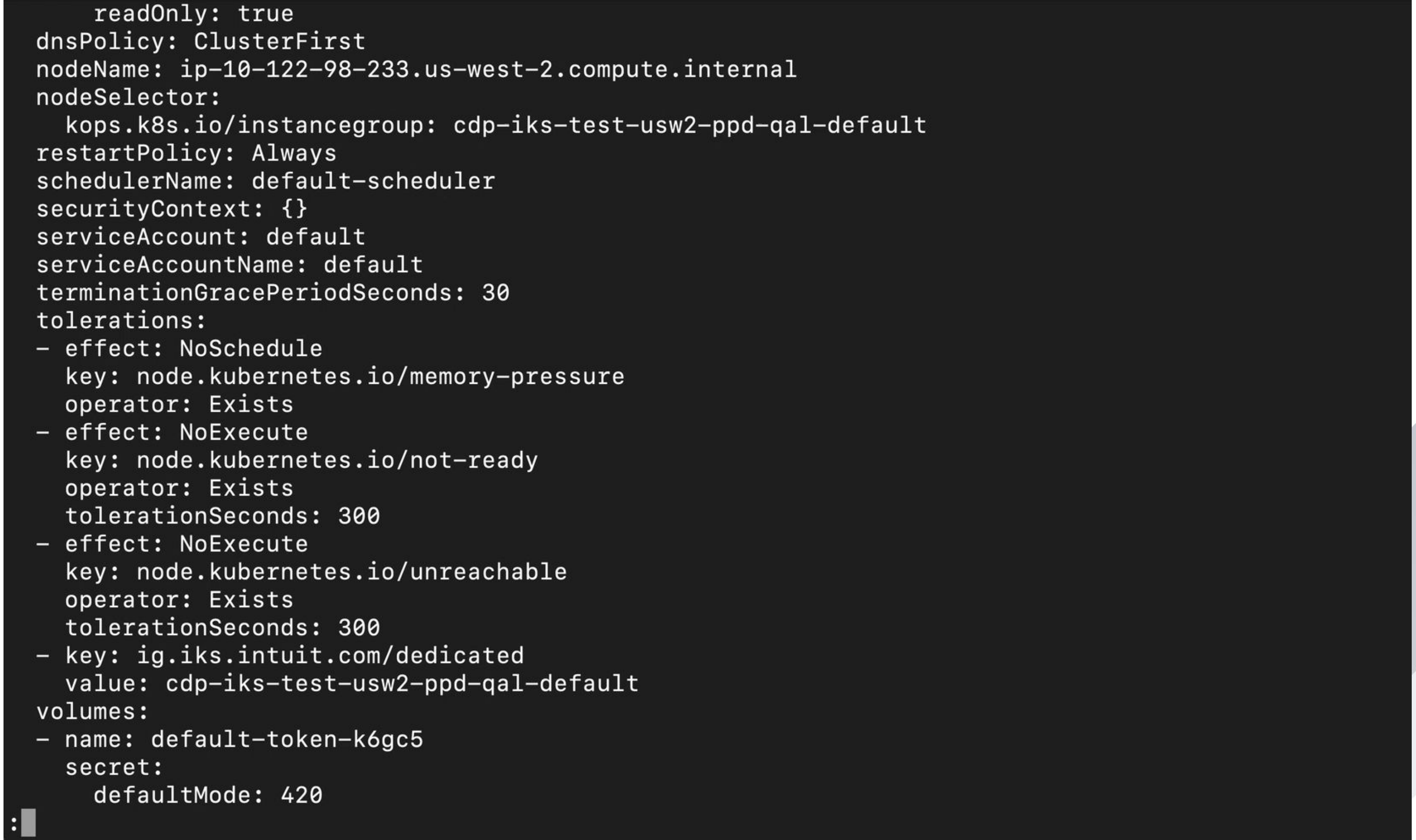
Would like “OPA CRD” to more easily apply policies

Vulnerability scanning data considered in ImagePolicyWebhook

More “Fail Fast” Checks

- Deny if new resource would exceed cloud limit
- Deny invalid Ingress objects (e.g. missing path and service)

Demo



```
      readOnly: true
  dnsPolicy: ClusterFirst
  nodeName: ip-10-122-98-233.us-west-2.compute.internal
  nodeSelector:
    kops.k8s.io/instancegroup: cdp-iks-test-usw2-ppd-qal-default
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoSchedule
    key: node.kubernetes.io/memory-pressure
    operator: Exists
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  - key: ig.iks.intuit.com/dedicated
    value: cdp-iks-test-usw2-ppd-qal-default
  volumes:
  - name: default-token-k6gc5
    secret:
      defaultMode: 420
:
```

Original Pod yaml vs Mutated Pod yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-server
spec:
  containers:
  - image: library/nginx:1.13
    imagePullPolicy: Always
    name: server
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-server
spec:
  containers:
  - image: library/nginx:1.13
    imagePullPolicy: Always
    name: server
  nodeSelector:
    kops.k8s.io/instancegroup: my-ig
  tolerations:
  - key: ig.iks.intuit.com/dedicated
    value: my-ig
```

Recognized as one of the world's leading companies:

MOST ADMIRED: SOFTWARE INDUSTRY

14 Years in a Row

2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 **2017**

FORTUNE WORLD'S MOST ADMIRED COMPANIES

Ranked #4

MOST INNOVATIVE COMPANIES

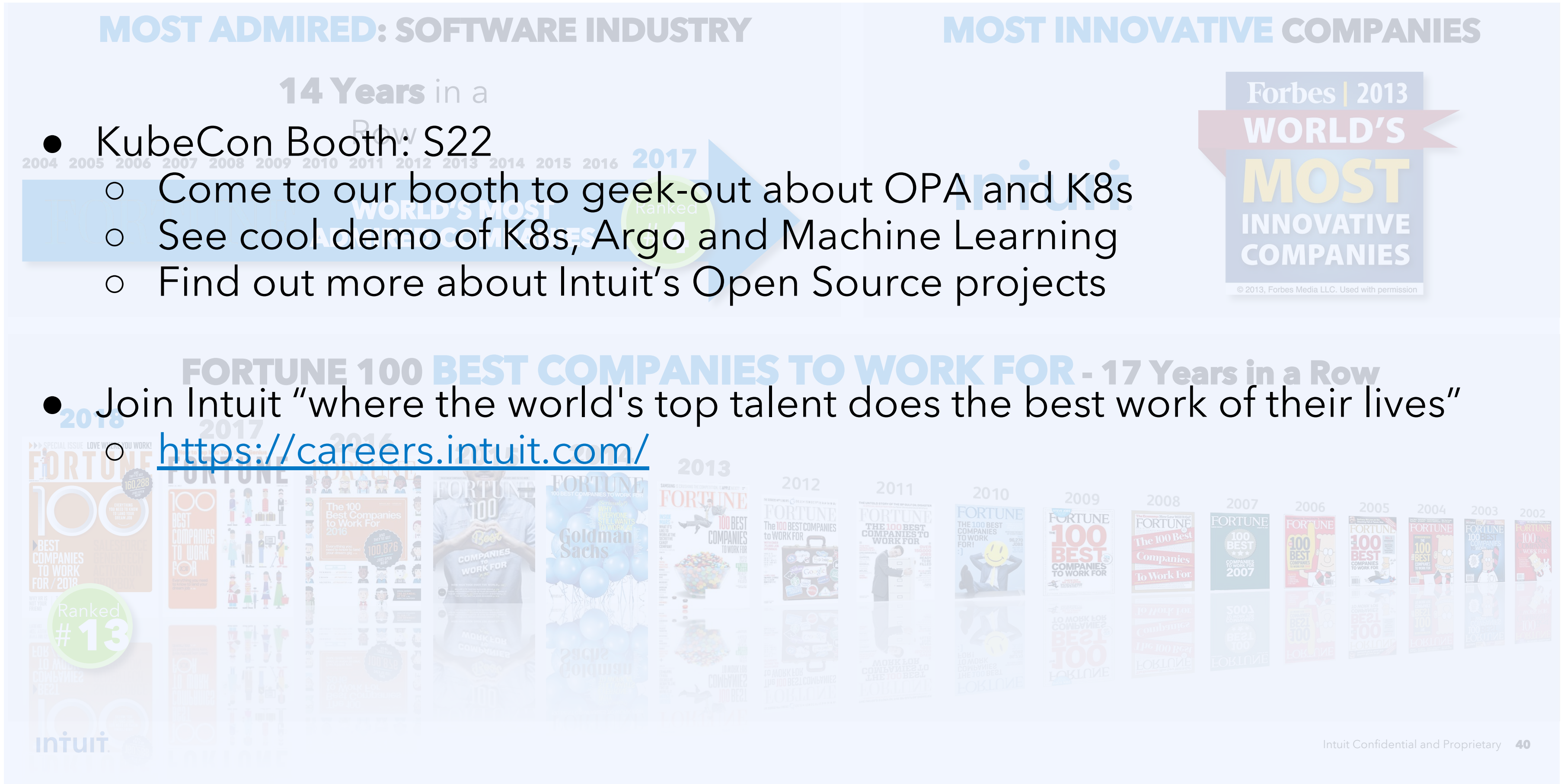
intuit

Forbes | 2013 WORLD'S MOST INNOVATIVE COMPANIES

© 2013, Forbes Media LLC. Used with permission

FORTUNE 100 BEST COMPANIES TO WORK FOR - 17 Years in a Row

Thank You


MOST ADMIRED: SOFTWARE INDUSTRY
14 Years in a Row
MOST INNOVATIVE COMPANIES
Forbes | 2013
WORLD'S MOST INNOVATIVE COMPANIES
© 2013, Forbes Media LLC. Used with permission
FORTUNE 100 BEST COMPANIES TO WORK FOR - 17 Years in a Row
Ranked #13
intuit

- KubeCon Booth: S22
  - Come to our booth to geek-out about OPA and K8s
  - See cool demo of K8s, Argo and Machine Learning
  - Find out more about Intuit’s Open Source projects

- Join Intuit “where the world's top talent does the best work of their lives”
  - https://careers.intuit.com/