

A photograph of a roller coaster at dusk. The tracks are silhouetted against a sky with soft, warm light from the setting sun. The coaster features several loops and drops. The overall mood is dramatic and adventurous.

# OPA Gatekeeper

**Policy and Governance for Kubernetes**

Max Smythe (@maxsmythe, Google)

Rita Zhang (@ritazzhang, Microsoft)

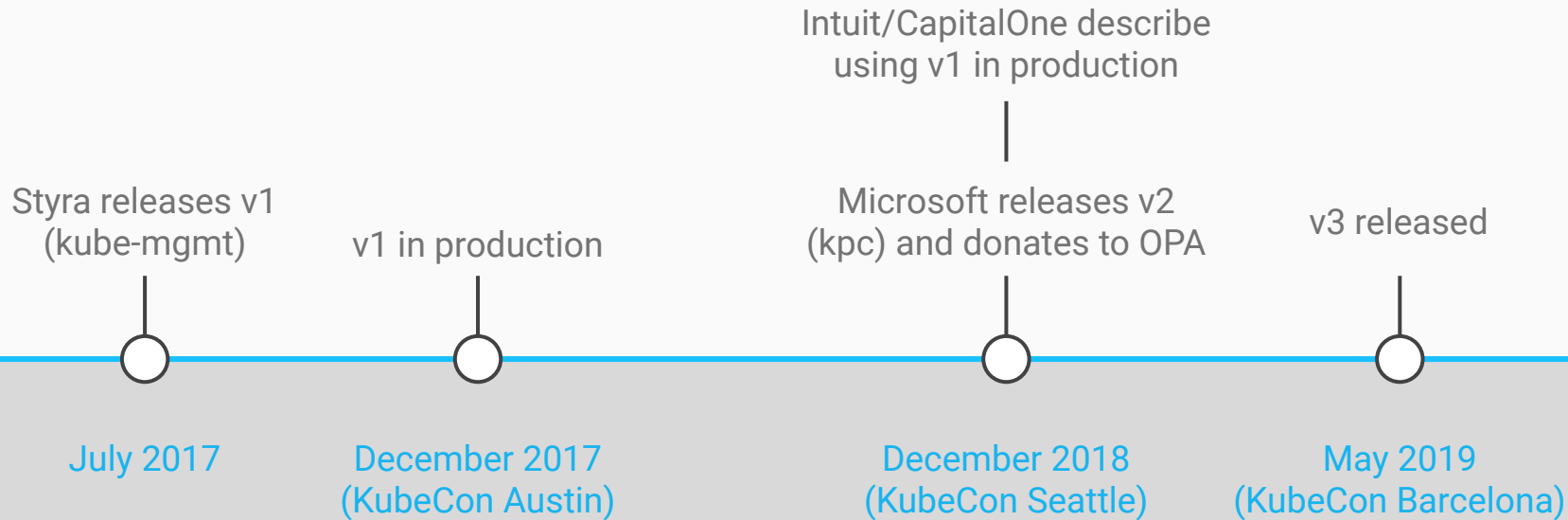
A customizable Kubernetes admission webhook that  
helps enforce policies and strengthen governance

# Motivation

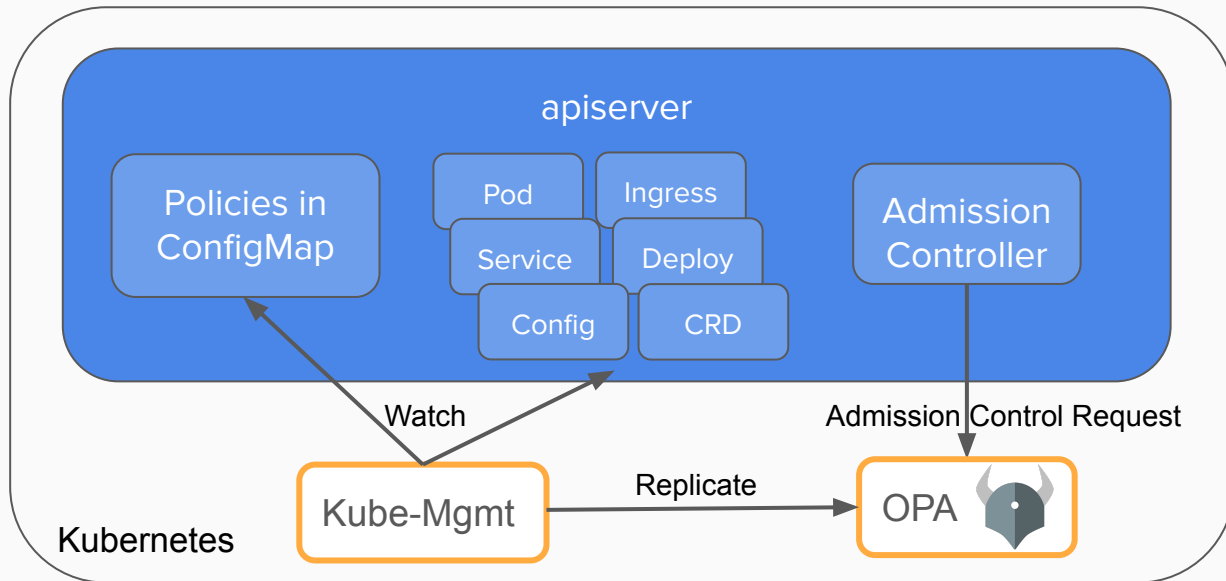
Primary user needs:

- Customizable admission via configuration (not code)
- Awareness of current cluster state (not just the 1 operation user is performing)

# How we got here

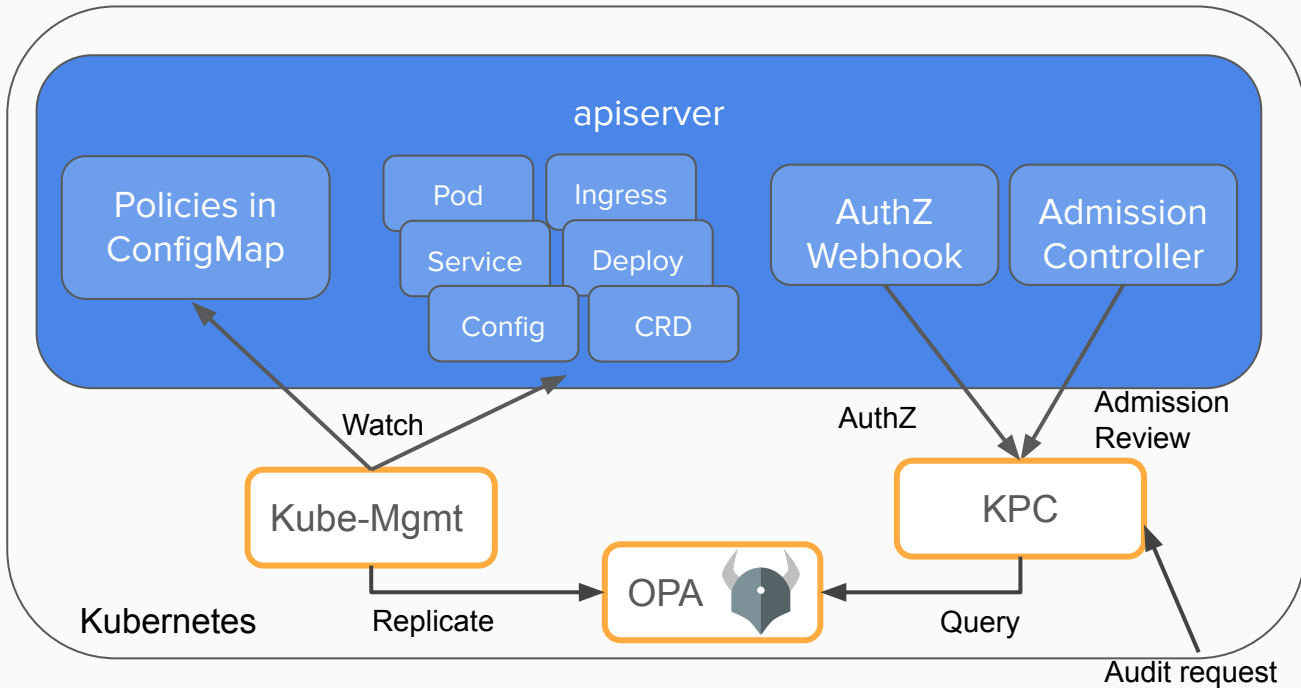


# OPA + kube-mgmt (aka OPA Gatekeeper v1.0)



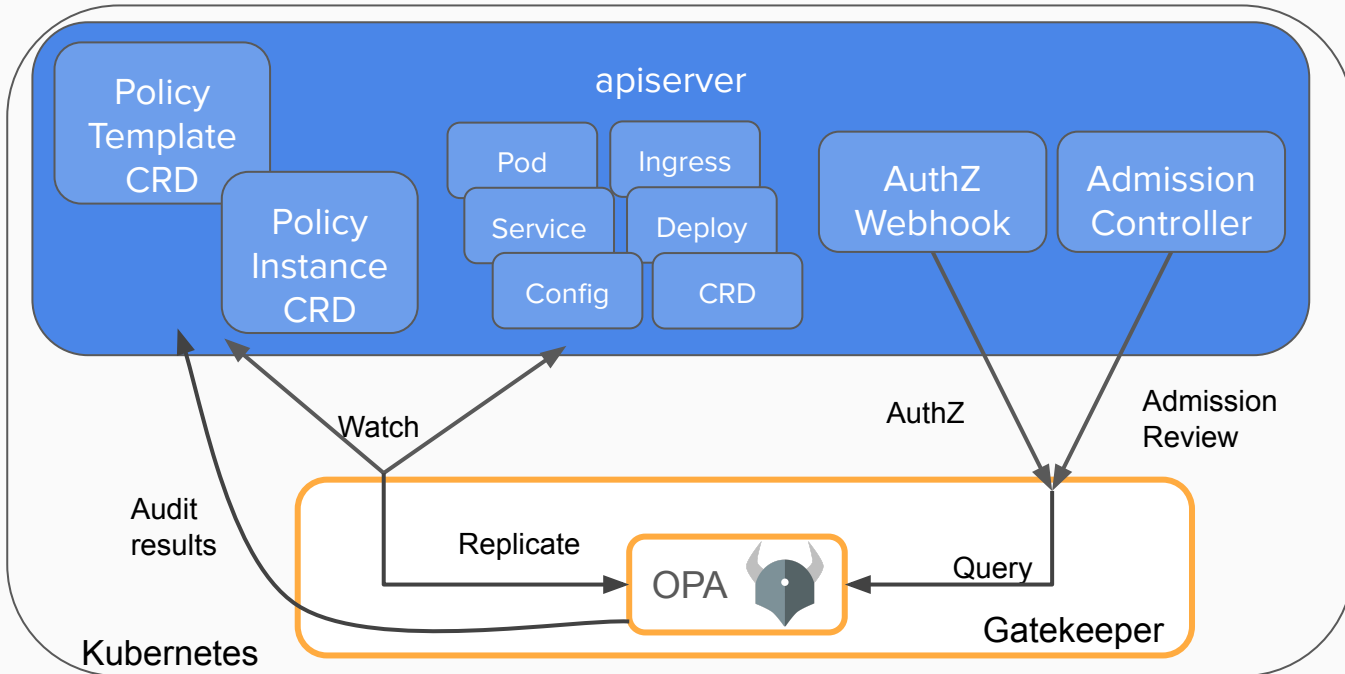
- Validating and mutating admission control.
- Policies in raw Rego (OPA's policy language)
- Policies stored in ConfigMaps with syntax-errors reported as annotations
- Donated by Styra
- Used by dozens of companies

# K8s Policy Controller (aka OPA Gatekeeper v2.0)



- Validating and **mutating** admission control. **Authorization. Audit.**
- Policies in raw Rego (OPA's policy language)
- Policies stored in ConfigMaps with syntax-errors reported as annotations
- [Full architecture](#)
- Donated by Microsoft to CNCF/OPA

# OPA Gatekeeper v3.0



- Validating admission. Audit. CI/CD.
- Policy templates (Rego) and instances
- Policies stored in CRDs
- Audit results stored on policy CRDs
- [Full architecture](#)
- Google, Microsoft, Redhat, CBA, Styra
- “Gatekeeper” donated by Replicated
- Built with kubebuilder

# Example Policies

- All namespaces must have a label that lists a point-of-contact
- All pods must have an upper bound for resource usage
- All images must be from an approved repository
- Services must all have globally unique selectors

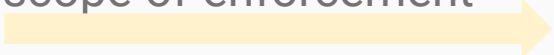


# Constraint Properties

- AND-ed together
  - Adding can only constrain, removing can only loosen
  - One rejection => whole request rejection
- Schema validation
  - Less error-prone
- Selection semantics
  - Only constrain certain kinds
  - Only constrain resources in a certain namespace
  - Use a label selector

# Constraints

Define  
scope-of-enforcement



```
apiVersion: constraints.gatekeeper.sh/v1alpha1
kind: K8sRequiredLabels
metadata:
  name: all-must-have-owner
spec:
```

```
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
```

Describe intent



```
  parameters:
    message: "All namespaces must have an `owner` label
that points to your company username"
    labels:
      - key: owner
        allowedRegex: "^[a-zA-Z]+.agilebank.demo$"
```

# Data Replication

- Constraints that compare against other objects in the cluster
  - Require replication of existing objects in the cluster
- Replication is defined via the config resource

```
apiVersion: config.gatekeeper.sh/v1alpha1
kind: Config
metadata:
  name: config
  namespace: gatekeeper-system
spec:
  sync:
    syncOnly:
      - kind: Service
        version: v1
      - kind: Pod
        version: v1
      - kind: Namespace
        version: v1
```

# Audit

- Periodically evaluates resources against constraints
- Allows for ongoing monitoring of cluster state to aid in detection and remediation of pre-existing misconfigurations
- Objects to be audited must be replicated
- Exposes results via the `status` field of the constraint

# Audit Results

```
apiVersion: constraints.gatekeeper.sh/v1alpha1
kind: K8sRequiredLabels
metadata:
  name: all-must-have-owner
spec:
  match:
    kinds:
      - apiGroups:
          - ""
          kinds:
            - Namespace
  parameters:
    labels:
      - allowedRegex: ^[a-zA-Z]+.agilebank.demo$
        key: owner
        message: All namespaces must have an `owner` label that points to your company username
  status:
    auditTimestamp: "2019-05-11T01:46:13Z"
    enforced: true
  violations:
    - kind: Namespace
      message: All namespaces must have an `owner` label that points to your company username
      name: default
    - kind: Namespace
      message: All namespaces must have an `owner` label that points to your company username
      name: gatekeeper-system
    - kind: Namespace
      message: All namespaces must have an `owner` label that points to your company username
      name: kube-public
    - kind: Namespace
      message: All namespaces must have an `owner` label that points to your company username
      name: kube-system
```

## Time of Audit



## Violations



# Constraint Templates

- Rego rule signature
  - If the rule matches, the constraint is violated
- Schema for Constraint Parameters
- Matching semantics provided by the system
  - Currently we can select off of Group/Version/Kind, namespace, or labelSelector
  - As more ways of specifying scope are discovered, existing constraints get them for free

```
apiVersion: templates.gatekeeper.sh/v1alpha1
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:
            message:
              type: string
            labels:
              type: array
              items:
                type: object
                properties:
                  key:
                    type: string
                  allowedRegex:
                    type: string
```

## Schema for input parameters

deny[{"msg": msg}]  
rules are executed

```
targets:
- target: admission.k8s.gatekeeper.sh
  rego: |
    package k8srequiredlabels

    # helper libraries and additional tests (e.g. regex match) not shown

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

# Code Reuse!

- Gatekeeper provides the admission system
- Policies/Rules are implemented through constraints
- Constraints are parameterized and easily configurable by admins
- ConstraintTemplates provide the source code for constraints
  - Easily shared
  - Testable
  - Developed internally or sourced from the community
- Portable to other systems
  - e.g. CI/CD pipelines





Demo

# Project Status

- Alpha
- Come help!
  - Issues
  - Feedback
  - User stories
  - Development



Photo by [Tikkho Maciel](#) on [Unsplash](#)

Cooking... but tasty

# Potential Growth

- Mutation
- External Data
- Authorization? (likely separate project, same general semantics)
- More audit features
- Metrics
- Developer tooling
- Dry run

# Thank You

Open Policy Agent Community

Andrew Block (Red Hat)

Craig Hooper (Commonwealth Bank AU)

Lachlan Evenson (Microsoft)

Nikhil Bhatia (Confluent, ex-Microsoft)

Marc Campbell (Replicated)

Dexter Horthy (Replicated)

Tim Hinrichs (Styra)

Torin Sandall (Styra)

# Join Us!



## Open Policy Agent

[openpolicyagent.org](https://openpolicyagent.org)  
[github.com/open-policy-agent/opa](https://github.com/open-policy-agent/opa)

## OPA Gatekeeper

[github.com/open-policy-agent/gatekeeper](https://github.com/open-policy-agent/gatekeeper)



## Community

[slack.openpolicyagent.org](https://slack.openpolicyagent.org)  
[#kubernetes-policy](#)

[Meetings](#) Tue @2p Pacific