KubeCon
CloudNativeCon
North America 2024

KubeCon | CloudNativeCon

North America 2024

Open Policy Agent (OPA) Intro, Deep Dive & 1.0 Update

Charlie Egan (@charlieegan3) – Styra

Rita Zhang (@ritazh) – Microsoft

What is Policy?

Policy defines allowed operations and controls system behaviors

- Authorize users & workloads based on organizational requirements
- Grant Kubernetes access to teams
- Implement API access controls
- Define dataset & filtering permissions
- Control actions taken by CI/CD jobs

What is Policy?

“Only currently on-call members of the support team can trigger deployments to production without review on the weekend.”

What is Policy as Code?

What is Policy?

"Only currently on-call members of the support team can trigger deployments to production without review on the weekend."

What is Policy as Code?


```
package deployments.production

import rego.v1

default allow := false

allow if {
    is_weekend

    "support" in input.user.roles

    input.user.id in data.oncall_users
}

# ...
```

What is Policy as Code?


```
package deployments.production

import rego.v1

default allow := false

allow if {
    is_weekend

    "support" in input.user.roles

    input.user.id in data.oncall_users
}

# ...
```

Why does **Policy as Code** work?

Developers can focus on delivering business value

- Decouple policy decision logic from applications

Version controlled configurations aid rollbacks and audits

- Use code review to control changes and create versioned policy artefacts

Security settings can be shared and managed centrally

- Easily share and supplement organizational policies with team-specific rules

Code can be collaborated on, tested and statically analyzed

- Policy logic benefits from all the modern coding tools we have

What is Open Policy Agent ?

It’s the building blocks of your Policy as Code Platform

What is OPA?

- Open source, general purpose policy engine
- Graduated CNCF project since January 2021
- Unified toolset and framework for working with policy across the stack
- Decouples policy from application logic
- Separates policy decision from enforcement
- Policies written in declarative language Rego

What is OPA?

Domain Specific Language for Policy

```
package example

import rego.v1

default allow := false

allow if user.role == "admin"
```

Rego

+

Policy Server

APIs

- Policy Evaluation
- Policy Reloading
- Decision Logging

...

=

OPA!

OPA's Decision Model

OPA's Decision Model

See the Ecosystem page on the OPA website!

OPA's Decision Model

Input → Policy → Decision

```
{
    "request": {
        "method": "PUT",
        "path": ["users", "anders"]
    },
    "user": {
        "name": "peter",
        "roles": [
            "developer"
        ]
    }
}
```

```
package policy

default allow := false

allow if "admin" in input.user.roles

allow if {
    input.request.method == "GET"
    input.request.path[0] == "users"
}

allow if {
    input.request.method == "PUT"
    input.request.path == ["users", input.user.name]
}
```

```
{
    "allow": false
}
```

OPA's Decision Model

OPA's Responsibilities

OPA Community

- **Vibrant community** of users and contributors
- **100+** Integrations
- **9K+** GitHub Stars
- **8K+** Slack Users
- **Hundreds of millions** of downloads
- **OPA Gatekeeper, Conftest**
- **Editor integrations**

OPA Works!

“OPA provides us a generic way to apply policy consistently across all our services and systems”

Yao Weng, Bloomberg

(2024)

https://www.youtube.com/watch?v=zPRoybQm_d4

OPA 1.0

- **The target release date for OPA 1.0 is December 2nd!**
- Milestone release that consolidates OPA functionality for years to come.
- OPA 1.0 will still have a backwards compatible mode.

0.65.0 0.66.0 0.67.0 0.68.0 0.69.0 0.70.0 **1.0.0!**

OPA 1.0

Users will have two upgrade options:

- Update their Rego to be v1 compatible
  - Recommended for most users
  - We have CLI tools to make the process largely automatic
- Use the `--v0-compatible` functionality after upgrading
  - Recommended only for those running 3rd party Rego policies
  - Users are still encouraged to upgrade policies where possible
  - Come to the KubeCon Kiosk to discuss in detail

This presentation will show you how!

OPA 1.0

- The `if` and `contains` keywords will be mandatory
  - Resolve some ambiguities in the Rego language
- “Future” keywords become the new Rego v1 syntax
- Deprecated built-in functions will no longer work
- Many strict mode violations will be defaults
  - Duplicate imports are prohibited
  - `input` and `data` keywords are reserved
  - Use of deprecated functions is an error

https://www.openpolicyagent.org/docs/latest/opa-1/

OPA 1.0

Current OPA

```
package example

import future.keywords.every
import future.keyword.in

l := [1, 2, 3]

default allow := false

allow {
  count(violations) == 0
}

violations[msg] {
  every x in l {
    x > 0
  }
  msg := "no negative entries"
}

```

OPA 1.0

```
package example

l := [1, 2, 3]

default allow := false

allow if count(violations) == 0

violations contains "no negative entries" if every x in l {
    x > 0
}

```

Get ready today with

`import rego.v1`

https://www.openpolicyagent.org/docs/latest/opa-1/

OPA 1.0 - Get Your Rego Ready

Step 1 `opa check`

- This will:
  - Find parse errors
  - Raise compilation errors

Checkout this OPA errors index for help if you get stuck!

OPA 1.0 - Get Your Rego Ready

Step 2 `opa check --strict`

- This will help you find:
  - Duplicate imports & Unused imports
  - Unused local assignments
  - Use of deprecated functions
  - Overriding reserved keywords input and data
- `check --strict` will exit early, run this until there are no more errors.
- Once you’ve done this you’re nearly there!

OPA 1.0 - Get Your Rego Ready

Step 3 `opa fmt --write --rego-v1`

- This will re-write your Rego to use the v1 syntax

```
package app.authz

allow = count(deny) == 0

deny[message] {
    input.role != "admin"
    message := "You are not an admin"
}
```

(You running `opa fmt`!)

```
package app.authz

import rego.v1

allow if count(deny) == 0

deny contains message if {
    input.role != "admin"
    message := "You are not an admin"
}
```

OPA 1.0 - Get Your Rego Ready

Step 4 `regal lint`

- This will help you:
  - Identify bugs and mistakes
  - Find performance improvements
  - Write more consistent and idiomatic Rego

(You running `regal lint`!)

Regal docs CLI and other editors

https://marketplace.visualstudio.com/items?itemName=tsandall.opa

Regal comes with the OPA VS Code Extension!

Need Help?

- The **#help** channel in the OPA slack is the best starting point for OPA help.
  - Use **#opa-gatekeeper** for GK queries
- You can also open discussions at github.com/orgs/open-policy-agent/discussions
- If you find a bug or have a feature request, open an issue! github.com/open-policy-agent/opa/issues/new

Read the “Renovating Rego” Blog

OPA Slack can be joined instantly at:
slack.openpolicyagent.org

OPA Use-Case: Kubernetes Admission Webhook

https:gatekeeper.sh

What's new in v3.17?

- K8sNativeValidation (CEL) engine is graduated to beta and enabled by default.
  - Use enable-k8s-native-validation feature flag to disable.
- Kubernetes Validating Admission Policy (VAP) generation is alpha!
  - Use `default-create-vap-for-templates` and `default-create-vap-binding-for-constraints` flags to enable generation.
- Turn on and off specific action for Gatekeeper: webhook, audit, gator, or VAP in the same constraint
  - scopedEnforcementActions
- Validation of CONNECT operations.

What's new in v3.17?

VAP included as `enforcementPoint`

```yaml
apiVersion:
constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
Spec:
  enforcementAction: deny
```

```yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
Spec:
  enforcementAction: scoped
  scopedEnforcementActions:
    - action: deny
      enforcementPoints:
      - name: vap.k8s.io
      - name: gator.gatekeeper.sh
      - name: audit.gatekeeper.sh
```

What's new in v3.17?

Generating VAP has transitioned from an annotation to a field under `K8sNativeValidation` engine.

```
targets:
    - target: admission.k8s.gatekeeper.sh
      code:
        - engine: K8sNativeValidation
          source:
            generateVAP: true
            validations:
            - expression: '[object, oldObject].exists(obj, obj != null &&
has(obj.metadata) && variables.params.labels.all(entry, has(obj.metadata.labels) &&
entry.key in obj.metadata.labels))'
              messageExpression: '"missing required label, requires all of: " +
variables.params.labels.map(entry, entry.key).join(", ")'
```

Demo

Checkout demo with VAP enforcement point

https://qr-codes.io/Qu5Gap

What's coming in v3.18?

- K8sNativeValidation (CEL) engine is GA.
  - Use enable-k8s-native-validation feature flag to disable.
- Added wait for VAPB generation
- gator sync test
- gator verify now supports ExpansionTemplates
- operation=generate is added to isolate and run generation processes for CRDs and VAPs as a singleton pod
  - Set --operation=generate on a singleton pod, e.g. as part of the audit pod.
- Gatekeeper Config resource reports error through status

What's coming in v3.18?

Support for `ExpansionTemplate` in gator verify

```yaml
apiVersion: test.gatekeeper.sh/v1alpha1
kind: Suite
tests:
- name: foo-is-bar-expansion
  template: template.yaml
  constraint: constraint.yaml
  expansion: expansion.yaml
  ...
```

```yaml
apiVersion: expansion.gatekeeper.sh/v1alpha1
kind: ExpansionTemplate
metadata:
  name: expand-foo
spec:
  applyTo:
  - groups: [ "apps" ]
    kinds: [ "Deployment" ]
    versions: [ "v1" ]
  templateSource: "spec.template"
  generatedGVK:
    kind: "Pod"
    group: ""
    version: "v1"
```

CEL Library

Check out gatekeeper policy library with CEL rules

https://qr-codes.io/8KGclv

Questions & Feedback

Please ask questions!

You can also come to the OPA Kiosk **(11A)** in the **Project Pavillion**

(tomorrow until 14:00)

View these slides

Please leave feedback about this session so we can improve for the next OPA talk at **KubeCon London**!