
KubeCon
CloudNativeCon
Europe 2025
LONDON BUS

KubeCon | CloudNativeCon
Europe 2025

Open Policy Agent (OPA) Introduction and Deep-Dive

Charlie Egan
Styra

Anders Eknert
Styra

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

OPA is about Policy as Code

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

```
package deployments.production

default allow := false

allow if {
    is_weekend
    "support" in input.user.roles
    input.user.id in data.oncall_users
}

# ...
```

Title

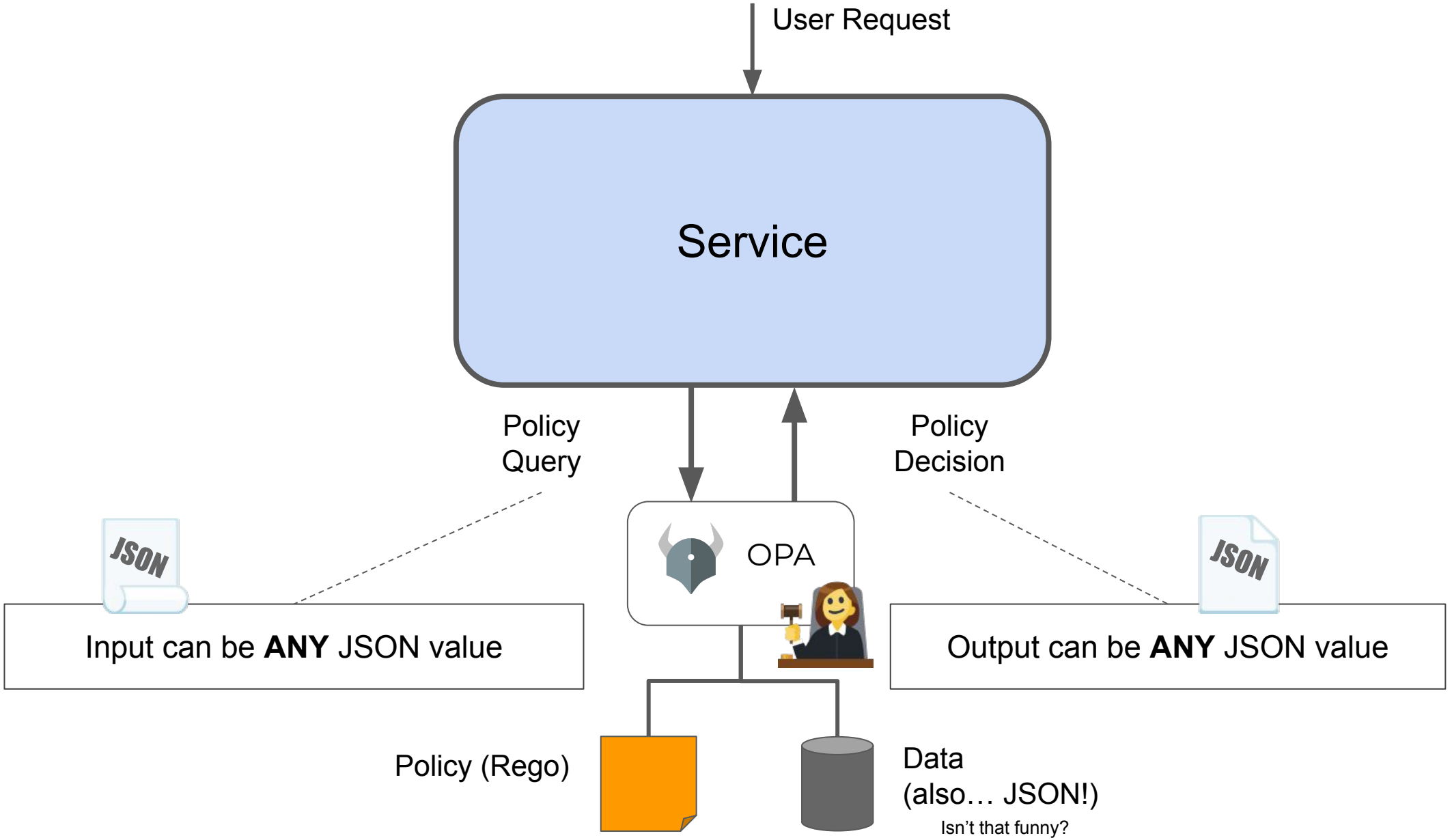
Input → **Policy** → **Decision**

```
{
    "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]
}
```

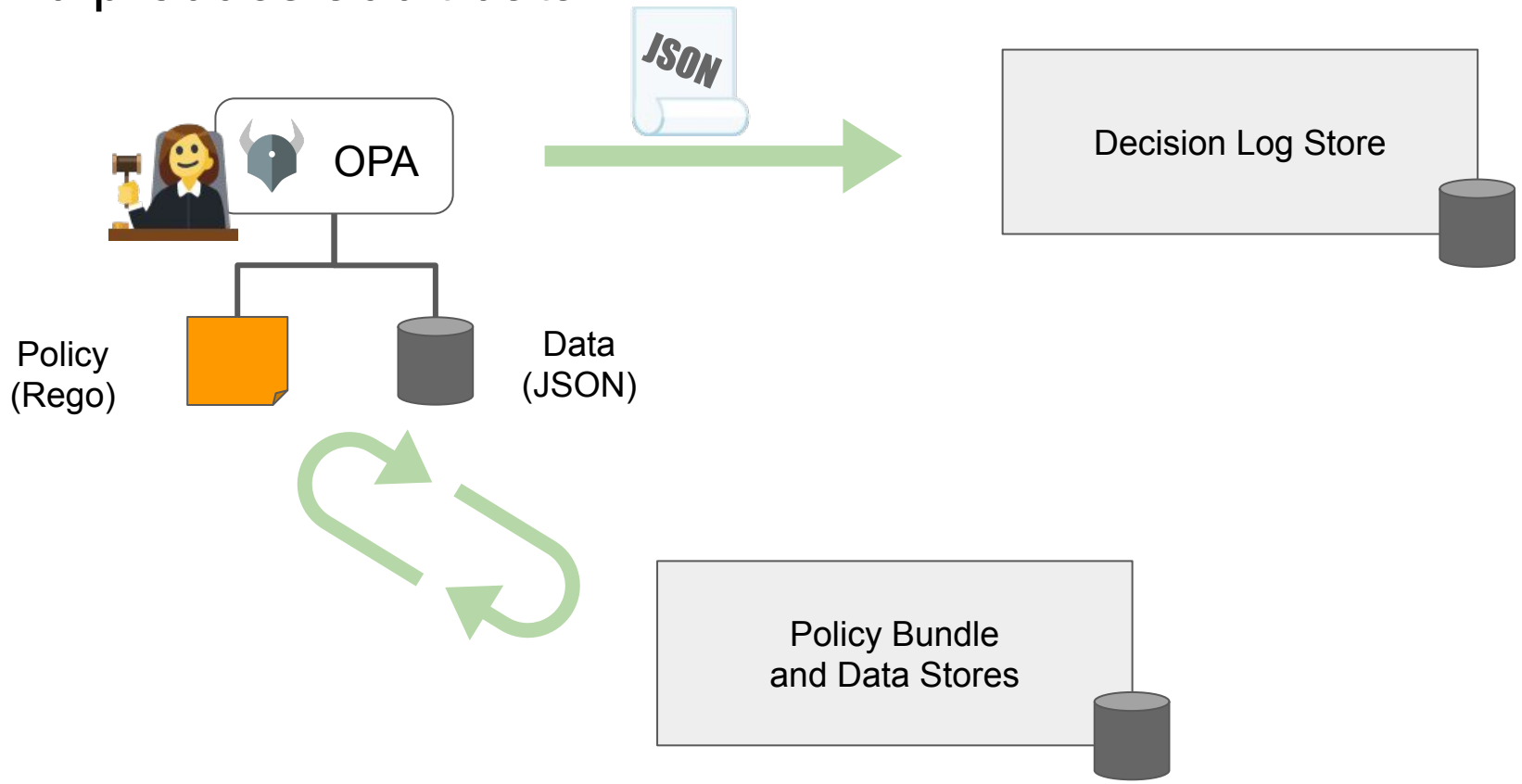
```
{
    "allow": false
}
```

OPA


User Request
Service
Policy Query
Policy Decision
JSON
OPA
JSON
Input can be **ANY** JSON value
Output can be **ANY** JSON value
Policy (Rego)
Data (also… JSON!)
Isn't that funny?

OPA's Responsibilities

OPA works continuously to maintain
up-to-date policies and produce audit data.



Why does Policy as Code work?

Developers can focus on delivering business value

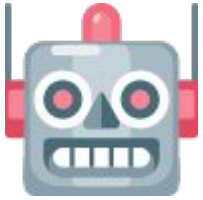
- Decouple policy decision logic from applications

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

Why does Policy as Code work?

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

Yao Weng, Bloomberg

(KubeCon Istio Day, 2024)

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

Why does Policy as Code work?

KubeCon CloudNativeCon Europe 2025

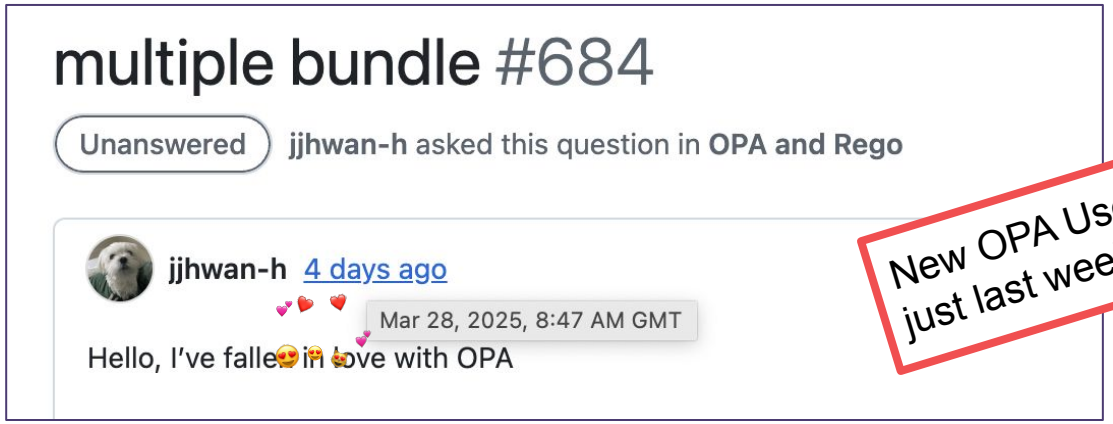
“OPA decouples access control logic from the rest of the service”

Sung Yun & Aki Sukegawa, Bloomberg

(yesterday!)

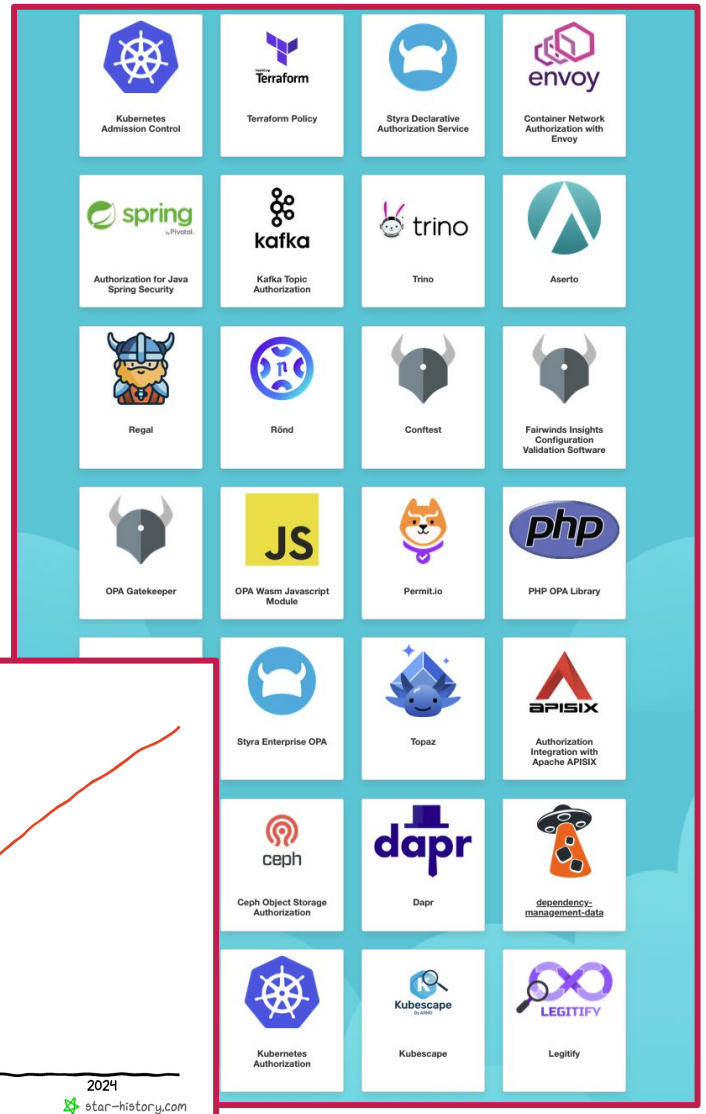
https://kccnceu2025.sched.com/event/1txF1/trino-and-data-governance-on-kubernetes-sung-yun-aki-sukegawa-bloomberg

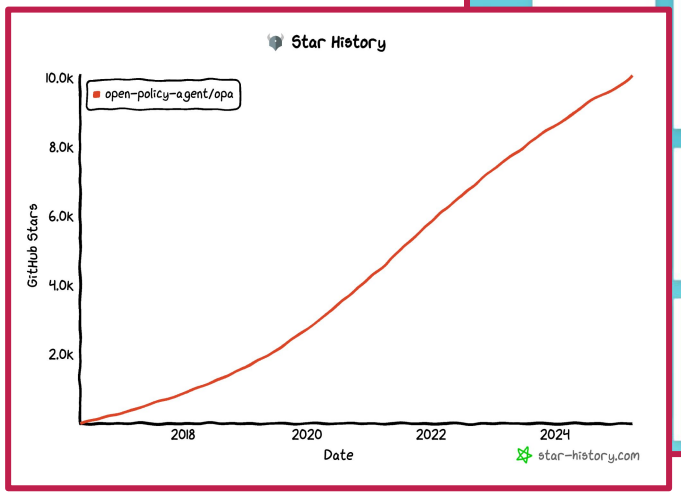
Community Highlights



New OPA User, just last week!

- **Vibrant community** of users and contributors
- **100+** Integrations
- **10K+** GitHub Stars
- **9K+** Slack Users
- **Hundreds of millions** of downloads
- **OPA Gatekeeper, Conftest**
- **Editor integrations**

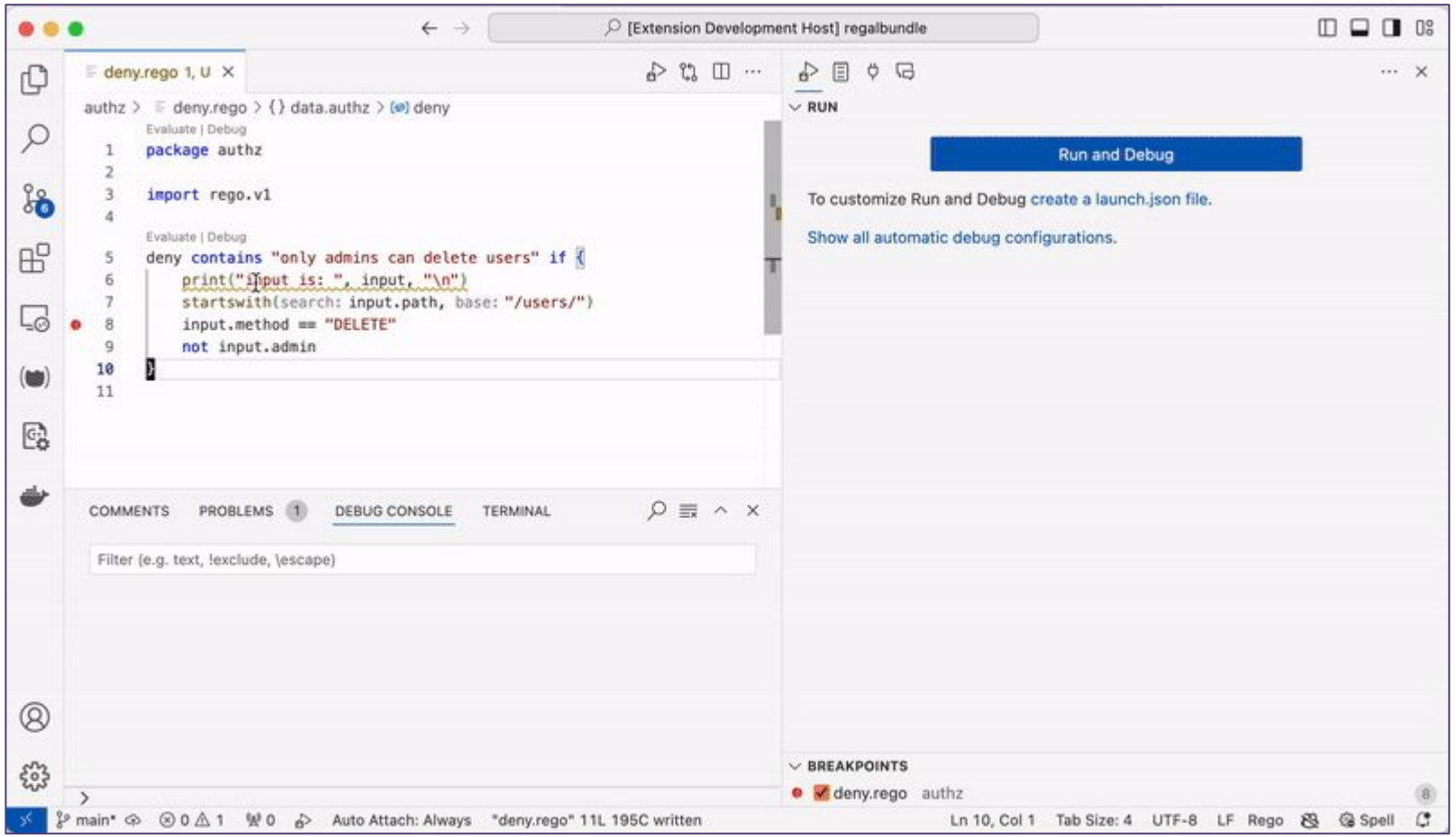




Community Highlights

KubeCon CloudNativeCon Europe 2025

Regal Linter Debug Adapter Protocol Support



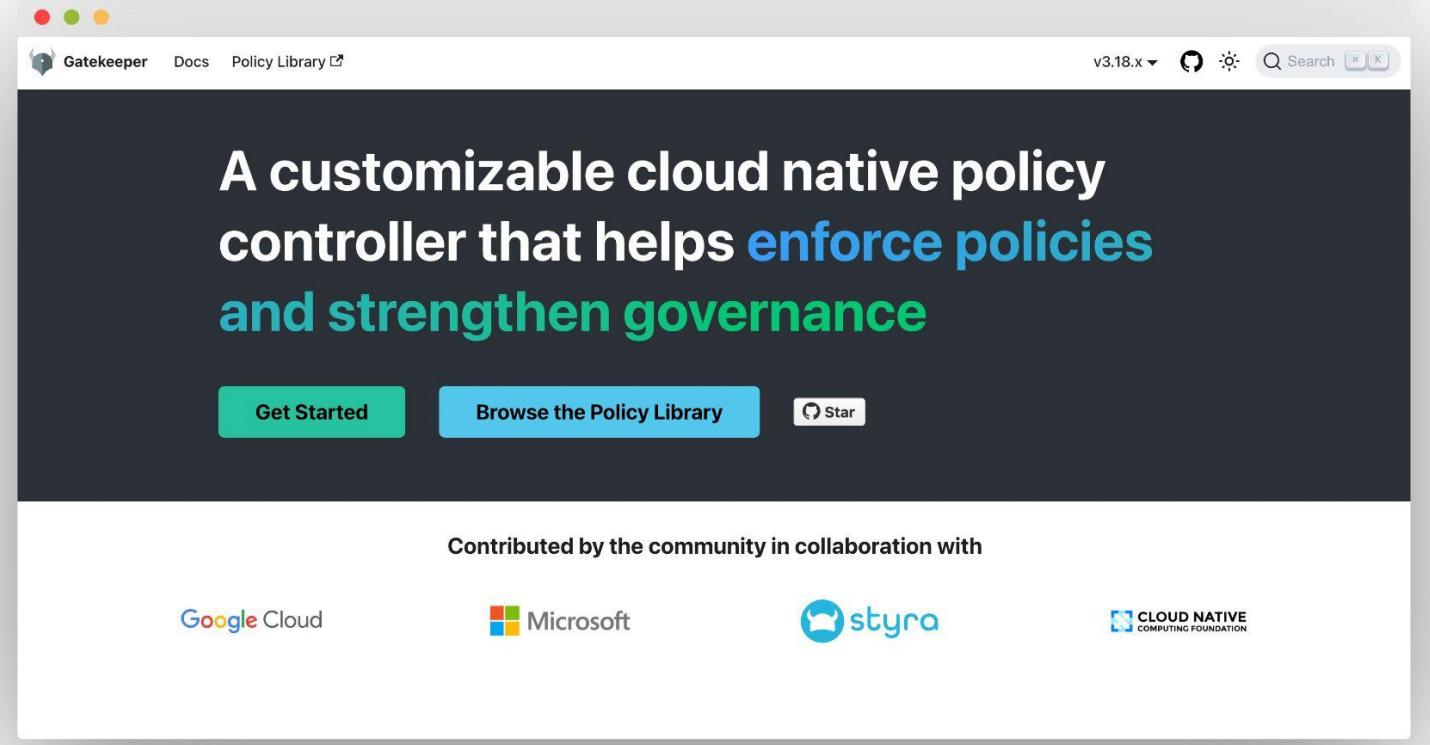
github.com/open-policy-agent/vscode-opa

neovim

github.com/rinx/nvim-dap-rego

OPA Gatekeeper

KubeCon | CloudNativeCon
Europe 2025



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

What's new since v3.17 (v3.18, v3.19)

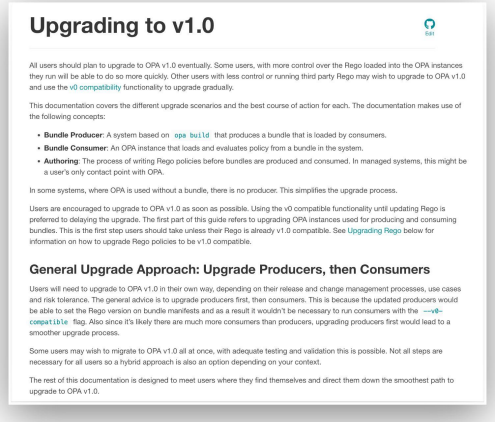
- OPA Rego v1 syntax is available in **`ConstraintTemplates`**.
  - Use source.version: v1 to enable v1 support for **`rego`** engine.
- Pub/Sub interface is replaced with generic export interface.
  - Integration with different backends to export violations is available.
- **`K8sNativeValidation`** (CEL) engine is graduated to GA.
  - Use enable-k8s-native-validation feature flag to disable.
- Gator CLI updates
  - **`ExpansionTemplate`** support in gator verify
  - gator sync test to verify **`ConstraintTemplates`** that required data replication and syncing resources.
- Operation generate to guard CRD and VAP/VAPB generation.
- Only output deny messages in gator test with --deny-only flag.

What’s upcoming?

- A driver to export all violations to disk.
- Graduating VAP integration to beta.
- Performance improvement for exporting violations.

OPA 1.0

We're on v1.3.0 now! Missed v1.0? review the releases or the blog



Upgrading to v1.0

All users should plan to upgrade to OPA v1.0 eventually. Some users, with more control over the Rego loaded into the OPA instances they run will be able to do so more quickly. Other users with less control or running third party Rego may wish to upgrade to OPA v1.0 and use the v0 compatibility functionality to upgrade gradually.

This documentation covers the different upgrade scenarios and the best course of action for each. The documentation makes use of the following concepts:

- **Bundle Producer**: A system based on `opa build` that produces a bundle that is loaded by consumers.
- **Bundle Consumer**: An OPA instance that loads and evaluates policy from a bundle in the system.
- **Authoring**: The process of writing Rego policies before bundles are produced and consumed. In managed systems, this might be a user's only contact point with OPA.

In some systems, where OPA is used without a bundle, there is no producer. This simplifies the upgrade process.

Users are encouraged to upgrade to OPA v1.0 as soon as possible. Using the v0 compatible functionality until updating Rego is preferred to delaying the upgrade. The first part of this guide refers to upgrading OPA instances used for producing and consuming bundles. This is the first step users should take unless their Rego is already v1.0 compatible. See Upgrading Rego below for information on how to upgrade Rego policies to be v1.0 compatible.

General Upgrade Approach: Upgrade Producers, then Consumers

Users will need to upgrade to OPA v1.0 in their own way, depending on their release and change management processes, use cases and risk tolerance. The general advice is to upgrade producers first, then consumers. This is because the updated producers would be able to set the Rego version on bundle manifests and as a result it wouldn't be necessary to run consumers with the `--v0-compatible` flag. Also since it's likely there are much more consumers than producers, upgrading producers first would lead to a smoother upgrade process.

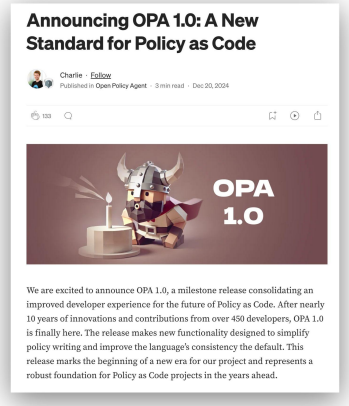
Some users may wish to migrate to OPA v1.0 all at once, with adequate testing and validation this is possible. Not all steps are necessary for all users so a hybrid approach is also an option depending on your context.

The rest of this documentation is designed to meet users where they find themselves and direct them down the smoothest path to upgrade to OPA v1.0.

Upgrade Documentation

www.openpolicyagent.org/docs/latest/v0-upgrade

If you have already updated, have some free love and stay tuned!



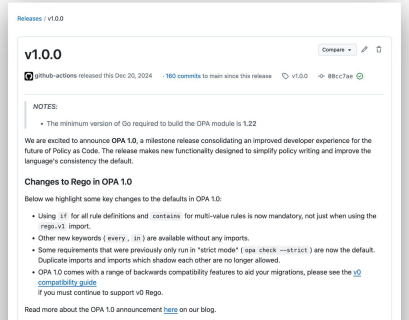
Announcing OPA 1.0: A New Standard for Policy as Code

Charlie · Follow
Published in Open Policy Agent · 3 min read · Dec 20, 2024

133

We are excited to announce OPA 1.0, a milestone release consolidating an improved developer experience for the future of Policy as Code. After nearly 10 years of innovations and contributions from over 450 developers, OPA 1.0 is finally here. The release makes new functionality designed to simplify policy writing and improve the language's consistency the default. This release marks the beginning of a new era for our project and represents a robust foundation for Policy as Code projects in the years ahead.

Blog.openpolicyagent.org/announcing-opa-1-0-a-new-standard-for-policy-as-code-a6d8427ee828



Releases / v1.0.0

v1.0.0

Compare

github-actions released this Dec 20, 2024 · 160 commits to main since this release · v1.0.0 · 80cc7ae

NOTES:

- The minimum version of Go required to build the OPA module is 1.22

We are excited to announce **OPA 1.0**, a milestone release consolidating an improved developer experience for the future of Policy as Code. The release makes new functionality designed to simplify policy writing and improve the language's consistency the default.

Changes to Rego in OPA 1.0

Below we highlight some key changes to the defaults in OPA 1.0:

- Using `if` for all rule definitions and `contains` for multi-value rules is now mandatory, not just when using the `rego.v1` import.
- Other new keywords (`every`, `in`) are available without any imports.
- Some requirements that were previously only run in "strict mode" (`opa check --strict`) are now the default. Duplicate imports and imports which shadow each other are no longer allowed.
- OPA 1.0 comes with a range of backwards compatibility features to aid your migrations, please see the v0 compatibility guide
  If you must continue to support v0 Rego.

Read more about the OPA 1.0 announcement here on our blog.

OPA GitHub Releases

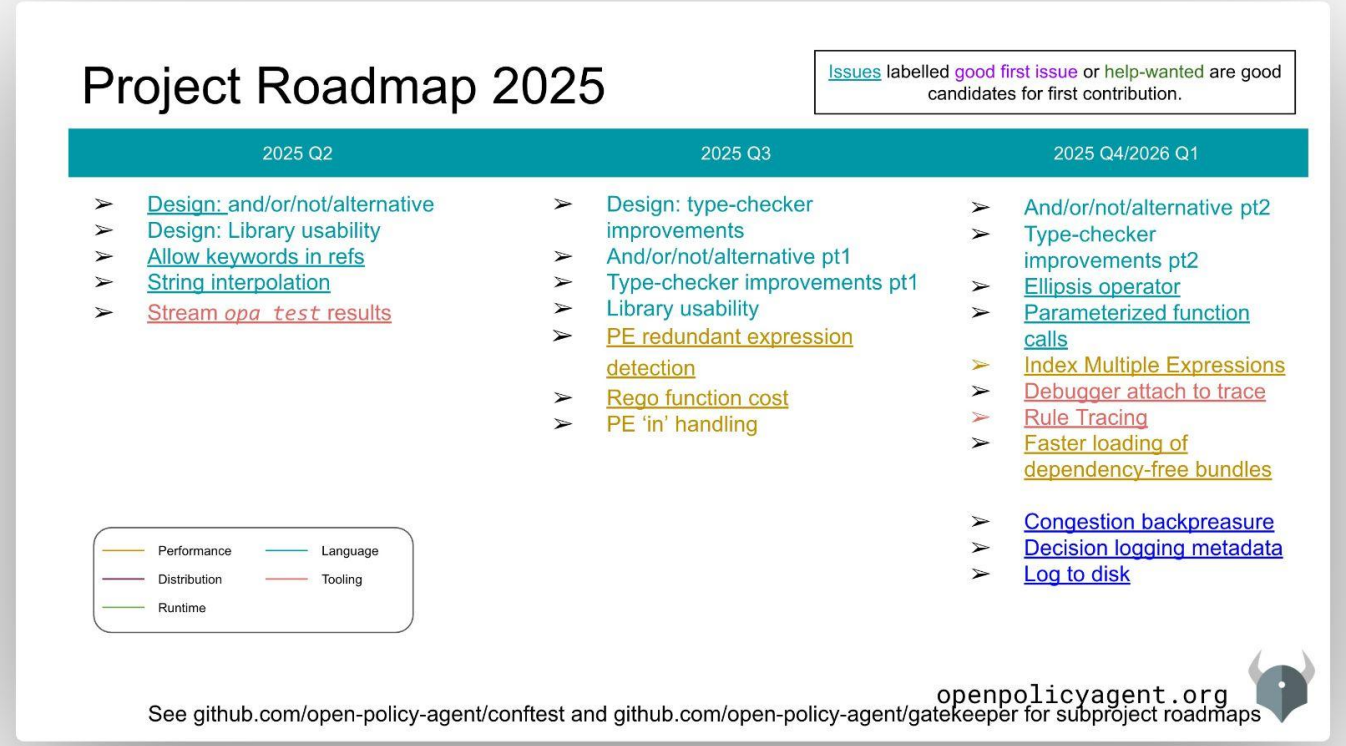
github.com/open-policy-agent/opa/releases

OPA Roadmap: Highlights

KubeCon CloudNativeCon Europe 2025

We are looking for feedback on the OPA Roadmap

Chat on issues and in Slack (slack.openpolicyagent.org)

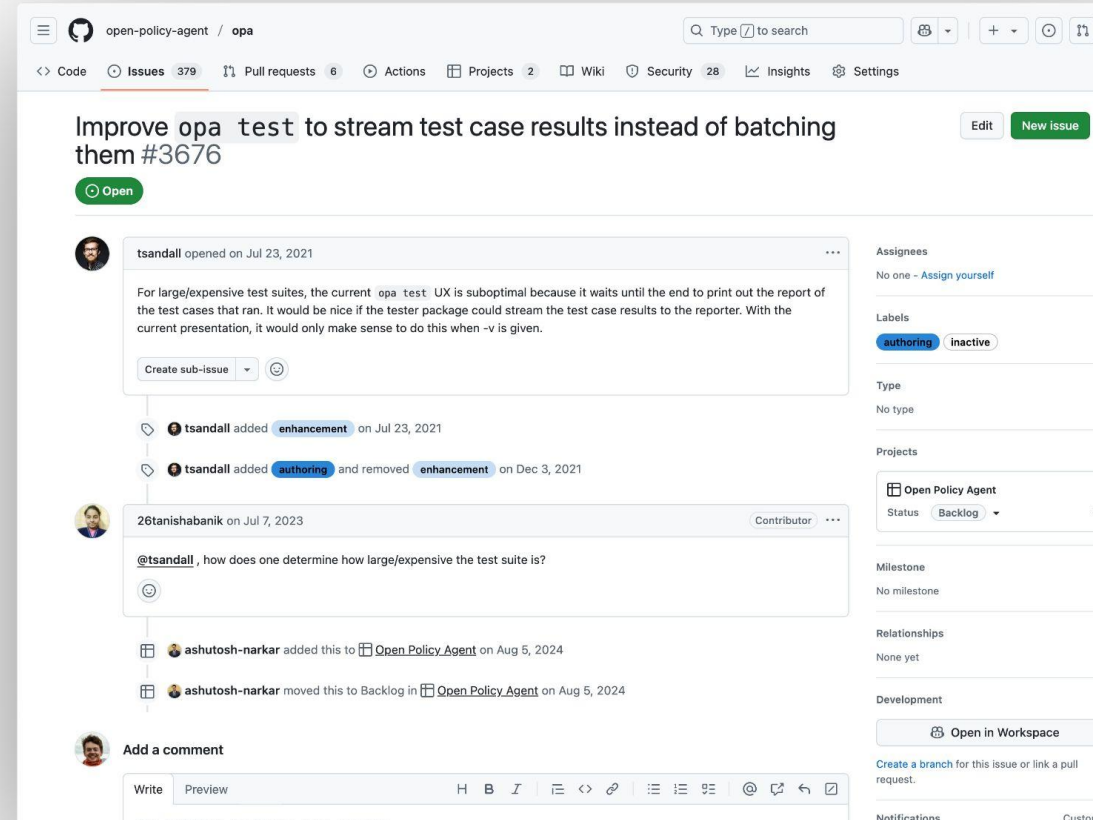


https://docs.google.com/presentation/d/16QV6gvLDOV3I0_guPC3_19g6jHkEg3X9xqMYgtoCKrs/edit

OPA Roadmap: Streaming Tests

- OPA waits before printing tests
- Also helpful for editors and IDE integrations too

Issue #3676



OPA Roadmap: Streaming Tests

```
../regal main ➜
```

```
../regal main ➜
```

OPA Roadmap: Logic Operators (Design)

KubeCon CloudNativeCon Europe 2025

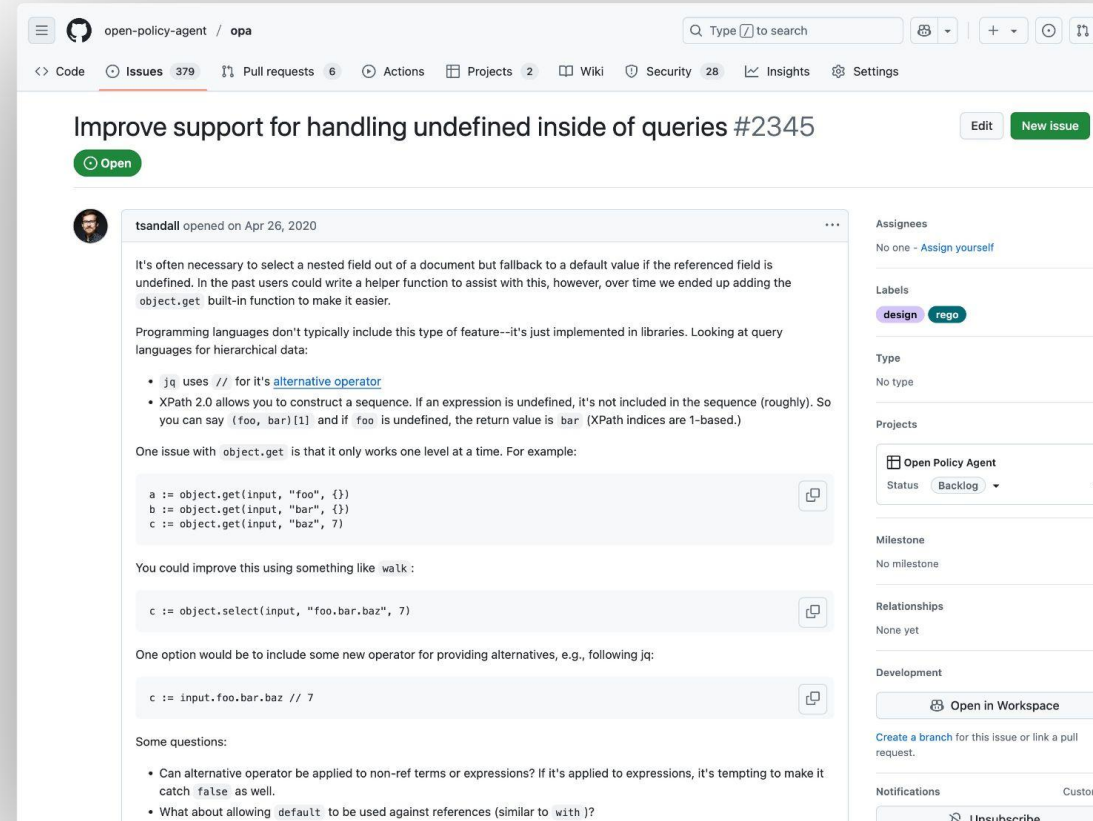
```rego
package example

# OR as of v1.3.0
allow if role == "admin"
allow if isParent(user, child)

# 'Alternative operator' as of v1.3.0
default user_role(_) := "guest"
user_role(user) := user.role
role := user_role(input.user)
```

the present

```rego
package example

# Or operator
allow if {
  ...
  role == "admin" || isParent(user, child)
}

# Alternative operator
role := input.user.role :| "guest"
```

the future?

Issue #2345

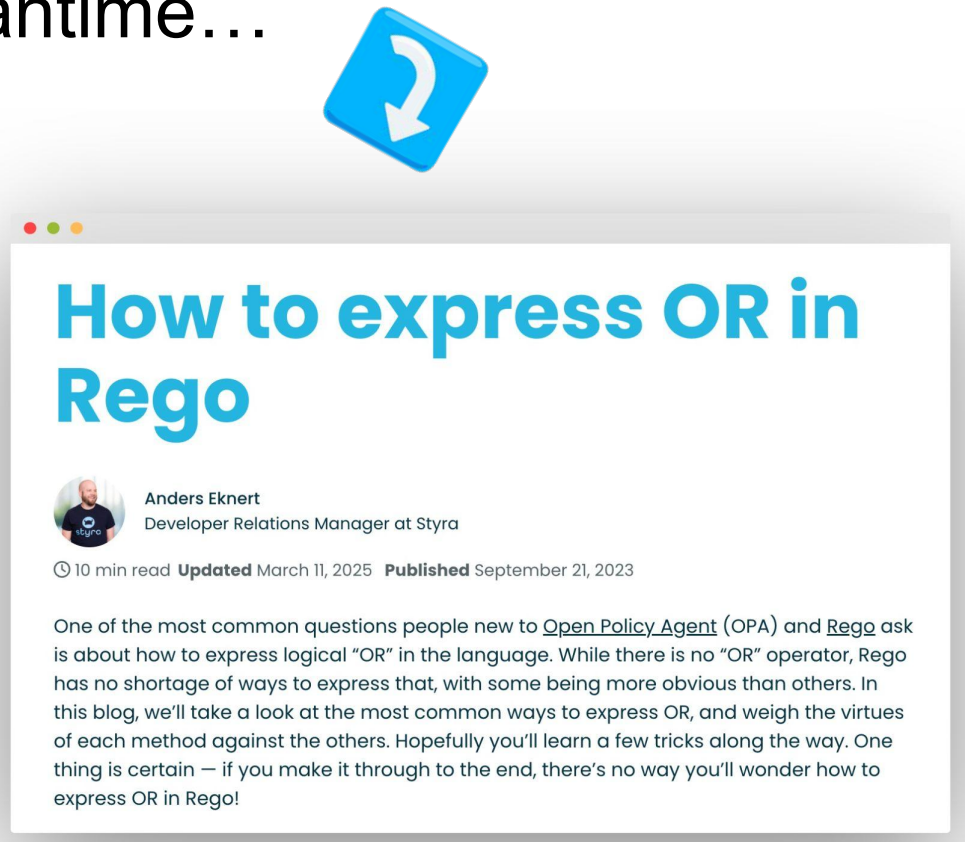


OPA Roadmap: Logic Operators (Design)


```
package example

# OR as of v1.3.0
allow if role == "admin"
allow if isParent(user, child)

# 'Alternative operator' as of v1.3.0
default user_role(_) := "guest"
user_role(user) := user.role
role := user_role(input.user)
```

the present

Read this in the meantime…



How to express OR in Rego

Anders Eknert
Developer Relations Manager at Styra

10 min read **Updated** March 11, 2025 **Published** September 21, 2023

One of the most common questions people new to Open Policy Agent (OPA) and Rego ask is about how to express logical "OR" in the language. While there is no "OR" operator, Rego has no shortage of ways to express that, with some being more obvious than others. In this blog, we'll take a look at the most common ways to express OR, and weigh the virtues of each method against the others. Hopefully you'll learn a few tricks along the way. One thing is certain — if you make it through to the end, there's no way you'll wonder how to express OR in Rego!

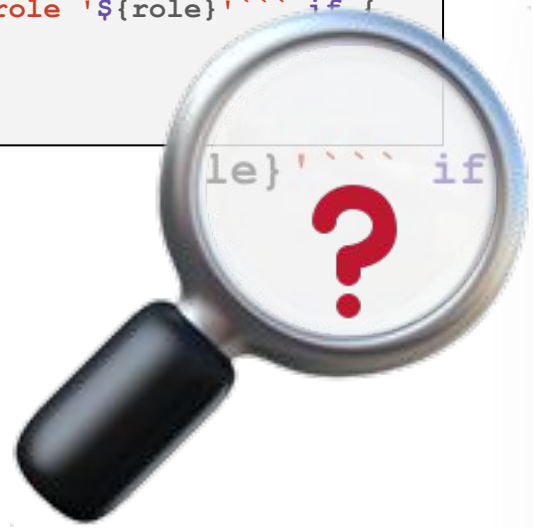
https://www.styra.com/blog/how-to-express-or-in-rego/

OPA Roadmap: String Interpolation

KubeCon CloudNativeCon Europe 2025

```rego
package example

deny contains sprintf("%s must have role '%s'", [input.name, role]) if {
  role := "developer"
  not role in input.groups
}
```

```rego
package example

deny contains ```${input.name} must have role '${role}'``` if {
  role := "developer"
  not role in input.groups
}
```



Issue #4733

open-policy-agent / opa

Code | Issues 379 | Pull requests 6 | Actions | Projects 2 | Wiki | Security 28 | Insights | Settings

String interpolation #4733

Open

Edit | New issue

anderseknert opened on Jun 3, 2022 · edited by anderseknert

Having string interpolation similar to Python f-strings in Rego would allow for more succint policy, while arguably improving readability by removing the ceremony associated with `sprintf`.

Another issue potentially addressed by this would be the common mistake of not handling undefined when referencing input/data in `sprintf` arguments. These will halt evaluation as any other undefined reference, but in the likely most common use case for `sprintf` — building a "message" or "reason" string to populate a partial rule set, whether a value is undefined or not is likely not interesting enough that you'd want it to affect the actual *outcome* of the evalutation.

```rego
deny[reason] {
    not "developer" in input.user.groups

    # fails if input.user.name is undefined, and the anonymous user will not be denied
    # this is likely not what the policy author intended
    reason := sprintf("%v must have role 'developer'", [input.user.name])
}
```

```rego
# less ceremony, and always evaluates if conditions in the body holds
# interpolation step could handle undefined (similar to `print`)
deny["{input.user.name} must have role 'developer'"] {
    not "developer" in input.user.groups
}
```

This would arguably improve string handling not just with regards to `sprintf`, but improve things like simple concatenation, which is currently somewhat verbose:

```rego
name := concat(", ", [input.first_name, input.last_name])

# vs.

name := "{input.first_name}, {input.last_name}"
```

Some design decisions would obviously need to be made here, like how to differentiate an "interpolation string" from a regular

Assignees: No one - Assign yourself

Labels: feature-request, inactive, rego

Type: No type

Projects: No projects

Milestone: No milestone

Relationships: None yet

Development: Open in Workspace

Create a branch for this issue or link a pull request.

Notifications: Custo

Unsubscribe

You're receiving notifications because you're subscribed to this thread.

OPA Roadmap: String Interpolation

KubeCon CloudNativeCon Europe 2025

```
package example

deny contains sprintf("%s must have role '%s'", [input.name, role]) if {
  role := "developer"
  not role in input.groups
}
```

1. `Input: {"name": "Alice", "groups": ["tester"]}:`

   ```
   Output: {
     "deny": ["Alice must have role 'developer'"]
   }
   ```

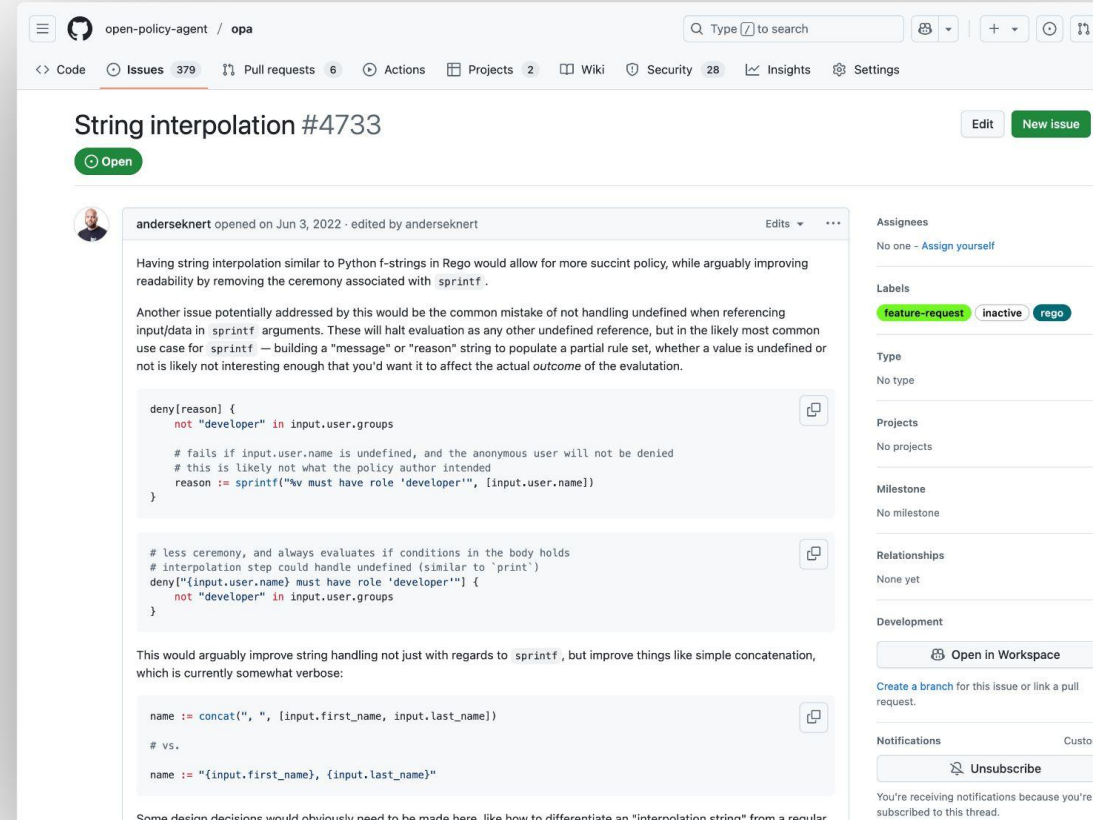
2. `Input: {"groups": ["tester"]}:`

   ```
   Output: {}
   ```

⚠ Name is missing

⚠ Output is undefined

Issue #4733



OPA Roadmap: String Interpolation

- Undefined values needed for string interpolation can be handled differently.
- Non breaking for existing rules.

Issue #4733

```
package example

deny contains ```${input.name} must have role '${role}'``` if {
  role := "developer"
  not role in input.groups
}
```

1. Input: {"name": "Alice", "groups": ["tester"]}:

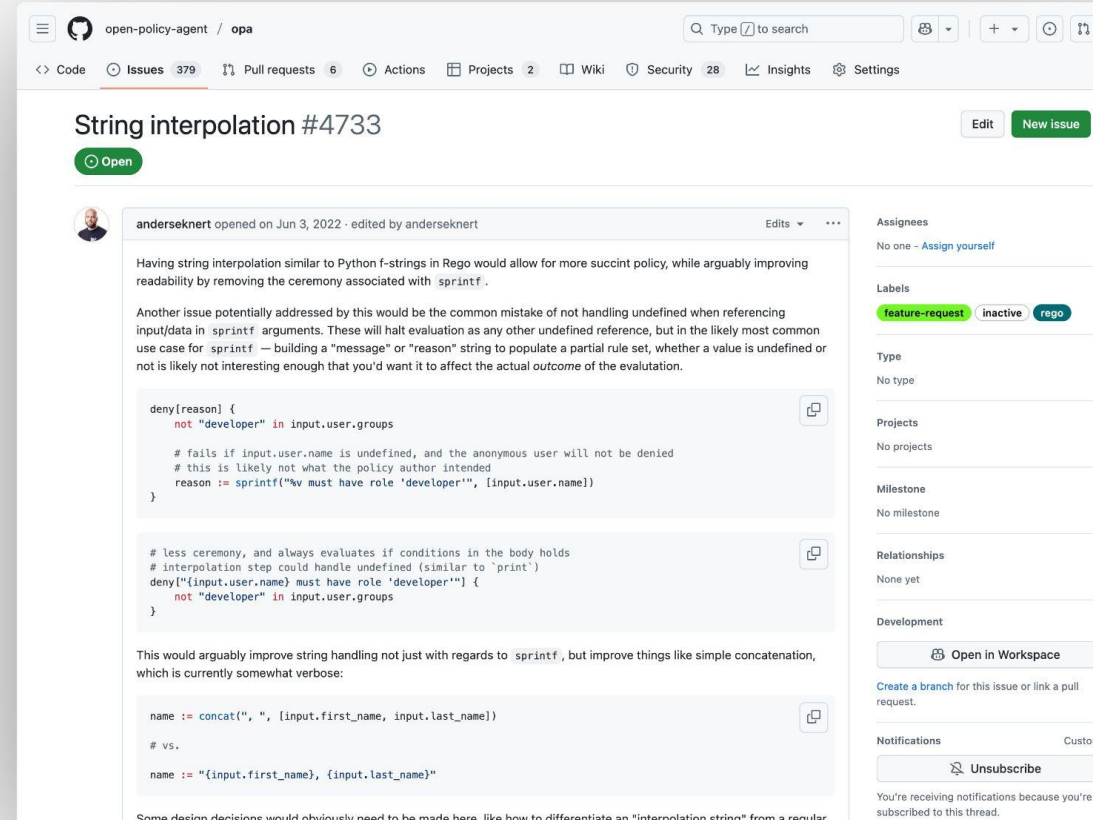
   Output: {
     "deny": ["Alice must have role 'developer'"]
   }

2. Input: {"groups": ["tester"]}:

   Output: {
     "deny": ["must have role 'developer'"]
   }

Name is missing

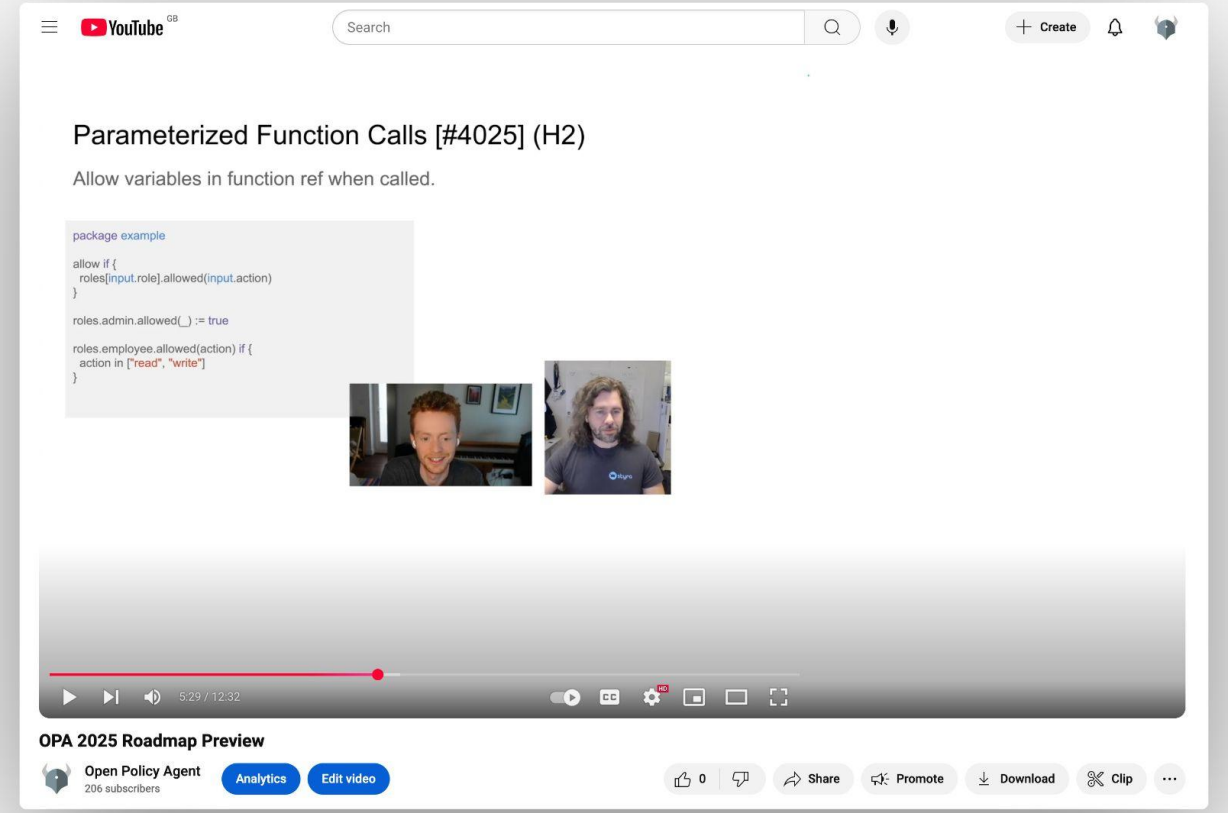
Output Message Still Set



OPA Roadmap: Highlights

KubeCon CloudNativeCon Europe 2025

Checkout this video to see the roadmap in more detail

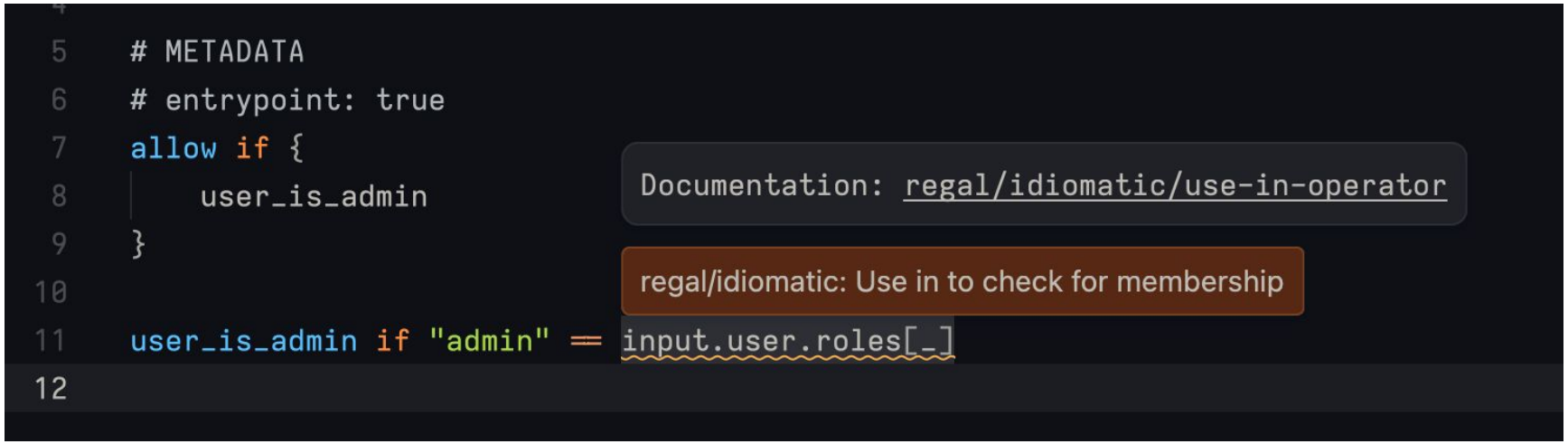


https://www.youtube.com/watch?v=3KYUkV4eg7s

KubeCon | CloudNativeCon
Europe 2025

OPA Performance

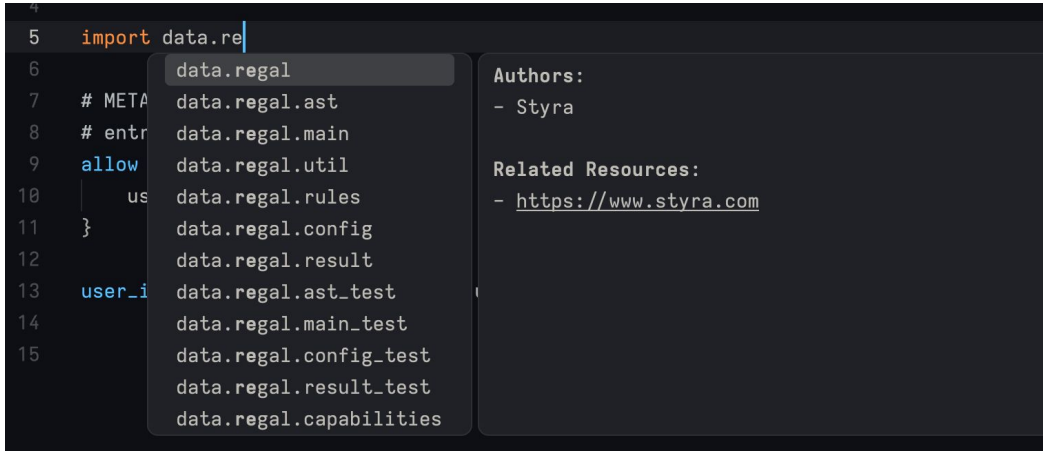
Why?



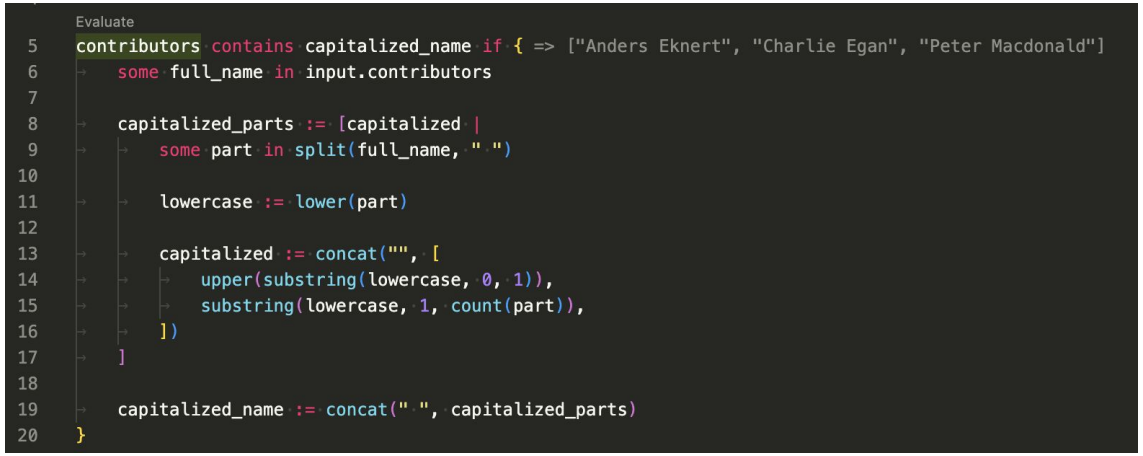
We built a linter for Rego

in Rego

Why?



Auto-completion



Click-to-evaluate

And later, a language server

in Rego

What?

- **Regal** — a linter for Rego written in Rego
- ~100 linters rules, and a language server == ~15K lines of Rego
- Evaluated for each input file. 200 files == ~3 million lines of Rego
- Linting some of the largest repos (10K+ lines of Rego) in seconds
- Highly parallel & CPU bound == much slower in CI
- Language server lints Rego on every key press
- If we want a faster Regal, **we need a faster OPA**
- So that's what we did

How

- Identify hot paths in policy evaluation
- Go application: reducing memory allocations == faster evaluation
- Optimize OPA's built-in functions, particularly common ones
- Optimize existing data structures to avoid allocating more than required
- A few optimizations particularly beneficial for Regal
  - But most of them help speed up **all** policy evaluation!
- Performance improvements from OPA v0.70.0 to OPA v1.2.0

How

We can’t go into them all!

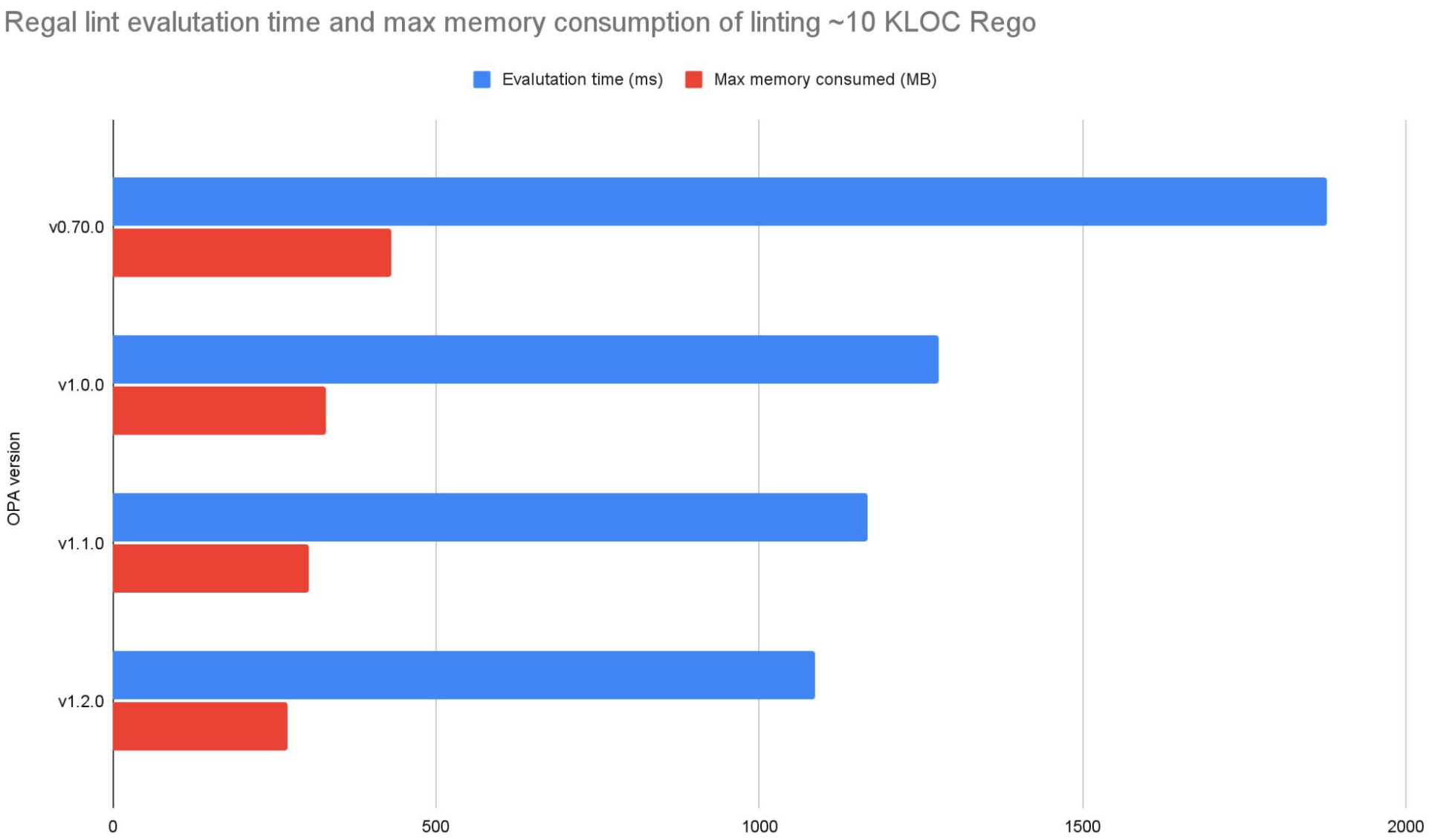
#7168 Optimize biunify for term slices
#7172 Tweaks to reduce number of allocations in regal lint hot path
#7190 Even less allocs
#7193 More reduced allocations
#7222 Reduce allocations, chapter III 
#7281 Remove jsonOptions from AST nodes and terms
#7284 Perf: improvements to terms and built-in functions
#7288 eval: reduce allocations in hot path
#7319 Eliminate allocation in Value.Find
#7327 eval: optimize iteration
#7350 perf: Add ref.CopyNonGround
#7365 perf: intern annotation terms
#7367 perf: eval optimizations
#7368 perf: slightly more efficient policy scanning
#7370 perf: cost of indexing greatly reduced
#7372 perf: use GetByValue to avoid boxing to interface{}

Title

#7172 Tweaks to reduce number of allocations in regal lint hot path

```
14  14    func builtinCount(_ BuiltinContext, operands []*ast.Term, iter func(*ast.Term) error) error {
15  15        switch a := operands[0].Value.(type) {
16  16        case *ast.Array:
17      -         return iter(ast.IntNumberTerm(a.Len()))
        17    +         return iter(ast.InternedIntNumberTerm(a.Len()))
18  18        case ast.Object:
19      -         return iter(ast.IntNumberTerm(a.Len()))
        19    +         return iter(ast.InternedIntNumberTerm(a.Len()))
20  20        case ast.Set:
21      -         return iter(ast.IntNumberTerm(a.Len()))
        21    +         return iter(ast.InternedIntNumberTerm(a.Len()))
22  22        case ast.String:
23      -         return iter(ast.IntNumberTerm(len([]rune(a))))
        23    +         return iter(ast.InternedIntNumberTerm(len([]rune(a))))
24  24        }
25  25        return builtins.NewOperandTypeErr(1, operands[0].Value, "array", "object", "set", "string")
26  26    }
          @@ -178,26 +178,26 @@ func builtinAll(_ BuiltinContext, operands []*ast.Term, iter func(*ast.Term) err
178 178        switch val := operands[0].Value.(type) {
179 179        case ast.Set:
180 180            res := true
181     -          match := ast.BooleanTerm(true)
        181   +          match := ast.InternedBooleanTerm(true)
182 182            val.Until(func(term *ast.Term) bool {
183 183                if !match.Equal(term) {
```

Result — 70% faster evaluation



KubeCon isn't over yet!

KubeCon CloudNativeCon Europe 2025

Please ask questions!

You can also come to the OPA Kiosk **(14A)** 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 Atlanta**!

Anders Eknert
anders@styra.com
@anderseknert@swecyb.com

Charlie Egan
charlie@styra.com
@charlieegan3@hachyderm.io