KubeCon | CloudNativeCon

North America 2017

How Netflix is Solving Authorization Across Their Cloud

Manish Mehta - Engineer @ Netflix
Torin Sandall - Tech Lead @ OPA Project

Manish Mehta

Senior Security Engineer @Netflix
mmehta@netflix.com

Projects:

- Bootstrapping Identities
- Secrets Management
- PKI
- Authentication
- Authorization

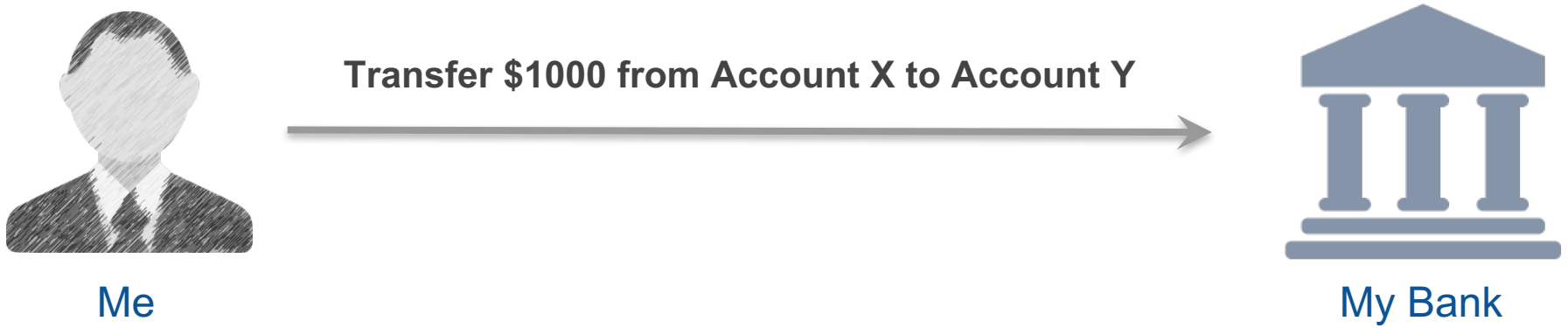
Torin Sandall

Software Engineer @ OPA project
 @sometorin

Projects:

- Open Policy Agent
- Kubernetes
- Istio (security SIG)
- Likes: Go, Quality, Good abstractions

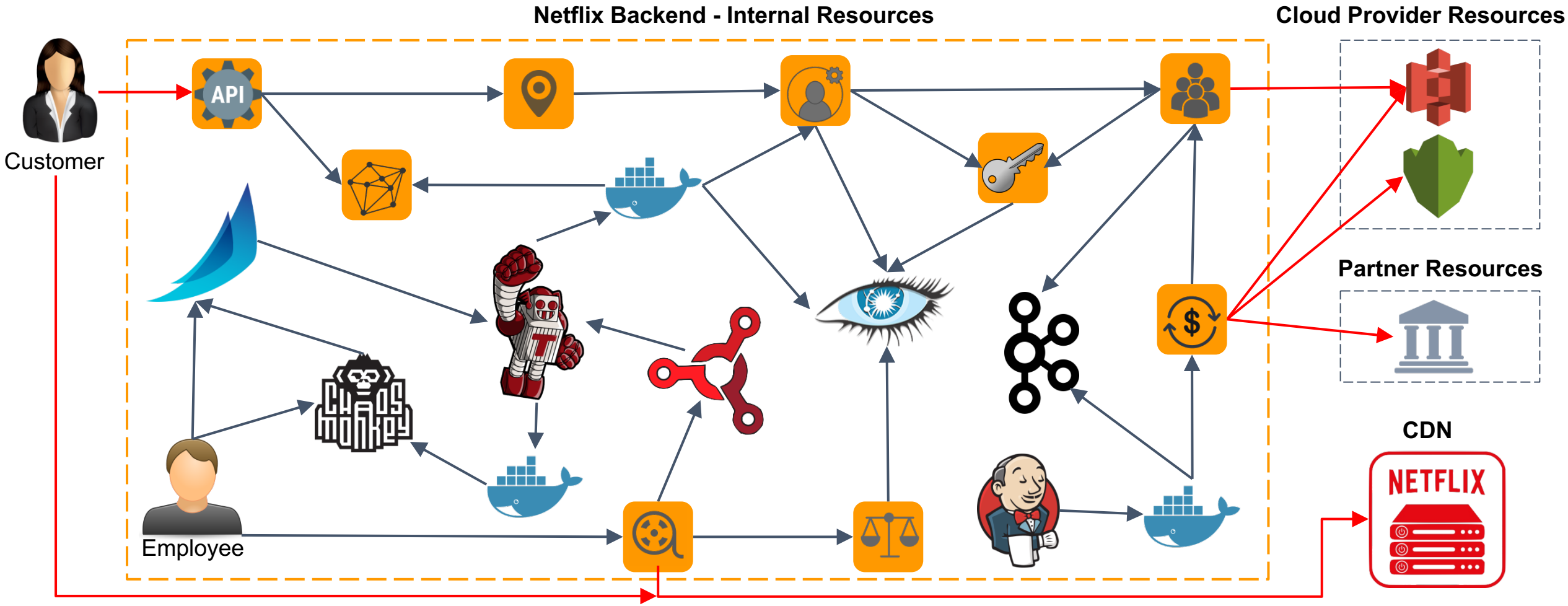
Background - Definitions



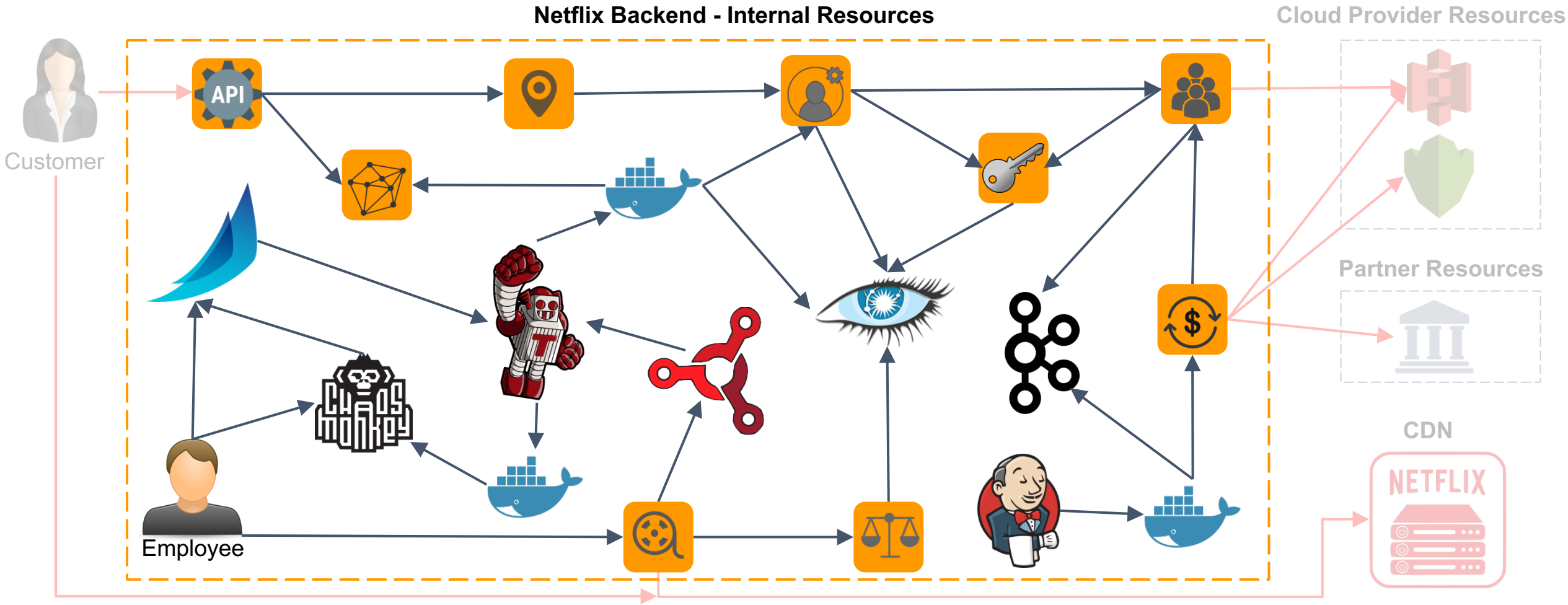
1. Verify the Identity of the Requester (Authentication or AuthN)
2. Verify that the Requestor is authorized to perform the requested operation (Authorization or AuthZ)

These 2 steps do not need to be tied together !!

Background - Netflix Architecture



Background - Netflix Architecture



AuthZ Problem

A (simple) way to **define** and **enforce** rules that read

```
Identity I
can/cannot perform
Operation O
on
Resource R
```

For **ALL** combinations of *I*, *O*, and *R* in the ecosystem.

Design Considerations

Company Culture
- Freedom and Responsibility

Resource Types
- REST endpoints, gRPC methods, SSH, Crypto Keys, Kafka Topics, …

Identity Types
- VM/Container Services, Batch Jobs, Employees, Contractors, …

Underlying Protocols
- HTTP(S), gRPC, Custom/Binary, ...

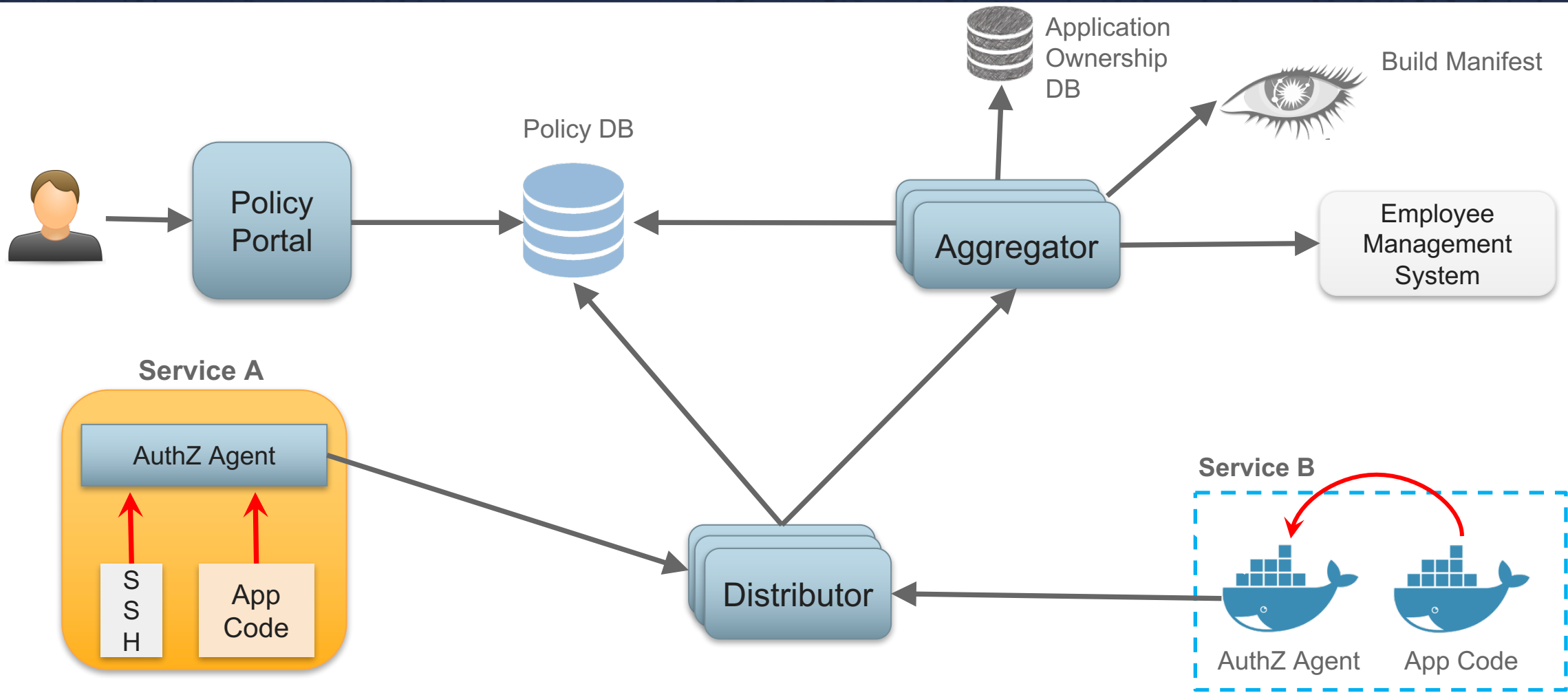
Implementation Languages
- Java, Node JS, Python, Ruby, …

Latency
- Call depth and Service rate

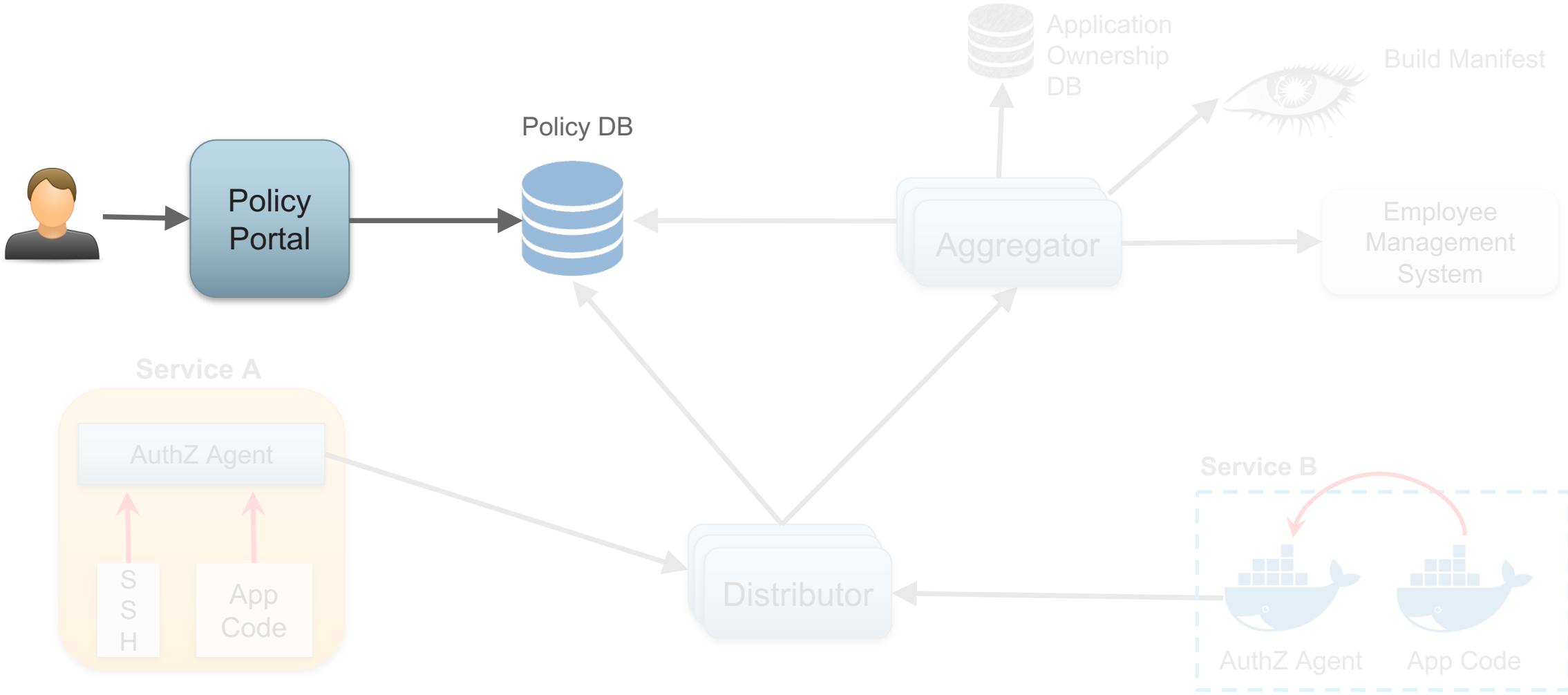
Flexibility of Rules
- Hard-coded structure vs. language-based

Capture Intent
- Did you actually do what you think you did?
- Don’t just trust, verify !!

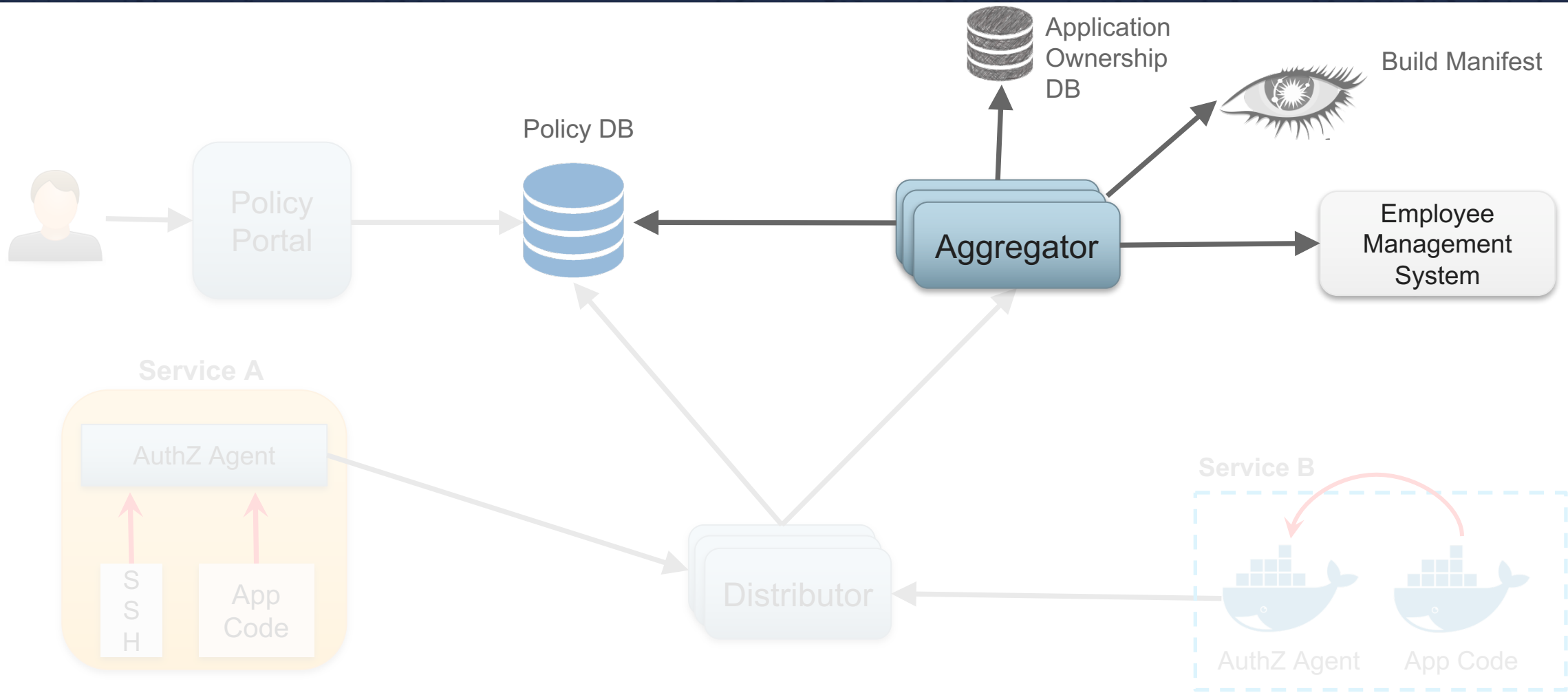
High-level Architecture



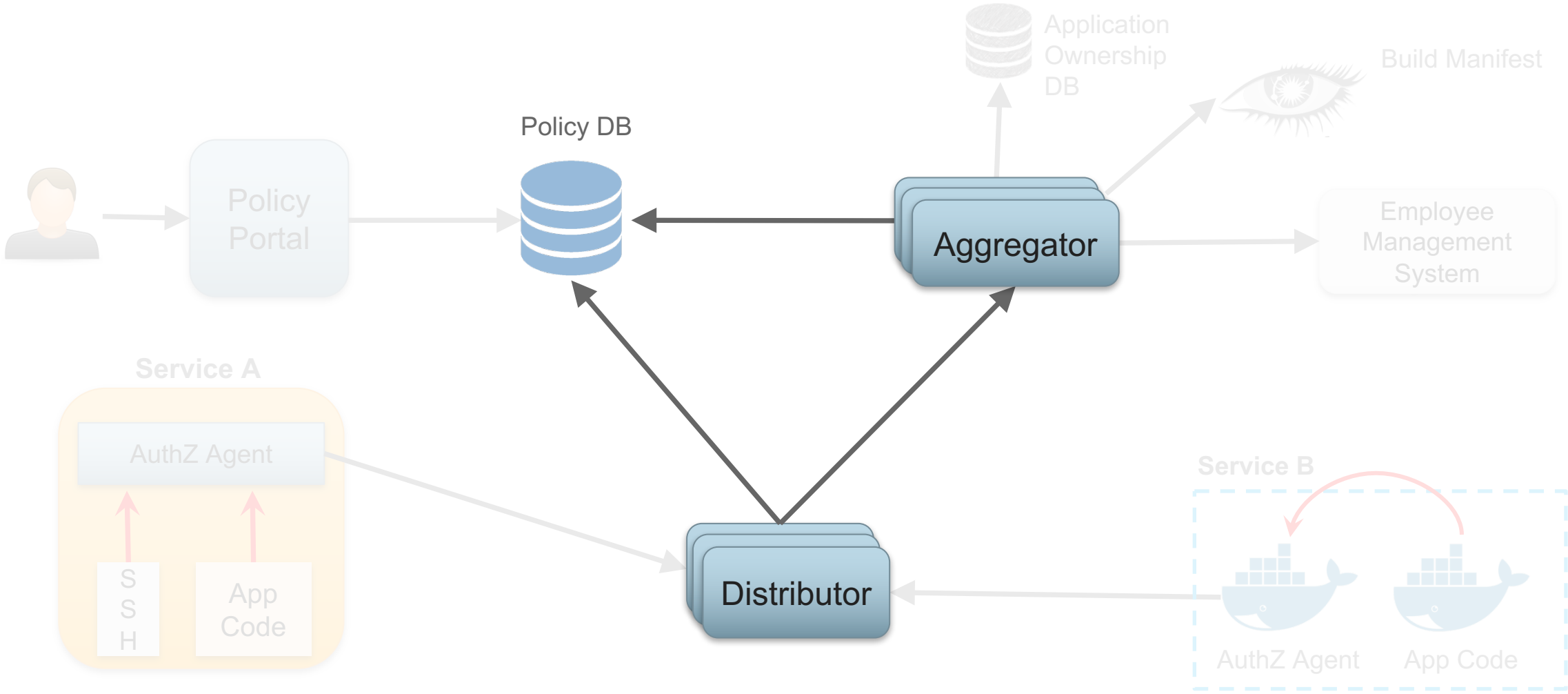
High-level Architecture



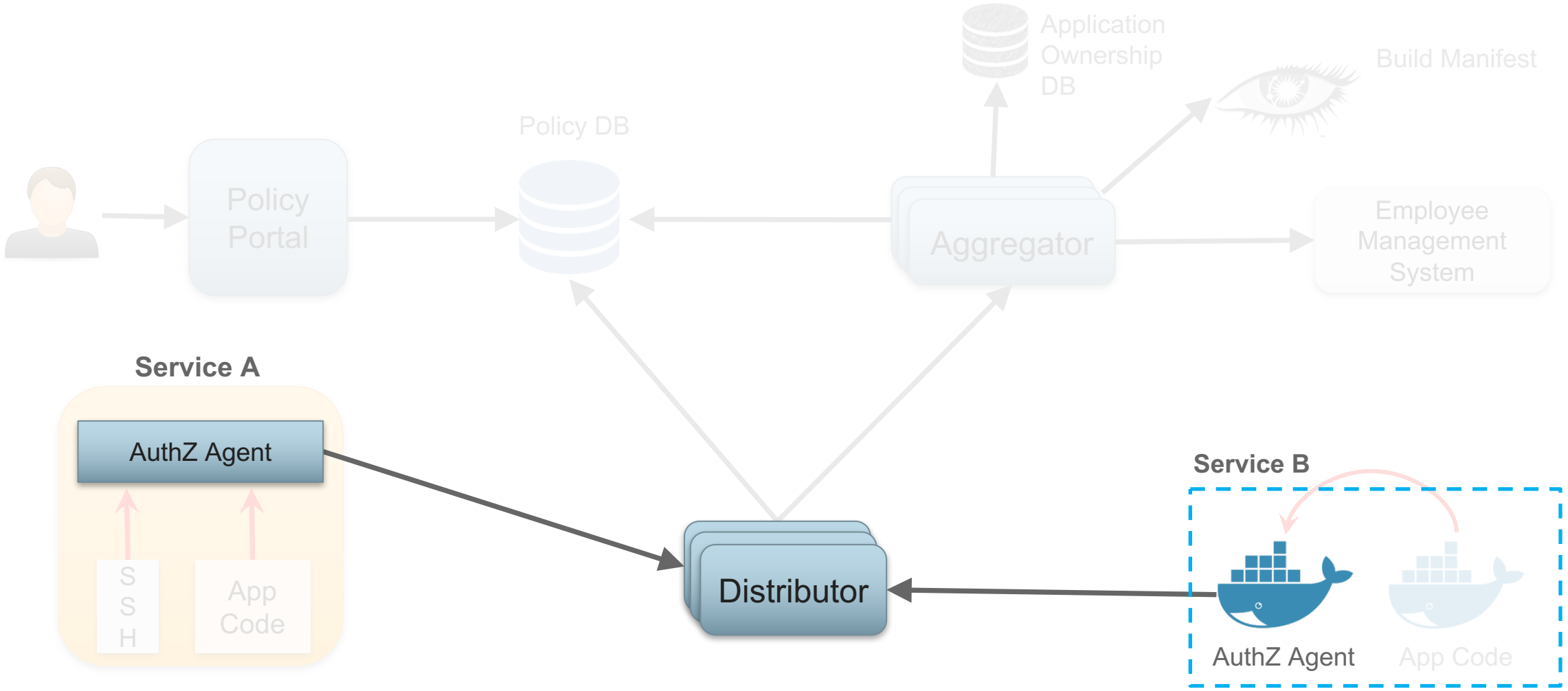
High-level Architecture



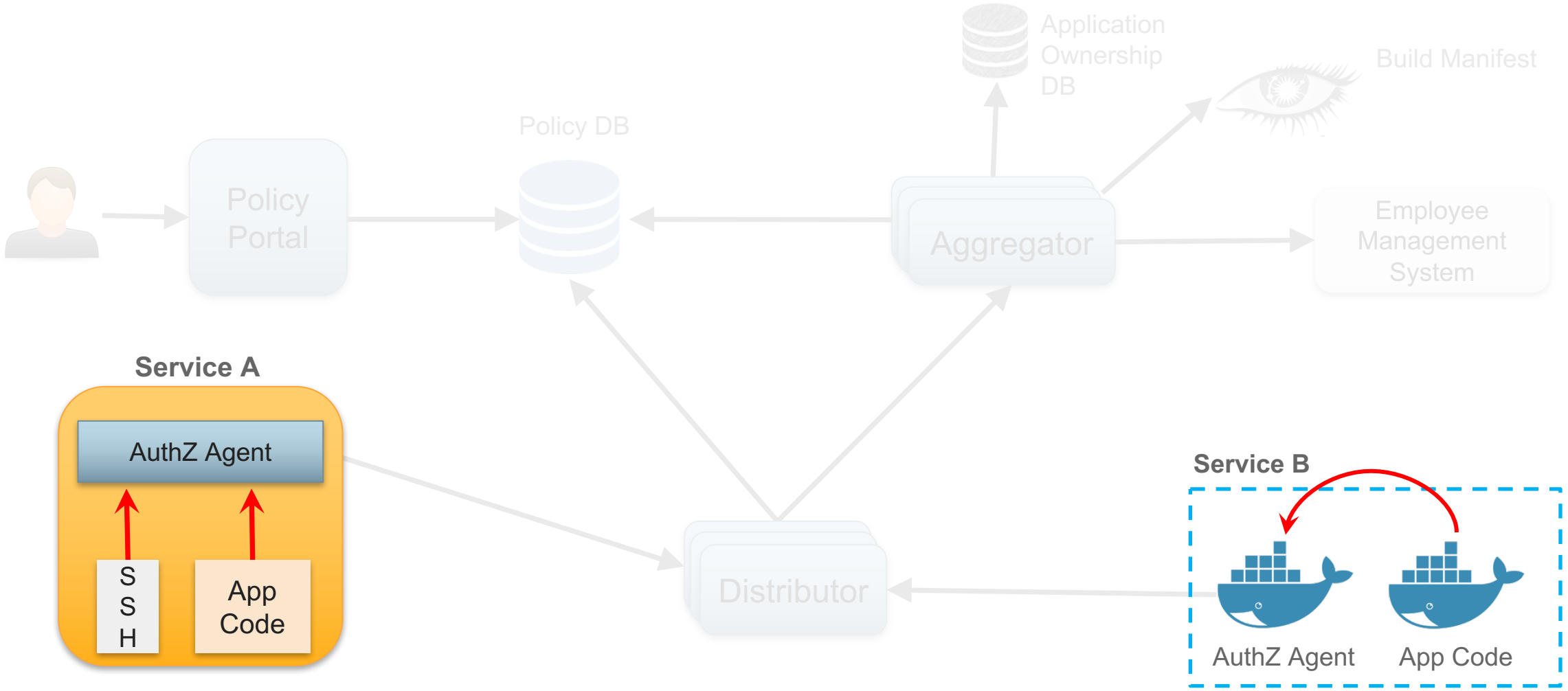
High-level Architecture



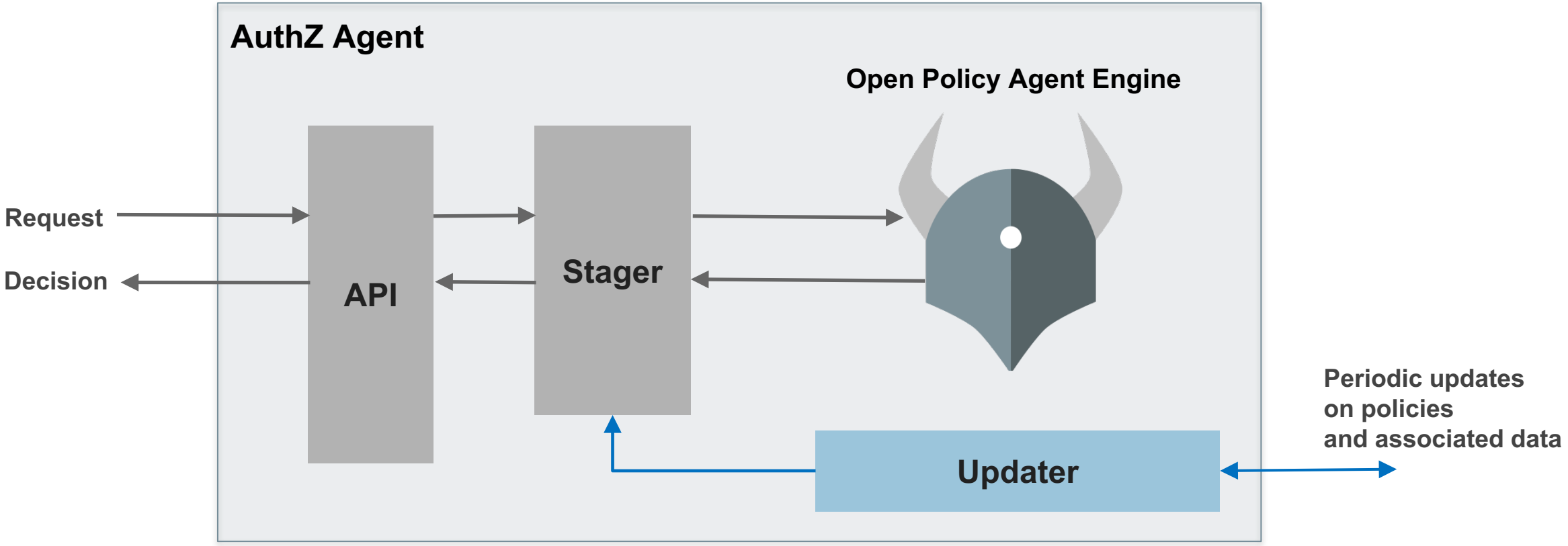
High-level Architecture



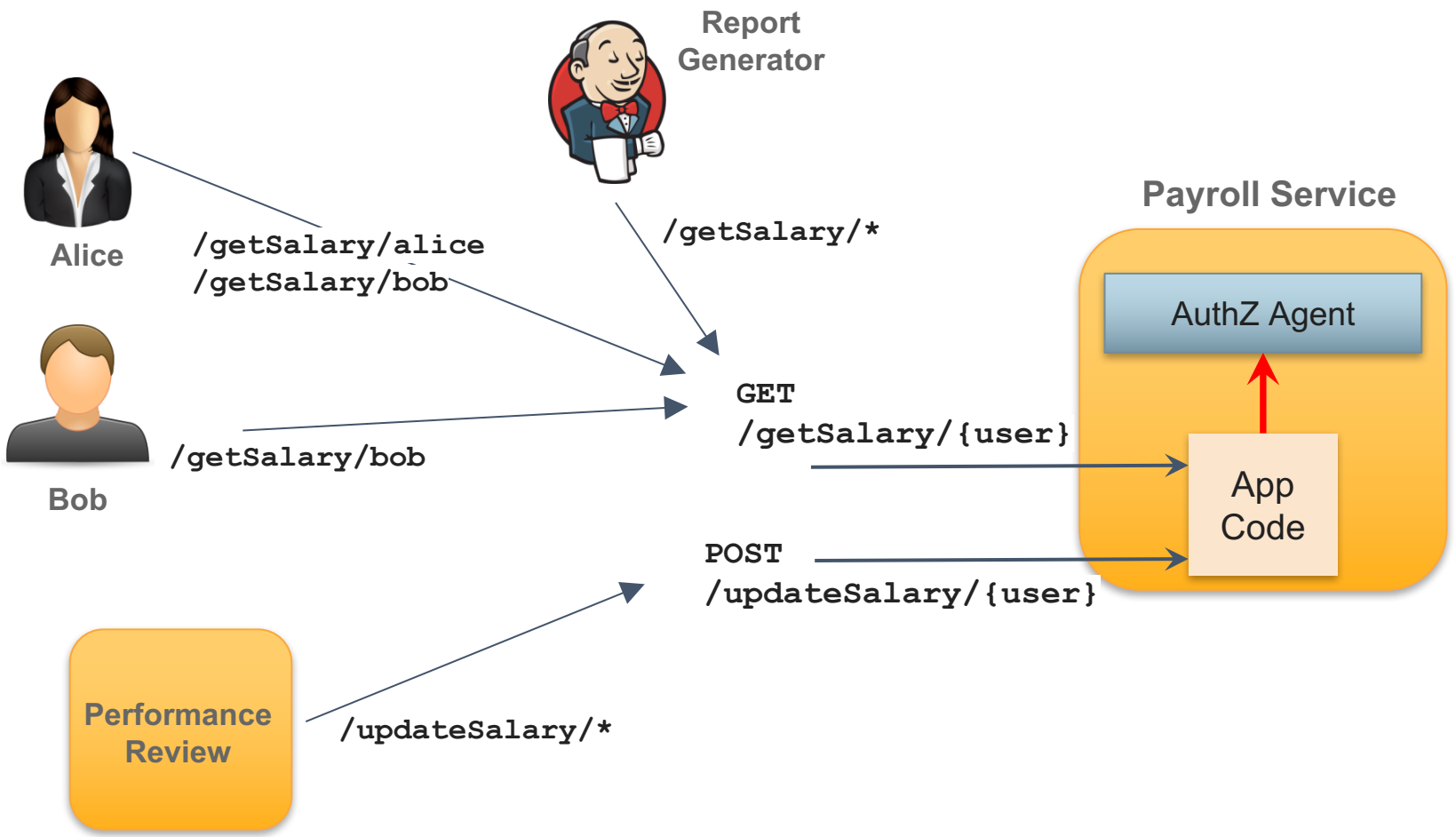
High-level Architecture



AuthZ Agent Internals



Example Setup

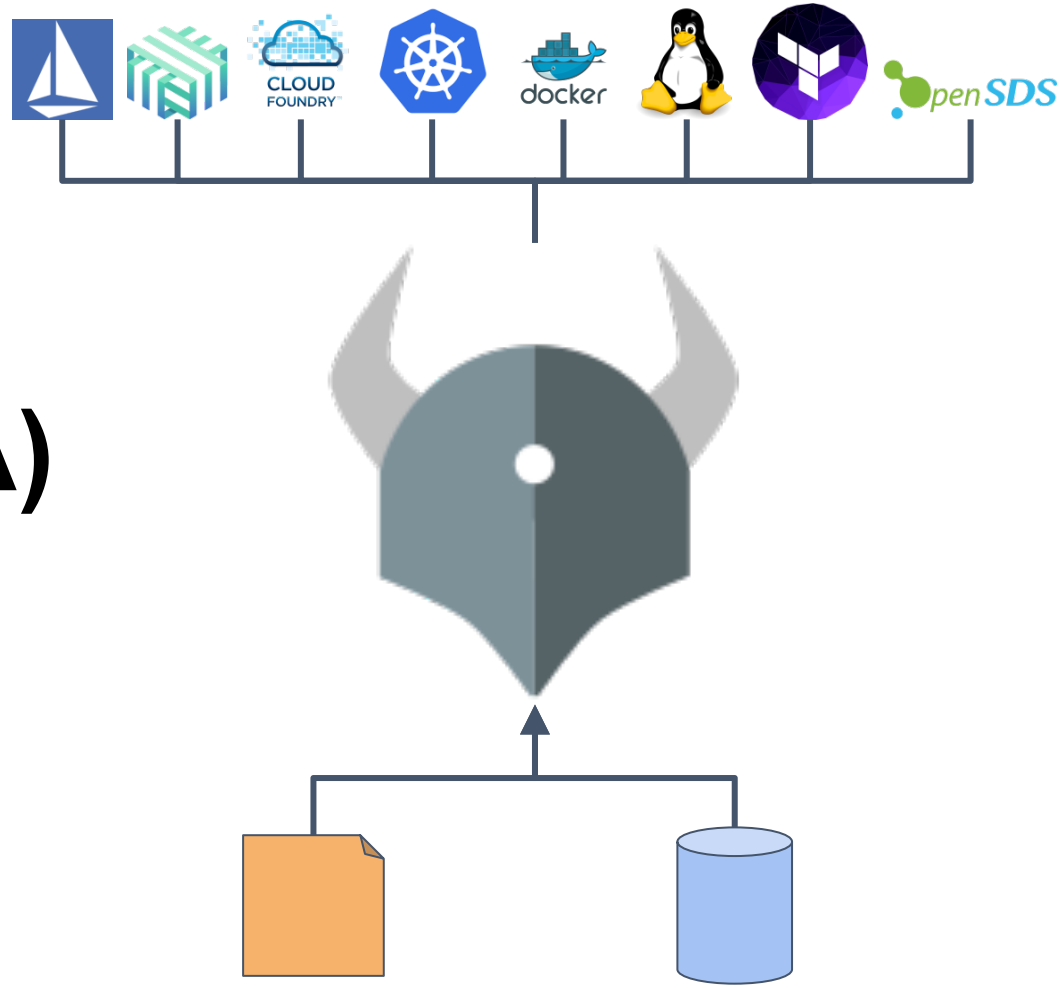


Authorization Policy

1. Employees can read their own salary and the salary of anyone who reports to them.
2. `Report Generator` Job should be able to Read all users' salaries
3. `Performance Review` Application should be able to update all users' salaries

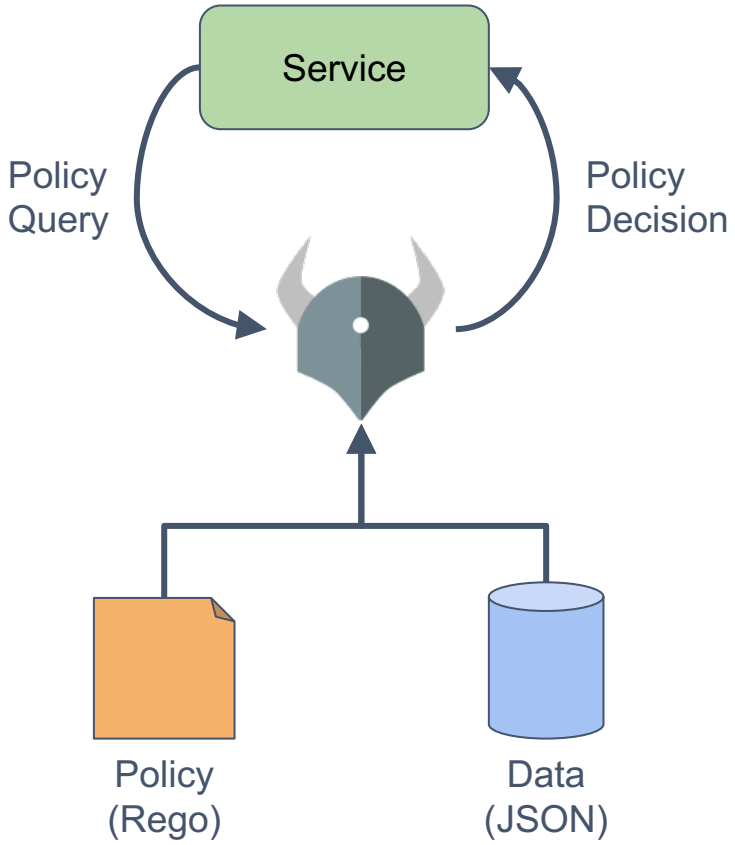
Open Policy Agent (OPA)

Policy-based control for the cloud native environments



Open Policy Agent (OPA)

- **Solve policy enforcement for any I, O, and R**
  - **I = Identity**
  - **O = Operation**
  - **R = Resource**
- Open source, general-purpose policy engine
  - Written in Go, library/host-local agent, in-memory
- Declarative Policy Language (Rego)
  - Can user X perform operation Y on resource Z?



Example Policy (in English)

“Employees can read their own salary and the salary of anyone who reports to them.”

OPA: Rules

*"**Employees can read their own salary** and the salary of anyone who reports to them."*

input

```
{
  method: "GET",
  path: ["getSalary", "bob"]
  user: "bob"
}
```

```
allow = true {
  input.method = "GET"
  input.path = ["getSalary", id]
  input.user = id
}
```

OPA: Context-aware

*"**Employees can read** their own salary and **the salary of anyone who reports to them.**"*

data

```
{
  managers: {
    bob: ["alice", "ken"]
    alice: ["ken"]
  }
}
```

```
allow = true {
  input.method = "GET"
  input.path = ["getSalary", id]
  managers = data.managers[id]
  input.user = managers[_]
}
```

OPA: Composable

*"**Employees can read** their own salary and **the salary of anyone who reports to them.**"*

api_authz.rego

```
allow = true {
  input.method = "GET"
  input.path = ["getSalary", id]
  is_manager_of(input.user, id)
}
```

org_chart.rego

```
is_manager_of(a, b) = true {
  managers = data.managers[b]
  a = managers[_]
}
```

OPA: Resource-agnostic

input (http)

```
{
  method: POST
  path: /salary/bob
  body: 1,000,000
}
```

input (kafka)

```
{
  action: publish
  service: reviews
  topic: audit_log
}
```

input (ssh)

```
{
  login: root
  host: i-012983928
  identity: bob@acme.com
}
```

http_authz.rego

```
allow = true {
  input.method = “POST”
  input.path = “...”
}
```

kafka_authz.rego

```
allow = true {
  input.action = “publish”
  input.topic = “...”
}
```

ssh_authz.rego

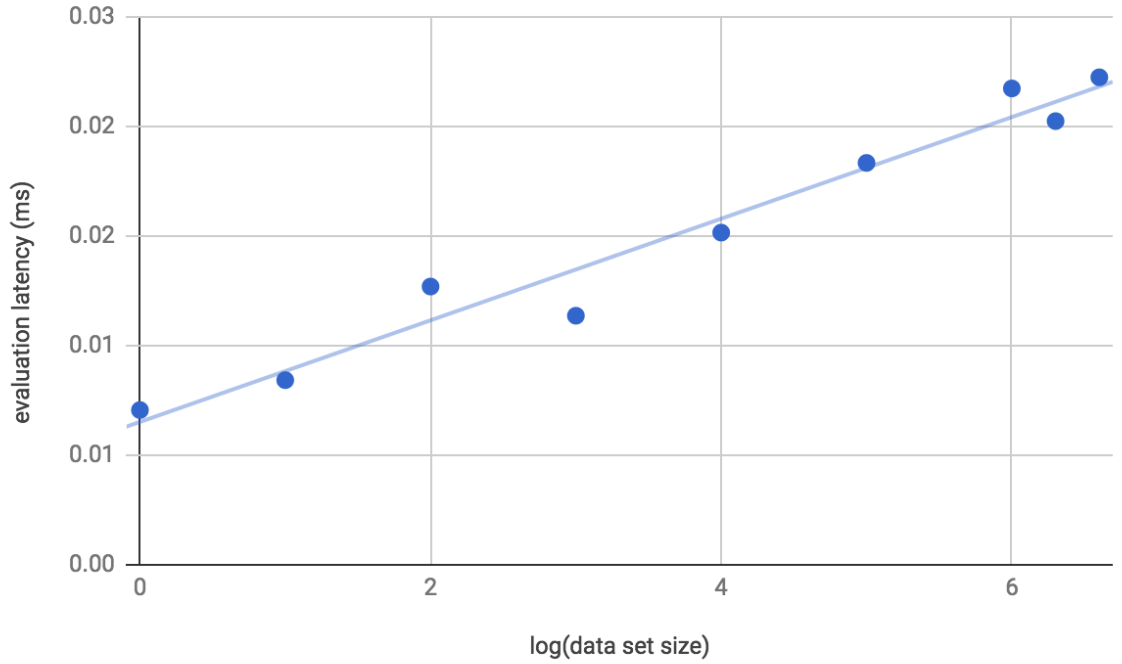
```
allow = true {
  input.login = “root”
  input.host = “...”
}
```

OPA: Performance

- Example: RBAC policy

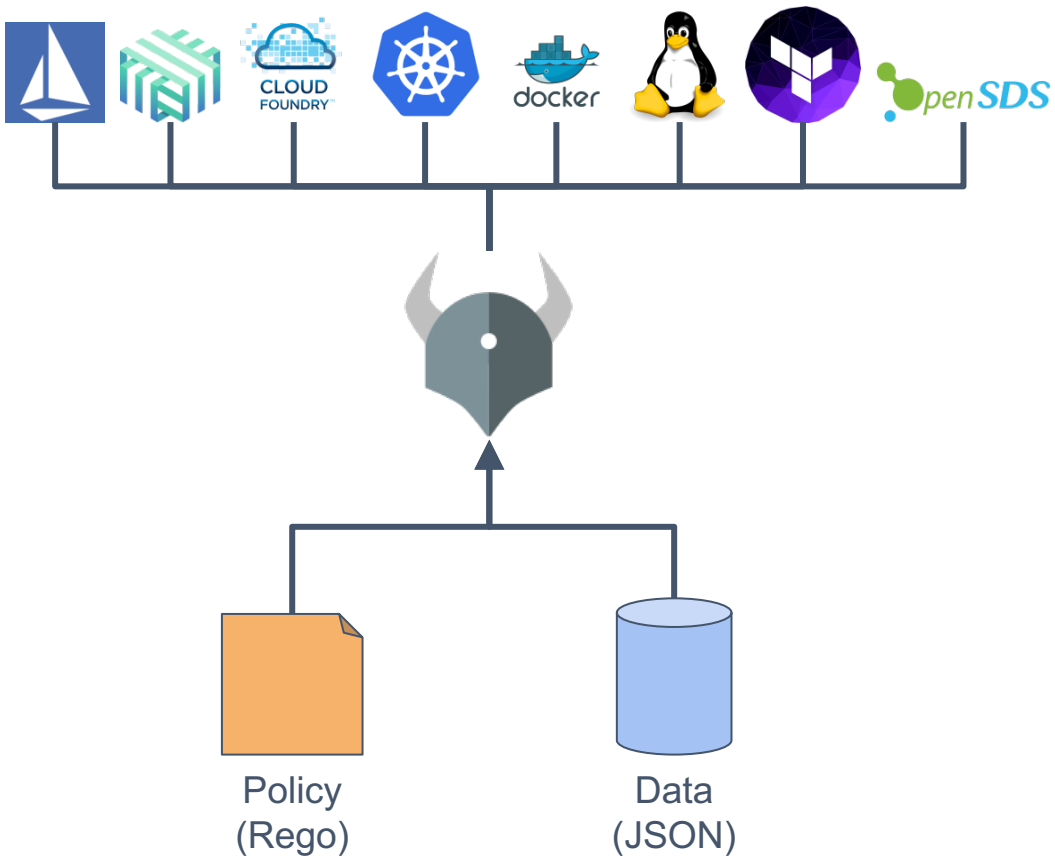
Roles	Bindings	Latency (us) [worst]
20	20	~10
2,000	2,000	~20

Intel Core i7 @ 2.9 GHz



Open Policy Agent

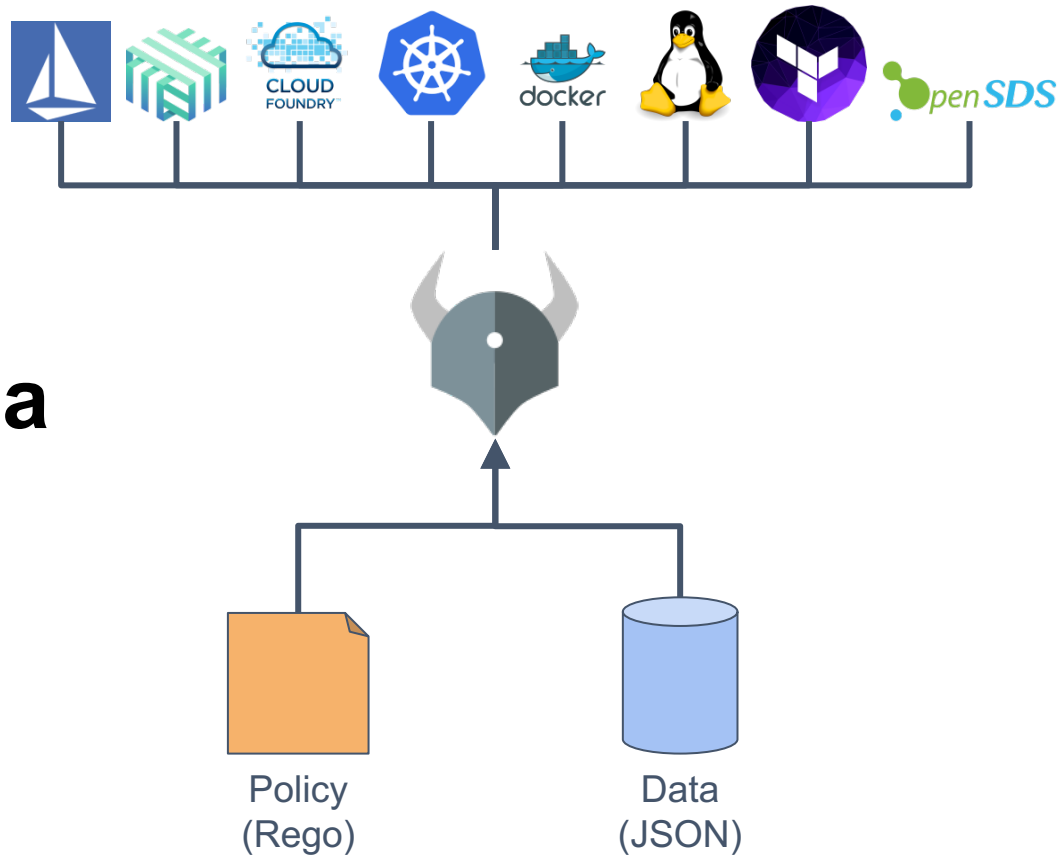
- OPA provides a reusable building block for enforcing policy **across the stack**
- OPA helps solve fundamental security problems like authorization



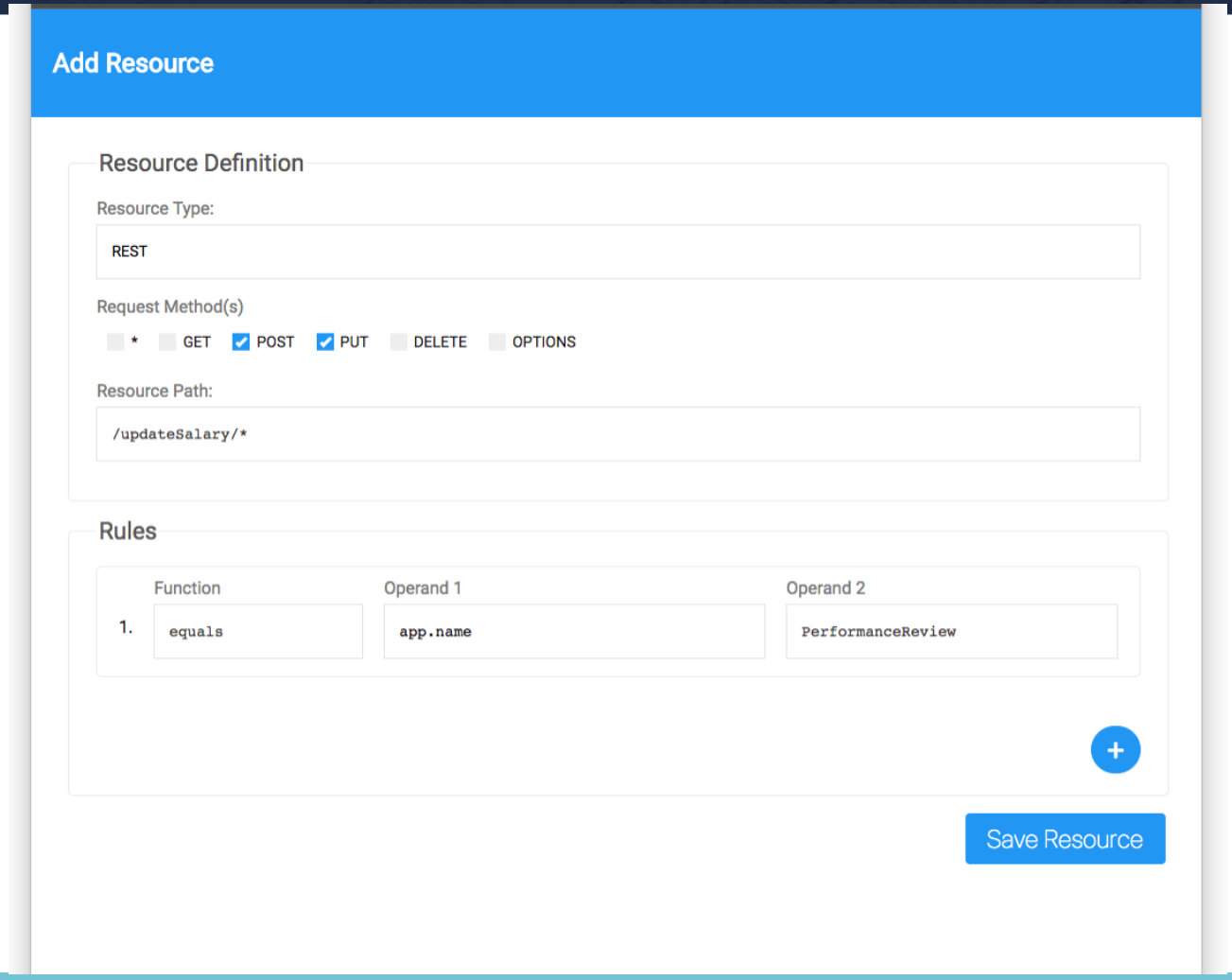
Open Policy Agent

github.com/open-policy-agent/opa

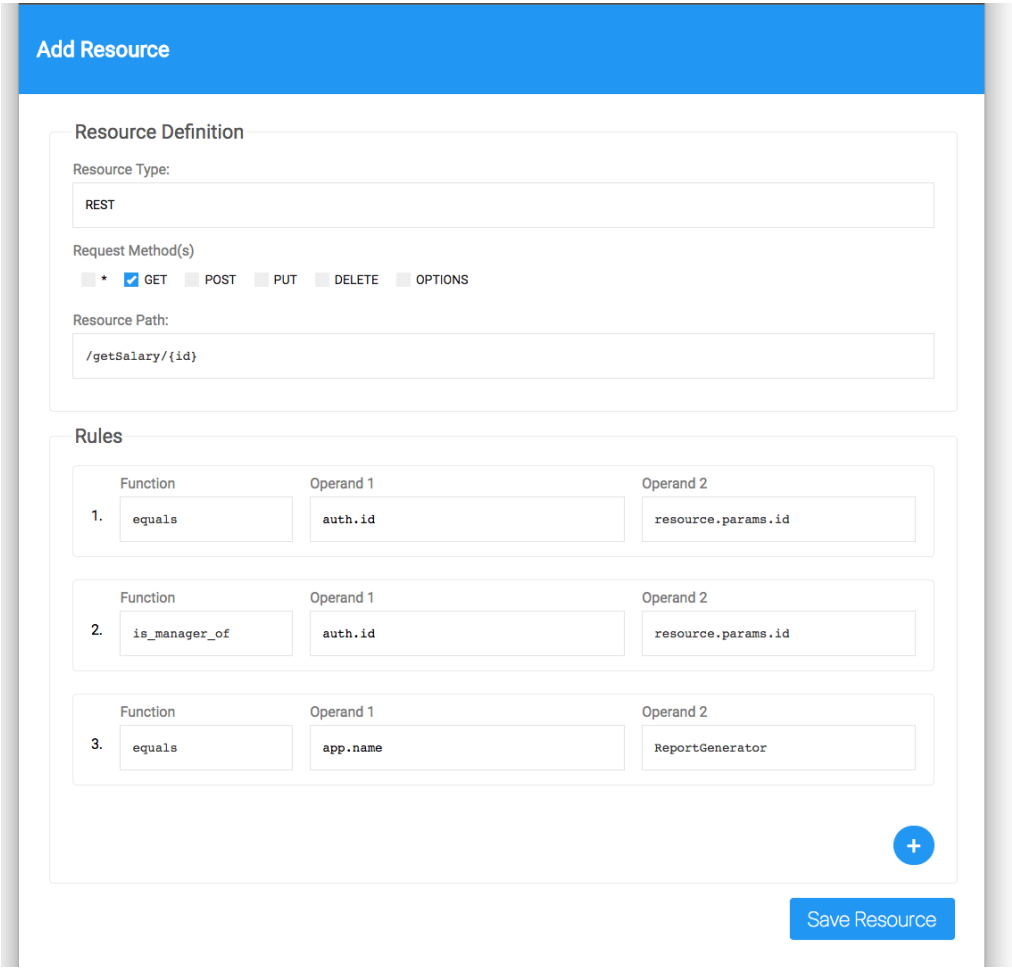
Check out our demo booth!



Capturing Intent



Capturing Intent



Summary

Resource types	REST, gRPC method, SSH Login, Keys, Kafka Topics
Identity types	VM/Container Services, Batch Jobs, FTEs, Contractors
Underlying Protocols	HTTP, gRPC, SSH, Kafka Protocol
Implementation Languages	Java, Node JS, Ruby, Python
Latency	< 0.2 ms for basic policies
Flexibility of Rules	OPA Policy Engine
Company Culture	Policy Portal - Exercising Freedom, Responsibly
Capture Intent	Policy Portal UI hides Policy Syntax

Take Away

- AuthZ is a fundamental security problem
- Comprehensive solution gives better Control and Visibility
- Get there faster with Open Source Tools (like OPA)
- Get involved in communities (like PADME)

Questions?

(We are hiring!)

Manish Mehta
mmehta@netflix.com

Torin Sandall
 **@sometorin**