

OpenPolicyAgent.jl

Policy Enforcement in Julia Enterprise Applications with OPA

JuliaCon 2024

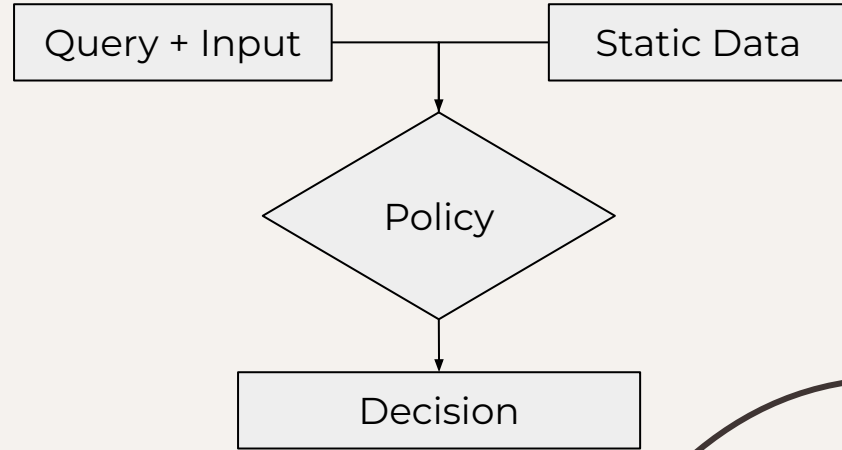
Slides: <https://bit.ly/460JkyT>



Tanmay | JuliaHub Inc. | @tanmaykm | tanmaykm@gmail.com | tanmay@juliahub.com

What is a Policy?

Policies are rules,
which when queried,
examine data & conditions
(input/static)
and arrive at a decision.



Example

- Only logged in users are allowed

Example

- Only logged in users are allowed
- No one is allowed to use the accounting system outside office hours

Example

- Only logged in users are allowed
- No one is allowed to use the accounting system outside office hours
- Except “the boss”, who is allowed at all times

Example

- Only logged in users are allowed
 - No one is allowed to use the accounting system outside office hours
 - Except “the boss”, who is allowed at all times
 - No one except auditors are allowed during audits, not even the boss
-

Need of Policy Engine

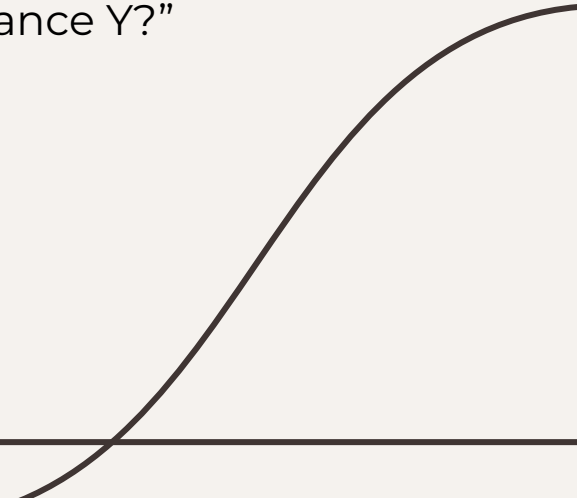
Rules are:

- likely not made by the product developers
- may change over time.

We simply want to ask “Are we allowed X under circumstance Y?”

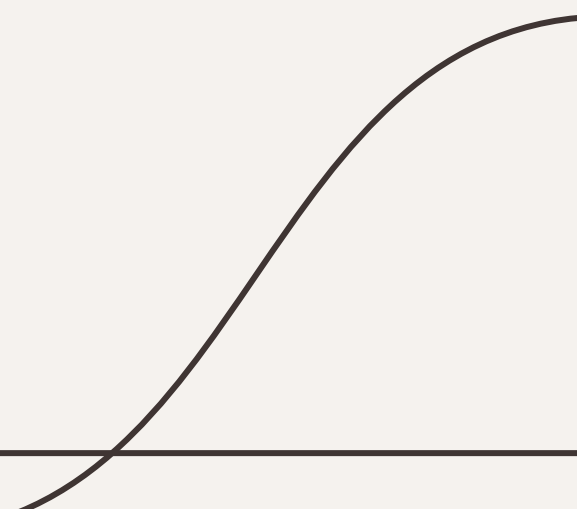
And let policy engine decide.

E.g. Is access allowed given

- time now, office hours
 - is user the boss?
 - are audits on?
- 

What is Open Policy Agent (OPA)?

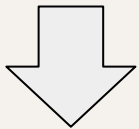
- Rules Engine
- High level declarative language: Rego
- Policy as Code
- Simple APIs



How does that help?

Offload Decision Making

Retain Enforcement



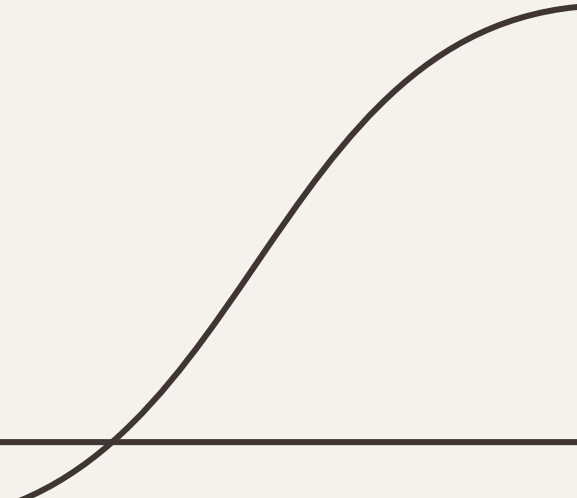
Simplification

Simpler

- **Compliance**
 - Centralized Policies
 - Centralized Decision Making
- **Audits**
- **Integration**
 - API based. As a library or a service
 - Easy to integrate at any enforcement point.

OpenPolicyAgent.jl

Encapsulates OPA for Julia applications

1. Service (decision making)
 2. Client (enforcement)
 3. CLI (use opa executable as tool)
 4. Plugin to consume partial eval results
- 

Service

- Start OPA as a service
- Monitor failures, restart

```
using OpenPolicyAgent.Server: MonitoredOPAServer, start!, stop!  
  
opa_server = MonitoredOPAServer("config.yaml")  
  
start!(opa_server)  
  
...  
  
stop!(server::MonitoredOPAServer)
```

Client

- Management & Enforcement APIs
- OpenAPI (REST)

```
opa_client = OpenPolicyAgent.Client.DataApi(openapi_client)
response, _ = OpenPolicyAgent.Client.get_document(opa_client,
    "policies/my_policy/allowed"
);
allowed = response.result
```

CLI

- For command line tasks
 - E.g. evaluate policy, test rego expr, query data
- Invaluable for
 - policy development
 - administration
 - troubleshooting

CLI

```
julia> using OpenPolicyAgent
```

```
julia> import OpenPolicyAgent: CLI
```

```
julia> ctx = CLI.CommandLine();
```

```
julia> CLI.opa(ctx; help=true);
```

An open source project to policy-enable
your service.

Usage:

opa [command]

Available Commands:

bench Benchmark a Rego query

build Build an OPA bundle

capabilities Print the capabilities of OPA

check Check Rego source files

...

Flags:

-h, --help help for opa

Use "opa [command] --help" for more information
about a command.

Partial Evaluation

Query provides only part of info that policy needs.

OPA can come up with either:

- Concrete decision (to consume directly)
- Simplified rules to evaluate
 - later with more info
 - elsewhere (maybe in a different language)

OpenPolicyAgent.jl includes

- interface to plug in processors for partially processed rules
 - implementation to process these rules to SQL where conditions.
-

Q&A

Slides:

<https://bit.ly/460JkyT>



Tanmay | JuliaHub Inc. | @tanmaykm | tanmaykm@gmail.com | tanmay@juliahub.com

Thanks

Tanmay | JuliaHub Inc. | @tanmaykm | tanmaykm@gmail.com | tanmay@juliahub.com

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**