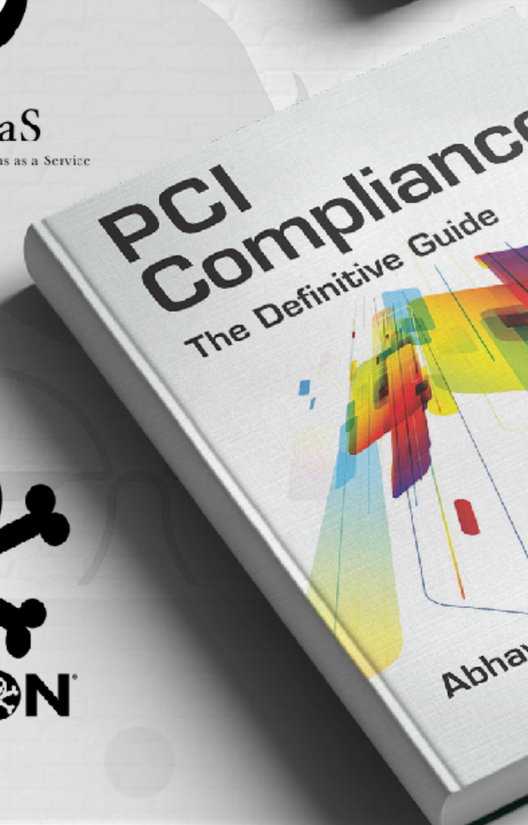
Policy-as-Code: Across the Stack

Abhay Bhargav

abhaybhargav

Yours Truly

- Founder @ we45
- Founder @ AppSecEngineer
- AppSec Automation Junkie
- Trainer/Speaker at DEF CON, BlackHat, OWASP Events, etc world-wide
- Co-author of Secure Java For Web Application Development
- Author of PCI Compliance: A Definitive Guide



abhaybhargav

My talk...

abhaybhargav

My talk...

abhaybhargav

Agenda

Agenda

- The need for decoupled security controls

Agenda

- The need for decoupled security controls
- The need for “Policy-as-Code”

Agenda

- The need for decoupled security controls
- The need for “Policy-as-Code”
- PaC across the StaCk

Agenda

- The need for decoupled security controls
- The need for “Policy-as-Code”
- PaC across the StaCk
- Application and API Gateway: Policy-as-Code

Agenda

- The need for decoupled security controls
- The need for “Policy-as-Code”
- PaC across the StaCk
- Application and API Gateway: Policy-as-Code
- Cloud-Native Control Planes - Policy-as-Code

Agenda

- The need for decoupled security controls
- The need for “Policy-as-Code”
- PaC across the StaCk
- Application and API Gateway: Policy-as-Code
- Cloud-Native Control Planes - Policy-as-Code
- Workload Policy-as-Code

Agenda

- The need for decoupled security controls
- The need for “Policy-as-Code”
- PaC across the StaCk
- Application and API Gateway: Policy-as-Code
- Cloud-Native Control Planes - Policy-as-Code
- Workload Policy-as-Code
- Conclusions

Trends on the Application Delivery Front

abhaybhargav

Some numbers...

- An Average of 15,564 APIs in use per Enterprise - Noname Security Report 2022
- Large Enterprises have 25,592 APIs (Noname Security Report 2022)
- 67% of Enterprises reported a slowed down deployment of security issues related to Kubernetes - Redhat State of Kubernetes Security Report 2023
- Kubernetes: Platform for moving workloads to public cloud. Grew 127+% in 2022 - Dynatrace Report

Dev-First Workflows!

abhaybhargav

APPSEC engineer we45

~~Dev-First Workflows!~~

abhaybhargav

APPSEC engineer we45

^

~~Dev–First Workflows!~~

abhaybhargav

Workflows that support iterative and continuous delivery of apps

^

~~Dev–First Workflows!~~

This means...

- Dev has consumed Ops (Infrastructure-as-Code, Continuous Integration, Continuous Deployment)
- Dev has consumed QA (Test Automation)
- Dev is halfway through consuming security (Security-as-code)
- Dev is coming for policy, compliance, etc next

Why is this good?

- ⬆️ Automation!
- ⬇️ Human Intervention
- ⏭ Faster delivery of features
- 🌤 Highly Scalable

DevSecOps

Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

SAST as Code

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

SAST as Code

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

SAST as Code

DAST/Regression as Code

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

SAST as Code

DAST/Regression as Code

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines

SAST
Source Composition Analysis
Secure Defaults
Build Security Processes

DAST
IAST,
InfraSec,
Sec Regression

Infrastructure Security,
Cloud Hardening,
Secrets Management

Security monitoring
& attack detection,
Threat Hunting,
Attack Simulation/RedTeam

SAST as Code

DAST/Regression as Code

Decoupled Security Controls
/Policy-As-Code

abhaybhargav

DevSecOps

Plan	Code	Build	Test	Release	Deploy	Operate	Monitor
Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes		DAST IAST, InfraSec, Sec Regression		Infrastructure Security, Cloud Hardening, Secrets Management		Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

Threat Models as Code

SAST as Code

DAST/Regression as Code

Decoupled Security Controls /Policy-As-Code

abhaybhargav

DevSecOps

Plan	Code	Build	Test	Release	Deploy	Operate	Monitor
Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes		DAST IAST, InfraSec, Sec Regression		Infrastructure Security, Cloud Hardening, Secrets Management		Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

Threat Models as Code

SAST as Code

DAST/Regression as Code

Decoupled Security Controls /Policy-As-Code

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines

SAST
Source Composition Analysis
Secure Defaults
Build Security Processes

DAST
IAST,
InfraSec,
Sec Regression

Infrastructure Security,
Cloud Hardening,
Secrets Management

Security monitoring
& attack detection,
Threat Hunting,
Attack Simulation/RedTeam

Threat Models as Code

SAST as Code

DAST/Regression as Code

Decoupled Security Controls
/Policy-As-Code

Detection Engineering

abhaybhargav

DevSecOps

Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam
Threat Models as Code	SAST as Code	DAST/Regression as Code	Decoupled Security Controls /Policy-As-Code	Detection Engineering

abhaybhargav

Decoupled Security Controls and Policy as Code

abhaybhargav

Need and Motivation

Need and Motivation

- The idea is to NOT hardcode security rules in app that have rapidly evolving and changing requirements

Need and Motivation

- The idea is to NOT hardcode security rules in app that have rapidly evolving and changing requirements

- Customisable and Purpose-Built

Need and Motivation

- The idea is to NOT hardcode security rules in app that have rapidly evolving and changing requirements
- Customisable and Purpose-Built
- Testable

Need and Motivation

- The idea is to NOT hardcode security rules in app that have rapidly evolving and changing requirements
- Customisable and Purpose-Built
- Testable
- Scalable

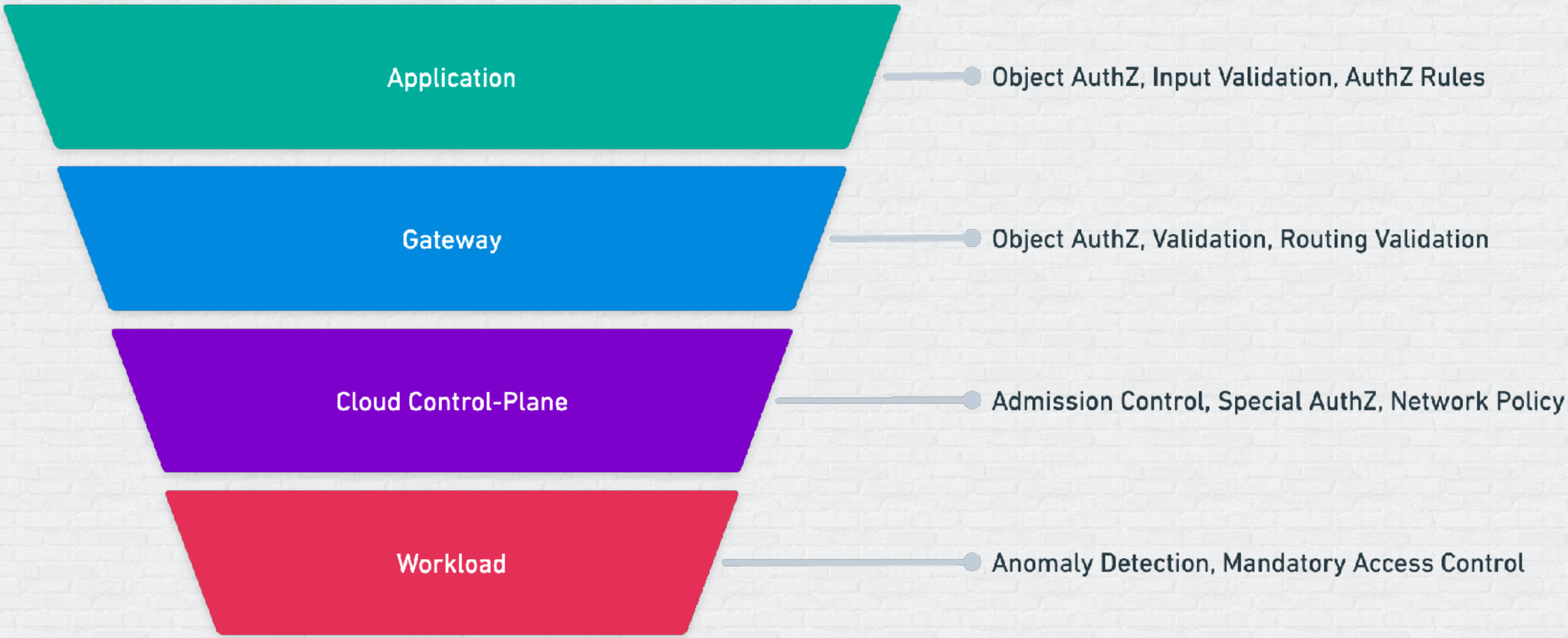
Need and Motivation

- The idea is to NOT hardcode security rules in app that have rapidly evolving and changing requirements
- Customisable and Purpose-Built
- Testable
- Scalable
- Create a “Paved Road” for Product Engineering Teams

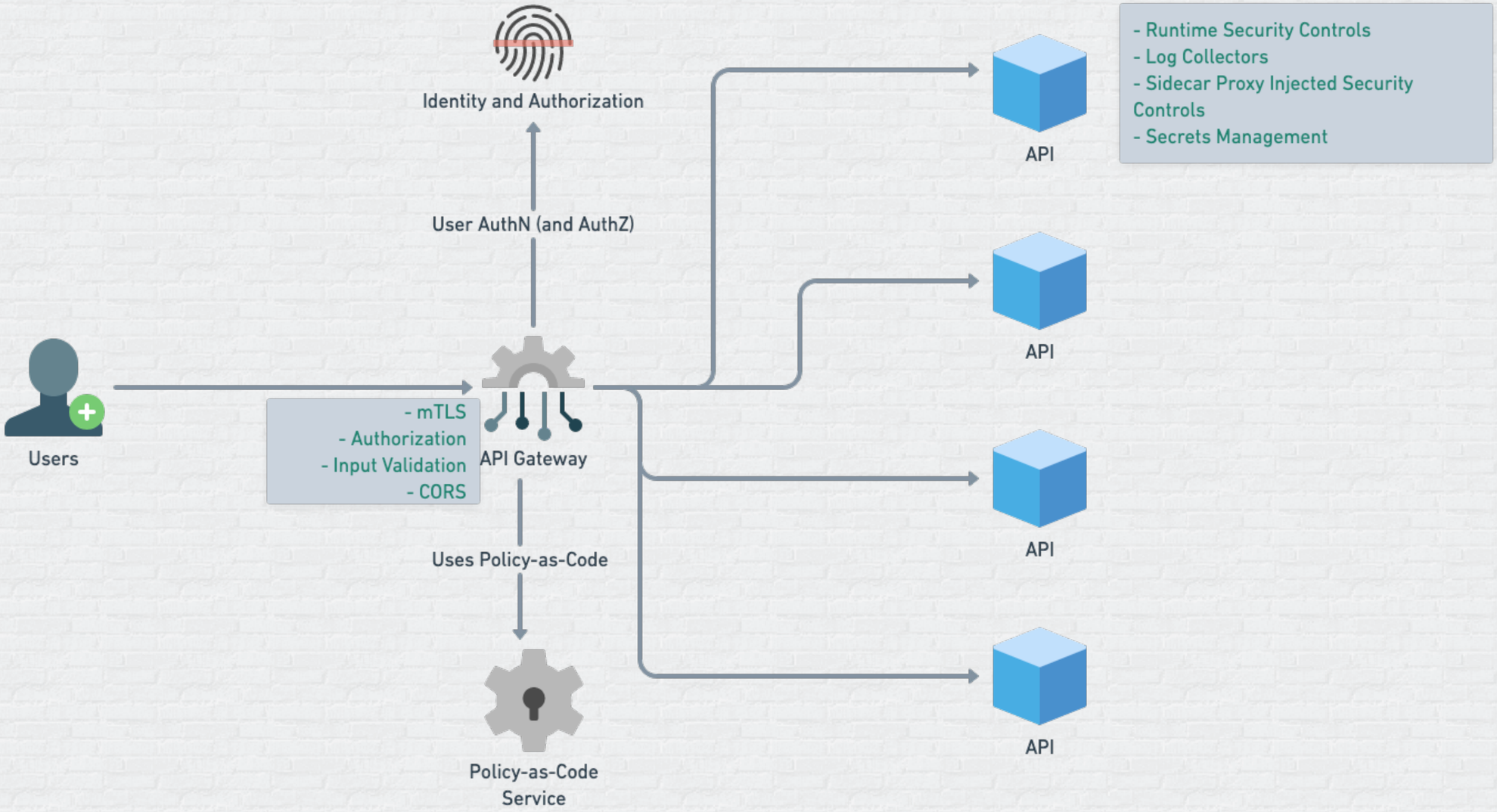
Typical Use-Cases

- Syscall Profiling, Seccomp, AppArmor and eBPF for Runtime Security enforcement
- Authorization, CORS, Rate-Limiting, mTLS and others on the API Gateway
- Log Collection and aggregation of services from Cloud-Native environments
- Input Validation, Access Control with Policy-as-Code Frameworks

Across the Stack



Security Model – An Example



abhaybhargav

Application + Gateway

abhaybhargav

BOLA

Broken Object Level AuthZ

BOLA

abhaybhargav

Broken Object Level AuthZ

BOLA

EDE

abhaybhargav

Broken Object Level AuthZ

BOLA

Excessive Data Exposure

EDE

abhaybhargav

Broken Object Level AuthZ

BOLA

BFLA

Excessive Data Exposure

EDE

abhaybhargav

Broken Object Level AuthZ

BOLA

Broken Function Level AuthZ

BFLA

Excessive Data Exposure

EDE

abhaybhargav

Broken Object Level AuthZ

BOLA

BA

Broken Function Level AuthZ

BFLA

Excessive Data Exposure

EDE

abhaybhargav

Broken Object Level AuthZ

BOLA

Broken AuthN

BA

Broken Function Level AuthZ

BFLA

Excessive Data Exposure

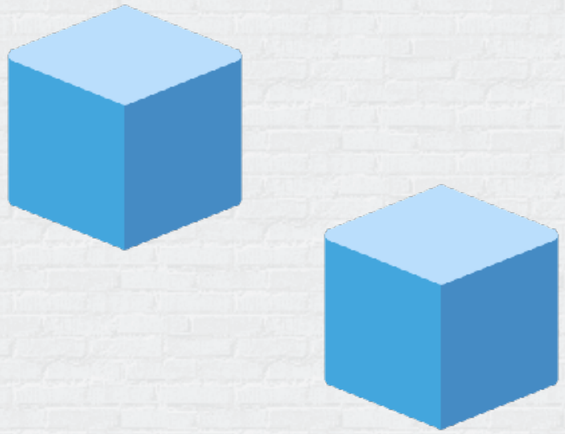
EDE

abhaybhargav

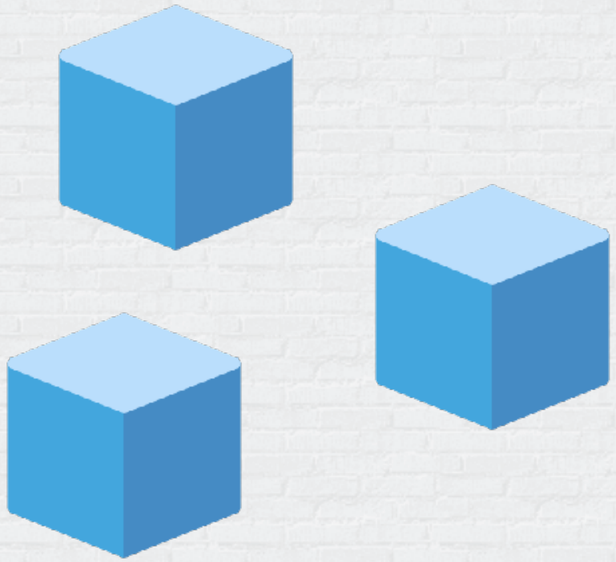
Imagine...

Imagine...

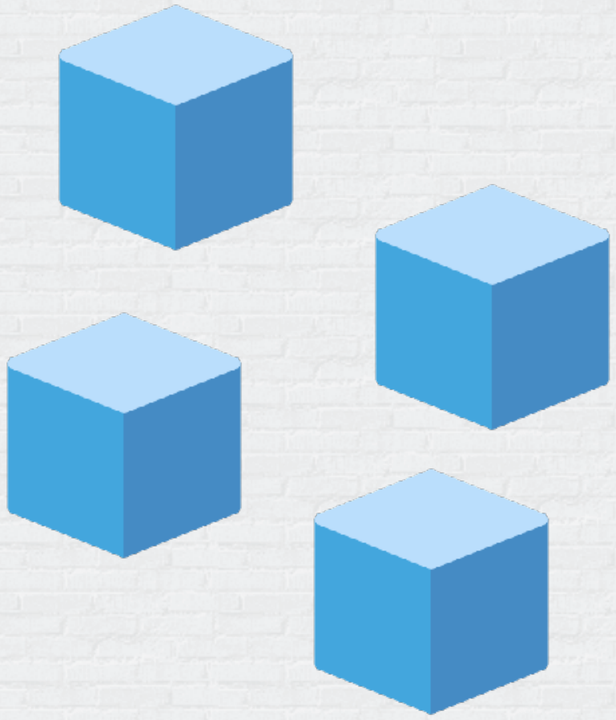
Imagine...



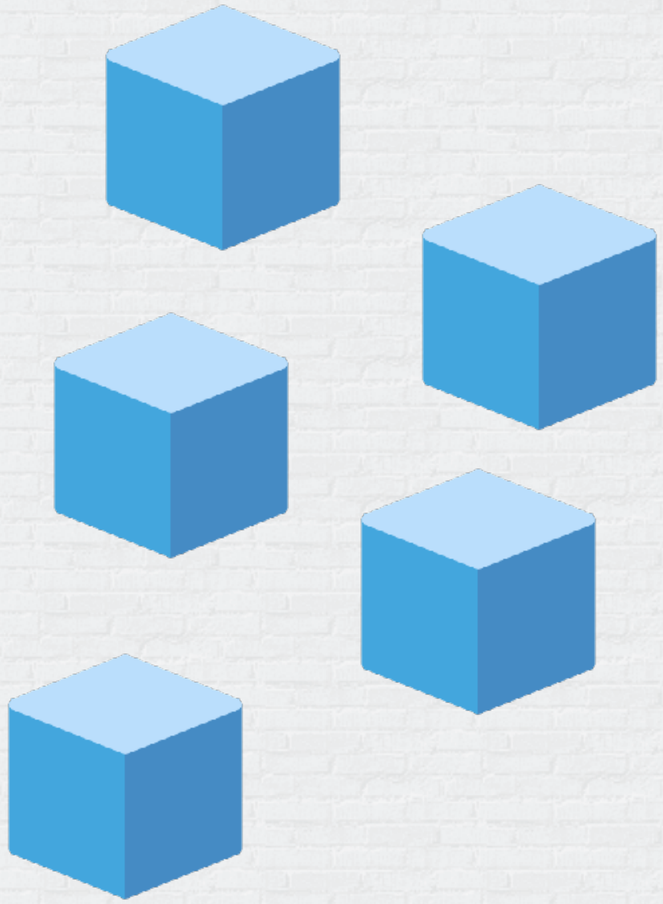
Imagine...



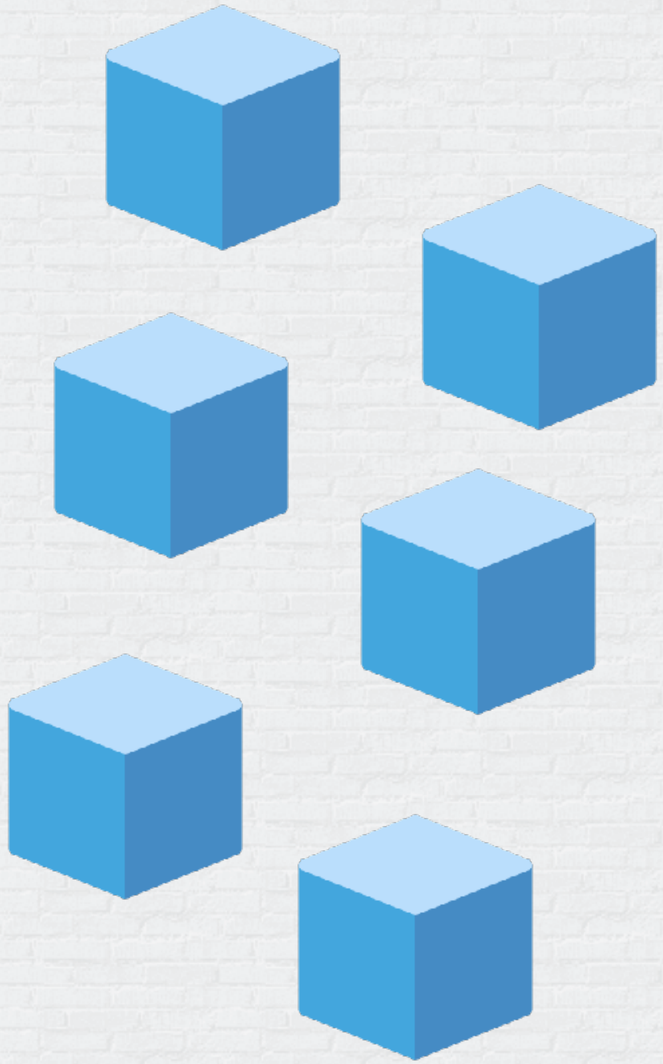
Imagine...



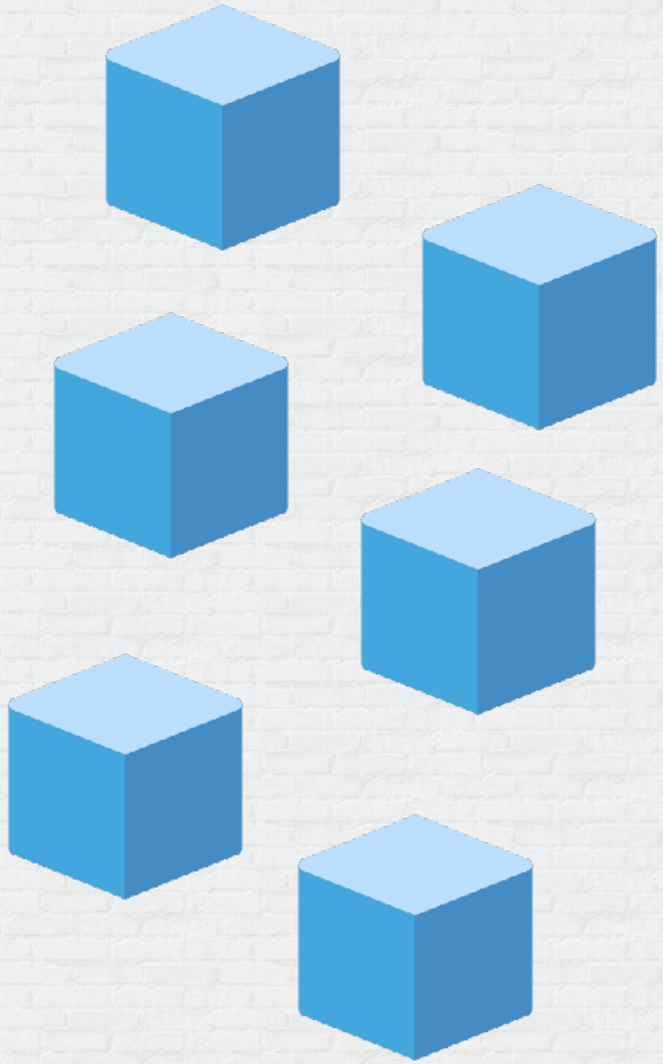
Imagine...



Imagine...

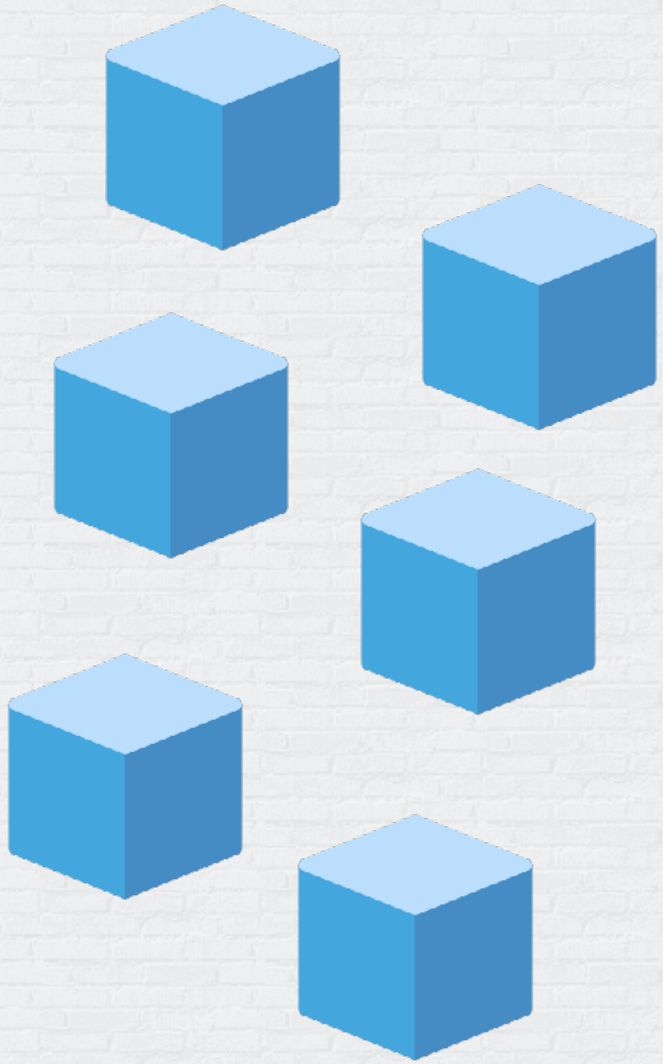


Imagine...



Your Service Business Logic

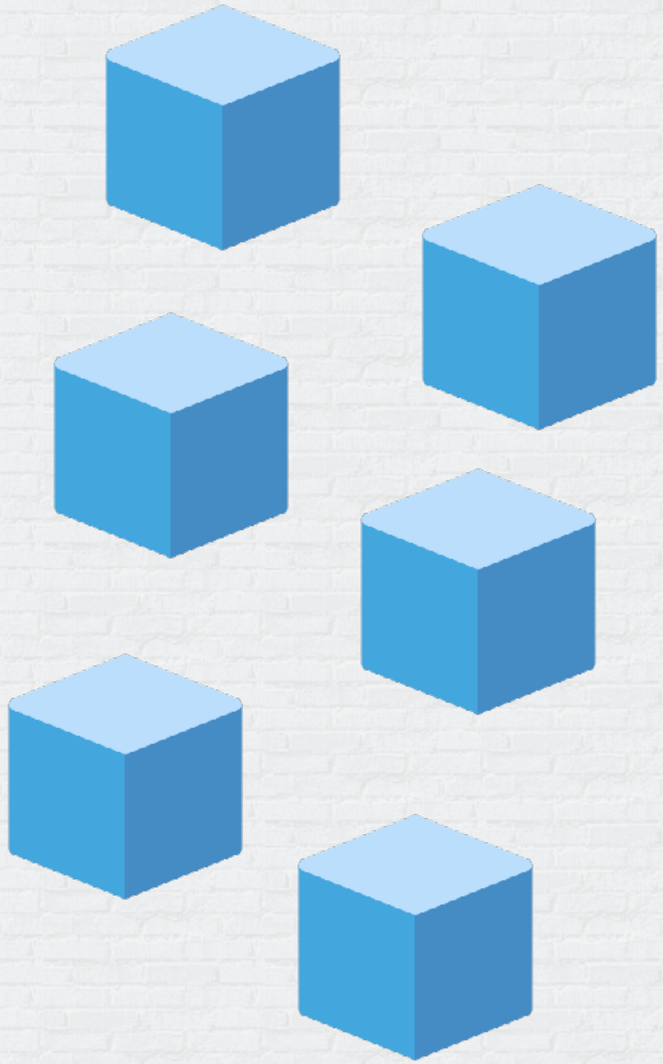
Imagine...



JWT Authorization

Your Service Business Logic

Imagine...



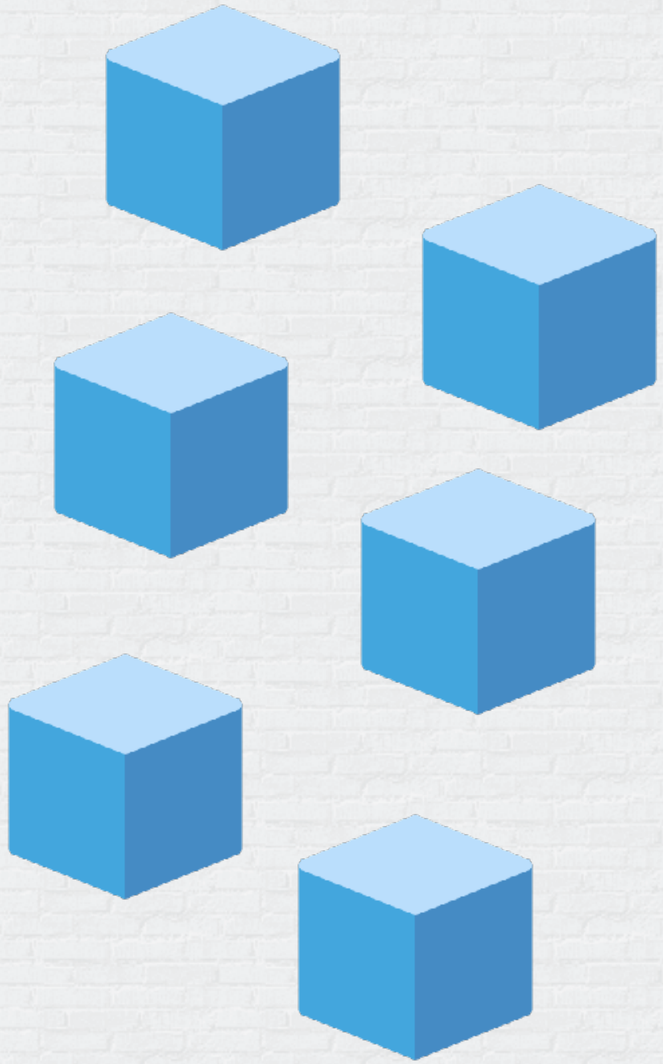
Input Validation

JWT Authorization

Your Service Business Logic

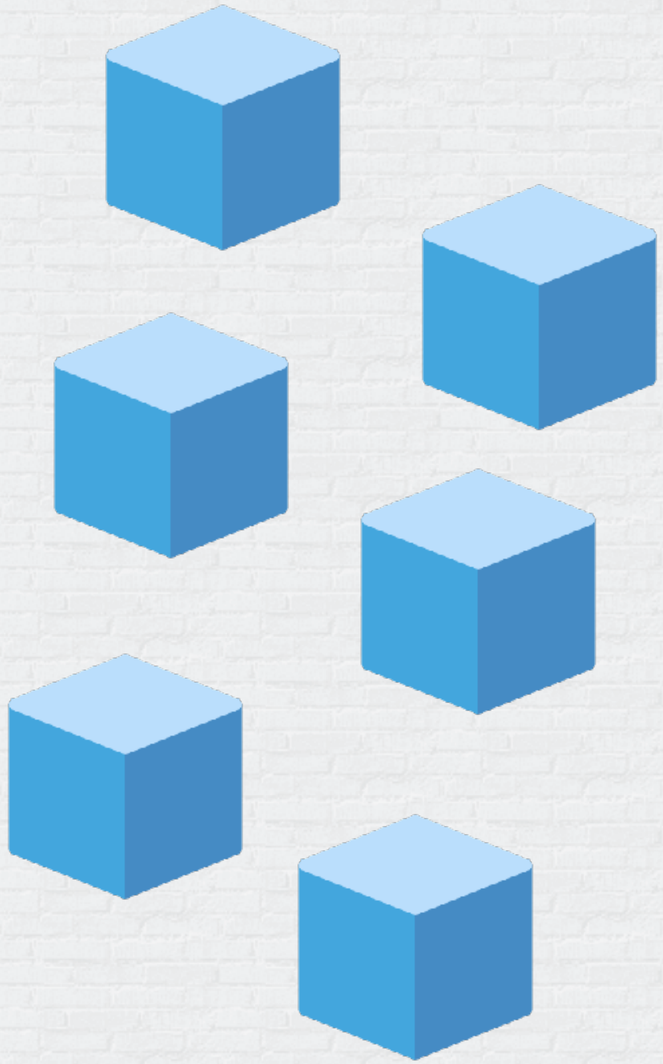
abhaybhargav

Imagine...



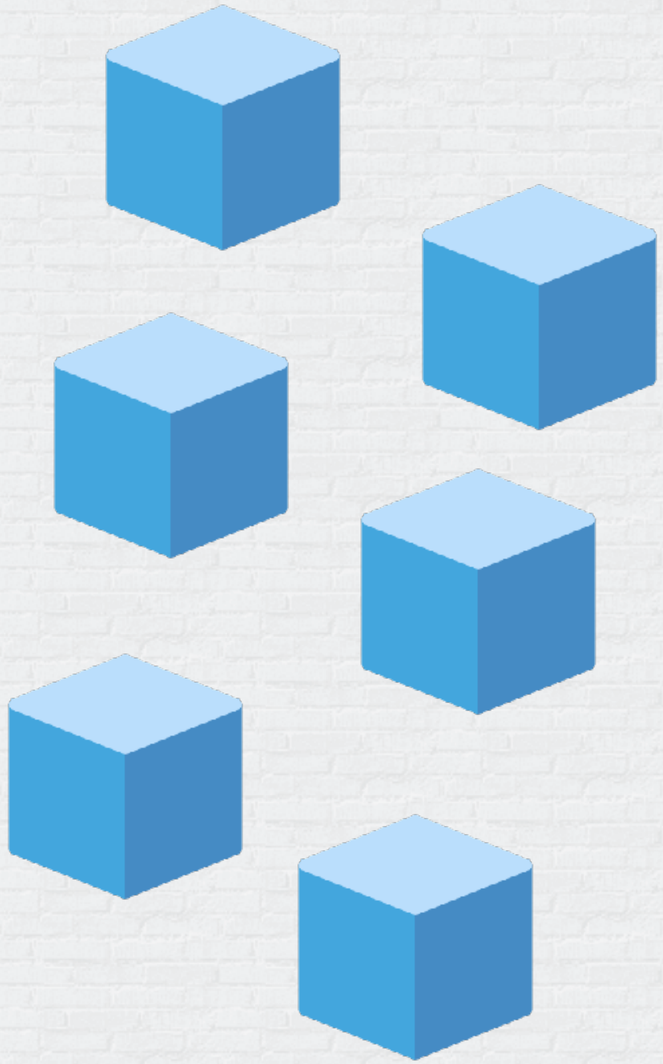
- Object Access Control
- Input Validation
- JWT Authorization
- Your Service Business Logic

Imagine...



- Authentication
- Object Access Control
- Input Validation
- JWT Authorization
- Your Service Business Logic

Imagine...



- Logging
- Authentication
- Object Access Control
- Input Validation
- JWT Authorization
- Your Service Business Logic

PaC – Applicability

PaC – Applicability

- Input Validation at Gateway

PaC – Applicability

- Input Validation at Gateway
- JWT Validation at Gateway + Claims

PaC – Applicability

- Input Validation at Gateway
- JWT Validation at Gateway + Claims
- Function Level AuthZ at App + Gateway

PaC – Applicability

- Input Validation at Gateway
- JWT Validation at Gateway + Claims
- Function Level AuthZ at App + Gateway
- Object Level AuthZ at App + Gateway

Frameworks we'll use

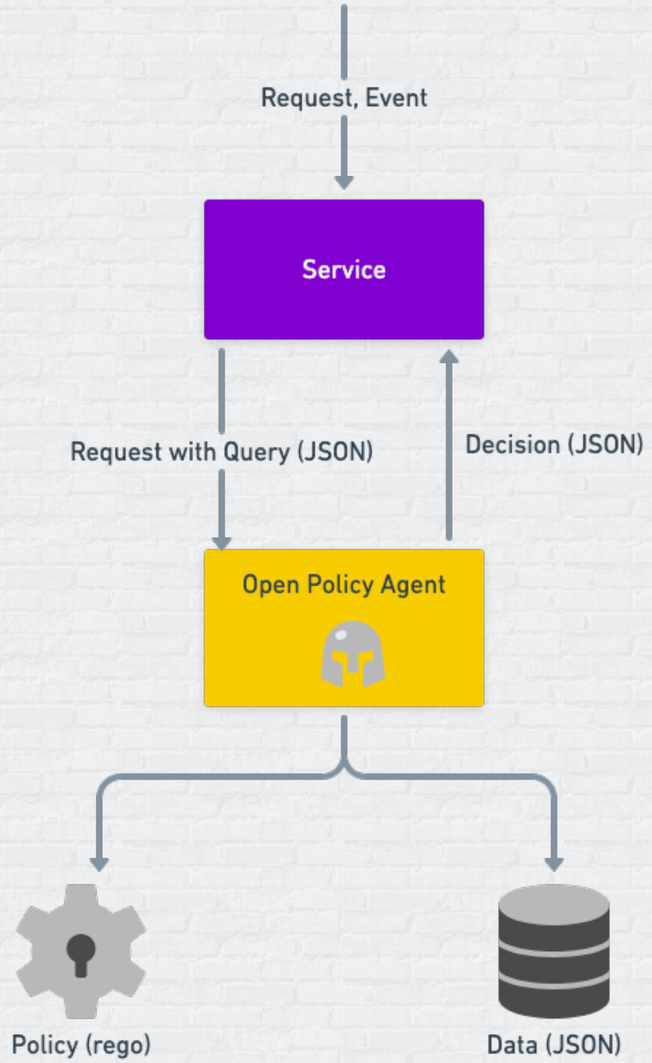
- Open Policy Agent and Rego
- Casbin/Oso/Permify, etc

Open-Policy-Agent

- Policy Management Framework for “any” environment
- Allows you to define policies that can be enforced based on generic json input and output parameters
- Uses a DSL (domain specific language) called “rego” that is used to define policies

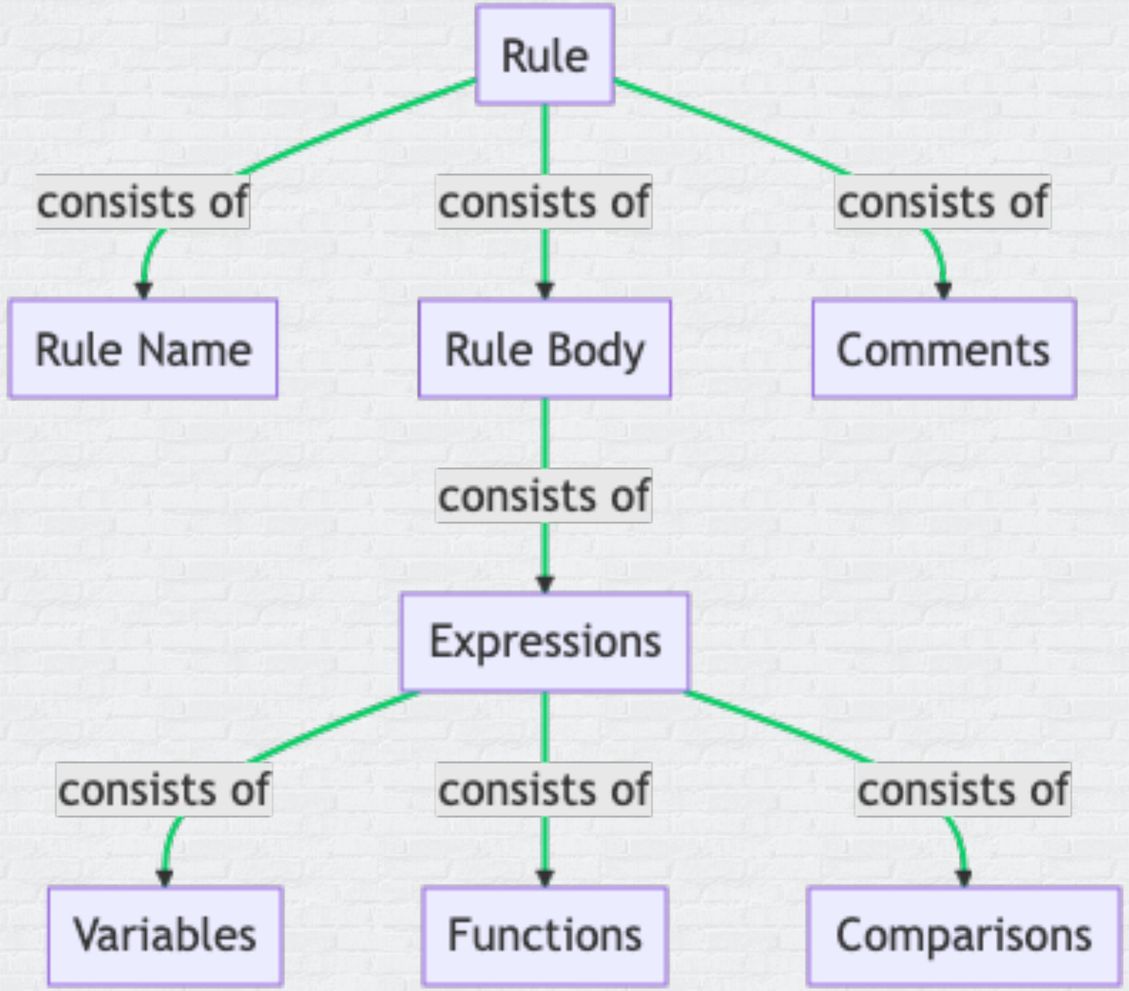
Open Policy Agent

Open Policy Agent – Operation



abhaybhargav

Rego Rule Syntax



Rego Rule Syntax

abhaybhargav

Rego Rule Syntax

```
allow {
    "read" == input.method
    input.resource == "/protected"
}
```

abhaybhargav

Rego Rule Syntax

abhaybhargav

Rego Rule Syntax

```
default allow = false

allow {
    "read" == input.method
    input.resource == "/protected"
}
```

abhaybhargav

Rego Rule Syntax

abhaybhargav

Rego Rule Syntax

```
allow = true {
    "read" == input.method
    input.resource == "/protected"
}
```

abhaybhargav

Rego Rule Syntax

abhaybhargav

Rego Rule Syntax

```
is_admin(user) {
    user.role == "admin"
}
```

abhaybhargav

Rego Rule Syntax

abhaybhargav

Rego Rule Syntax

```
valid_email(email) {
    startswith(email, "user@")
    endswith(email, ".com")
}
```

Rego Rule Syntax

abhaybhargav

Rego Rule Syntax

```
//function level access control

allow {
    split(input.path, "/")[2] == input.user.id
}
```

abhaybhargav

Rego Rule Syntax

abhaybhargav

Rego Rule Syntax

```
default allow = false

allow {
    input.path = ["users"]
    input.method = "POST"
    claims.user
}

claims := payload {
    io.jwt.verify_hs256(token, "password")
    [_,payload,_] := io.jwt.decode(token)
}

token := t {
    t := input.token
}
```

abhaybhargav

Rego Rule Syntax

abhaybhargav

OPA Use-Cases

- Kubernetes Policy Management
- API AuthZ and Policy Management
- OS Policy Management - SSH and Access Control
- Kafka Topic Authorization
- Many more...

OPA – API AuthZ

```
package authz

allow {
    input.path == ["users"]
    input.method == "POST"
}

allow {
    some profile_id
    input.path = ["users", profile_id]
    input.method == "GET"
    profile_id == input.user_id
}
```

abhaybhargav

OPA Demos

abhaybhargav

AuthZ-as-Code

abhaybhargav

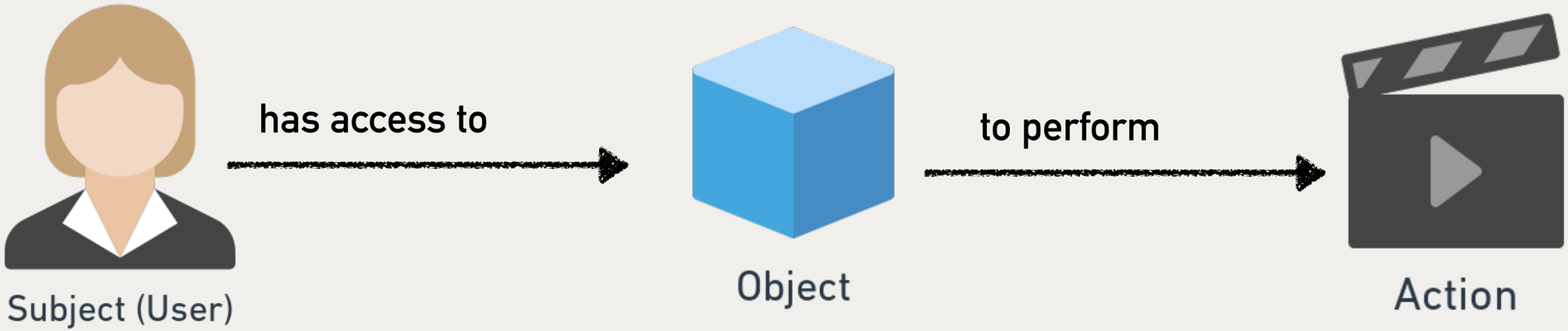
Let's look at most AuthZ flaws

- Inconsistent implementation of Object Level Authorization
- Access Control code strewn across multiple services
- Lack of standardization and expressive capability for AuthZ frameworks
- Heavily design dependent - which gets complex at scale

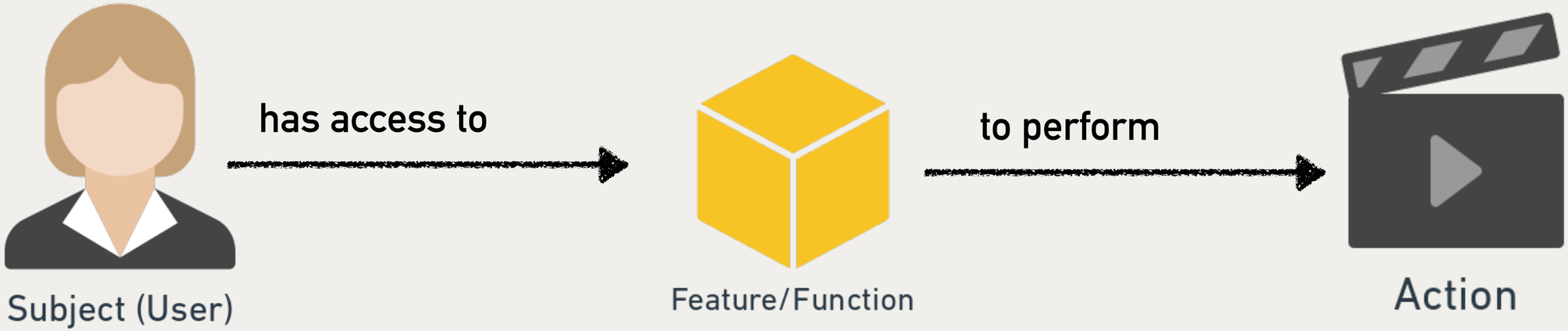

abhaybhargav

AuthZ-as-Code Frameworks

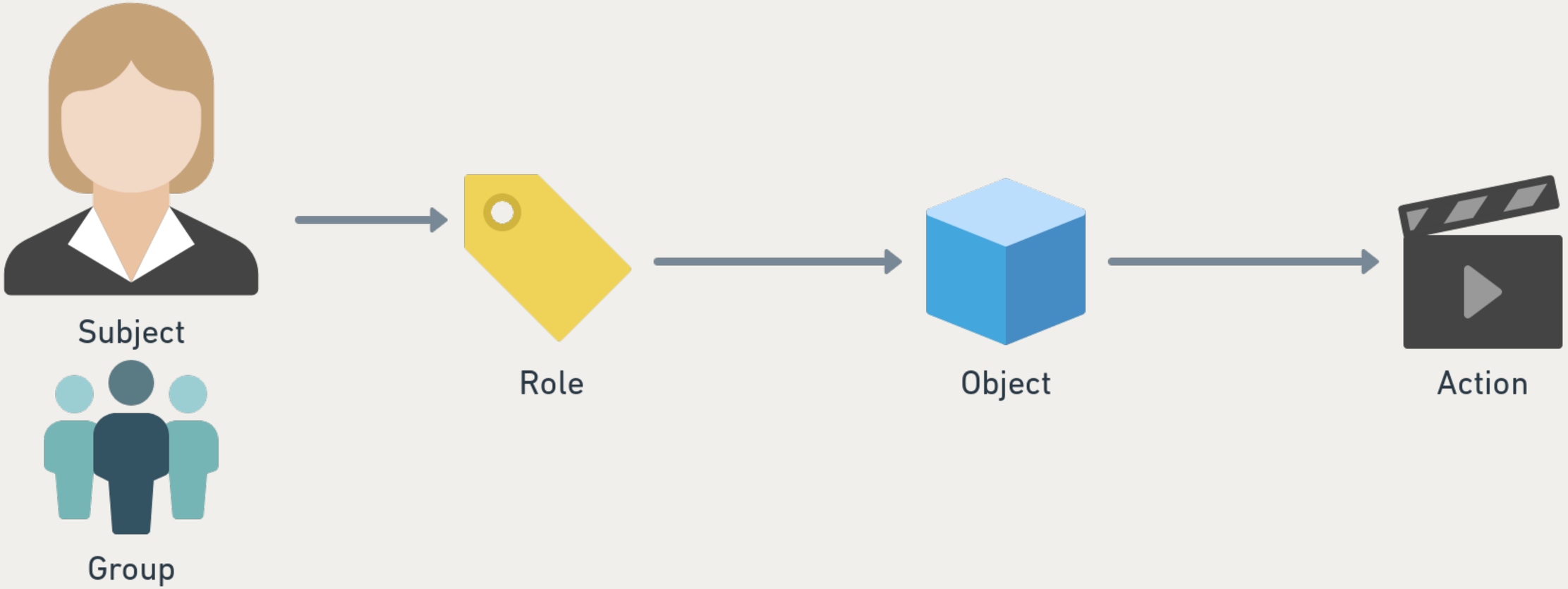
Object Level AuthZ



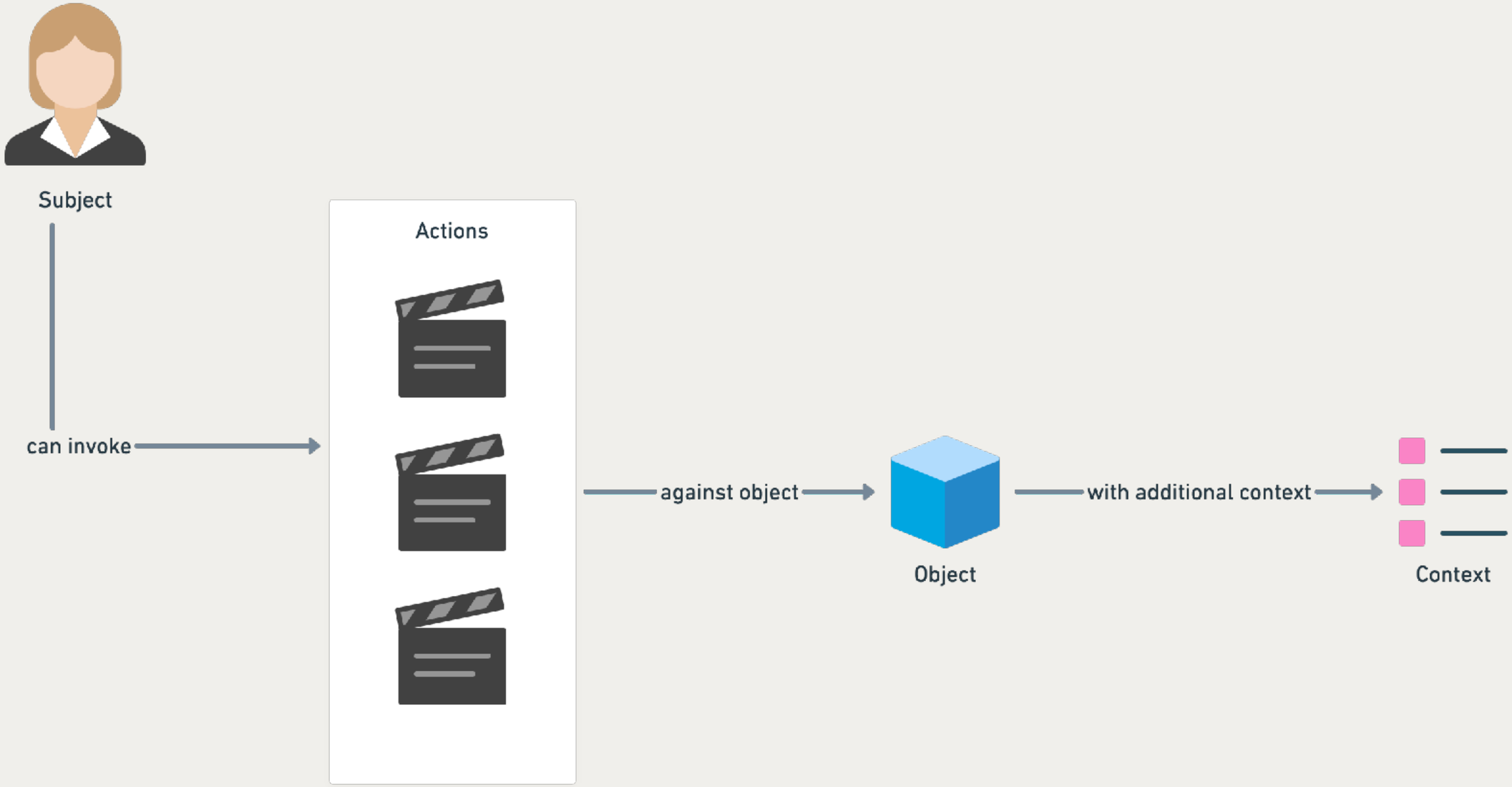
Functional AuthZ



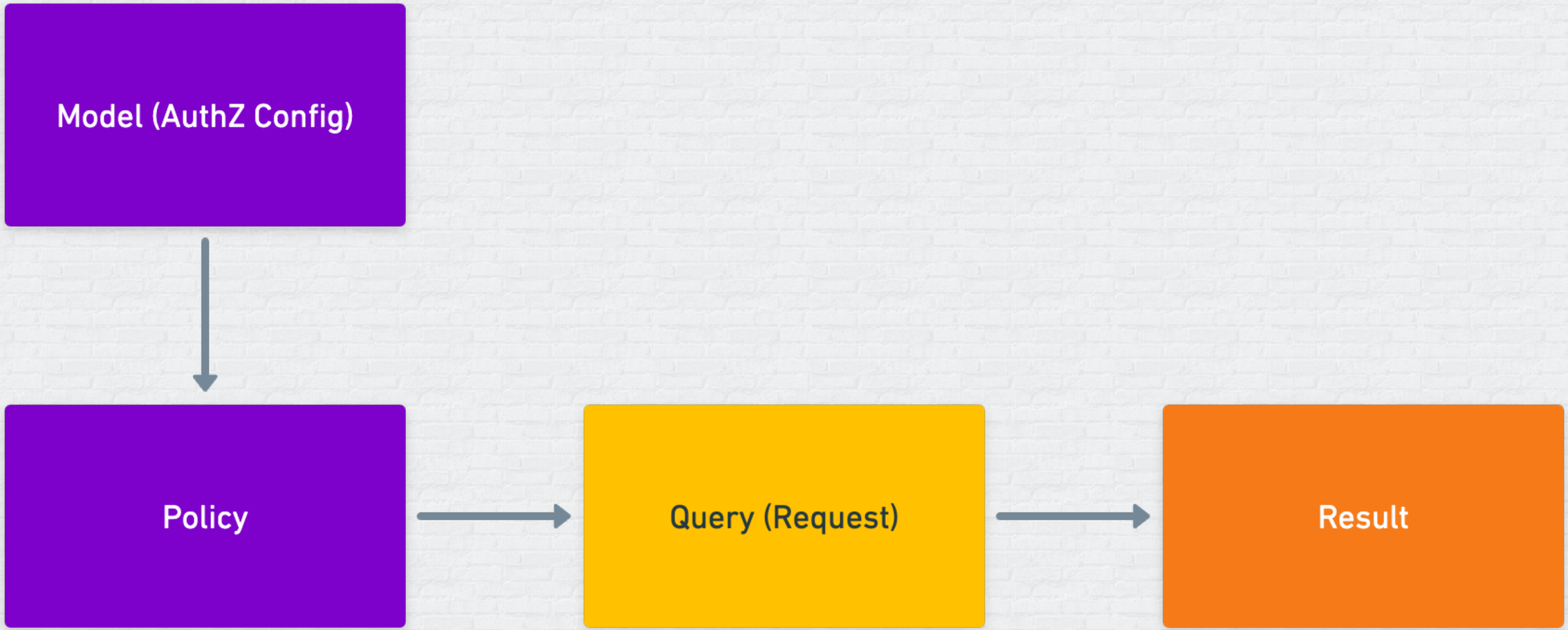
RBAC – Role Based Access Control



ABAC – Attribute Based Access Control



Google Zanzibar approach



Approach

Approach

```
definition user {}

/**
 * resource is an example resource.
 */
definition resource {
    relation writer: user
    relation viewer: user

    permission write = writer
    permission view = viewer + writer
}
```

Approach

Approach

```
// Some example relationships
resource:someresource#viewer@user:somegal
resource:someresource#writer@user:anotherguy
resource:anotherresource#writer@user:somegal
```

Approach

Casbin


abhaybhargav

Casbin

```
[request_definition]
r = sub, obj, act

[policy_definition]
p = sub, obj, act

[role_definition]
g = _, _

[policy_effect]
e = some(where (p.eft == allow))

[matchers]
m = g(r.sub, p.sub) && r.obj == p.obj && r.act == p.act
```

abhaybhargav

Casbin


abhaybhargav

Casbin

```
# Roles definition.
p, admin, data1, read
p, admin, data1, write
p, user, data2, read

# Assign roles to users.
g, alice, admin
g, bob, user
```

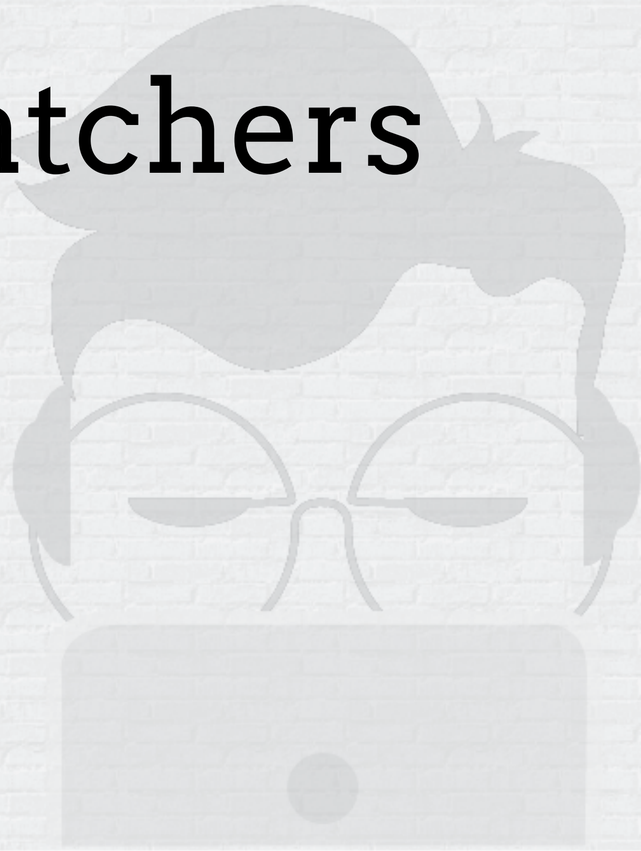
Casbin


abhaybhargav

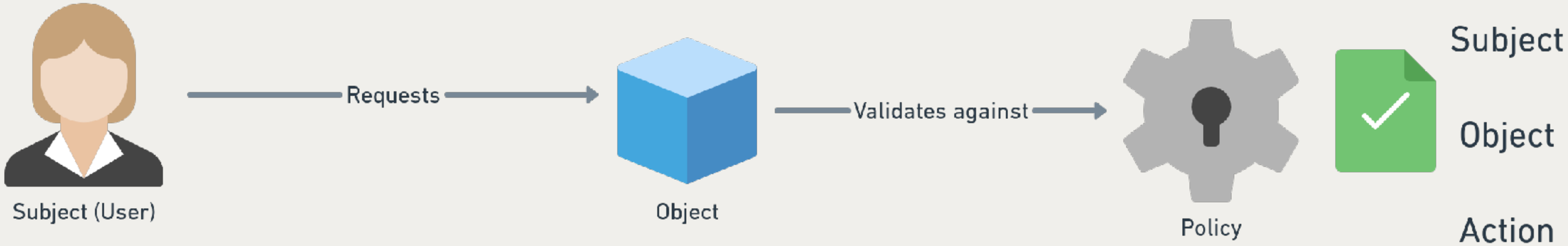
PERM

PERM

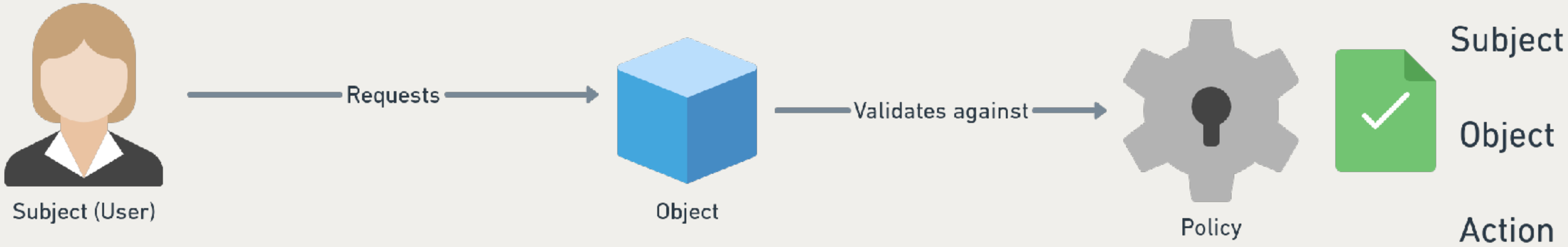
Policy, Effect, Request, Matchers



What is PERM?

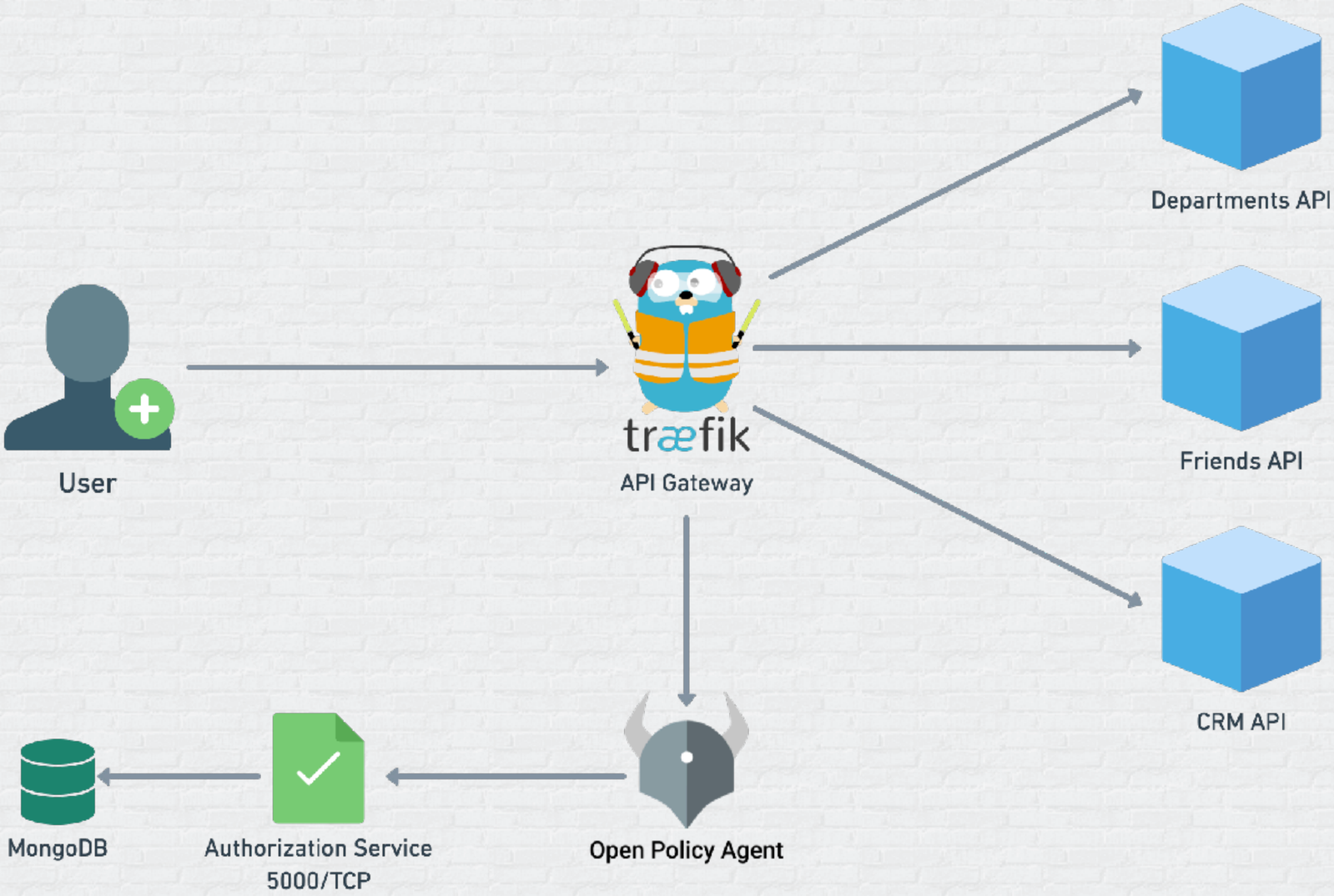

Subject (User)
Requests
Object
Validates against
Policy
Subject
Object
Action

What is PERM?



Request Attributes must MATCH Policy Attributes

Lab: OPA, Traefik and Decentralized security Controls



PaC on Cloud Control-Planes

abhaybhargav

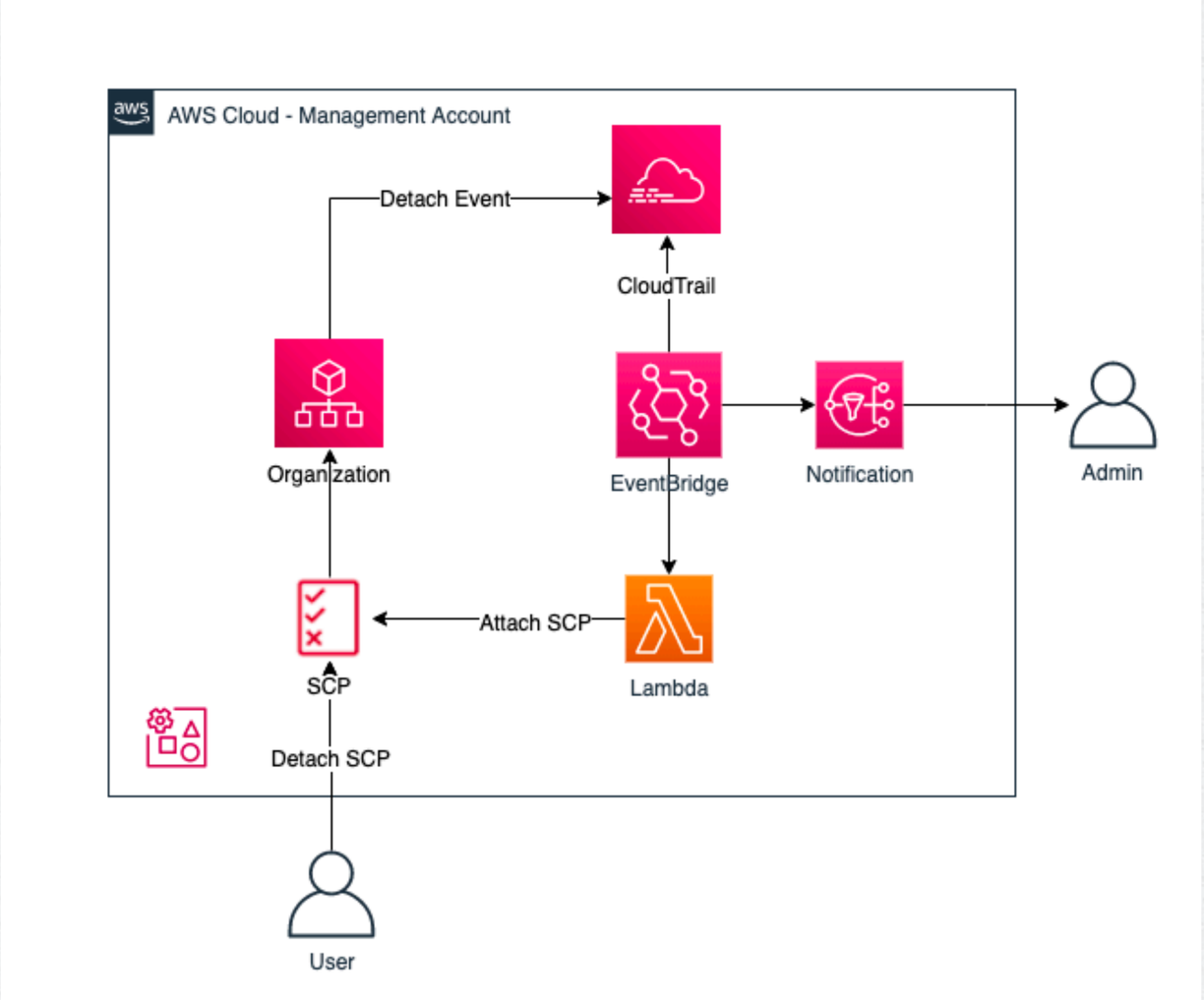
PaC Applicability

- PaC is already important for enforcing policies across Cloud and Cloud-Native Control-Planes
- Can be leveraged for Access Control, Admission Control
- Common Use-Cases: Network Policy, Service Policies, Admission Control Policies

PaC – Cloud Control–Planes

abhaybhargav

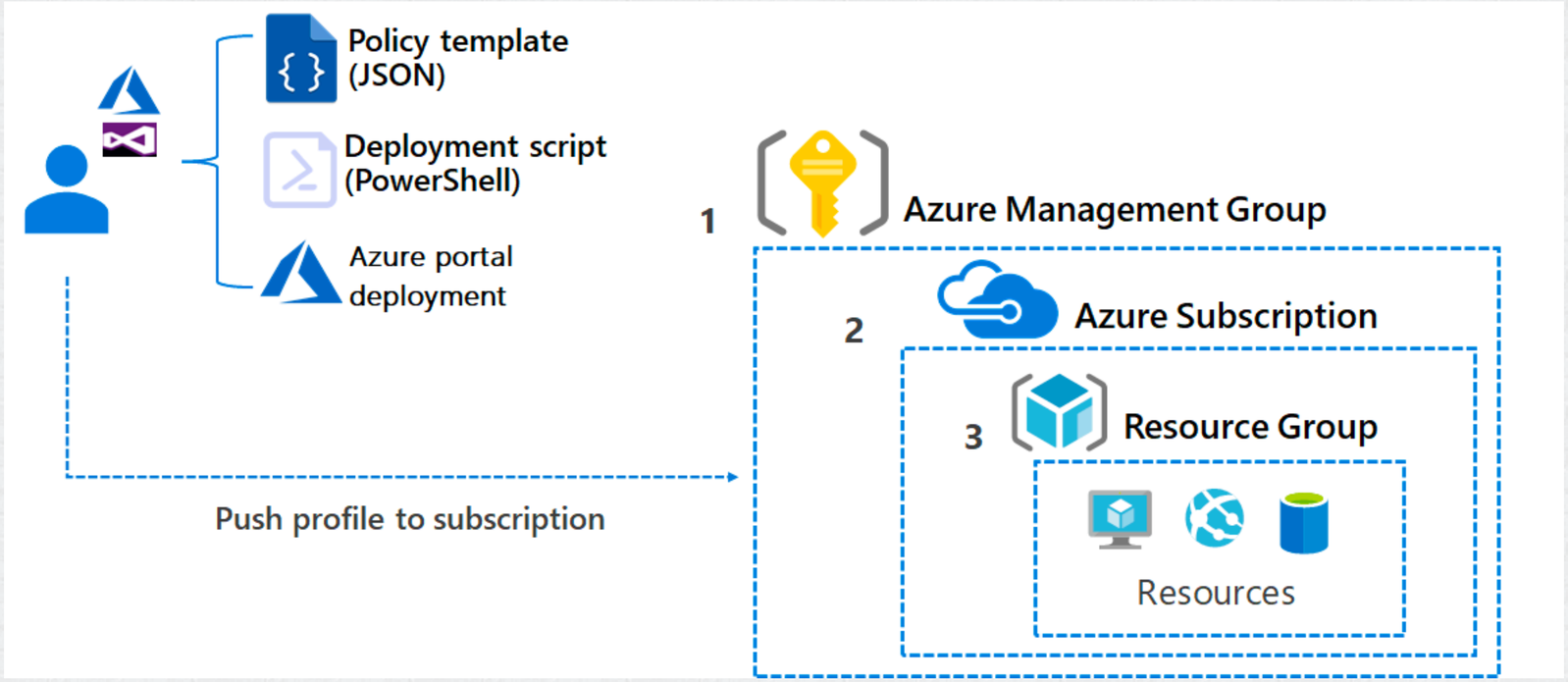
PaC – Cloud Control–Planes



PaC – Cloud Control–Planes

abhaybhargav

PaC – Cloud Control–Planes

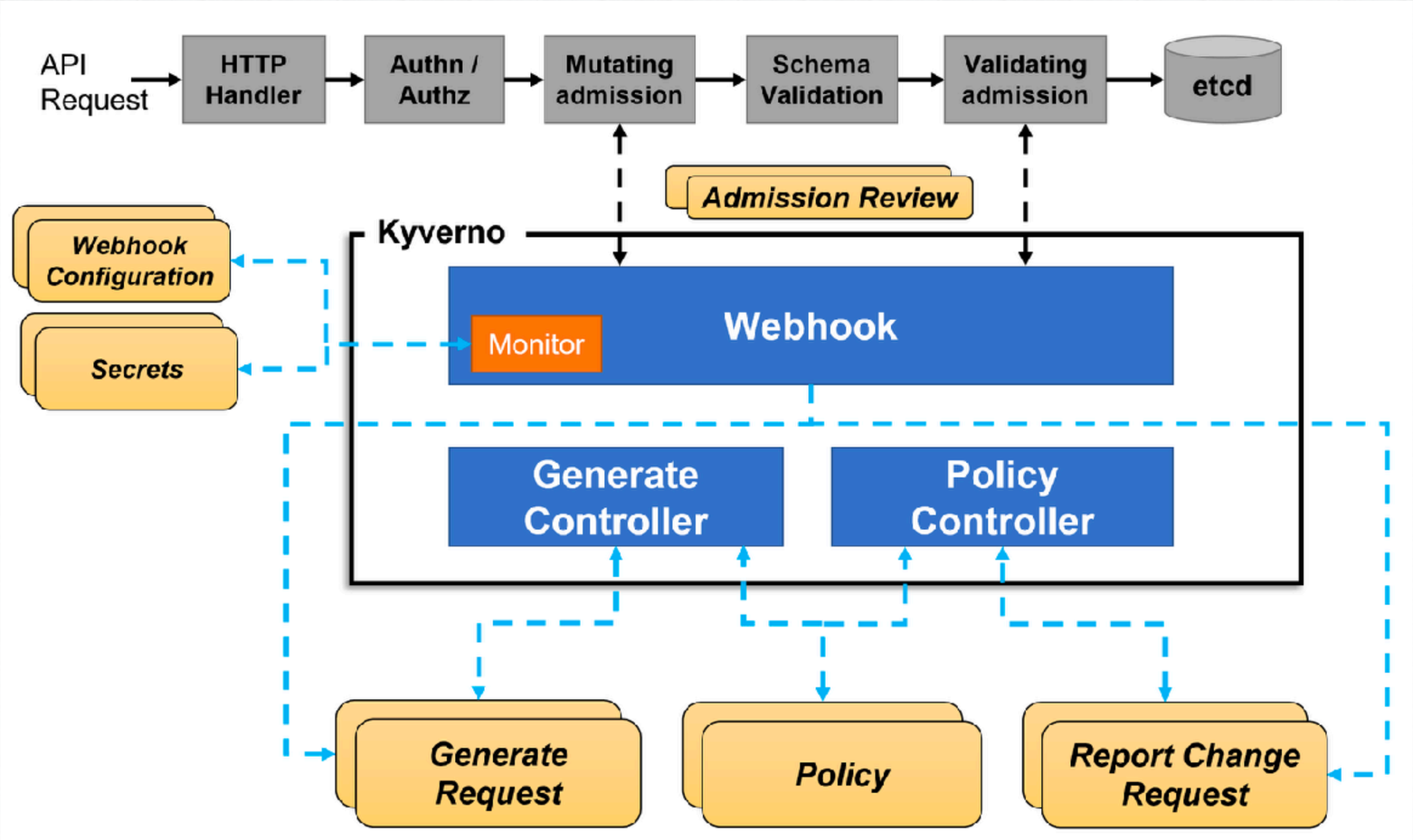


abhaybhargav

PaC – Cloud Control–Planes

abhaybhargav

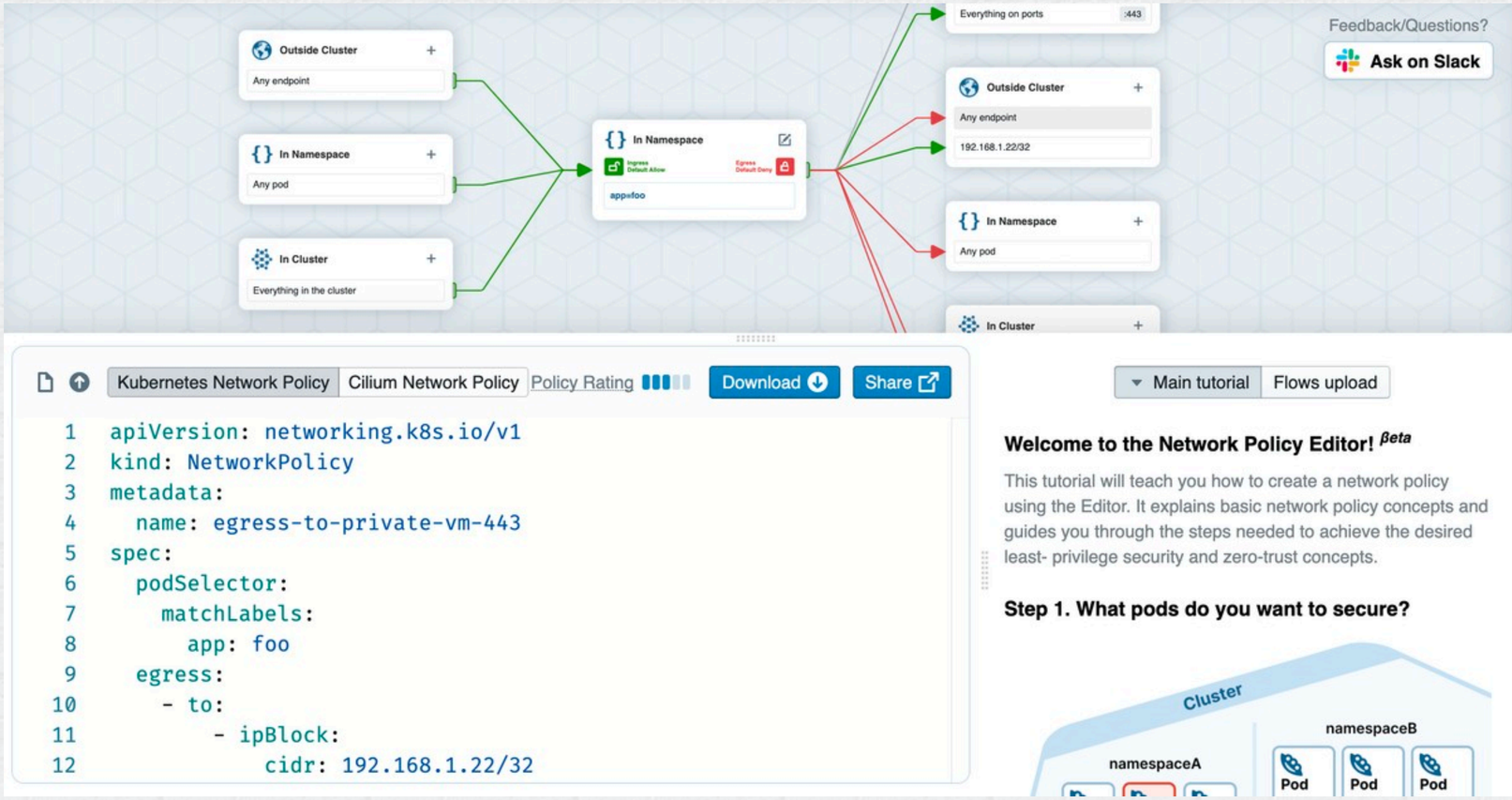
PaC – Cloud Control–Planes



PaC – Cloud Control–Planes

abhaybhargav

PaC – Cloud Control-Planes



abhaybhargav

PaC – Cloud Control–Planes

abhaybhargav

Policy Management with Kyverno

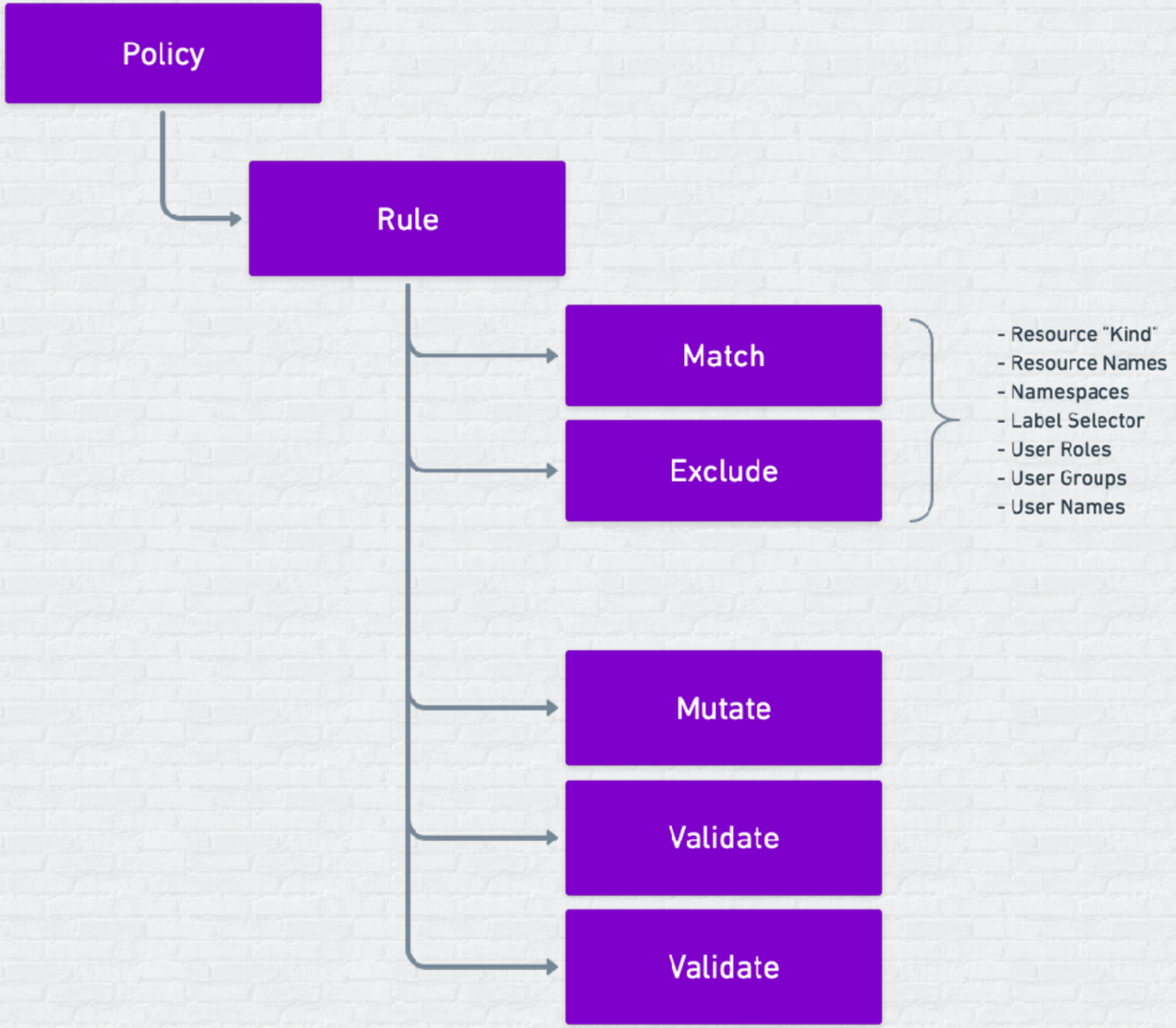
What is Kyverno?

- Policy-Engine specifically designed for Kubernetes
- Policies are created and managed as native Kubernetes resources and authored in YAML
- Validating and Mutating Policies and Webhooks are Supported by Kyverno

Kyverno Concepts

- Install Kyverno CRDs, Webhooks, Service Accounts and Namespaces
- Policies => Validating or Mutating Policy Definitions
- Selectors => Matches Resources in Request based on Policy

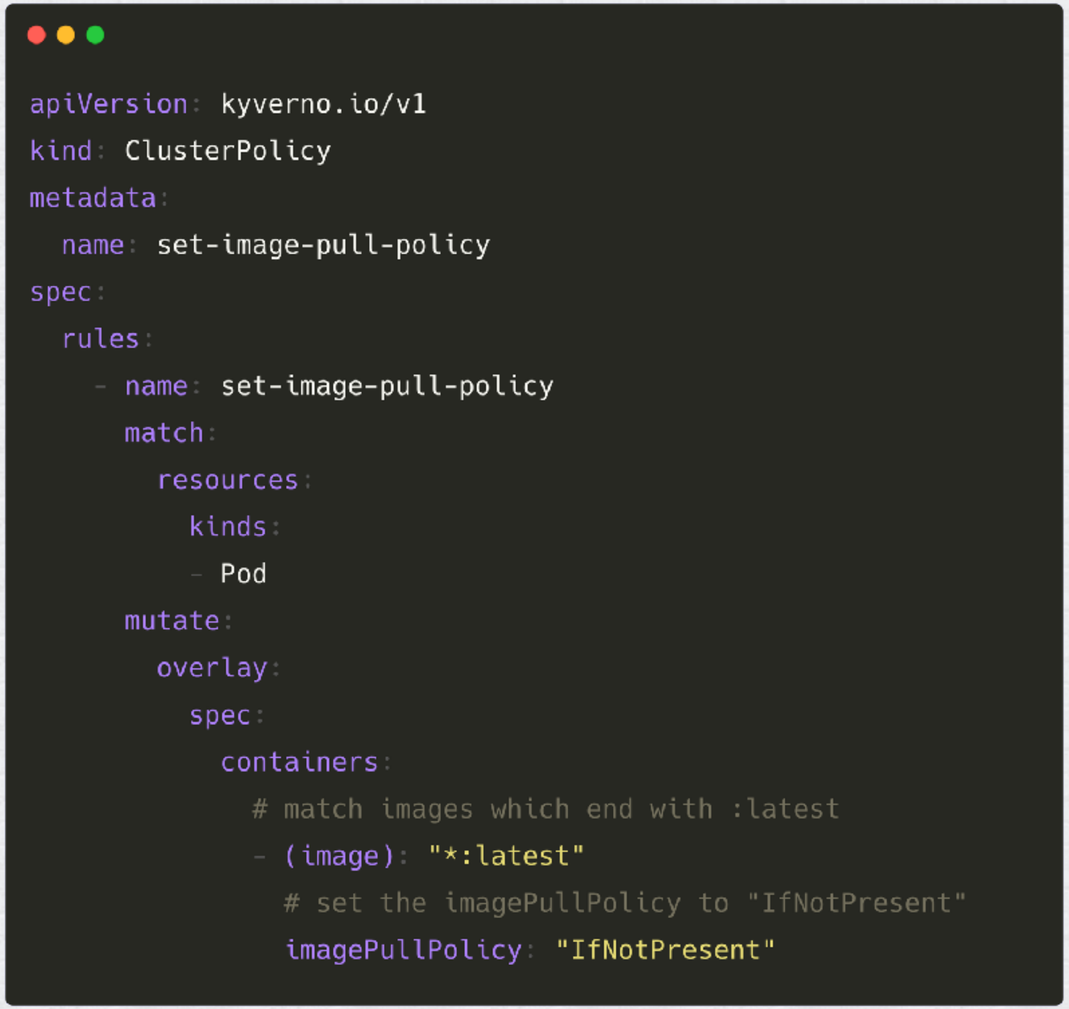
Kyverno Policy Structure



Basic Kyverno Validate Policy


```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-labels
spec:
  validationFailureAction: enforce
  rules:
  - name: check-for-labels
    match:
      resources:
        kinds:
        - Namespace
    validate:
      message: "The label `purpose` is required."
      pattern:
        metadata:
          labels:
            purpose: "?*"
```

Kyverno Mutate Policy



```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: set-image-pull-policy
spec:
  rules:
    - name: set-image-pull-policy
      match:
        resources:
          kinds:
          - Pod
      mutate:
        overlay:
          spec:
            containers:
              # match images which end with :latest
              - (image): "*:latest"
                # set the imagePullPolicy to "IfNotPresent"
                imagePullPolicy: "IfNotPresent"
```

Kyverno Generate Policy

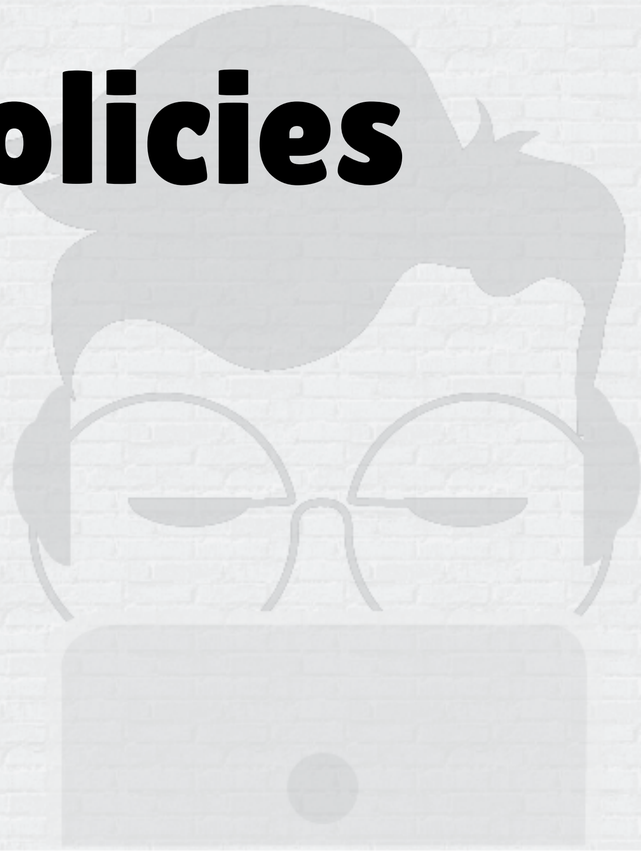
```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: default
spec:
  rules:
  - name: deny-all-traffic
    match:
      resources:
        kinds:
        - Namespace
    exclude:
      resources:
        namespaces:
        - kube-system
        - default
        - kube-public
        - kyverno
    generate:
      kind: NetworkPolicy
      name: deny-all-traffic
      namespace: "{{request.object.metadata.name}}"
      data:
        spec:
          # select all pods in the namespace
          podSelector: {}
          policyTypes:
          - Ingress
          - Egress
```

Kyverno Benefits

- No additional DSL required.
- Mutate, Validate AND Generate
- Background Capabilities
- Audit/Enforce
- Reporting - Out of the box

Lab: Kyverno

Kubernetes Network Policies



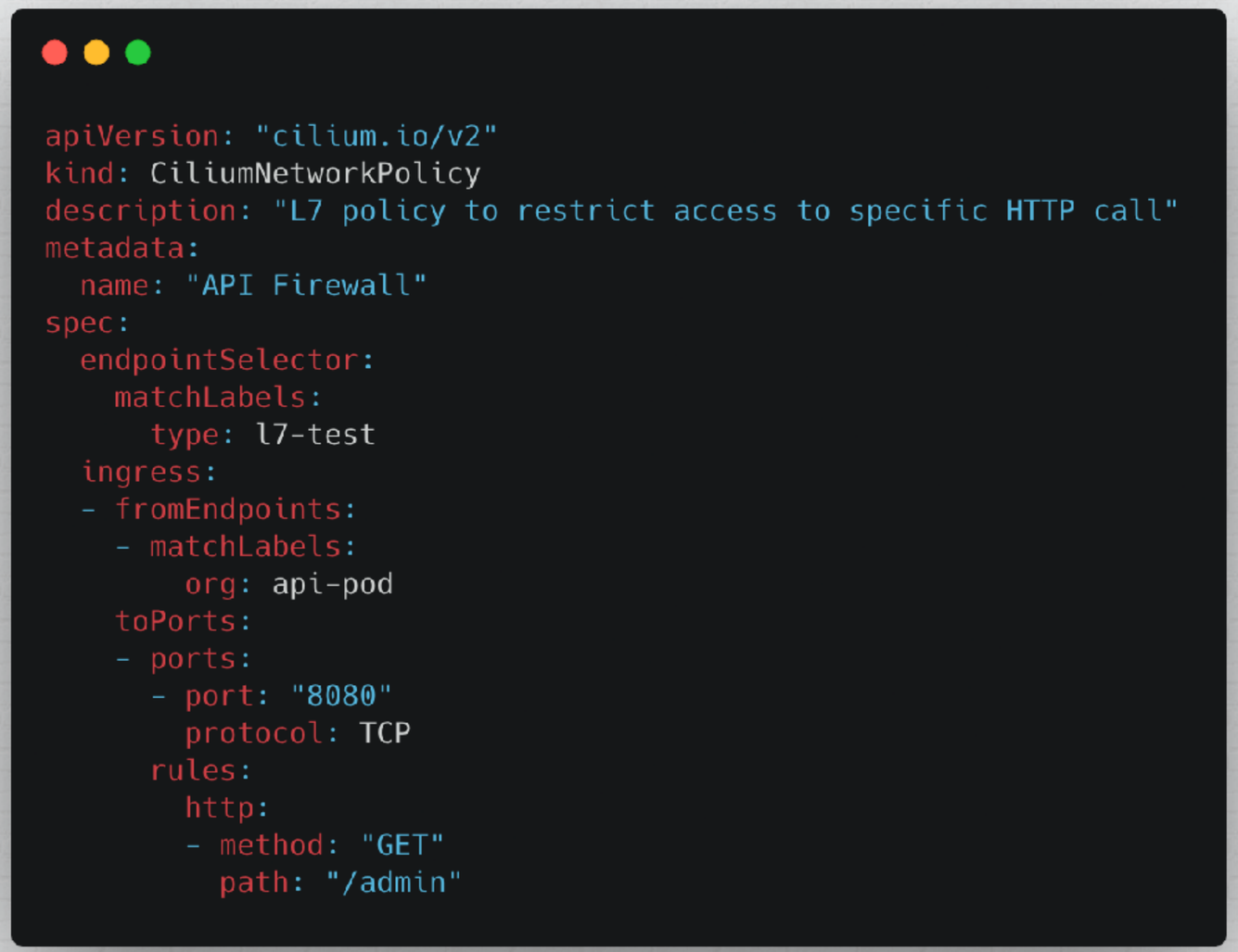
NetworkPolicy – Local and Global

- NetworkPolicy is namespaced (local)
- Certain CNIs provide Global variants of Network Policy that can be applied across the cluster
- NetworkPolicy cannot do more than L3/L4
- NetworkPolicy cannot log security events

Cilium – Network Security

- Cilium is a popular OSS CNI (with an Enterprise Option)
- Leverages BPF for Network Security, Routing and Observability
- Adds several features including:
  - Encryption
  - UI and Filter Panel
  - L7 Network Policy Control

Layer 7 Policy Example

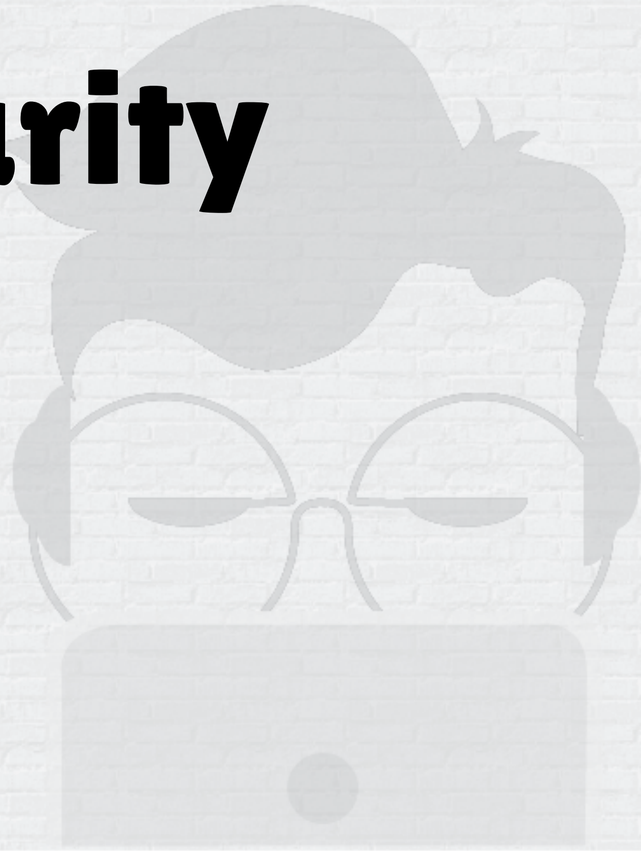


```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
description: "L7 policy to restrict access to specific HTTP call"
metadata:
  name: "API Firewall"
spec:
  endpointSelector:
    matchLabels:
      type: l7-test
  ingress:
  - fromEndpoints:
    - matchLabels:
        org: api-pod
    toPorts:
    - ports:
      - port: "8080"
        protocol: TCP
      rules:
        http:
        - method: "GET"
          path: "/admin"
```

Database Security Policy

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
description: "Allow only permitted requests to empire Cassandra server"
metadata:
  name: "secure-empire-cassandra"
specs:
  - endpointSelector:
      matchLabels:
        app: cass-server
    ingress:
    - fromEndpoints:
      - matchLabels:
          app: empire-outpost
      toPorts:
      - ports:
        - port: "9042"
          protocol: TCP
        rules:
          l7proto: cassandra
          l7:
          - query_action: "select"
            query_table: "system\\..*"
          - query_action: "select"
            query_table: "system_schema\\..*"
          - query_action: "insert"
            query_table: "attendance.daily_records"
    - fromEndpoints:
      - matchLabels:
          app: empire-hq
      toPorts:
      - ports:
        - port: "9042"
          protocol: TCP
        rules:
          l7proto: cassandra
          l7:
          - {}
```

PaC – Workload Security



abhaybhargav

PaC Applicability

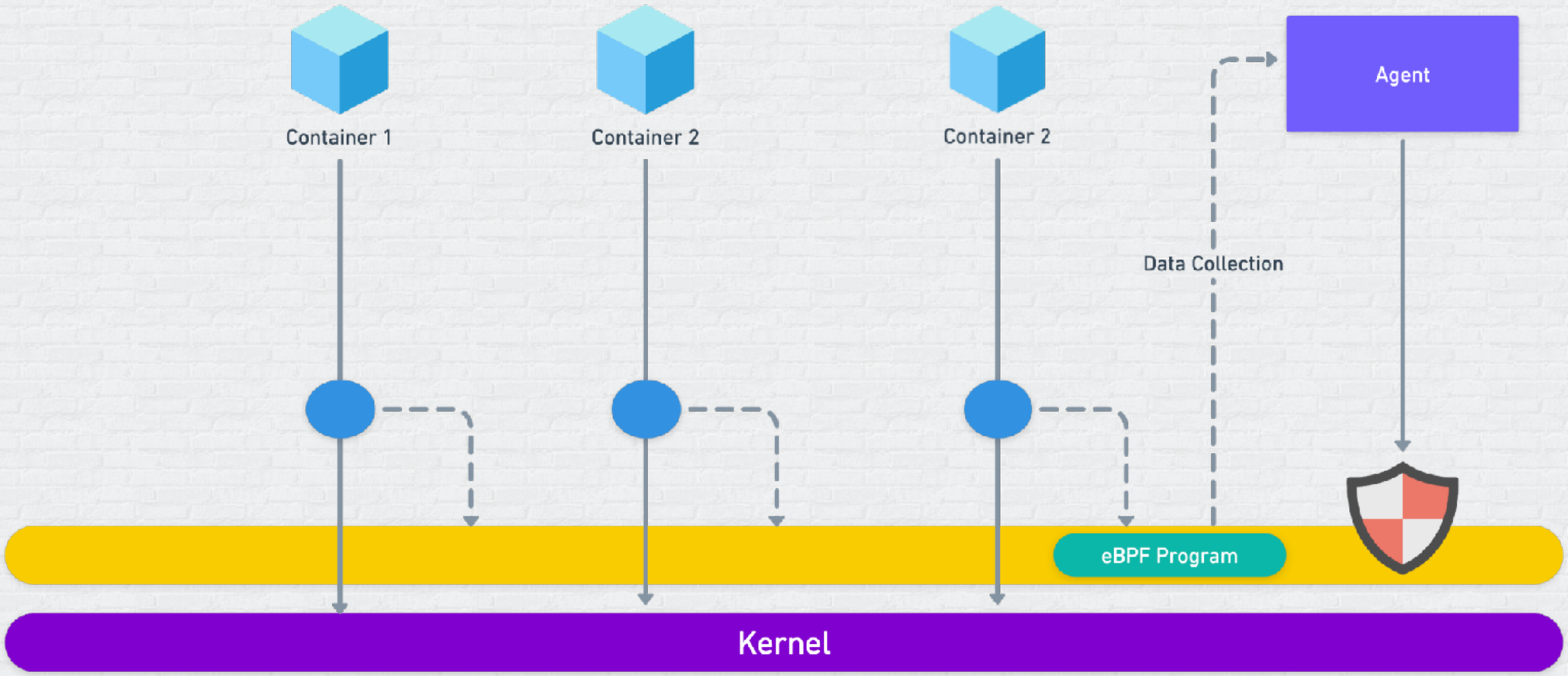
- PaC - Used to compose eBPF Signatures, LSM Configurations to protect workloads
- PaC Frameworks - help you abstract from the complex details

Tools and Frameworks

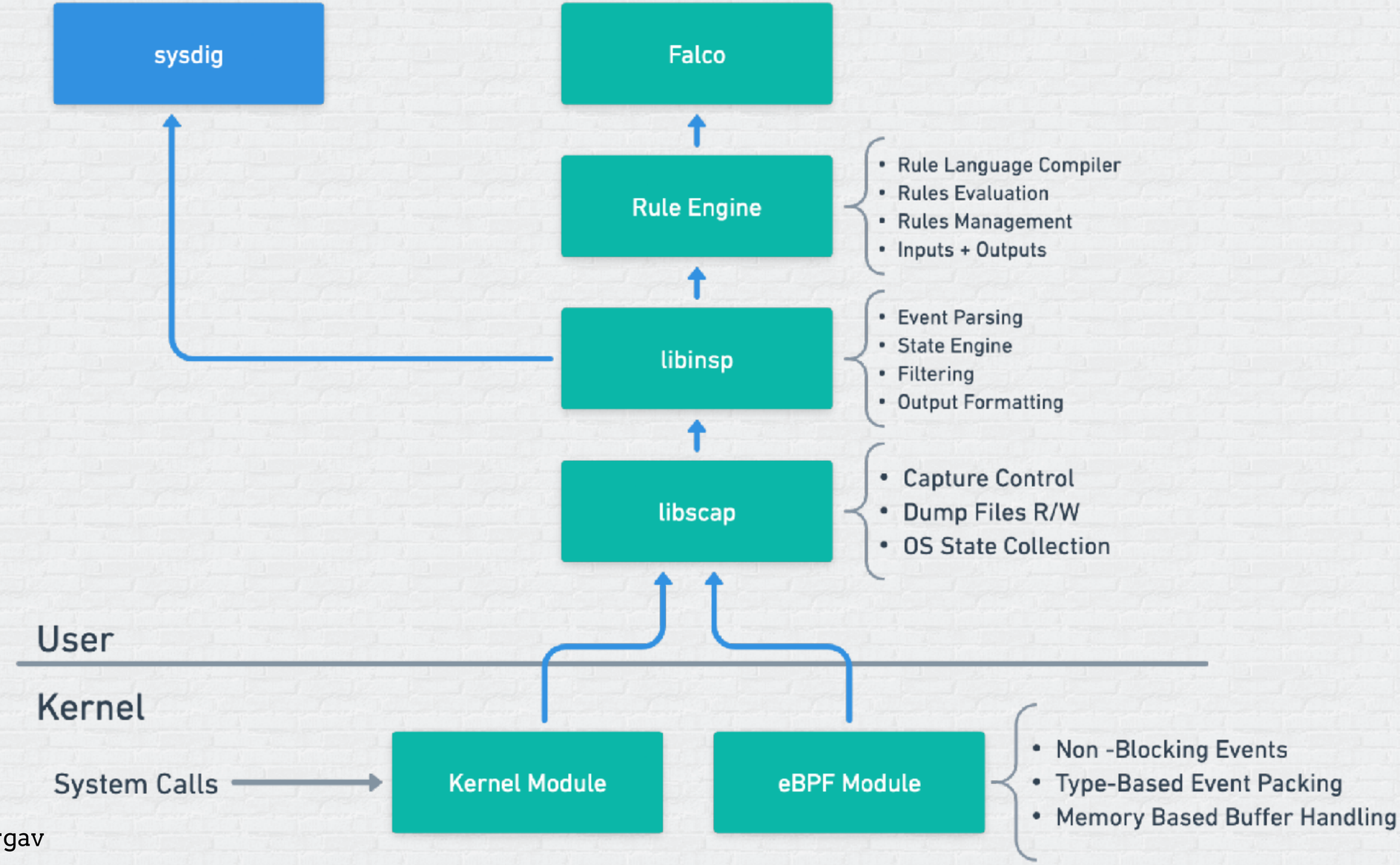
Monitoring Security Anomalies with Falco

abhaybhargav

How Falco works



Falco Under the hood



abhaybhargav

Falco Rules

```
- macro: open_read
  condition: (evt.type=open or evt.type=openat) and evt.is_open_read=true and fd.typechar='f' and
fd.num>=0
- list: sensitive_file_names
  items: [/etc/secret]
- rule: Secret file is accessed
  desc: >
    secret
condition: >
    fd.name in (sensitive_file_names) and open_read
output: "Sensitive Secret file accessed"
  priority: CRITICAL
  tags: [filesystem, secret]
```