Distributed Authorization System: A Netflix case study

Manish Mehta - Chief Security Architect @ Volterra

Torin Sandall - Co-founder of Open Policy Agent project
- Software Engineer @ Styra

Velocity 2018
June 12-14

Manish Mehta
~~Senior Security Engineer @ Netflix~~
Chief Security Architect @ Volterra
manish@ves.io

Projects:

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

Torin Sandall
Co-founder of the OPA project
Software Engineer @ Styra

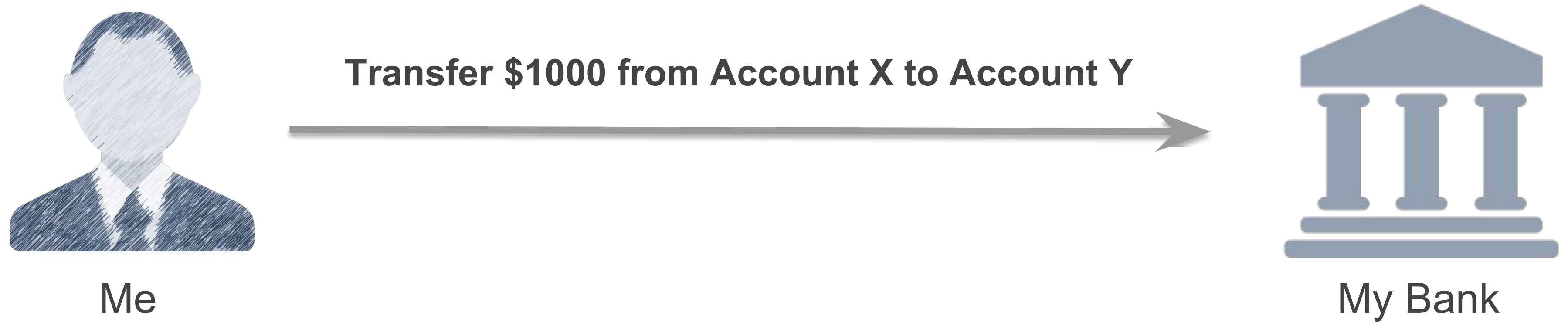
@sometorin

@OpenPolicyAgent

Projects:

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

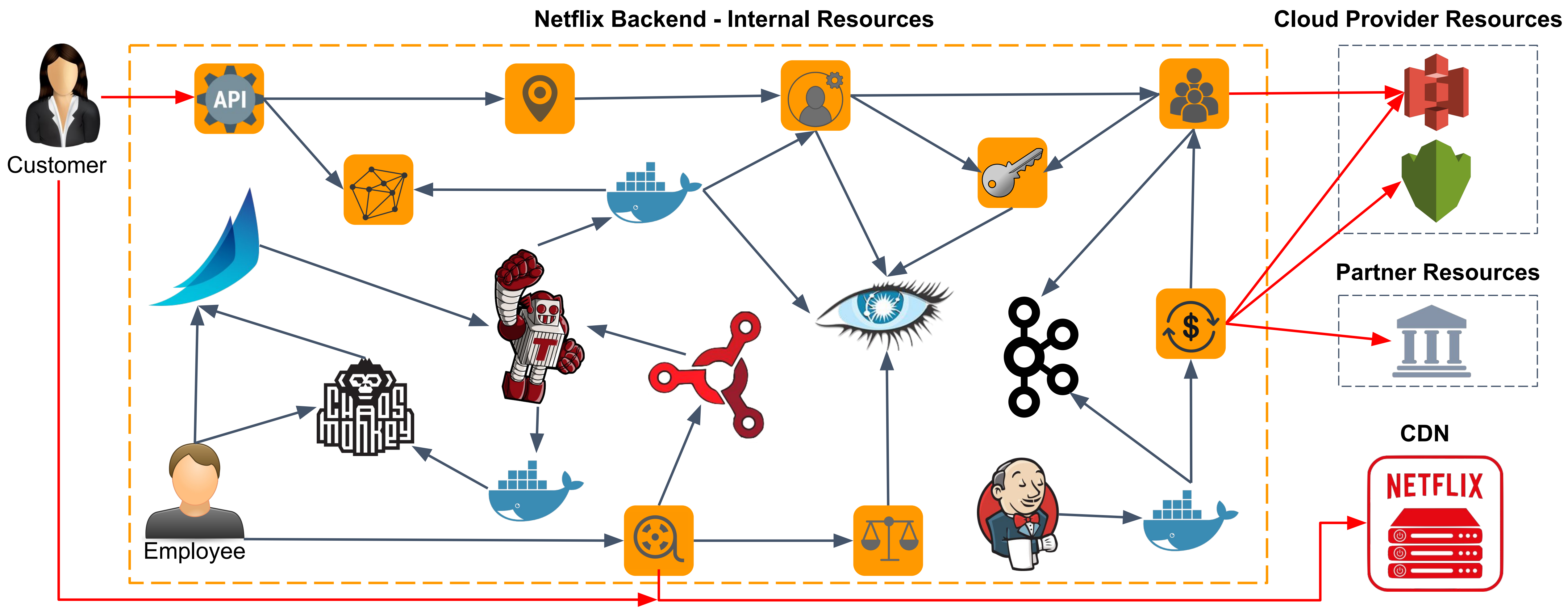
Background - Definitions



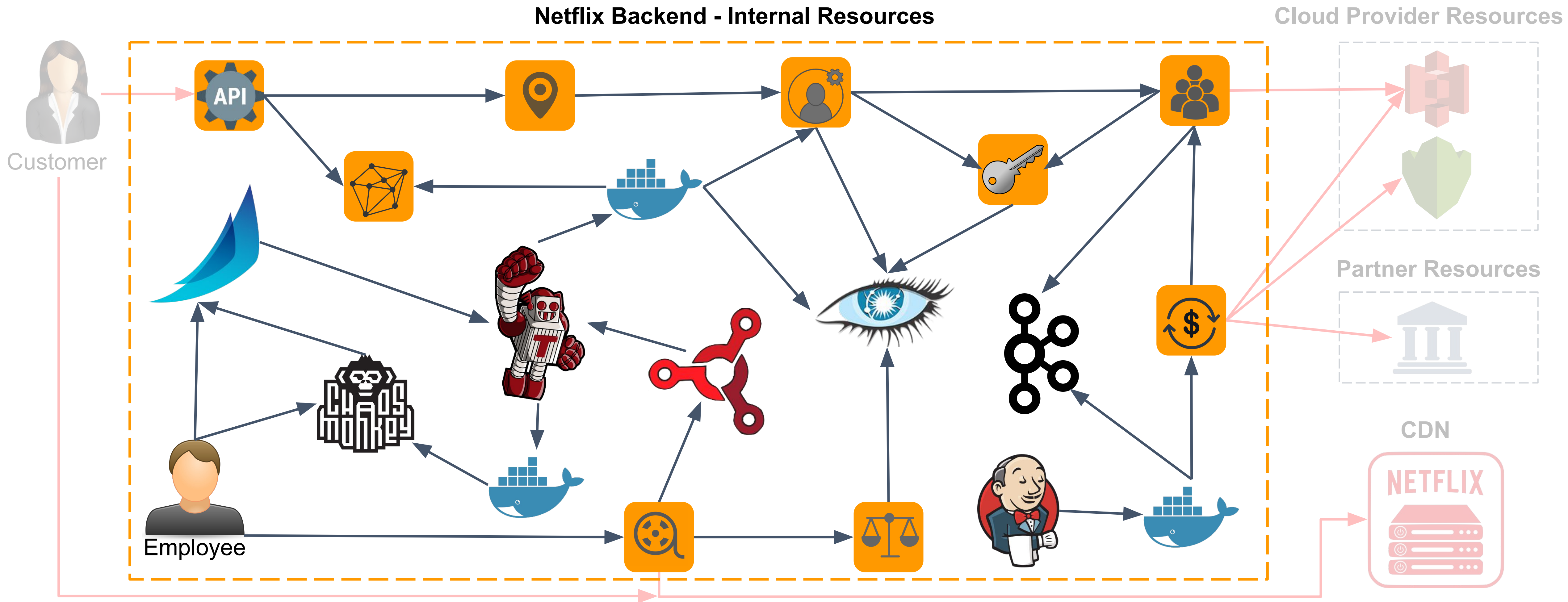
1. Verify the Identity of the Requester (Authentication or AuthN)
2. Verify that the Requestor is authorized to perform the requested operation (Authorization or AuthZ)

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

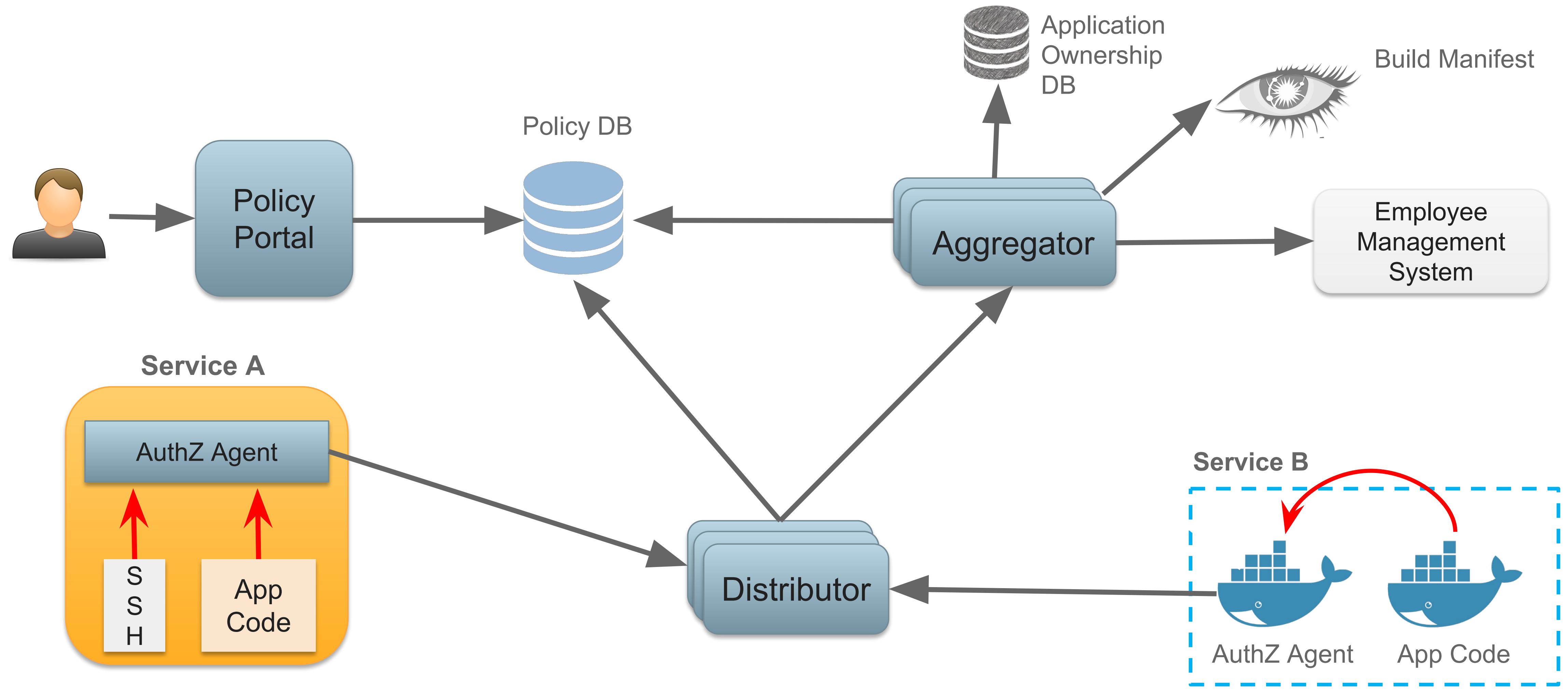
Flexibility of Rules

- Hard-coded structure vs. language-based

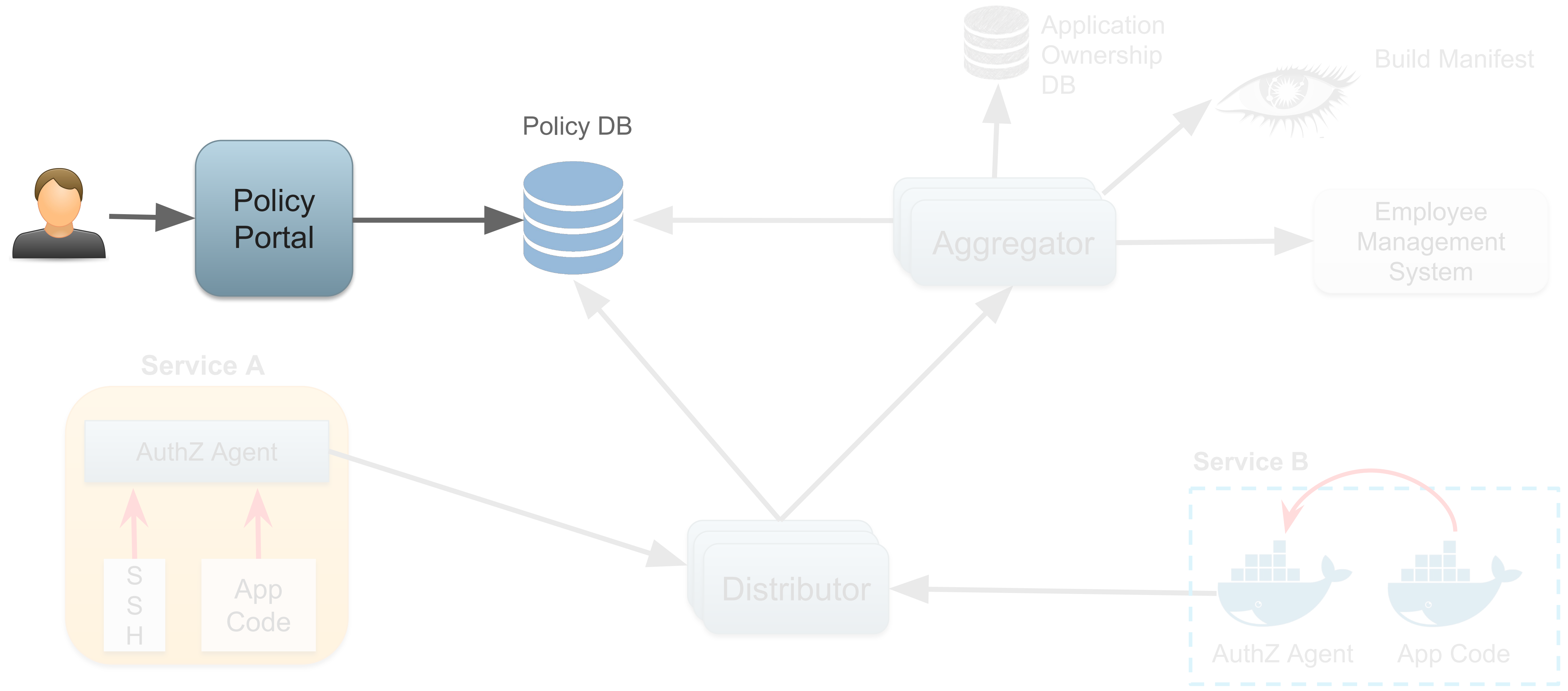
Capture Intent

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

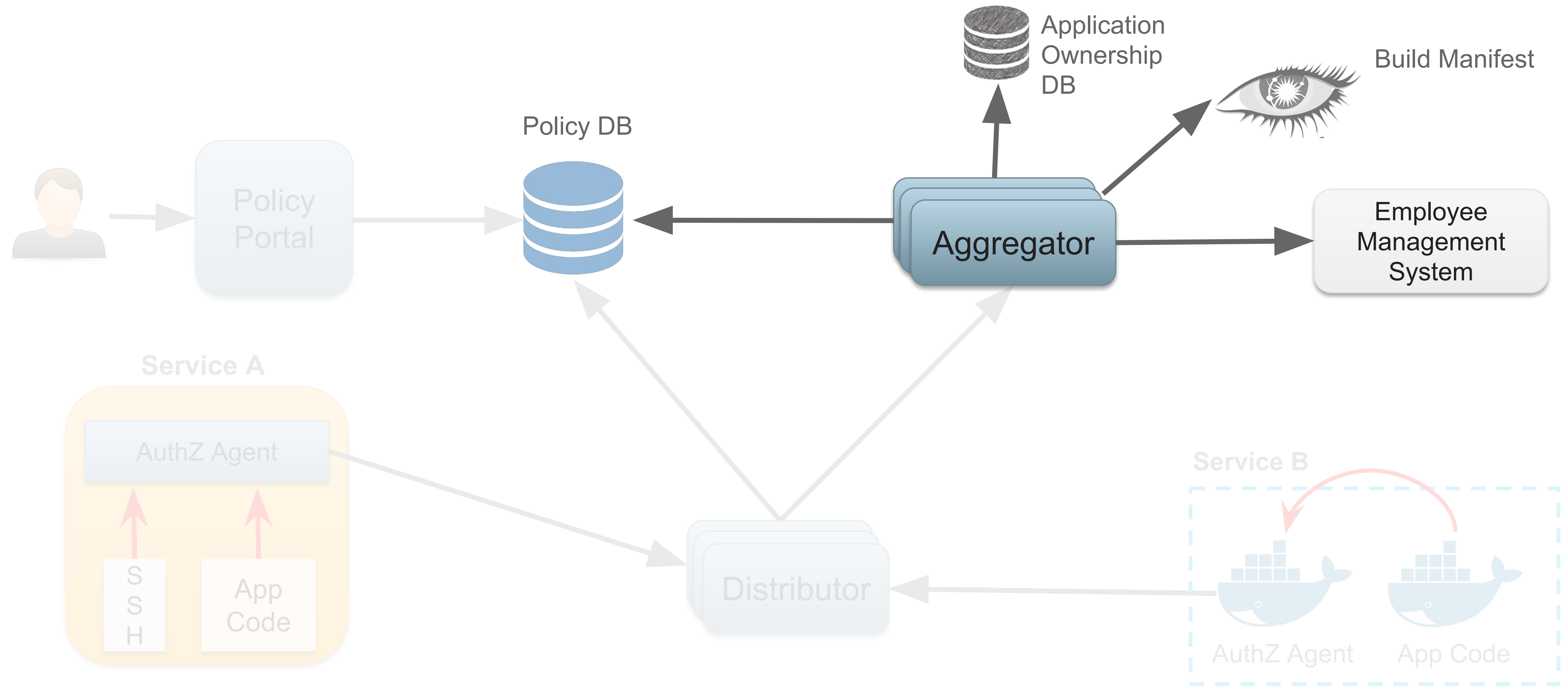
High-level Architecture



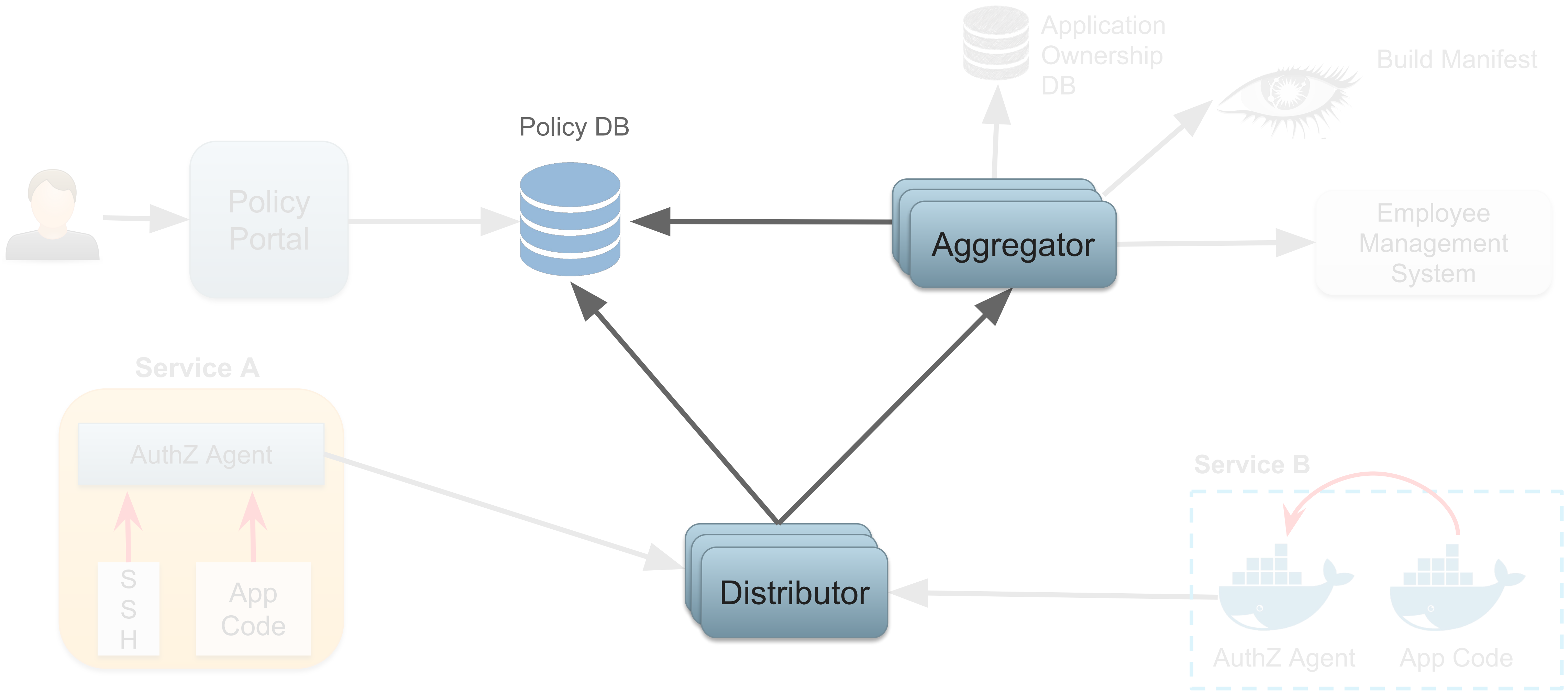
High-level Architecture



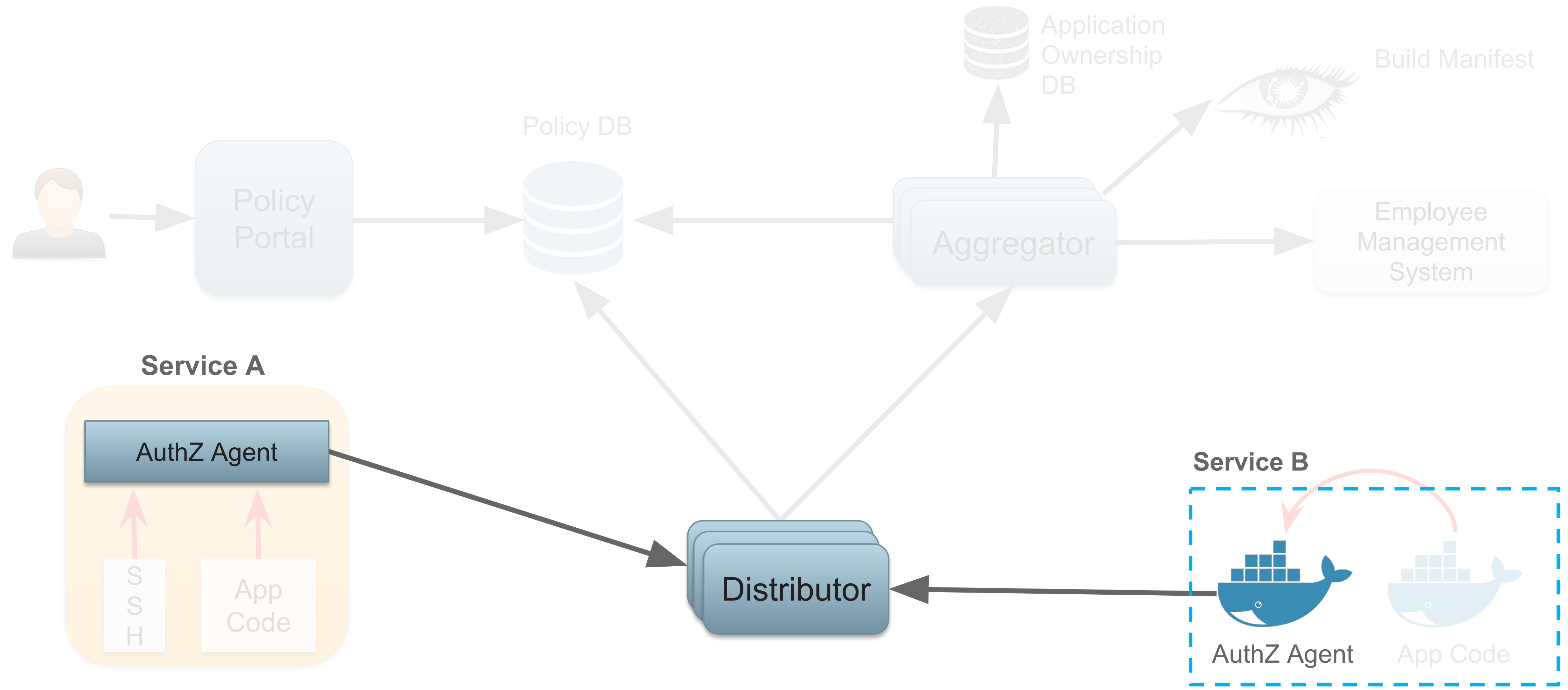
High-level Architecture



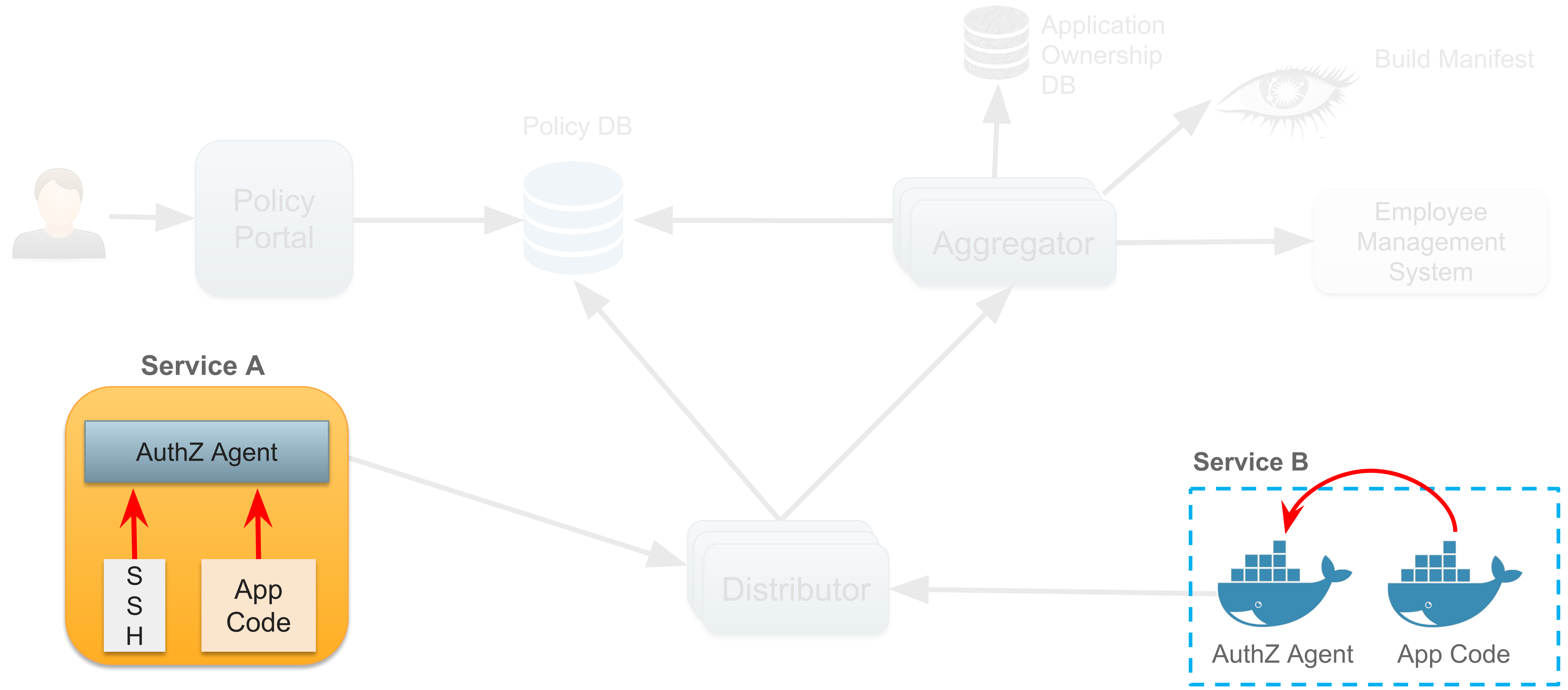
High-level Architecture



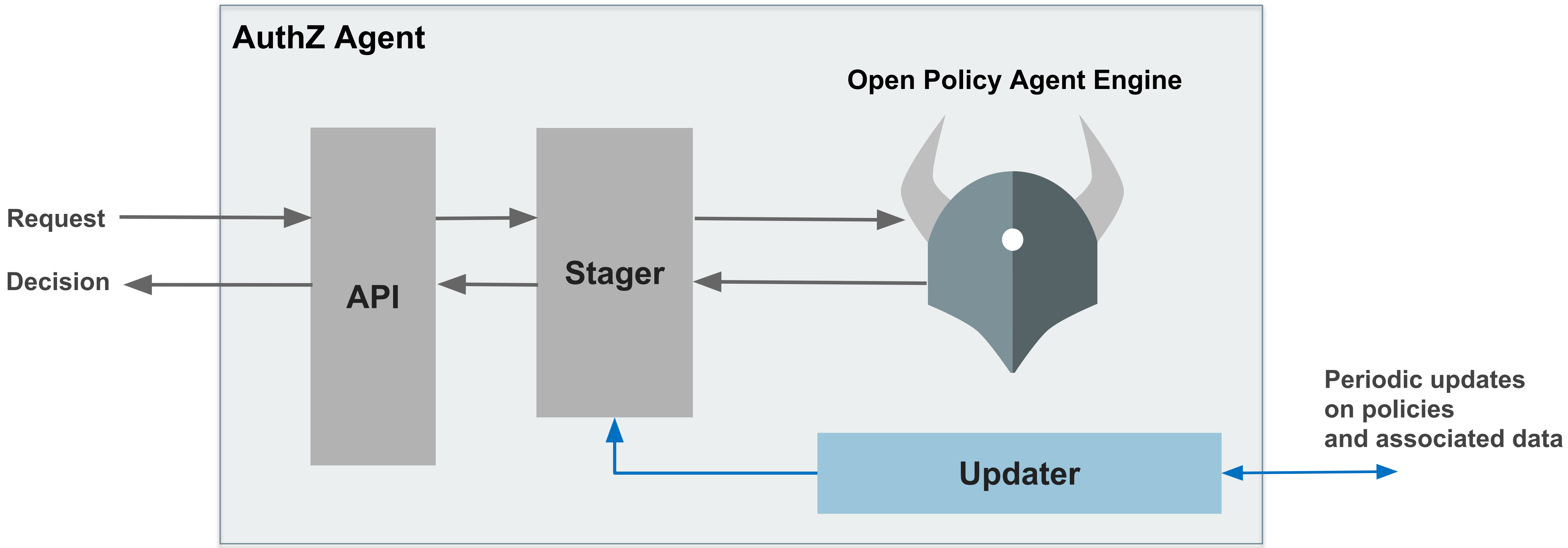
High-level Architecture



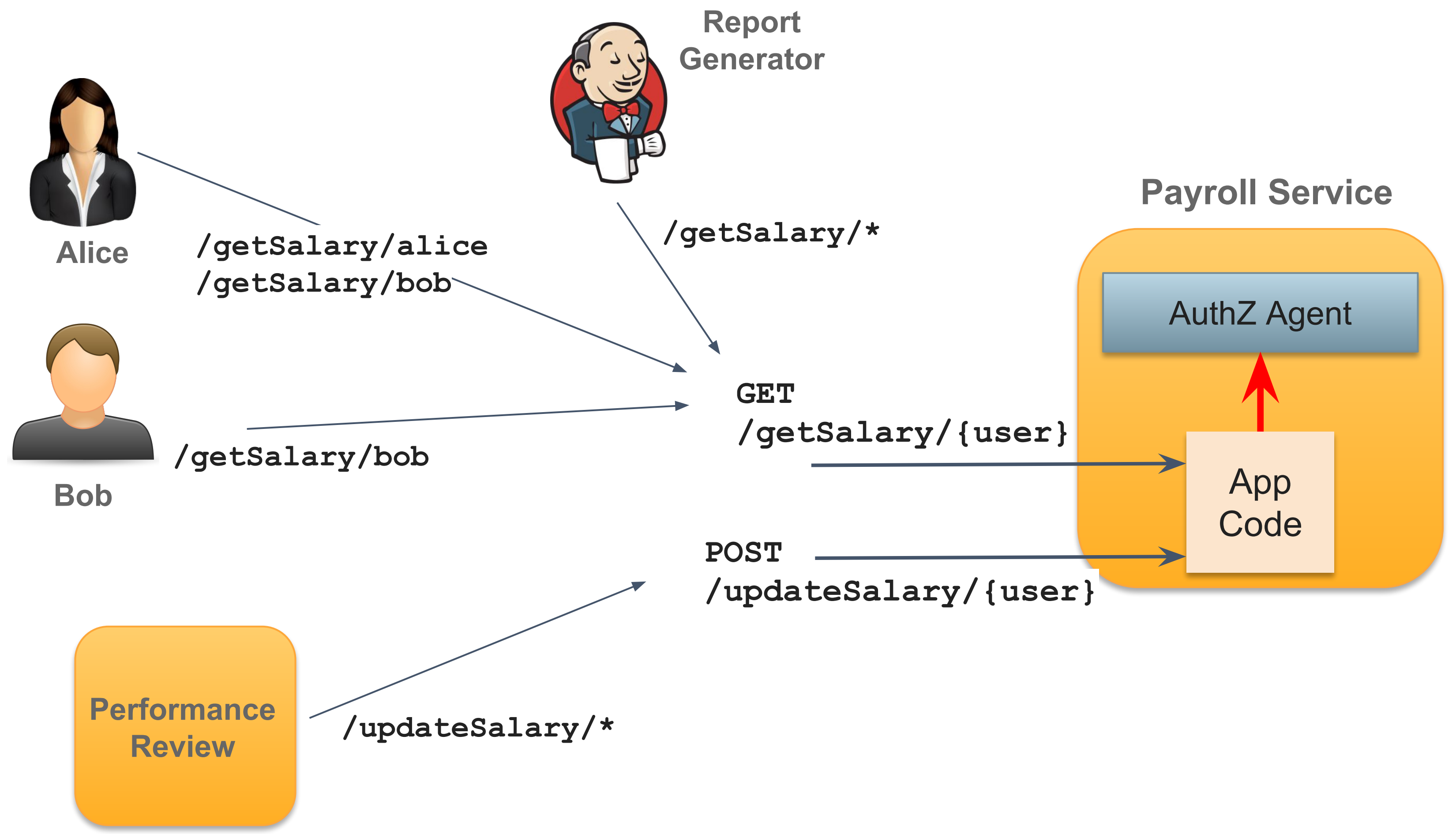
High-level Architecture



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

kafka
spring
amazon web services

CLOUD NATIVE SANDBOX

@sometorin

@OpenPolicyAgent

What about RBAC?

RBAC solves XX% of the problem.

"Allow all HTTP requests from 10.1.2.0/24."

"Restrict employees from accessing the service outside of work hours."

"QA must sign-off on images deployed to the production namespace."

"Restrict ELB changes to senior SREs that are on-call."

"Analysts can read client data but PII must be redacted."

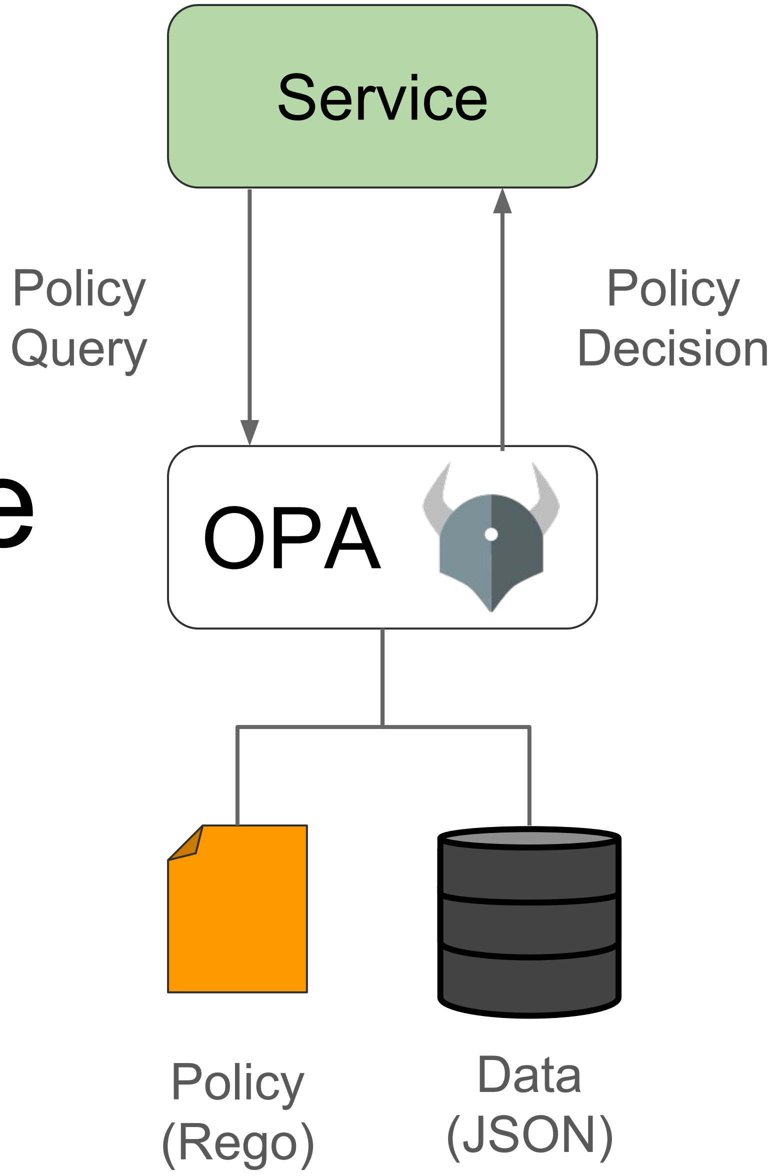
RBAC is not enough.

"Give developers SSH access to machines listed in JIRA tickets assigned to them."

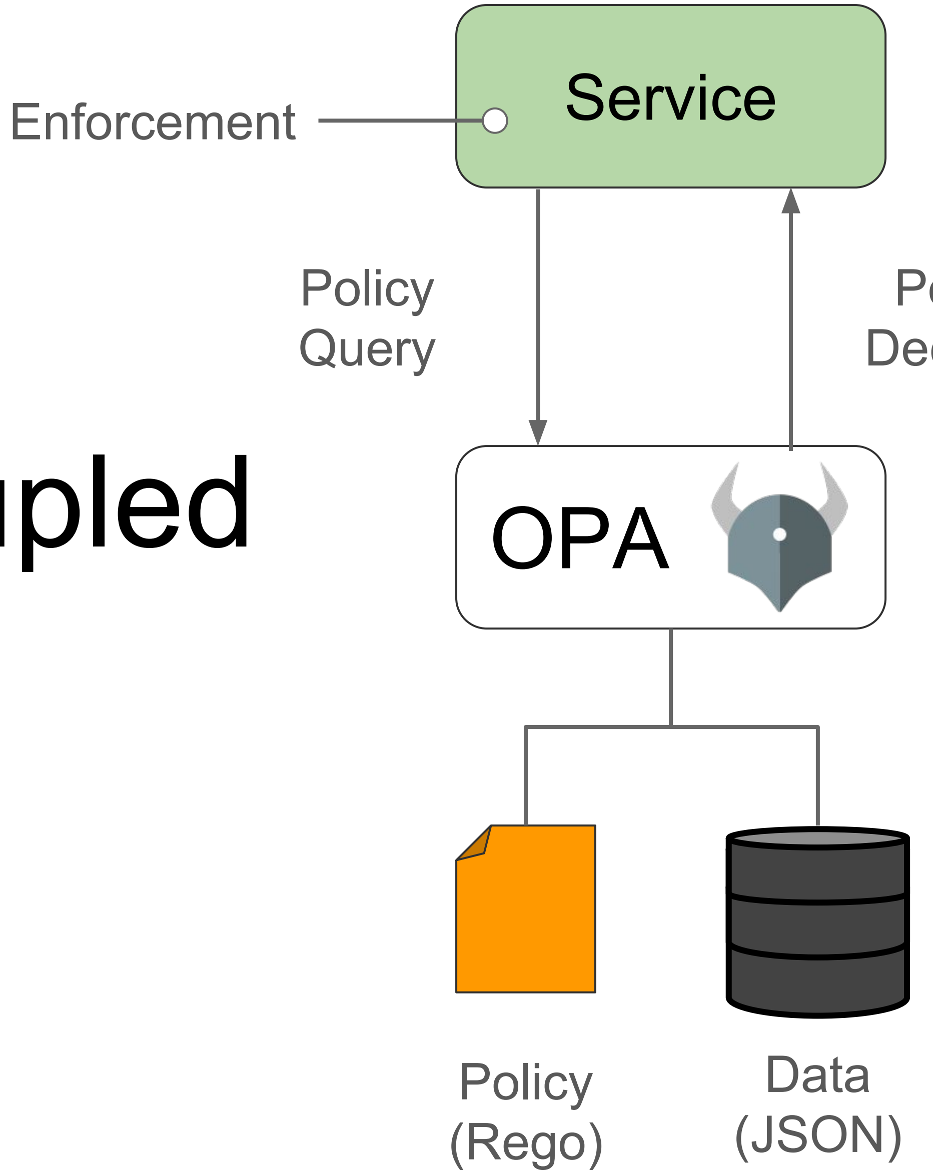
"Prevent developers from running containers with privileged security contexts in the production namespace."

"Workloads for euro-bank must be deployed on PCI-certified clusters in the EU."

OPA is a general-purpose policy engine.

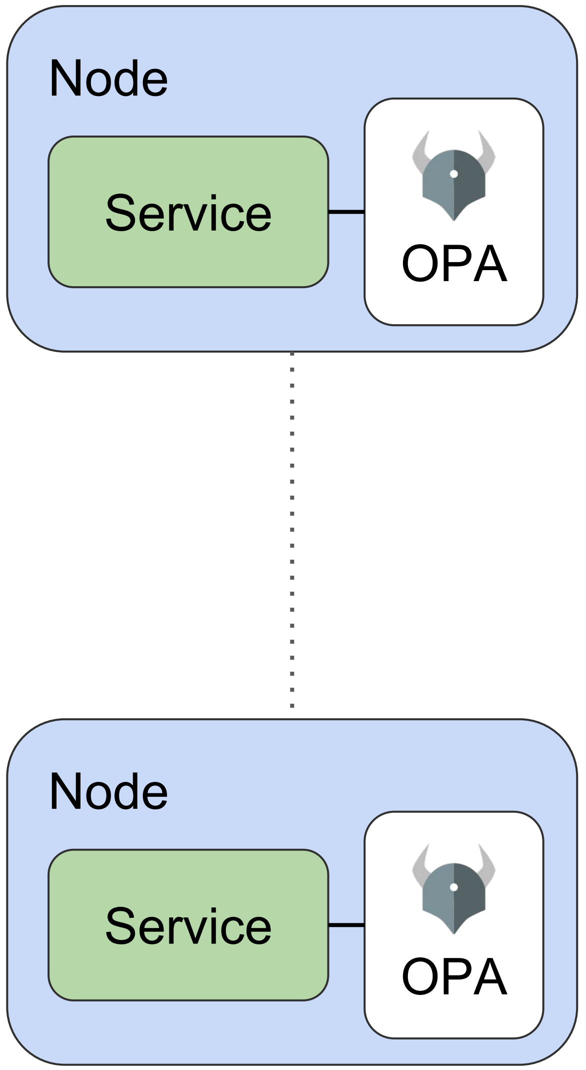


Decisions are decoupled from enforcement.



Evaluate policies locally.

- Daemon (HTTP API)
- Library (Go)
- Service Mesh (Istio)



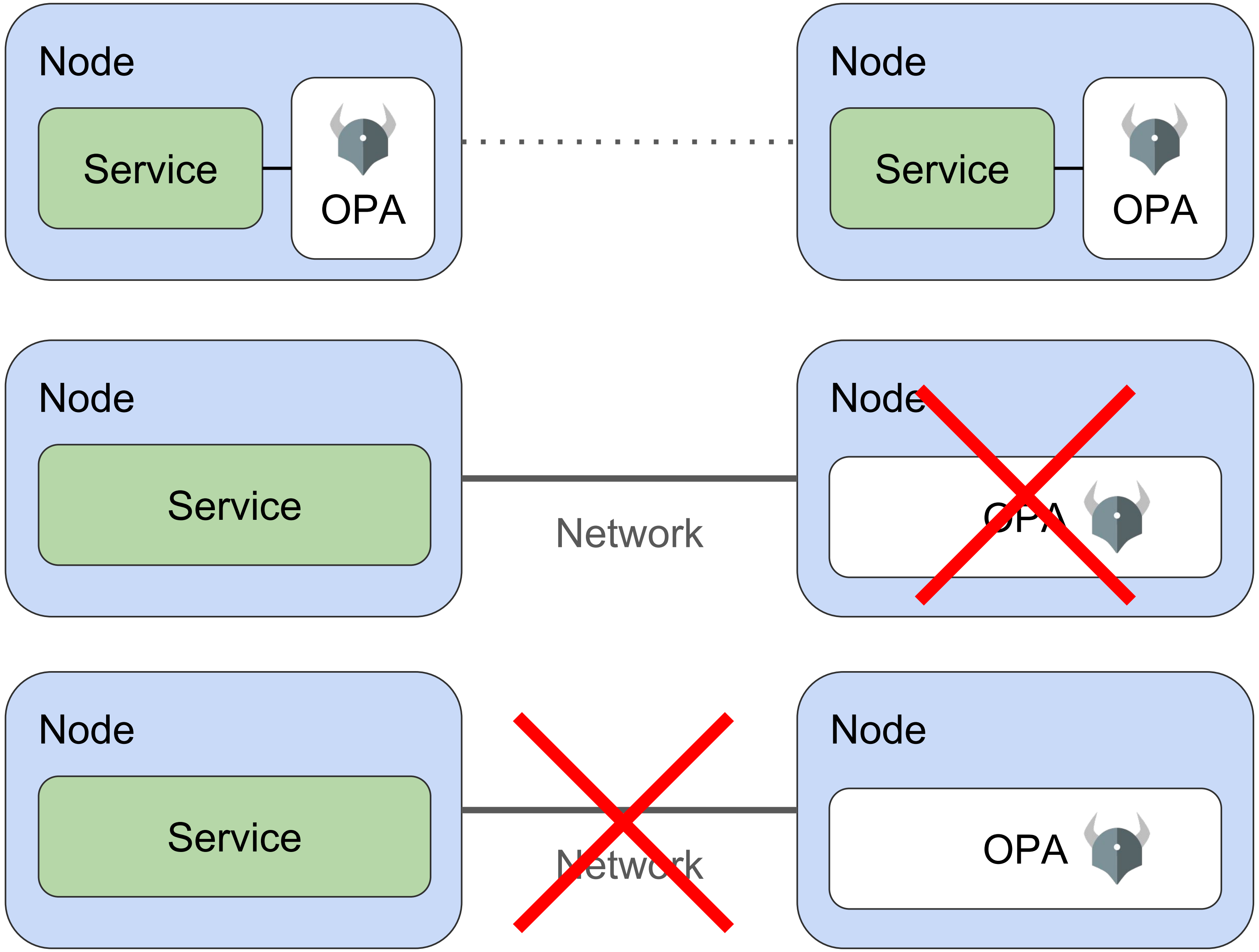
Fate Sharing

✔ Low latency

✔ High availability

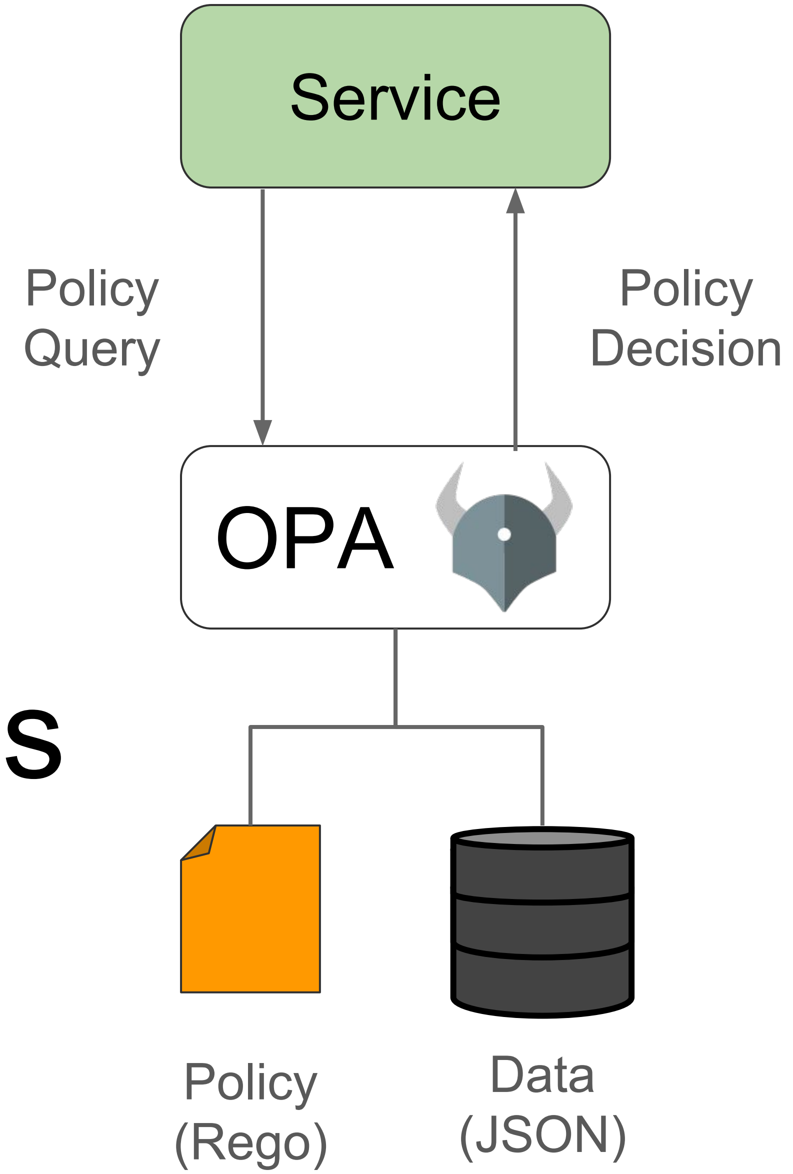
Host Failures

Network Partitions



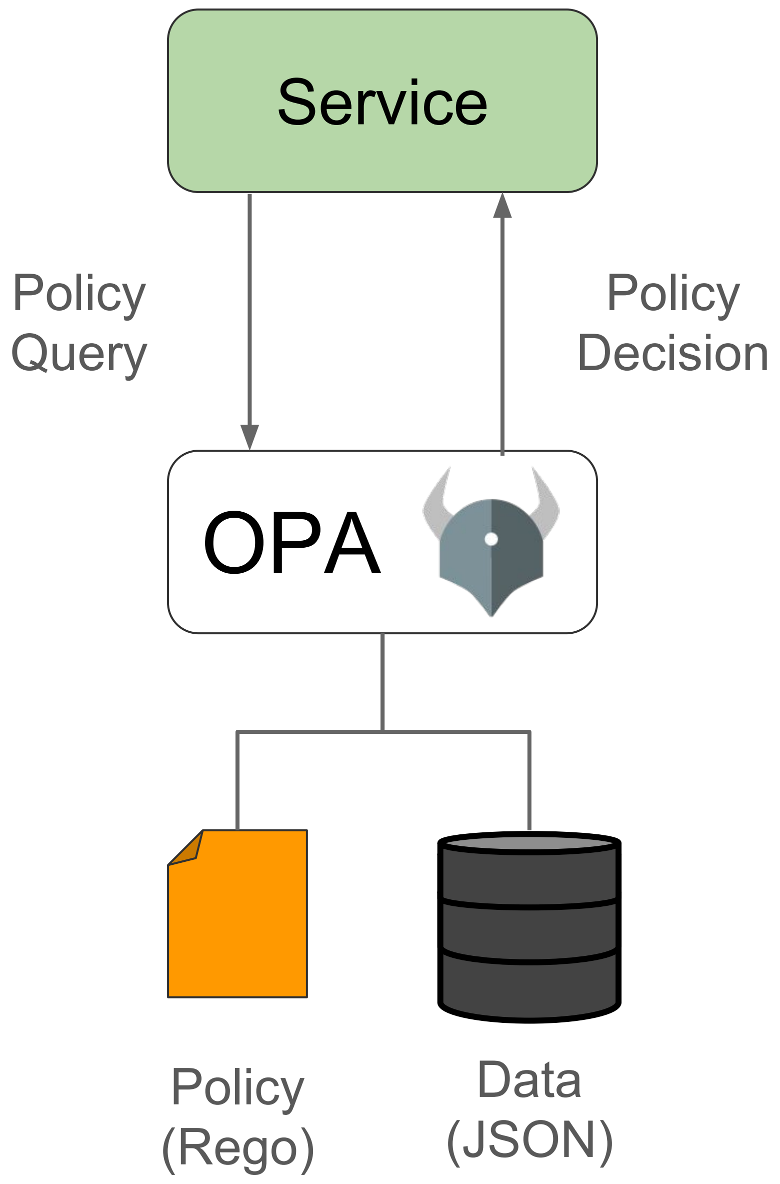
Policy and data are stored in-memory.

No external dependencies during enforcement.



Declarative Language (Rego)

- Is Identity I allowed to perform Operation O on Resource R?
- What labels must applied to Deployment X?
- Which users can SSH into production servers?



"Employees can read their own salaries and the salaries of their subordinates."

"Employees can read their own salaries [...]"

"Employees can read their own salaries [...]"

Input

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

"Employees can read their own salaries [...]"

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

Input

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

"Employees can read their own salaries [...]"

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

Input

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

"Employees can read their own salaries [...]"

```
allow = true {
  input.method = "GET"                  # OK
  input.path = ["salaries", "bob"] # OK
  input.user = "bob"                    # OK
}
```

Input

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

@sometorin

@OpenPolicyAgent

"Employees can read their own salaries [...]"

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

<u>Input</u>

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

↑

"alice" instead of "bob"

"Employees can read their own salaries [...]"

```
allow = true {
  input.method = "GET"                   # OK
  input.path = ["salaries", "bob"] # OK
  "alice" = "bob"                        # FAIL
}
```

Input

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

@sometorin

@OpenPolicyAgent

"Employees can read [...] the salaries of their subordinates."

```
allow = true {
  input.method = "GET"                     # OK
  input.path = ["salaries", "bob"] # OK
  "alice" = "bob"                          # FAIL
}
```

Input

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

@sometorin

@OpenPolicyAgent

"Employees can read [...] the salaries of their subordinates."

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

Input

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

Data (in-memory)

```
{"manager_of": {
   "bob": "alice",
   "alice": "janet"}}
```

"Employees can read [...] the salaries of their subordinates."

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

allow = true {
  input.method = "GET"
  input.path = ["salaries", employee_id]
  input.user = data.manager_of[employee_id]
}
```

Input

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

Data (in-memory)

```
{"manager_of": {
  "bob": "alice",
  "alice": "janet"}}
```

"Employees can read [...] the salaries of their subordinates."

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

allow = true {
  input.method = "GET"
  input.path = ["salaries", "bob"]
  input.user = data.manager_of["bob"]
}
```

<u>Input</u>

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

<u>Data (in-memory)</u>

```
{"manager_of": {
   "bob": "alice",
   "alice": "janet"}}
```

"Employees can read [...] the salaries of their subordinates."

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

allow = true {
  input.method = "GET"
  input.path = ["salaries", "bob"]
  input.user = "alice"
}
```

<u>Input</u>

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

<u>Data (in-memory)</u>

```
{"manager_of": {
   "bob": "alice",
   "alice": "janet"}}
```

"Employees can read [...] the salaries of their subordinates."

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

allow = true {
  input.method = "GET"                # OK
  input.path = ["salaries", "bob"] # OK
  input.user = "alice"                # OK
}
```

Input

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

Data (in-memory)

```
{"manager_of": {
   "bob": "alice",
   "alice": "janet"}}
```

OPA enables flexible

- RBAC
- ABAC
- Admission Control
- Data Protection
- Risk Management
- ...

OPA supports any

- Resource Type
- Identity Type
- Implementation Language
- Underlying Protocol

```
allow {
  input.method = "GET"
  input.path = ["salary", user]
  input.user = user
}
```

```
method: GET
path: /salary/bob
service.source:
  namespace: production
  service: landing_page
service.target:
  namespace: production
  service: details
user: alice
```

```
deny {
  not metadata.labels["qa-signoff"]
  metadata.namespace == "prod"
  spec.containers[_].privileged
}
```

```
metadata:
  name: nginx-149353-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    name: nginx
    securityContext:
      privileged: true
  nodeName: minikube
```

```
deny {
  is_read_operation
  is_pii_topic
  not in_pii_consumer_whitelist
}
```

```
operation: Read
resource:
  name: credit-scores
  resourceType: Topic
session:
  principal:
    principalType: User
    name:
CN=anon_producer,O=OPA
  clientAddress: 172.21.0.5
```

kafka

```
allow {
  risk_score <= risk_budget
  count(plan_names["aws_iam"]) == 0
  blast_radius < 500
}
```

```
aws_autoscaling_group.lamb:
  availability_zones#: '1'
  availability_zones.3205: us-west-1a
  desired_capacity: '4'
  launch_configuration: kitten
  wait_for_capacity_timeout: 10m
aws_instance.puppy:
  ami: ami-09b4b74c
  instance_type: t2.micro
```

- Submillisecond Latency
- Composition
- External Context
- Partial Evaluation
- Rule Indexing
- Tracing
- Interactive Shell (REPL)
- IDE Integrations (VS Code)
- Test Framework
- Coverage
- Dependency Analysis

```
allow {
  input.method = "GET"
  input.path = ["salary", user]
  input.user = user
}
```

```
method: GET
path: /salary/bob
service.source:
  namespace: production
  service: landing_page
service.target:
  namespace: production
  service: details
user: alice
```

```
deny {
  not metadata.labels["qa-signoff"]
  metadata.namespace == "prod"
  spec.containers[_].privileged
}
```

```
metadata:
  name: nginx-149353-bvl8q
  namespace: production
spec:
  containers:
  - image: nginx
    name: nginx
    securityContext:
      privileged: true
  nodeName: minikube
```

```
deny {
  is_read_operation
  is_pii_topic
  not in_pii_consumer_whitelist
}
```

kafka

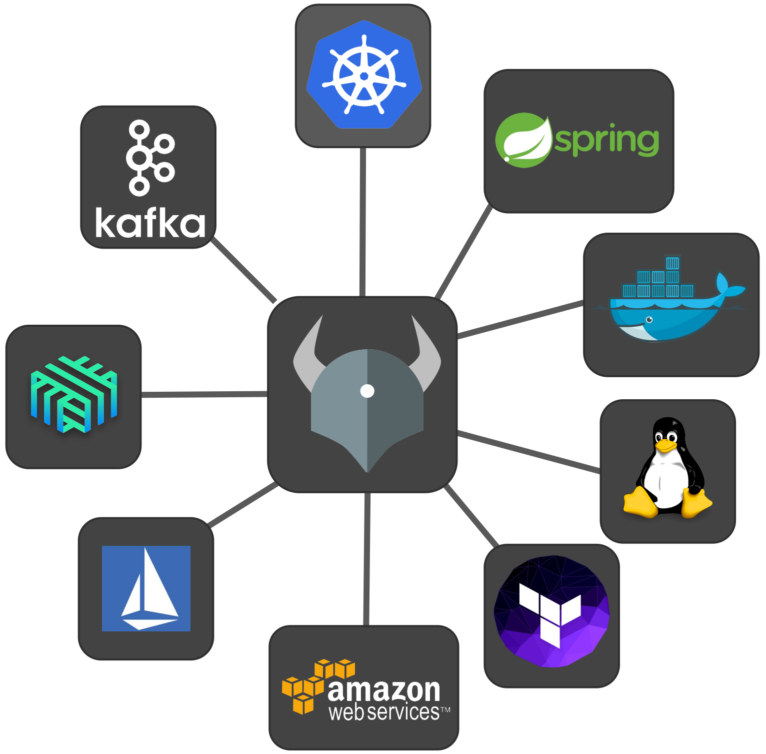
```
operation: Read
resource:
  name: credit-scores
  resourceType: Topic
session:
  principal:
    principalType: User
    name:
CN=anon_producer,O=OPA
  clientAddress: 172.21.0.5
```

```
allow {
  risk_score <= risk_budget
  count(plan_names["aws_iam"]) == 0
  blast_radius < 500
}
```

```
aws_autoscaling_group.lamb:
  availability_zones#: '1'
  availability_zones.3205: us-west-1a
  desired_capacity: '4'
  launch_configuration: kitten
  wait_for_capacity_timeout: 10m
aws_instance.puppy:
  ami: ami-09b4b74c
  instance_type: t2.micro
```

open-policy-agent/opa

★ Star 1,285

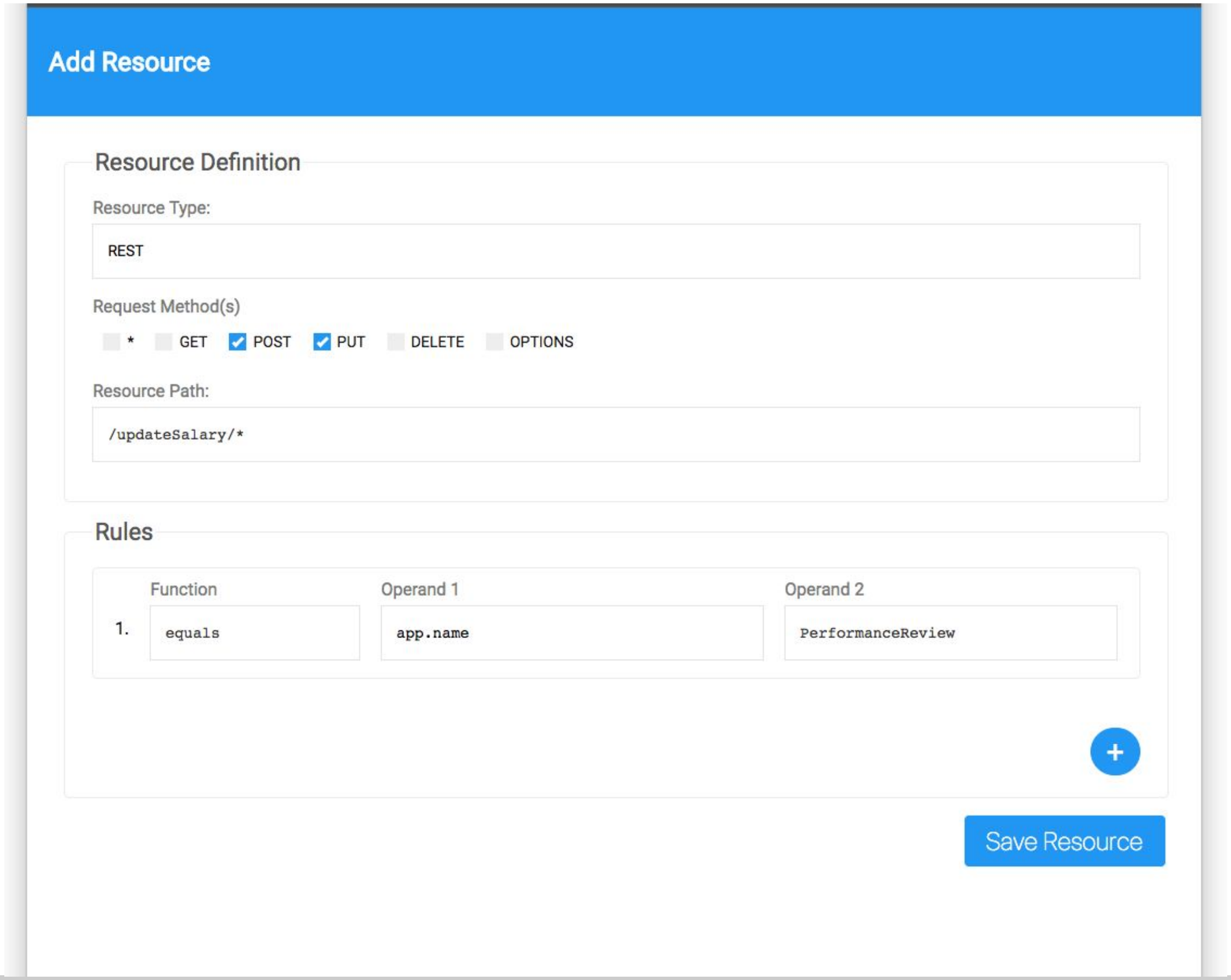


CLOUD NATIVE SANDBOX

@sometorin

@OpenPolicyAgent

Capturing Intent


Add Resource

Resource Definition

Resource Type: REST

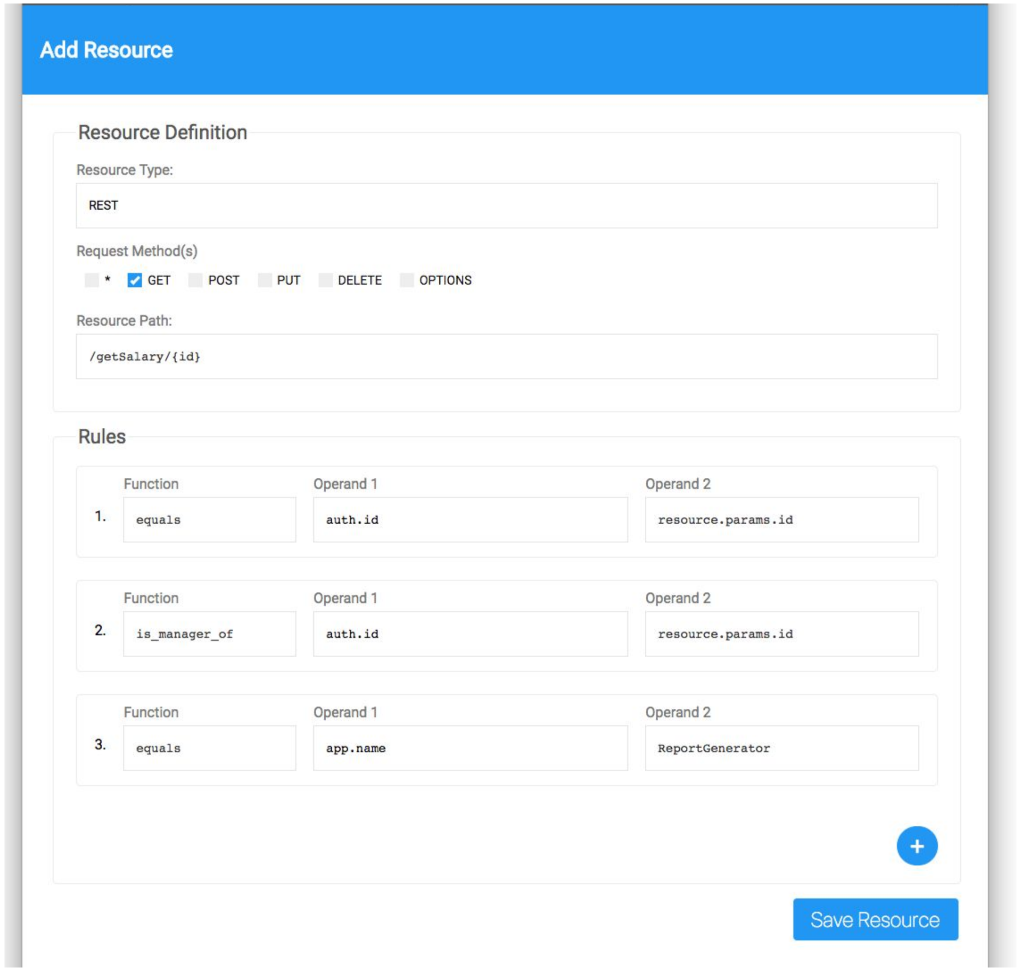
Request Method(s): [ ] * [ ] GET [✓ POST] [✓ PUT] [ ] DELETE [ ] OPTIONS

Resource Path: `/updateSalary/*`

Rules

	Function	Operand 1	Operand 2
1.	`equals`	`app.name`	`PerformanceReview`

Capturing Intent



Add Resource

Resource Definition

Resource Type:
REST

Request Method(s)
[ ] * [✓ GET] [ ] POST [ ] PUT [ ] DELETE [ ] OPTIONS

Resource Path:
/getSalary/{id}

Rules

	Function	Operand 1	Operand 2
1.	equals	auth.id	resource.params.id
2.	is_manager_of	auth.id	resource.params.id
3.	equals	app.name	ReportGenerator

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?

(Volterra is hiring!)

Manish Mehta

manish@ves.io

Torin Sandall

@sometorin