# Policy for Use-Case Awareness

Brian Sweeney

Production Engineering (SRE)

2023

# How did we get here?

- You're a multitenant platform SRE.
- You provide a range of features, but a power user wants a narrow "happy path"
- How can you let them know when somebody goes off path?

# Let's talk about policy!

- if/else in code, for one user, is ... suboptimal.
- My team should get out of the way of their data collection
- If only there were a language that would allow them to express their rules (and maybe for us to safely enforce those rules down the road)

There is - it's Rego.

# What's Rego?

- Rego is the language behind Open Policy Agent and conftest.
- "Rego was inspired .. [a] query language. Rego queries are assertions on data stored in OPA. These queries can be used to define policies that enumerate instances of data that violate the expected state of the system".[1]
- So if your API produces JSON, we can see what does (and does not) match your policy.

---

[1]https://www.openpolicyagent.org/docs/latest/policy-language

# How is this useful?

- Say your platform links workflows to some owner by their email address
- Did you validate the email is internal? Is the email owned by a Managed Service Provider?

# Letś validate email domains (the data)

```
name: Second
size: 72
accountFlows:
  - act_flow_001:
      responsibleEmail: vhhcjfjl@sharklasers.com
      FlowNodes:
        - name: Flow_001
          address: flow001.vcap.me
          region: US
          parallelism: 5
        - name: Flow_002
          address: flow002.vcap.me
          region: US
          parallelism: 5
  - act_flow_002:
      responsibleEmail: vhhcjfjj@sharklasers.com
  - act_flow_003:
      responsibleEmail: vhhcjfjj@vcap.me
```

# Lets validate email domains (the policy)

```
package main

deny[msg] {
    flow := input.accountFlows[_]
    # print(flow)
    some key
    m := is_internal_email(flow[key].responsibleEmail)
    not m
    # print(flow)
    msg = sprintf("External email addres in flow  %v: %v",
}

is_internal_email(email_address) = true {
    pattern := "@sharklasers.com$"
    matched := regex.match(pattern, email_address)
    matched == true
} else = false { true }
```

# Let's validate email domains (the results)

```
>conftest test --policy policies\domain \
    backend/data/tenants/Second.yaml
FAIL - backend/data/tenants/Second.yaml - main - \
  External email addres in flow act_flow_003: \
  vhhcjfjj@vcap.me

1 test, 0 passed, 0 warnings, 1 failure, 0 exceptions
```

# Letś validate email domains (the results)

```
>curl -s http://localhost:8000/tenant/Second|jq .|\
   conftest test --policy policies\domain -
FAIL - - main - External email addres in flow \
 act_flow_003: vhhcjfjj@vcap.me

1 test, 0 passed, 0 warnings, 1 failure, 0 exceptions
```