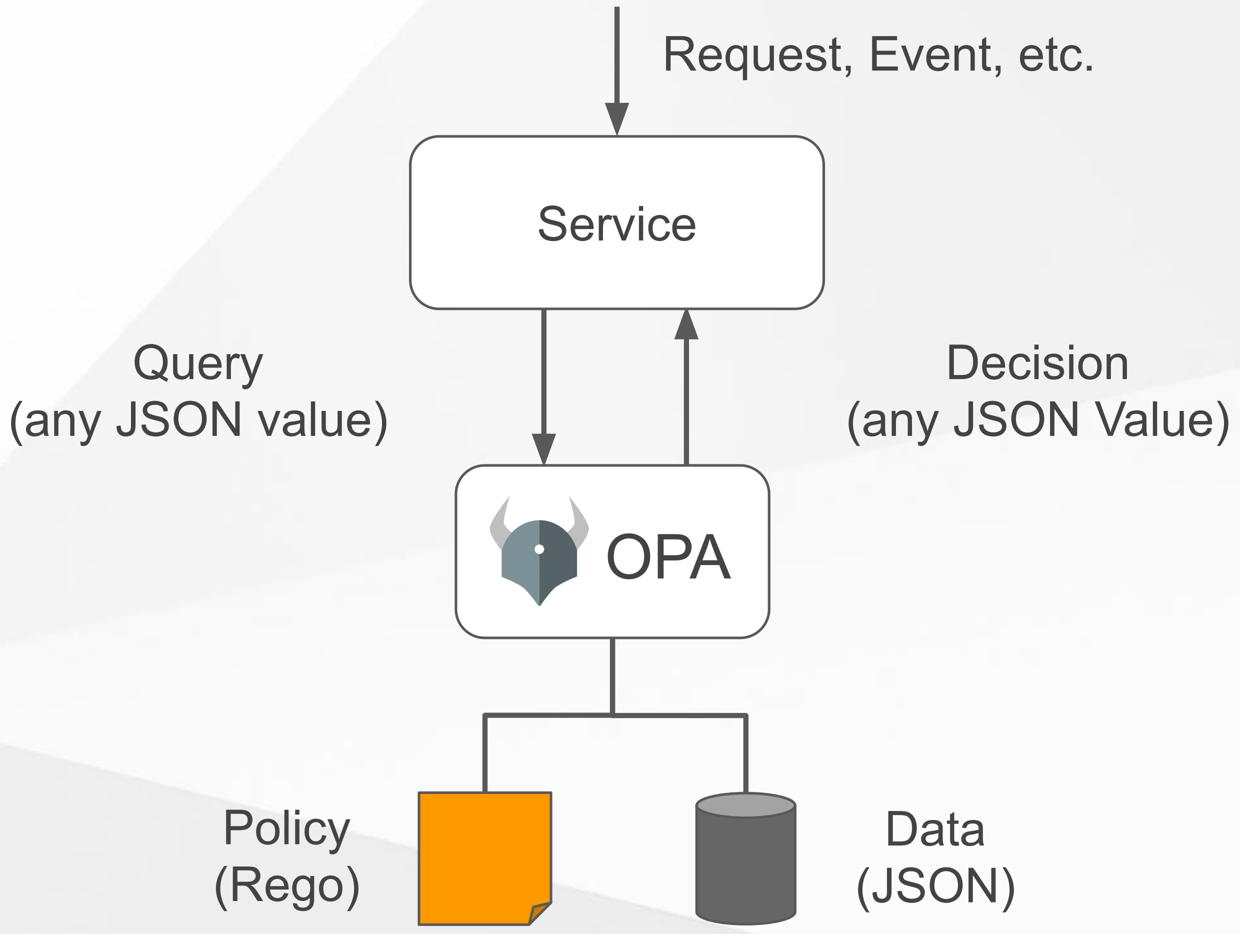
Open Policy Agent

Overview

- OPA
- OPA Syntax Demo
- KMO Use Case
- KMO - OPA Integration (Spring Security) Demo

OPA

Decouples access control policy **decision making** from **enforcement**



Why decouple?

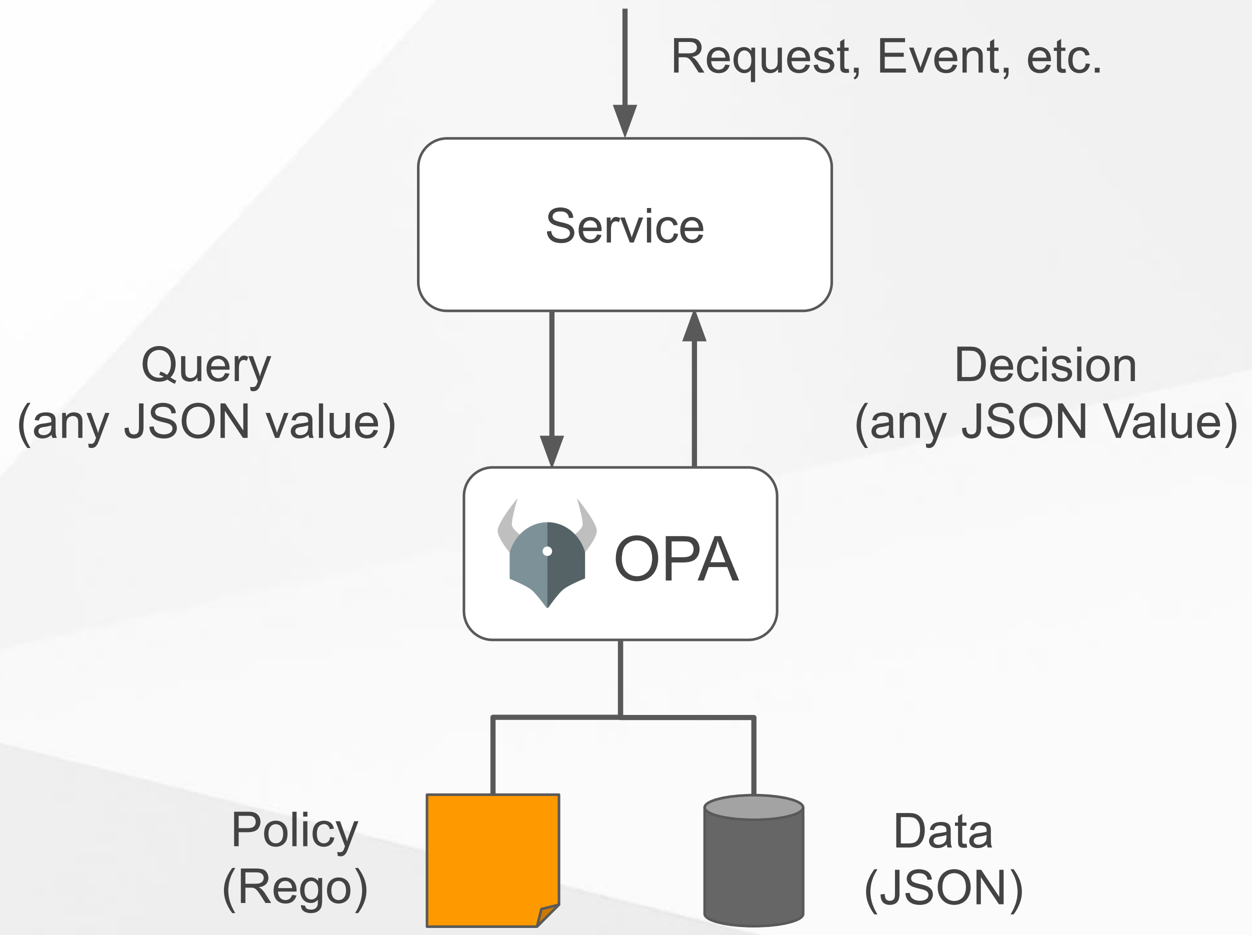
1. Abstract authorization away from business logic as much as possible (separation of concerns)
2. Different apps can use a common language/platform to specify their access control policy

Why OPA?

1. Rego provides a declarative syntax when it comes to specifying your policy
2. You can update your access control policy without restarting your application
3. OPA allows current policies to be queried centrally - sets us up for integration in future

Demo with VSCode

Query & Policy



KMO Recap

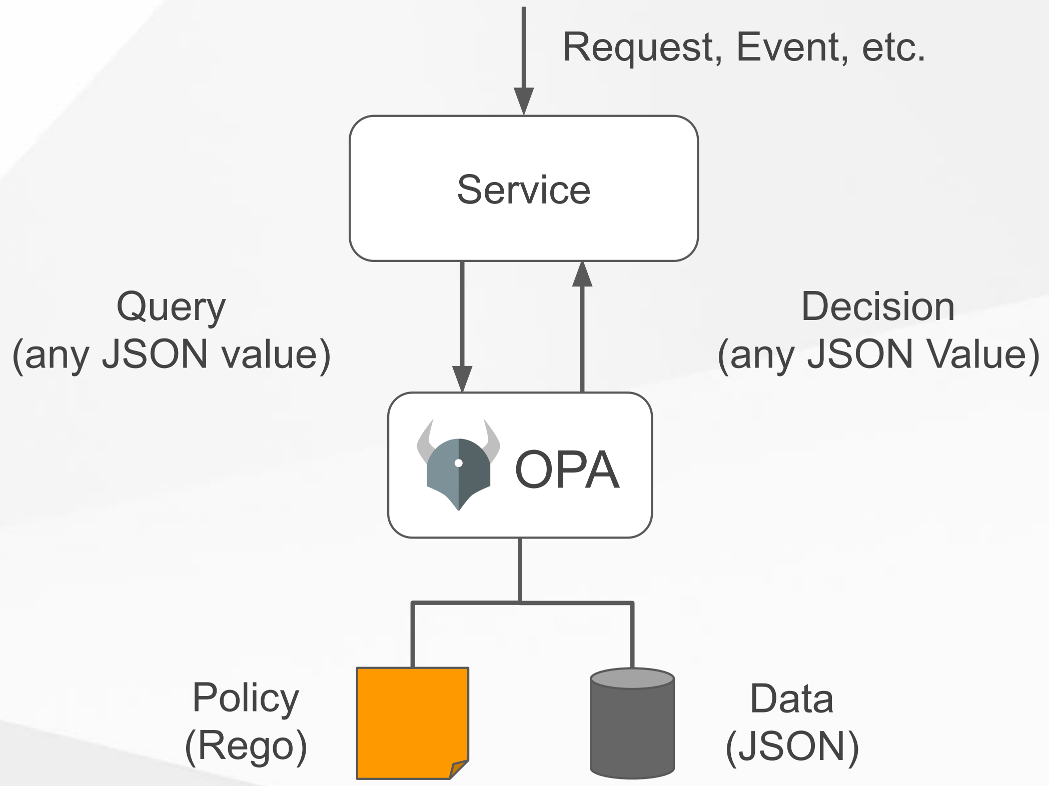
For each resource, there's owner team, collaborator teams, etc... (ACL of the resource)

A user can be part of many teams. These teams will be matched against the resource ACL to see if the user has access.

How will it be done?

In KMO, each resource is tied to an API endpoint (except for some).

For each access via the API endpoint, **Query** , and supply ACL for a **Decision**



Demo with Spring Security

1. Show unsecured access to general info and location info
2. Show access control information (next slide)
3. Secure the application, explain AccessDecisionManager & Voter
4. Access general info with *aqua_grunt*
5. Open OPA container logs
   ```
   docker logs -f <CONTAINER_NAME> 2>&1 >/dev/null | jq '.'
   ```
6. Access general info then location info with *magma_grunt*
7. Access location info with *magma_galatic_grunt*
8. Show tests

Team & Resource Information

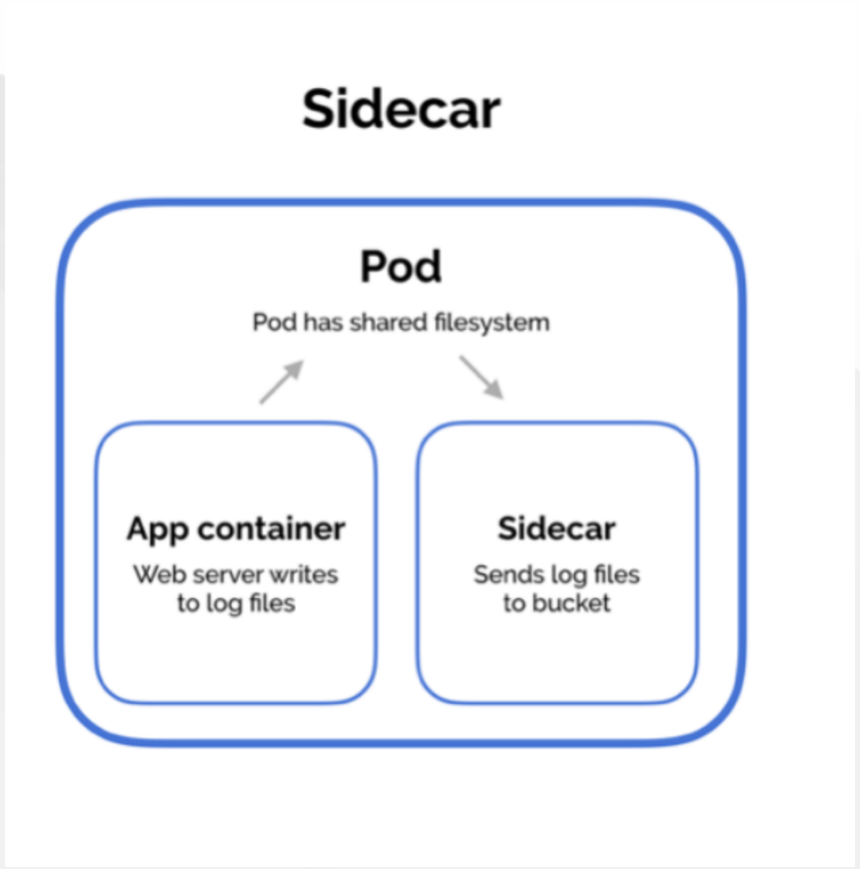
Resource ID	Owner	Collaborator	Viewer
1 (Bulbasaur)	Rocket	Galactic	Magma

Access Control Policy

	Owner	Collaborator	Viewer
General	READ	READ	READ
Location	READ	READ	

Deployment

Sidecar Pattern in Kubernetes


Sidecar
Pod
Pod has shared filesystem
App container
Web server writes to log files
Sidecar
Sends log files to bucket