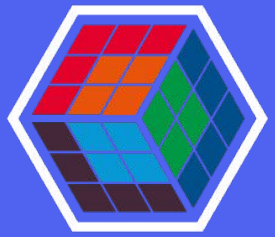


THE DEVELOPER'S CONFERENCE

Opa, posso entrar?

Autorização de APIs com Open Policy Agent

Thiago Pinto | **linkedin.com/in/thspinto**

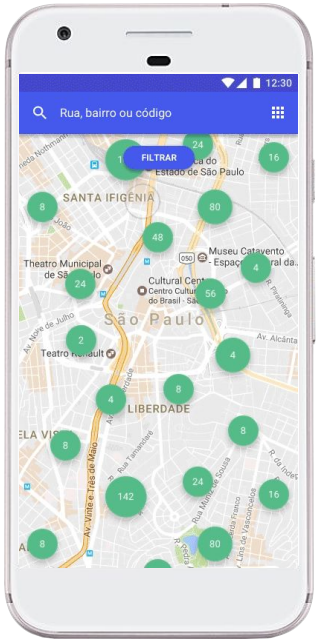
Agenda

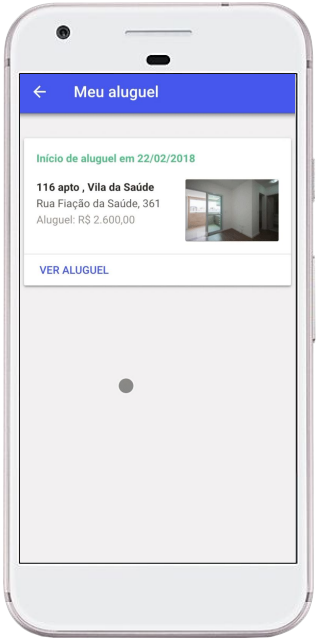
- Contexto
- Problema
- Conhecendo o Open Policy Agent
- Nossa proposta
- Resultados
- Próximos passos

Contexto

Sobre mim

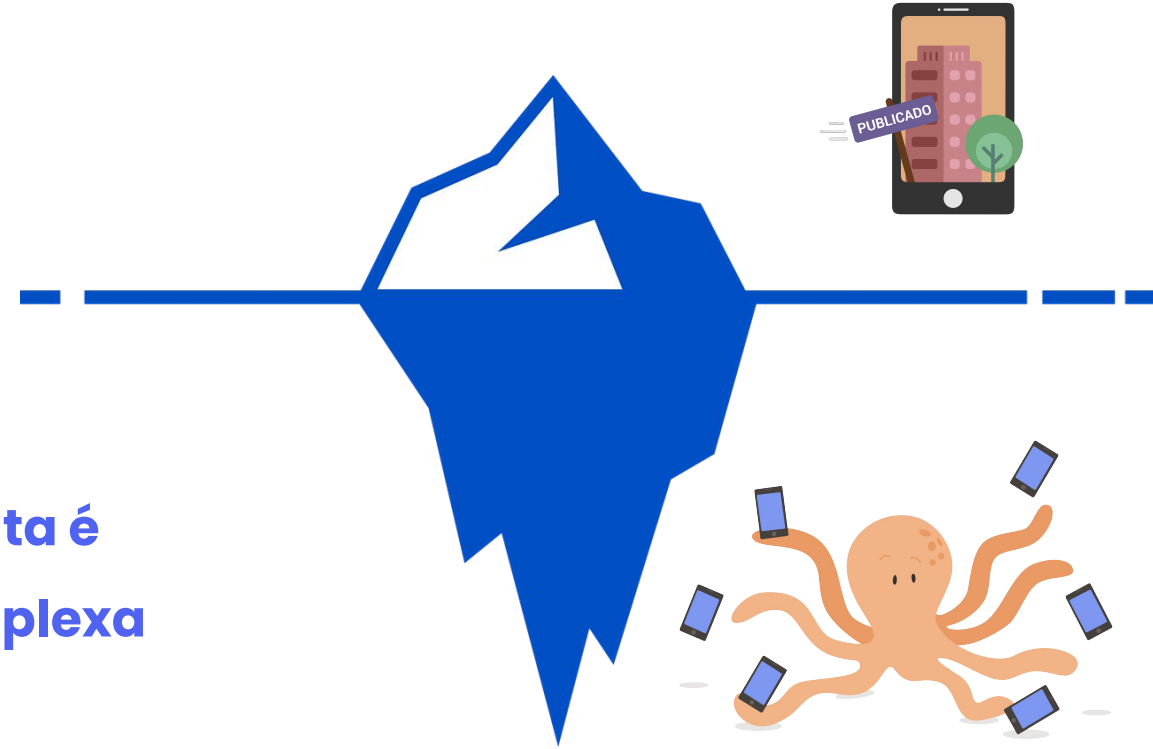
Sobre o QuintoAndar





Sobre o QuintoAndar

- ~300 Serviços
- 300+ Devs
- A jornada ponta-a-ponta é operacionalmente complexa



O problema

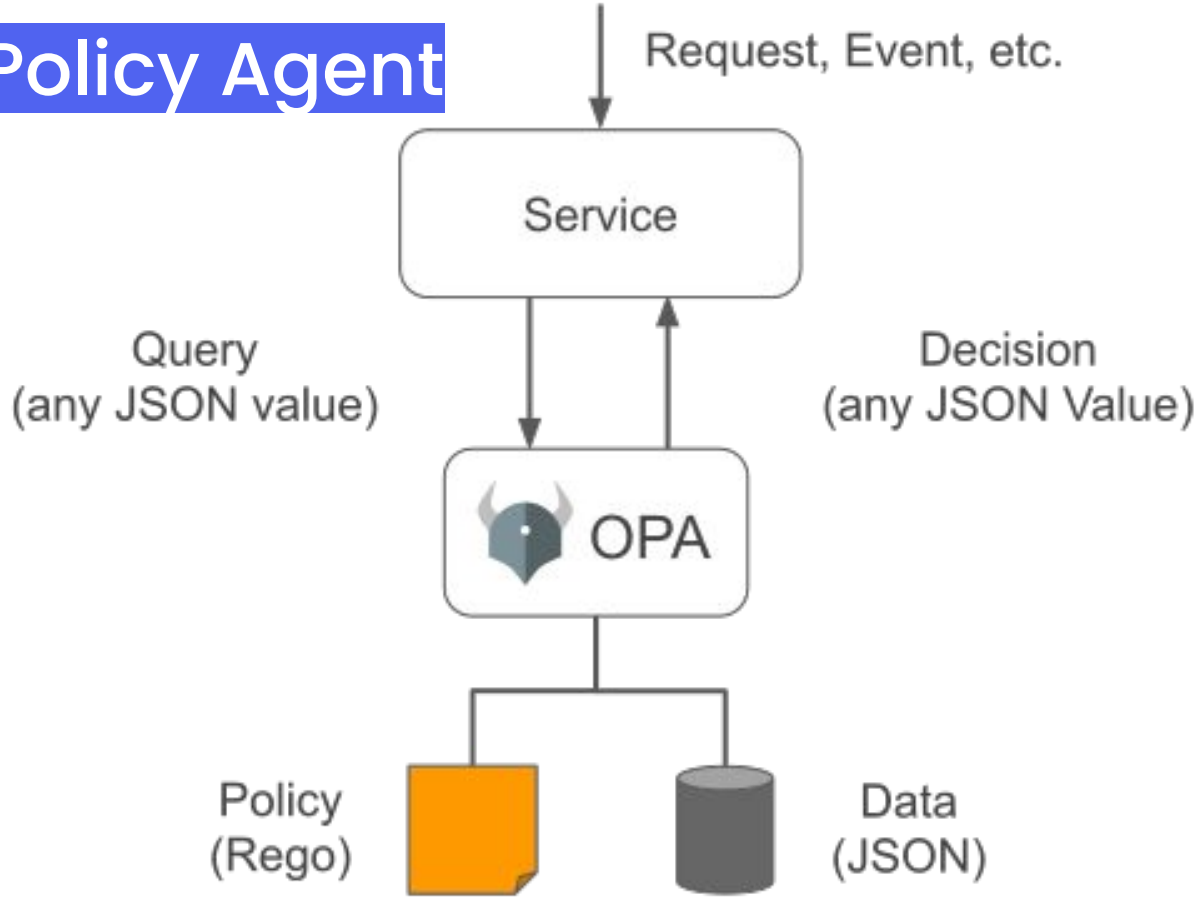
- Padronizar implementação de Autenticação e Autorização
- Gerir permissões de forma ágil
- Dar visibilidade da API para o time de governança

Conceitos

- **Autenticação: O processo de validar quem o usuário diz ser**
- **Autorização: Conceder ou negar acesso a um recurso**
- **RBAC (Role-Based Access Control):**
  Autorizar baseado em papéis designado ao usuário autenticado

Opa, tudo bem?

Open Policy Agent



Open Policy Agent

```
package demo.app

default allow = false

allow {
 input.method == "GET"
 input.path == "/health"
}
```

```
package demo.app

default allow = false

allow {
 input.method == "GET"
 input.path == "/health"
}

allow {
 input.method == "GET"
 input.path == "/test"
}
```

Open Policy Agent

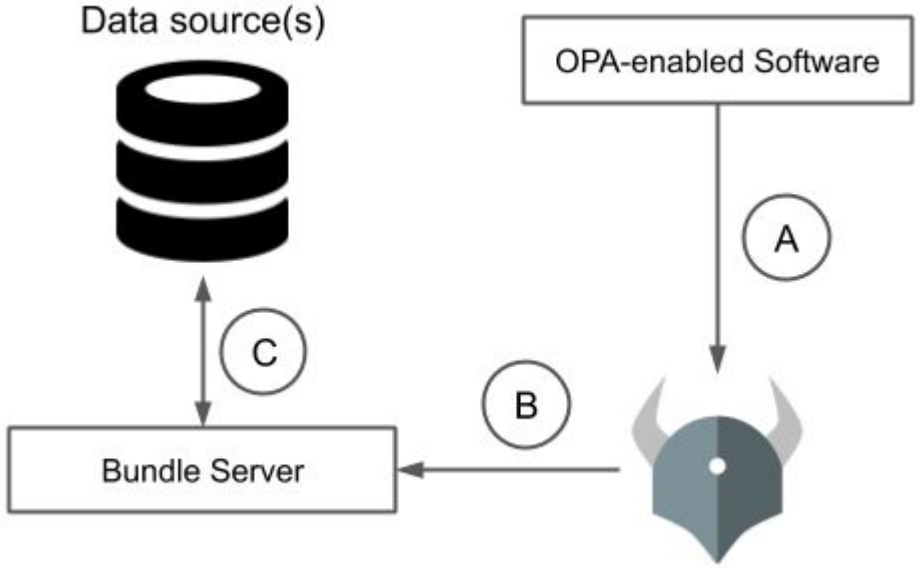
```
➜  curl localhost:8181/v1/data/demo/app -s -d '{"input": {"path":"/", "method":"GET"}}' | jq .
{
  "result": {
    "allow": false
  }
}
➜  curl localhost:8181/v1/data/demo/app -s -d '{"input": {"path":"/health",
"method":"GET"}}' | jq .
{
  "result": {
    "allow": true
  }
}
```

OPA com Testes

```
package demo.app


test_public_path {
 result := allow
    with input.path as "/health"
    with input.method as "GET"
 result == true
}
```

OPA external data

- **Polling**
- **Push**
- **At execution time**



OPA bundle

```
$ opa build -b ./
```

```
#config.yaml
services:
 - name: s3
   url: https://s3.amazonaws.com/demo
   credentials:
(...)
bundles:
 app:
   resource: bundle.tar.gz
   service: s3
```

OPA e Identidade

```
default allow = false


allow {
 data.admin_uers[claims.username]
}
```

```
admin_users:
  -   Toledo
  -   Fernando
```

Por que OPA?

- Centralizar a gestão de políticas de acesso em stack não padronizado

- Possibilidade de implementar políticas de controle para além de APIs
  - Kubernetes
  - Terraform

Nossa Proposta

RBAC com OPA

- Definir papéis para acessar recursos das APIs
- JWT com os grupos que o usuário pertence
- Grupos mapeam para papel de API

```
default allow = false

allow {
 input.method == "GET"
 input.path == "/podcasts"
 user_groups[_] == "podcasts:list"
}


user_groups := groups {
 data.groups[claims.groups[_]]
}
```

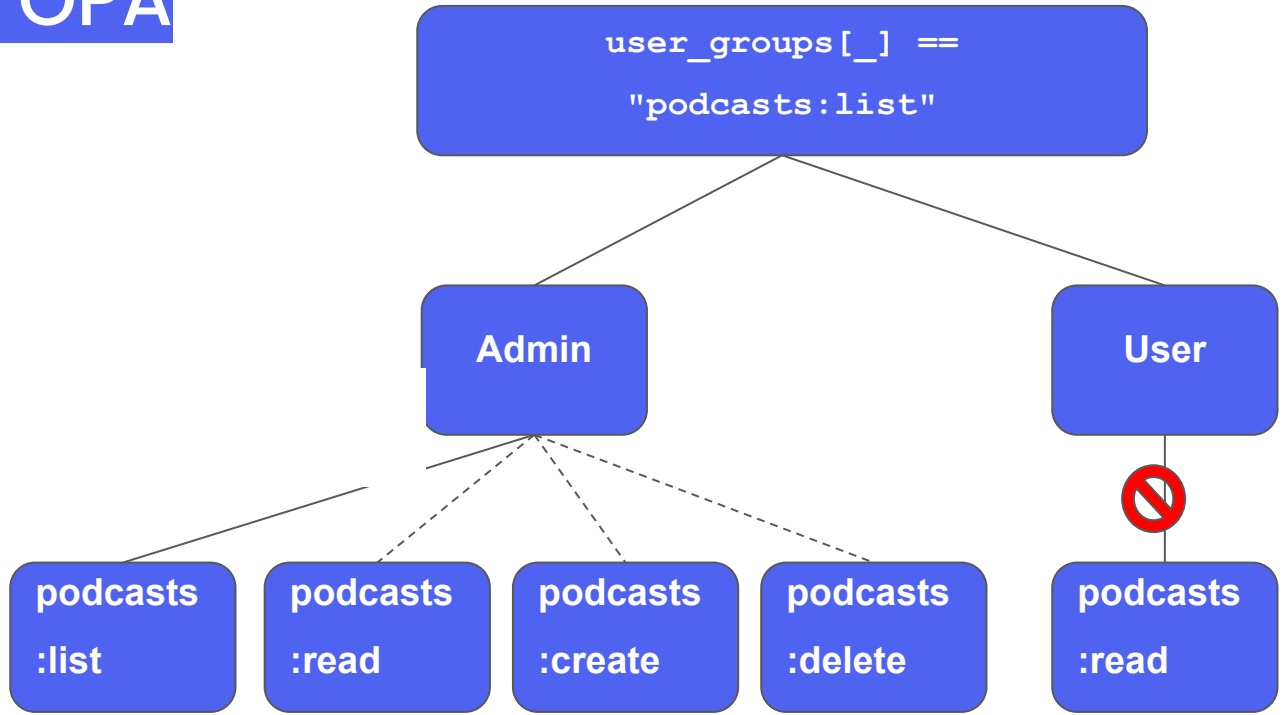
RBAC com OPA

```
groups:
 Admin:
   - podcasts:list
   - podcasts:read
   - podcasts:create
   - podcasts:delete
 User:
   - podcasts:read
```

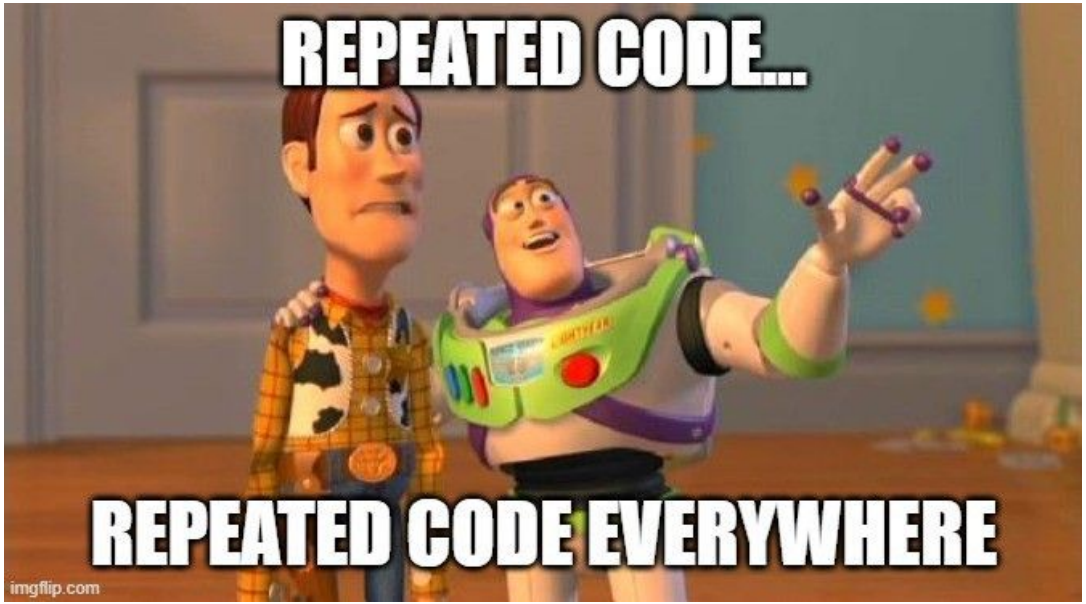
```
default allow = false

allow {
 input.method == "GET"
 input.path == "/podcasts"
 user_groups[_] == "podcasts:list"
}


user_groups := groups {
 data.groups[claims.groups[_]]
}
```

RBAC com OPA

```
groups:
 Admin:
   - podcasts:list
   - podcasts:read
   - podcasts:create
   - podcasts:delete
 User:
   - podcasts:read
```



RBAC com OPA



Toy Story (Pixar/Disney)

```
default allow = false

allow {
 input.method == "GET"
 input.path == "/podcasts"
 user_roles[_] == "podcasts:list"
}


user_roles := roles {
 data.groups[claims.groups[_]]
}
```

OpenAPI

- **Padronização**
- **Documentação**
- **Gerar código**
  - **clients**
  - **mocks**


```java
@GetMapping("/books")
@SecurityRequirement(name = "oauth2", scopes = "demo.books:list")
public List<String> getBooks() {
  return Arrays.asList("book1", "book2");
}
```


```
 "/api/books":
   get:
     security:
     - oauth2:
       - demo.books:list
     operationId: getBooks
     responses:
{...}
```

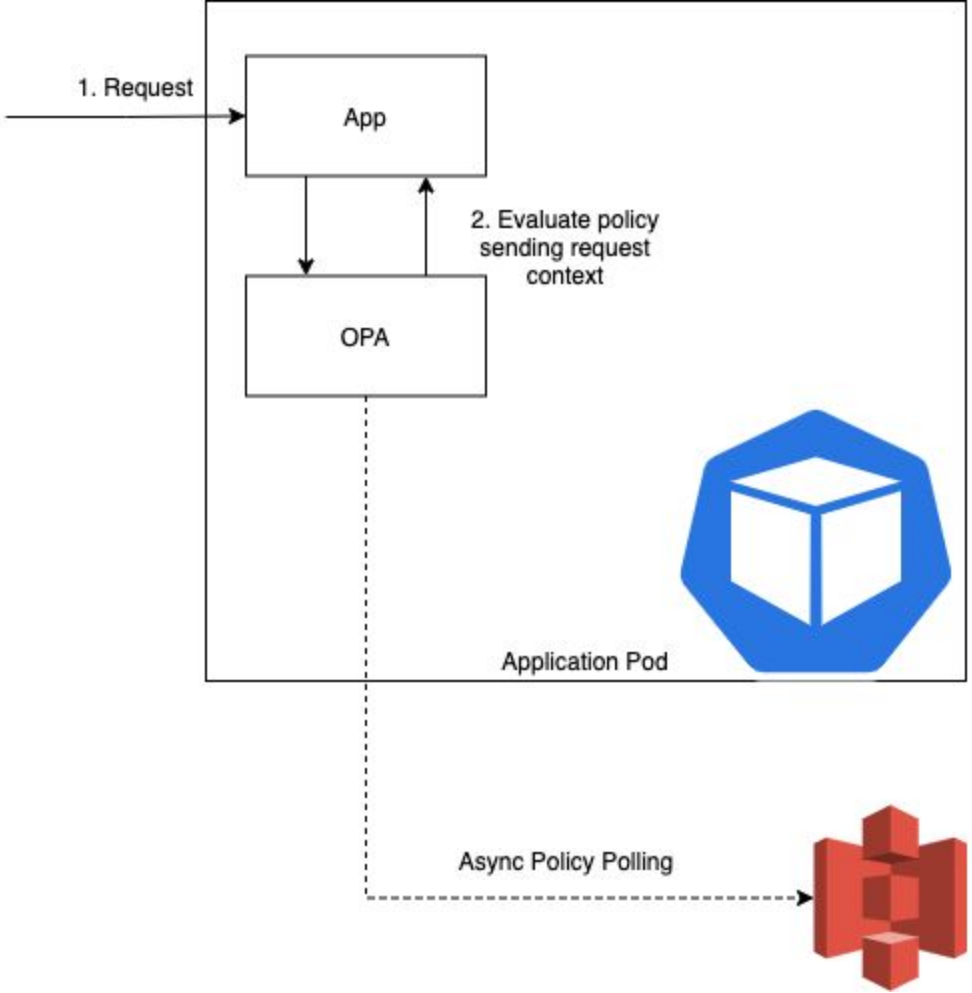
OPA e OpenAPI

```
allow {
 user_roles[_] == requestRoles[_]
}

requestRoles = roles {
 roles := data.openapi.paths[input.path][lower(input.method)].security[_].oauth2
}
```

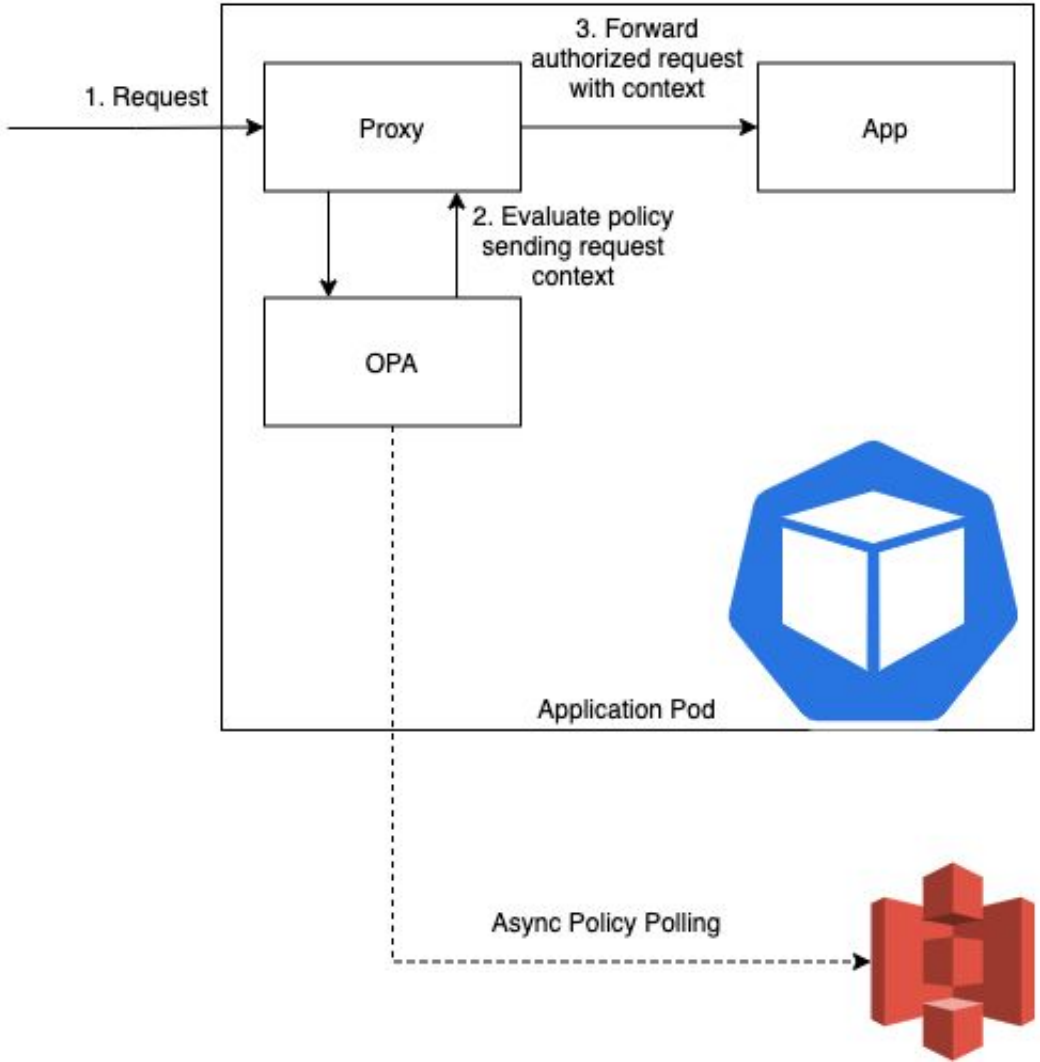
Organização das Políticas

```
policy/
├── apps/ # Application specific functions
|   ├── app_name/ # this will act as an adapter to convert the input sent from the
application to the commons functions expected input
├── commons/ # General purpose functions
|   ├── v1/ # v1 specific functions
|   |   ├── authentication/ # Auth specific functions
|   |   |   ├── auth.rego
|   |   |   ├── auth_test.rego
|   ├── data/ # Environment specific data
|   |   ├── prod.yaml
|   |   ├── staging.yaml
```

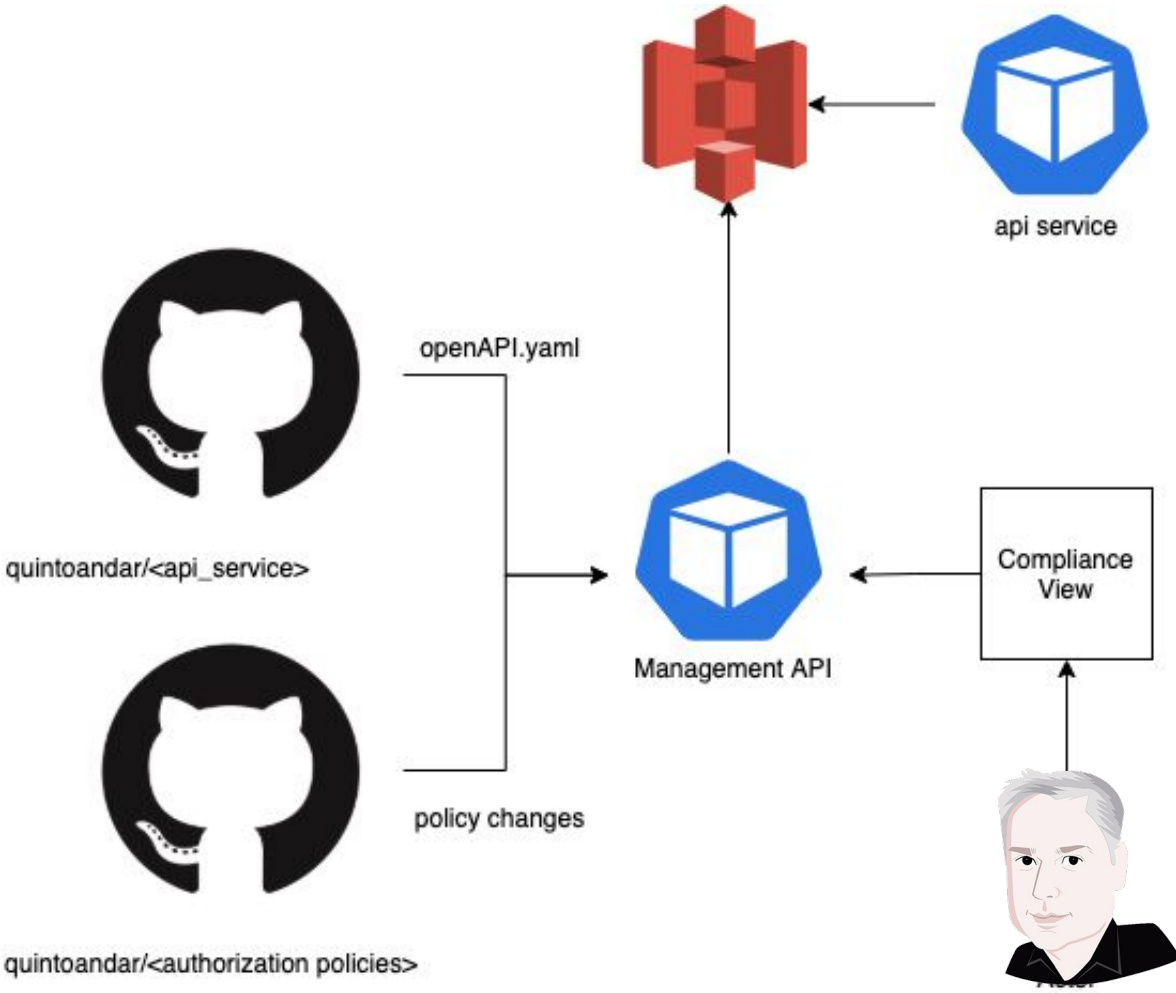
Uso distribuído



Uso distribuído



Gerenciamento



Resultados

- Visibilidade das regras em ambiente heterogêneo de linguagens
- Centralização da gestão de autorização
- Padronização da coleta de logs de autorização
- Diminuição da responsabilidade core dos serviços

Próximos passos

- **Otimização**
- **Escalar a adoção**
- **Sistema de gestão para time de governança**
- **Alertas de outlier baseados nos logs de auditoria**

Referências

- Demo repo
- Open Policy Agent
- OpenAPI

Vagas

https://carreiras.quintoandar.com.br/

Obrigadx!

Thiago Pinto | **linkedin.com/in/thspinto**