Policy as Code 101

Charles Daniels
August 22, 2024

About Me – Charles Daniels

- Computer Engineering B.S.E. – UofSC 2019
- Computer Science M.S. - UofSC 2021
- Backend Software Engineer – Styra Inc.

All views and opinions expressed are my own, and do not represent those of Styra.

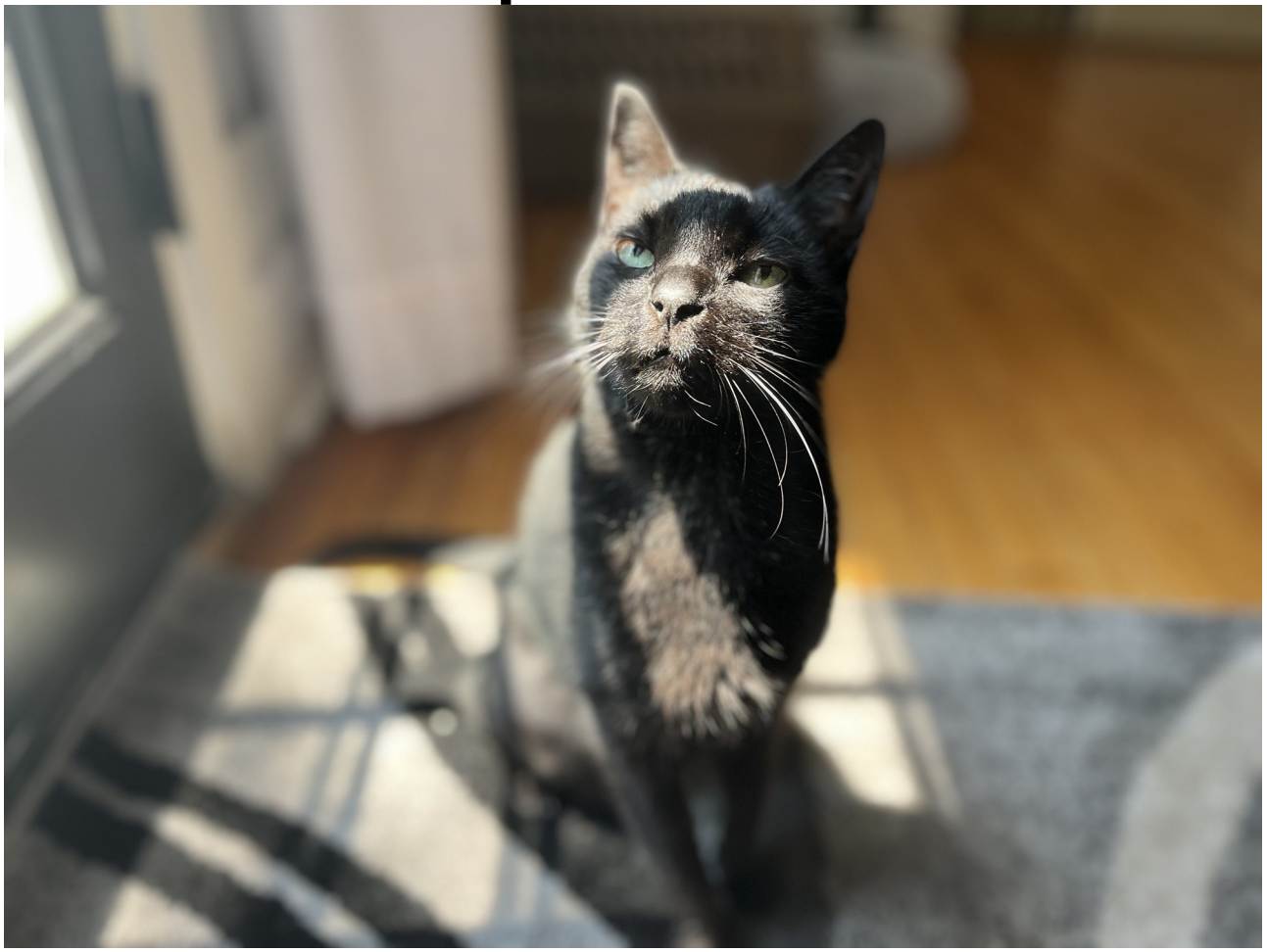
Want the Slides?

- https://cdaniels.net/talks

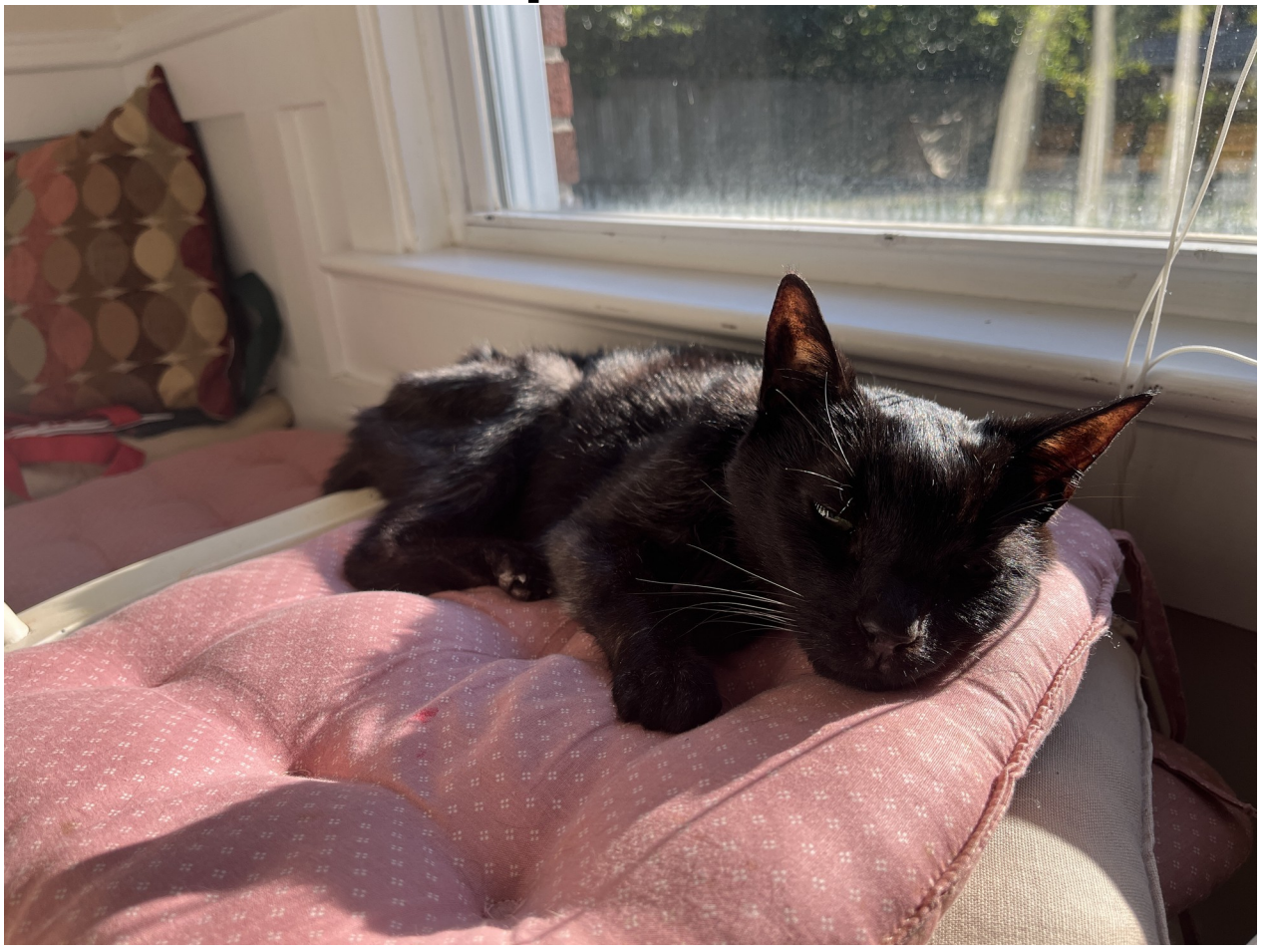
Swamp Creature

- Photographs of a mysterious local swamp creature, as is the tradition of Silicon Swamp...

Swamp Creature



Swamp Creature



Swamp Creature

Swamp Creature

Agenda

- What is policy as code?
- Introduction to Open Policy Agent
- Authoring policies in Rego
- Understanding the Authorization Maturity Model

What is a policy?

- These are *authorization questions*:
  - Is Alice allowed in the VIP lounge?
  - Can John withdraw $200 from account# 1234?
  - What actions can Tim do on an escalated support ticket?
- A *policy* is a process which answers authorization questions

What is Policy as Code?

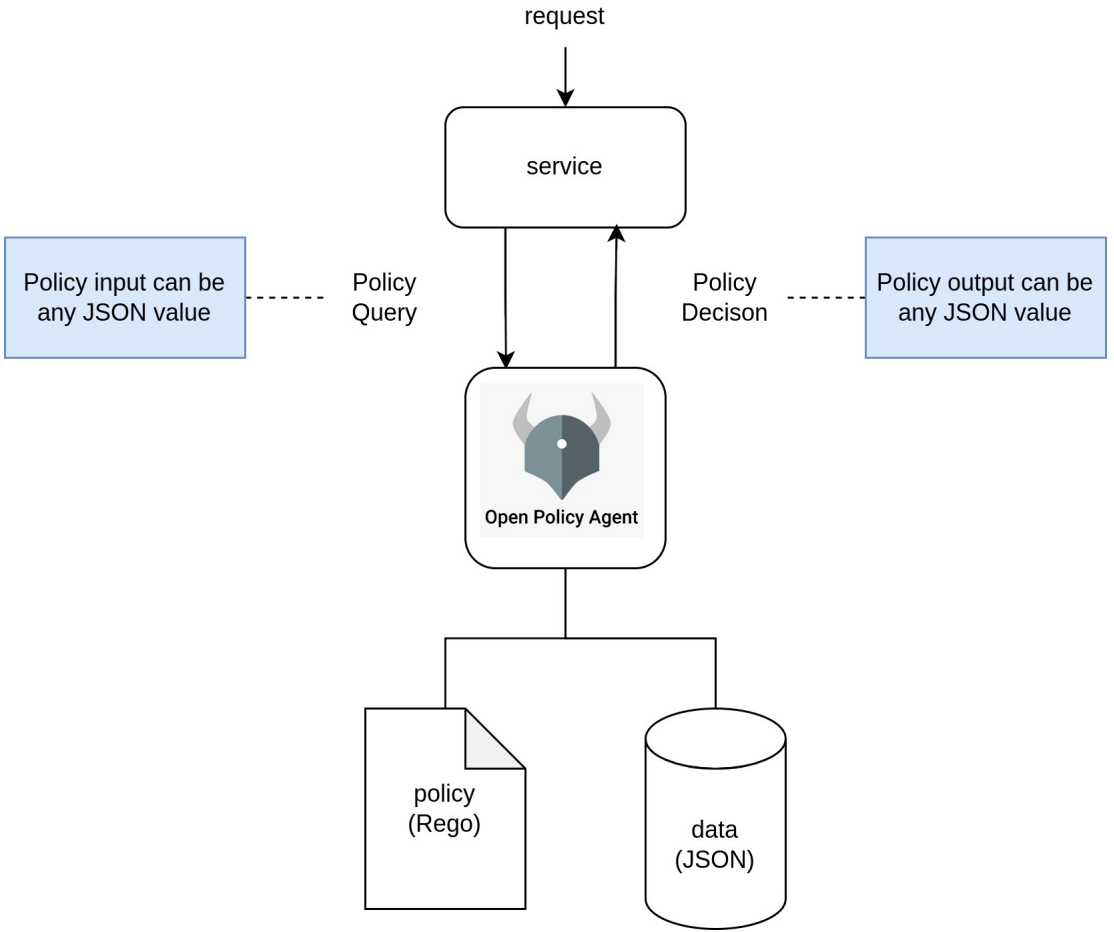
- Policy as code is a policy, realized as a reproducible, auditable, re-usable artifact.
- Definition of policy is decoupled from application logic.
- Policy can be used across the stack with many different languages, service, and tools.
- Policy *enforcement* is separated from policy *decision*.

Enter Open Policy Agent

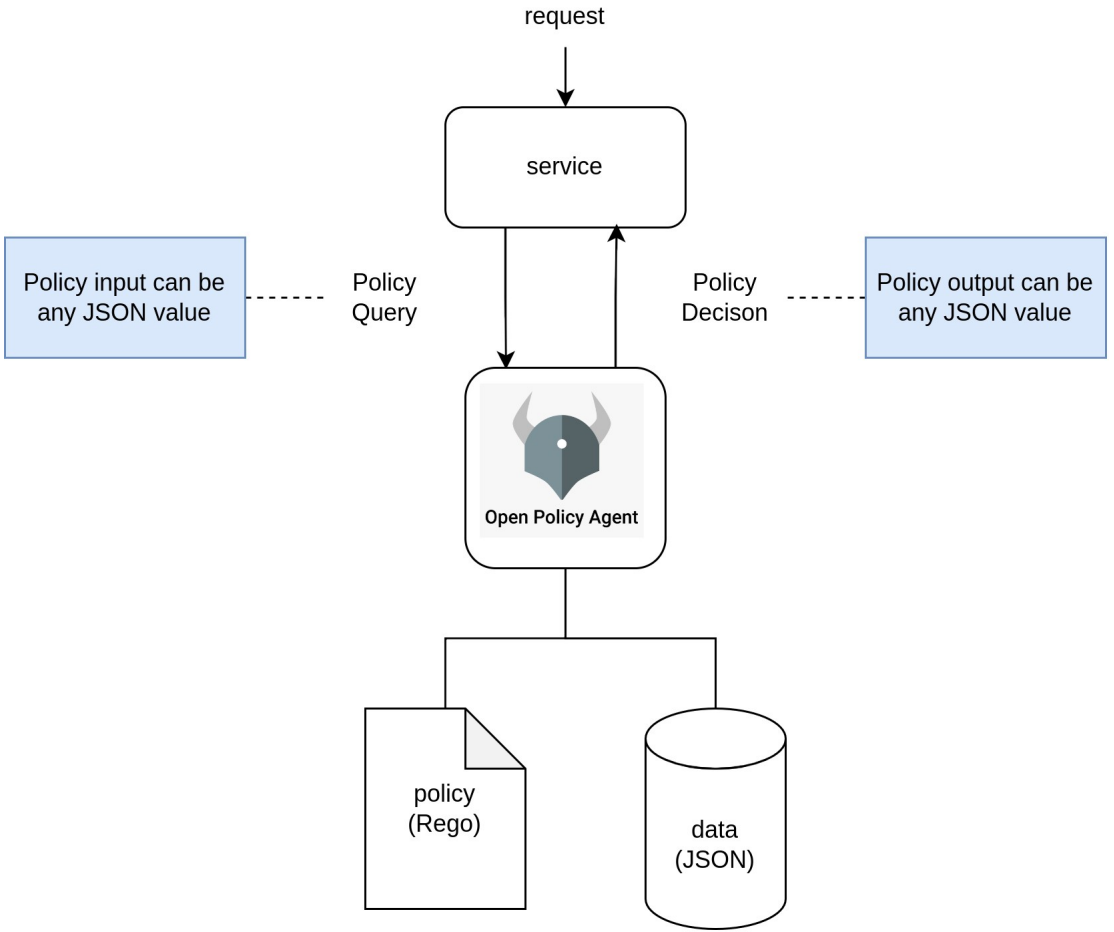
- Open Policy Agent (OPA) is a CNCF-graduated open-source runtime for policy-as-code
- 100+ integrations with other software: https://www.openpolicyagent.org/ecosystem/
- Policies are written in a Datalog-based DSL: **Rego**
- Rego allows expressing arbitrary transforms over JSON documents
- Try it in your browser: https://play.openpolicyagent.org/

Open Policy Agent

OPA Policy Decision Point

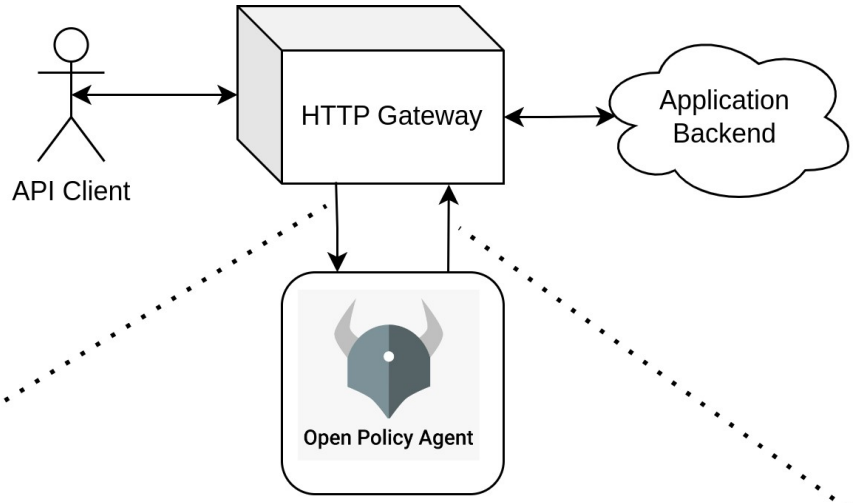


OPA Policy Decision Point



- OPA uses a Rego policy to transform JSON into JSON.
- Integrate with other services over HTTP or GRPC
- Supports control plane for policy+data distribution, decision logging

Scenario: HTTP Gateway Authorization



Policy Input:

```
{
    "method": "PUT",
    "headers": {
        "authorization": "bearer 123...",
        ...
    },
    "path": "/api/v1/files/budget.csv"
}
```

Policy Output:

```
{
    "decision": false,
    "reason_user": "access denied",
    "reason_admin": "missing role 'accounting'"
}
```

Rego Crash Course - Rules

```rego
package play

import rego.v1

default allow = false

# admins can do anything
allow if "admin" in data.roles[input.user]

allow if {
    input.action == "read"
    "reader" in data.roles[input.user]
}

allow if {
    input.action == "write"
    "writer" in data.roles[input.user]
}

```

INPUT

```json
{
    "user": "bob",
    "resource": "/documents/2024-Q3-report.xlsx",
    "action": "write"
}
```

DATA

```json
{
    "roles": {
        "alice": [
            "admin"
        ],
        "bob": [
            "reader"
        ],
        "tom": [
            "writer",
            "reader"
        ]
```

https://play.openpolicyagent.org/p/c6sGrNFCZc

Rego Crash Course - Iteration

```rego
package play

import rego.v1

default allow = false

allow if {
    input.amenity == "main-bar"
    "age-verified" in data.entitlements[input.patron]
}

allow if {
    input.amenity == "vip-bar"
    "age-verified" in data.entitlements[input.patron]
    some ent in ["vip-ticket", "backstage-pass"]
    ent in data.entitlements[input.patron]
}

allow if {
    input.amenity == "vip-lounge"
    some ent in ["vip-ticket", "backstage-pass"]
    ent in data.entitlements[input.patron]
}

allow if {
    input.amenity == "backstage"
    "backstage-pass" in data.entitlements[input.patron]
}
```

INPUT

```json
{
    "patron": "kate",
    "amenity": "main-bar"
}
```

DATA

```json
{
    "entitlements": {
        "alice": ["age-verified", "vip-ticket", "backstage-pass"],
        "bob": ["vip-ticket"],
        "jan": ["vip-ticket", "age-verified"],
        "mike": ["age-verified", "backstage-pass"]
        "kate": ["age-verified"]
    }
}
```

https://play.openpolicyagent.org/p/9zIw3YVQdh

Rego Crash Course – AND/OR

OR via iteration

```
12 allow if {
13     input.amenity == "vip-bar"
14     "age-verified" in data.entitlements[input.patron]
15     some ent in ["vip-ticket", "backstage-pass"]
16     ent in data.entitlements[input.patron]
17 }
```

OR via rules

```
12 allow if {
13     input.amenity == "vip-bar"
14     "age-verified" in data.entitlements[input.patron]
15     "vip-ticket" in data.entitlements[input.patron]
16 }
17
18 allow if {
19     input.amenity == "vip-bar"
20     "age-verified" in data.entitlements[input.patron]
21     "backstage-pass" in data.entitlements[input.patron]
22 }
```

https://play.openpolicyagent.org/p/DEGE5kF6qW

Running OPA Locally

- Binary releases are available here: https://github.com/open-policy-agent/opa/releases
- (demo video)
- Editor integrations
  - Regal – LSP server + linter - https://github.com/StyraInc/regal/
  - VSCode extension - https://github.com/open-policy-agent/vscode-opa
  - Vim plugin - https://github.com/tsandall/vim-rego
  - Find more at awesome-opa - https://github.com/StyraInc/awesome-opa

Adopting Policy as Code

- More than just integrating a library and transforming some JSON.
- Developers, IT, ops, & management need to buy into policy-as-code.
- Call to action: increasing regulation & cybersecurity risk means your organization will need to care about fine grained access control sooner rather than later.
- Authorization Maturity Model quantifies where your organization is in your authorization journey.

Authorization Maturity Model in a Nutshell

what?	← less mature	developing	more mature →
location	application code implements policy	externalized to repo or IAM system	centralized, re-usable, decoupled from app code
enforcement	application enforces policy	backend applications, gateway	full zero trust at all layers
governance	developers deploy permissions changes when asked	change controls in place for permissions	policies from centralized IAM team automatically take precedence over application policies
auditability	no reporting or logging	ad-hoc reporting or logging	access logs saved in external system, policy changes cause audit log events
people	IAM is a function of HR team	IAM is a function of security and/or eng team	central IAM team manages policies, guides and assists other teams with integration, non-technial users can implement buisness requirements as policies easily

Authorization Maturity Model

- https://registration.styra.com/rs/668-YOD-554/images/unified_authorization_maturity_model.pdf

Conclusion

- Policy-as-code makes answering authorization questions reproducible, language-agnostic, and re-usable.
- Open Policy Agent (OPA) is a CNCF-graduated open source project with wide adoption, and lets you implement policy-as-code using the Rego language.
- When implemented properly, policy-as-code is a powerful tool to align what actually happens in the datacenter with high-level business requirements in a centralized, auditable way.

End.

- Questions?

- Want to connect?
  - contact@cdaniels.net
  - https://www.linkedin.com/in/charles-daniels-932767bb/