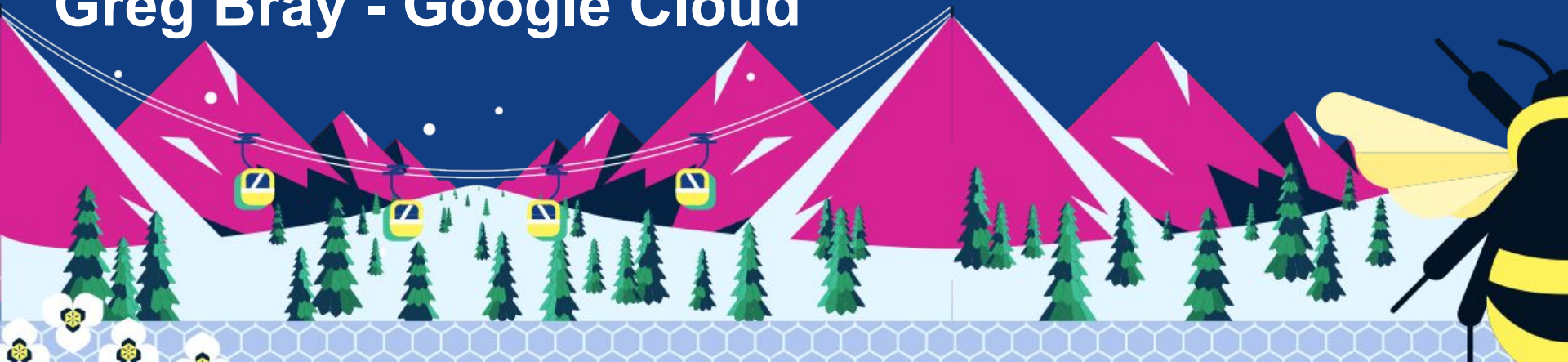


#KubeCon #CloudNativeCon
Social links at **https://a-z.dev**

Goodbye Webhooks, Hello CEL:
Using Native Validating and Mutating Admission Policies in Kubernetes

Greg Bray - Google Cloud



Session Outline

01 **Webhooks: D'oh!**
The cause of, and solution to, all Platform Engineering problems

02 **CEL: An Elegant Weapon for a More Civilized Age**
Common Expression Language fundamentals and safety guarantees

03 **VAPs and MAPs and Bindings (oh my!)**
The wonderful world of ValidatingAdmissionPolicy and MutatingAdmissionPolicy

04 **You're Going to Need a Bigger Policy Engine**
Using Gatekeeper or Kyverno as a frontend for VAP/MAP and more complex policies

05 **Down the Rabbit Hole**
Guidance and links to other sessions and admission control policy examples



`$ id gregbray  # whoami`

Google Cloud Customer Engineer
Lifelong Utahn & U of U Alum
Former SRE: Reddit | WalmartLabs | Stack Overflow
Social Links at **https://a-z.dev**

Kubernetes Admission Controls

More details in [Admission Control in Kubernetes](#) docs

Admission Control Plugins

A number of Kubernetes features are implemented as in-tree plugins that get executed when processing kube-api requests. Initially plugins were the only option, but now using CEL policies or webhooks are preferred.

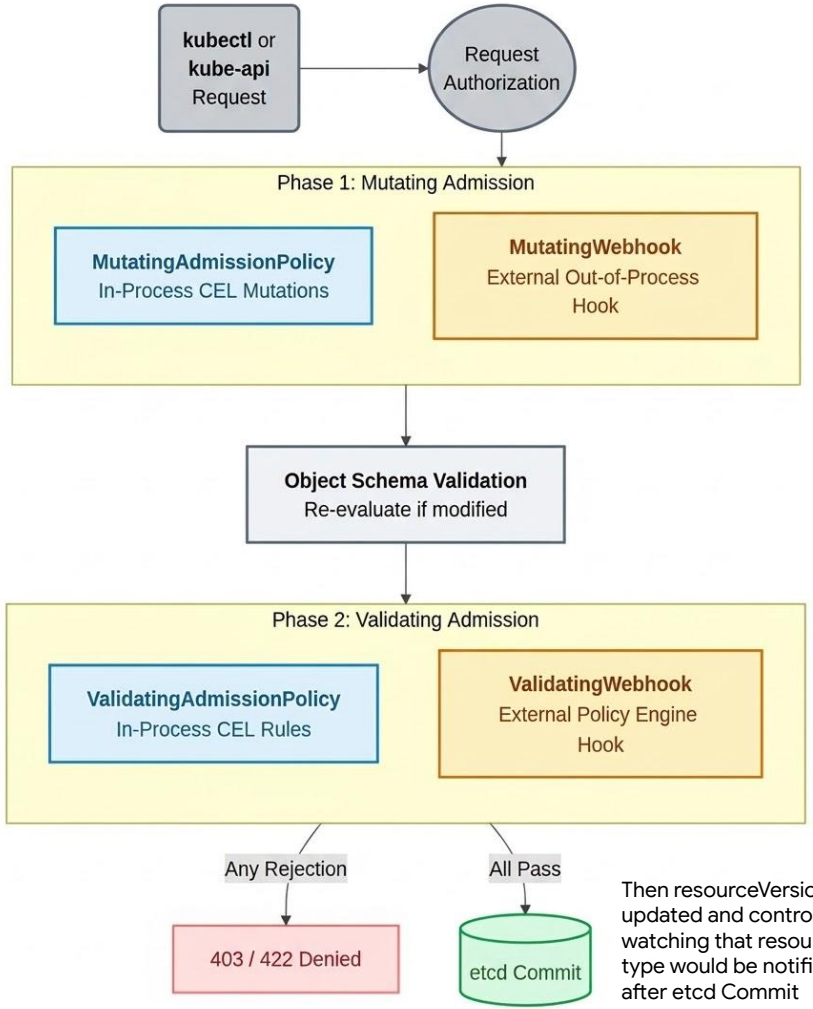
Mutation Policies and Webhooks

Allow you to modify resources used in CREATE, DELETE, or MODIFY API requests, or requests using custom verbs.

These run serially with possible reinvocation if requested.

Validation Policies and Webhooks

CEL validation policies run serially in a loop, and if those all pass then matching webhooks are called in parallel.



Pain Points of Webhooks: Latency, TLS Complexity, & Cluster Outages

Out-of-process webhooks introduce significant operational risks and performance overhead to the control plane.



Network Latency

- Every external webhook call can add **milliseconds** to API requests
- Slows down cluster scaling and large scale deployments



TLS Complexity

- Securing API-server to webhook communication requires (correctly!) managing TLS certs
- Rotating certificates is notoriously error-prone and easy to forget until things break



The 'Fail Closed' Risk

- If a webhook pod crashes or has networking issues the system may default to a 'fail closed' state as a safeguard
- Could block all cluster operations, taking down the entire Kubernetes control plane

Common Expression Language (CEL): an open source, non-Turing complete, memory-safe language that guarantees linear evaluation time
See Kubernetes API and Gateway API usage guide for boolean expression examples

```
expression: object.metadata.name.endsWith('.gateway.networking.k8s.io')
expression: '!("system:nodes" in request.userInfo.groups)'
expression: |
  has(object.metadata.annotations) && 'gateway.networking.k8s.io/channel' in object.metadata.annotations
  && object.metadata.annotations['gateway.networking.k8s.io/channel'] == 'standard'
```

similar to above but using CEL Optional Values (added v1.29)

```
matchConditions:
  - expression: object.metadata.?annotations["gke-gcsfuse/volumes"].orValue("") == "true"
validations:
  - expression: 'has(variables.sidecar.restartPolicy) && variables.sidecar.restartPolicy == "Always"'
variables:
  - expression: object.spec.initContainers.filter(c, c.name == "gke-gcsfuse-sidecar")[0]
    name: sidecar
```

CEL features	Included
Standard macros/functions	`has`, `all`, `exists`, `exists_one`, `map`, `filter` see cel.dev/reference/api-reference
Extended strings library	`charAt`, `indexOf`, `lastIndexOf`, `lowerAscii`, `upperAscii`, `replace`, `split`, `join`, `substring`, `trim`
Kubernetes CEL libraries	Custom libraries for lists, regex, urls, IP/CIDRs, formatting, quantities, semver, etc

Other projects using CEL

envoy

Istio

Kyverno

TEKTON

Google Cloud

The CEL Advantage: In-Process Evaluation Inside the API Server

Native CEL policies execute directly within the API server, eliminating network hops and external dependencies.

01

Zero Network Latency

Common Expression Language (CEL) rules are compiled and executed in-process, offering **sub-millisecond** execution times.

02

Safe and Bounded

CEL is strictly sandboxed, guaranteeing termination and bounding memory usage so a bad policy cannot crash the API server.

03

Simplified Infrastructure

By evaluating policies natively, you completely eliminate the need for external webhook servers, load balancers, and complex TLS bootstrapping.

Custom Resources for Admission, Bindings, and Parameters

`ValidatingAdmissionPolicy` (**VAP**) graduated to stable in v1.30 (April 2024)

- **Policy Definition** - The `ValidatingAdmissionPolicy` resource sections include:
  - *validations* - Core CEL expressions representing the policy requirements
  - *matchConditions* - CEL expression that **must be true** or policy will be skipped
  - *matchConstraints* - broadly what resources the policy **could** apply to
  - *failurePolicy* - *fail* for hard close or *ignore* for fail open
  - and optional sections for *variables* (reuse CEL derived values) or *paramKind* (static values from ConfigMap/etc or custom parameters CRD)
- **Policy Binding:** The `ValidatingAdmissionPolicyBinding` resource connects a VAP *policyName* to specific namespaces, users, or resource types using *matchResources* (further narrows scope down from *matchConstraints*), and includes *validationActions* (Deny, Warn, Audit, or [Warn,Audit] for both)

Custom Resources for Mutation, Bindings, and Parameters

`MutatingAdmissionPolicy` (**MAP**) graduated to stable in v1.36 (April 2026)
Same general shape as VAP but with a *mutations* section instead of *validations* section

- **Policy Definition -** Describes the MAP abstract CEL logic and schema requirements:
  - *mutations* - contains either a **declarative** *applyConfiguration* section using the server side apply merge strategy or an **imperative** RFC 6902 jsonPatch section
  - *reinvocationPolicy* - *Never* (default), or *IfNeeded* to re-run after other mutations
  - *Same failurePolicy, matchConditions, matchConstraints, paramKind,* and *variables*

- **Policy Binding:** The `MutatingAdmissionPolicyBinding` resource connects a MAP *policyName* to specific namespaces, users, or resource types again using *matchResources* (further narrows scope down from *matchConstraints*) and optional *paramRef* for the specific ConfigMap or CRD instance to use as *params.**

Full Example: Deny Anonymous Access Role Bindings

https://github.com/kubescape/cel-admission-library/blob/main/controls/C-0262/policy.yaml

```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "kubescape-c-0262-deny-anonymous-access-in-role-bindings"
  labels:
    controlId: "C-0262"
  annotations:
    controlUrl: "https://kubescape.io/docs/controls/c-0262/"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["rbac.authorization.k8s.io"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["rolebindings", "clusterrolebindings"]
  validations:
    # subjects is optional on a binding, so it needs the guard.
    - expression: >
```

```
  validations:
    # subjects is optional on a binding, so it needs the guard.
    - expression: >
        !has(object.subjects) || object.subjects.all(subject,
          subject.name != 'system:anonymous' &&
          subject.name != 'system:unauthenticated'
        )
      messageExpression: >
        object.kind + '/' + object.metadata.name + ' grants permissions to anonymous users, so
anyone who can reach the API server without credentials gets them. (see more at
https://kubescape.io/docs/controls/c-0262/)'
---
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "c-0262-deny-anonymous-role-binding"
spec:
  policyName: "kubescape-c-0262-deny-anonymous-access-in-role-bindings"
  matchResources:
    namespaceSelector:
      matchLabels:
        platform.a-z.dev/kubescape: "enabled"  # can also use MAP/VAP to apply/protect labels
  validationActions:
    - Deny
```

MutatingAdmissionPolicy: Inject Sidecar as initContainer

https://kubernetes.io/docs/reference/access-authn-authz/mutating-admission-policy/

```
spec:  (some lines omitted to fit on slide)
  matchConstraints:
    resourceRules:
    - operations:  ["CREATE"]
      resources:   ["pods"]
  matchConditions:
    - name: does-not-already-have-sidecar
      expression: "!object.spec.initContainers.exists(ic, ic.name == \"mesh-proxy\")"
  failurePolicy: Fail
  reinvocationPolicy: IfNeeded
  mutations:
    - patchType: "ApplyConfiguration"
      applyConfiguration:
        expression: >
          Object{spec: Object.spec{initContainers: [
                Object.spec.initContainers{
                  name: "mesh-proxy",
                  image: "mesh/proxy:v1.0.0",
                  args: ["proxy", "sidecar"],
                  restartPolicy: "Always"
                }]}}
```

New Kubernetes 1.37 Beta Feature: Manifest-Based Admission Control

https://kubernetes.io/docs/reference/access-authn-authz/manifest-admission-control/

Load admission webhooks and VAP/MAP policies immediately on startup from static files on KCP (similar to Static Pods)

The Issue

Admission policies registered through the Kubernetes API have several inherent limitations: **bootstrap gap** (not enforced until processed by API), **self-protection gap** (don't apply to webhooks/VAP/MAP resources), and a **dependency on etcd** being healthy.

The Fix

Manifest-based admission control fixes these limitations. The policies are active as soon as the API server is ready to serve requests and cannot be viewed or changed from the Kubernetes API. They are independent of etcd and can also intercept operations on webhook/VAP/MAP resources. Example: Deny delete/modify on protected policies.

The How

Use apiserver `AdmissionConfiguration`with `staticManifestsDir` section for each Admission Control plugin type you want to load from disk. Policy configurations should exist in isolation and cannot reference API resources. All objects **must have names** ending with the .static.k8s.io suffix like protect-admission-resources.static.k8s.io.

Full Example:
https://kubernetes.io/docs/reference/access-authn-authz/manifest-admission-control/#protecting-admission-resources

Architectural Tradeoffs: Native Policies

Native CEL offers unmatched simplicity and performance, but can fall short when you need more advanced policies

- Cannot reference external-datasource or even other Kubernetes resources (just the new/old resource in the current request)
- CEL Cost Unit budgets (~1,000,000) will limit your expression complexity to ensure runtime latency
- Mutation policies can only change the current resource, cannot spawn additional resources
- Difficult to manage VAP/MAP across fleet of clusters

RECOMMENDATION

Choose VAP/MAP for the >80% of common Admission Control use-cases that don't need to reference external data or use complex logic

vs. Kyverno, Gatekeeper, Etc

Much more mature enterprise focused systems, often with monitoring/testing/audit frameworks and a large existing libraries of common policies you can immediately start using

- Can cross-reference or even manipulate other Kubernetes or external resources as part of validation and mutation policy
- Can also do background/batch scanning of existing resources vs only checking items at admission time
- Often have much better testing and authoring tools
- Have access to better logic and libraries when using a full Turing-complete language
- Often will integrate with native VAP/MAP policies

RECOMMENDATION

As your requirements continue to grow, find a more advanced policy engine that supports native policies alongside complex CEL/Rego/Wasm/etc based policies

Down The Rabbit Hole

PDF of slides included on KubeCon schedule
My social links are listed at a-z.dev

Learn CEL and use it for basic policies or performance-critical systems. Evaluate full policy engines for complex enterprise governance or as a frontend to help manage both in-process and webhook based policies

Also when crafting policies add safeguards (exclusions for namespace/label/users) and runbooks for how to disable if needed

- Interesting Projects: Kyverno, OPA Gatekeeper, Kubewarden, Kubescape, Polaris, Cloud Custodian
- https://nirmata.com/2025/10/01/kyverno-vs-kubernetes-policies-how-it-complements-and-completes

- Kubescape library https://github.com/kubescape/cel-admission-library
- Kyverno policy library https://github.com/kyverno/policies
- Gatekeeper library https://github.com/open-policy-agent/gatekeeper-library

A few great sessions from KubeCon EU last year:

- EVAPorating Kubernetes Security Risk: Adopting Validating Admission Policy at Scale - Kaitlyn Lee & Jordan Conard, Datadog https://kccnceu2025.sched.com/event/1txFk
- A Practical Guide To Kubernetes Policy as Code https://kccnceu2025.sched.com/event/1tcxh
- Unlocking the Future of Kubernetes Policy as Code With Kyverno Vishal Choudhary & Frank Jogeleit https://kccnceu2025.sched.com/event/1td0G

Session Feedback Link