styra

Open Policy Agent

OPA: The cloud-native policy engine

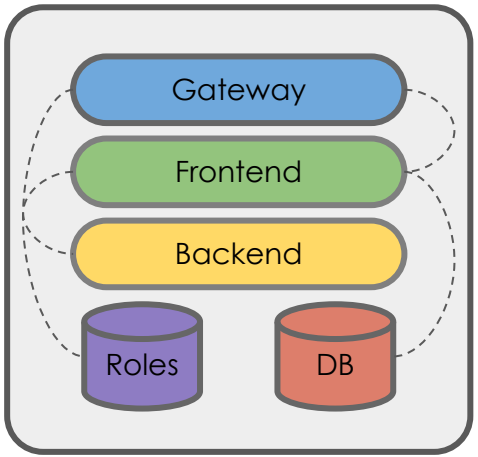
Intro to Open Policy Agent and Policy-as-code

Tim Hinrichs
CTO, co-founder Styra
co-creator OPA
@tlhinrichs

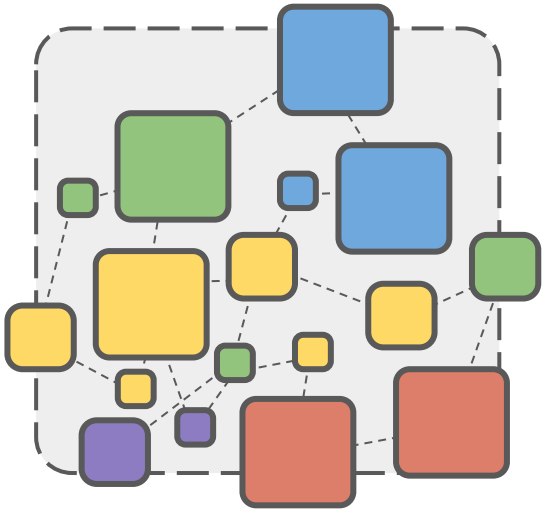
Software and therefore Security is Changing

- Monoliths
- Siloed Roles
- Manual
- Hardware

YESTERDAY



TOMORROW



- Microservices
- Dev(Sec)Ops
- Automation
- Self-service
- CICD
- Cloud

Impact on Security

- APIs → 10x larger attack surface
- Cloud → Less control over network / datacenter
- CICD/Automation → Dynamic environments
- DevOps/Self-service → 10x more people controlling production

Authorization is More Important than Ever

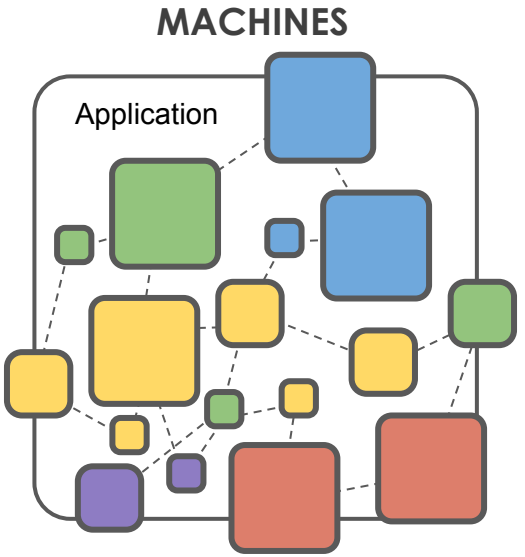
AUTHORIZATION

Which

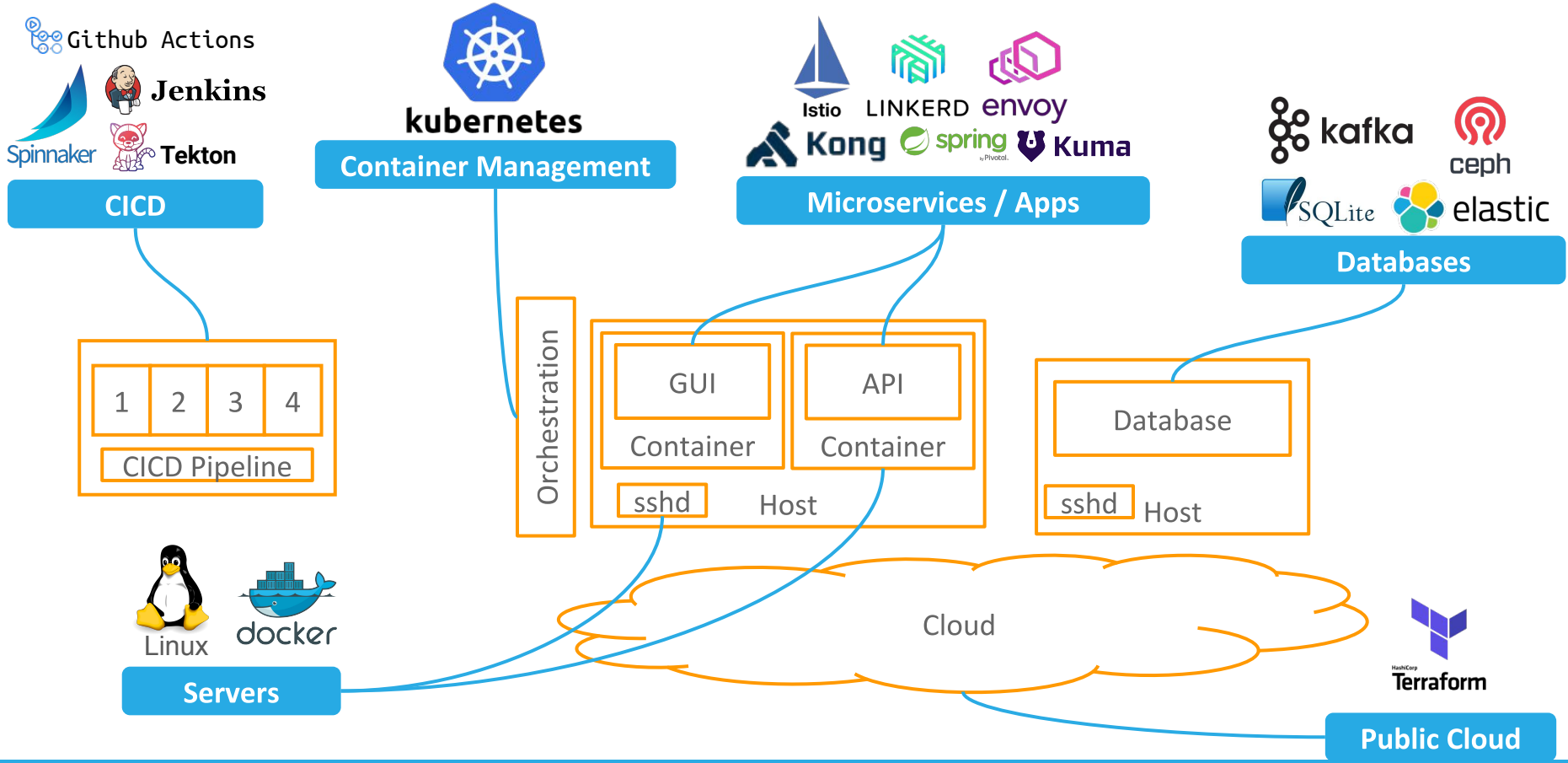
- **people/machines** can perform which
- **actions (APIs)** on which
- **software** in which
- **environments**

REQUIREMENTS

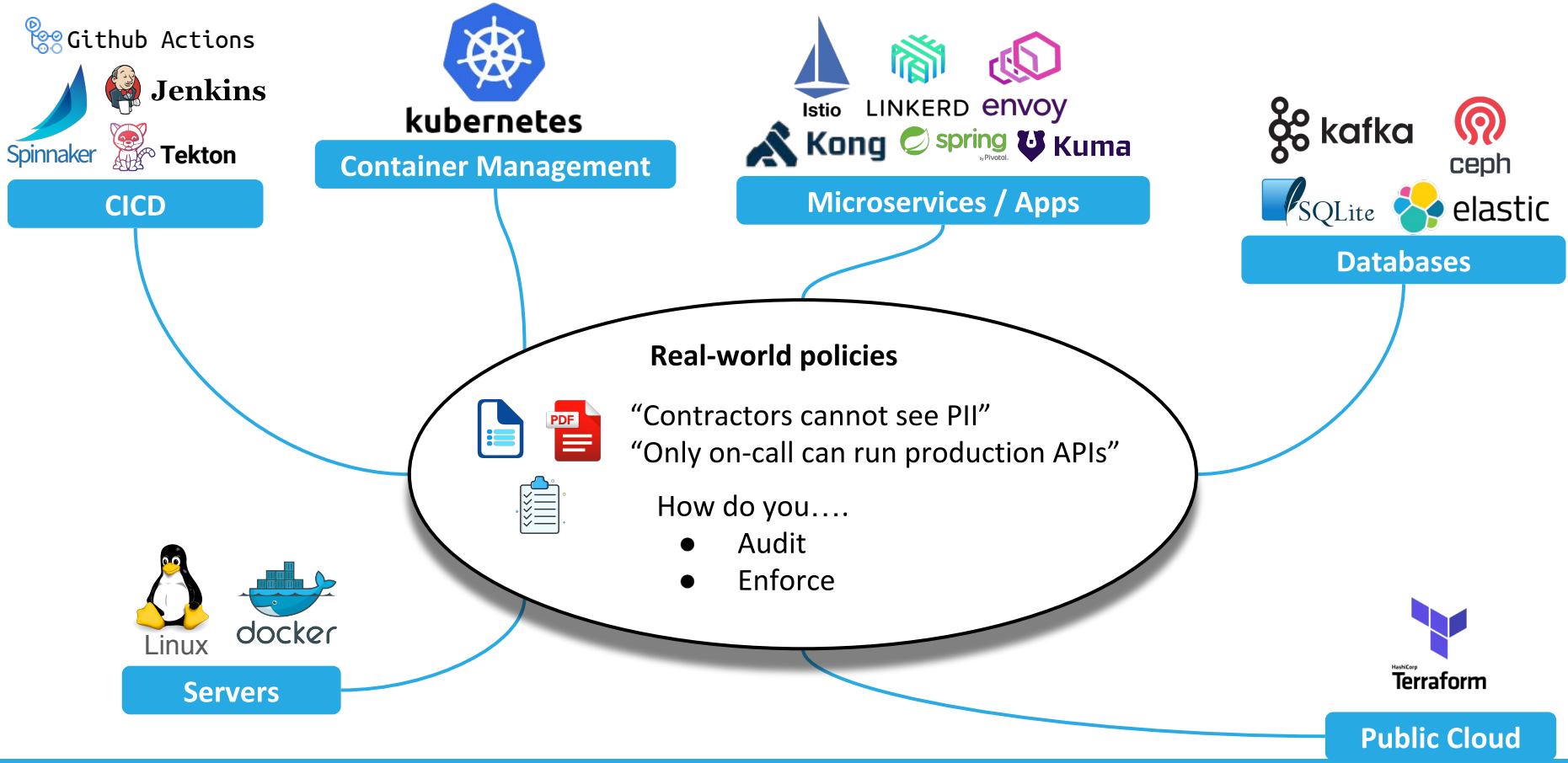
- **Sophisticated policies**: Space of decisions is larger than ever
- **Pervasive enforcement**: Can no longer rely on the environment to enforce authorization
- **Policy lifecycle**: Lifecycle of policy should be woven into the lifecycle of software



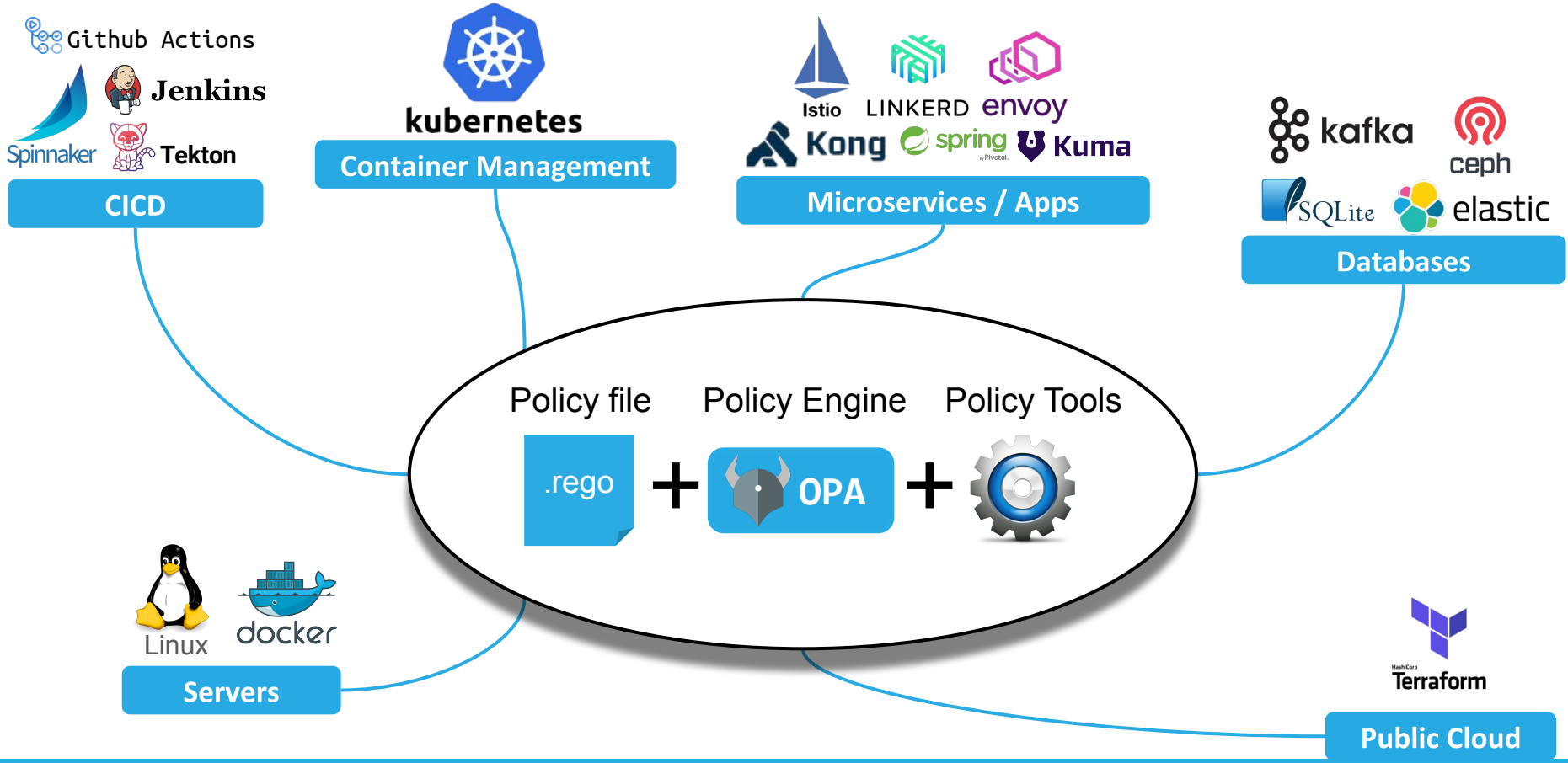
Authorization Happens Everywhere within Cloud-native



Yesterday: Train people || Hardcode policy || Many config languages



Open Policy Agent: Unified Authorization for Cloud-Native



OPA is ...

Open Policy Agent

General purpose policy-as-code solution designed for cloud-native.

1. Define authz policy
2. Evaluate any relevant context
3. Provide decision for enforcement

DECLARATIVE
Desired-state works for DevOps

DECOUPLED
Easy for change management

PORTABLE
Write once, enforce across the stack

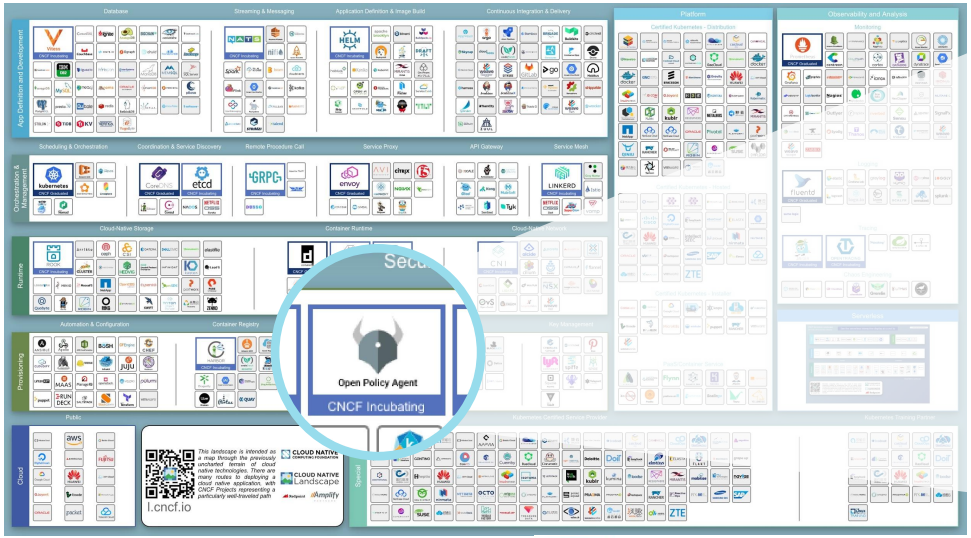
OPEN
Community supports & builds connectors for ubiquity

CONTEXT-AWARE
Business data used with policy

Community

Open Policy Agent: A CNCF Project

Founded by Styra (2016) / Sandbox (2018) / Incubating (2019)



CLOUD NATIVE
COMPUTING FOUNDATION

Open Policy Agent (OPA)

Cloud-native policy engine

Contributors: 30+ companies, 150+ devs

Users: Netflix, Chef, Medallia, Atlassian, Cloudflare, Pinterest, Intuit, Capital One, ABN AMRO, Goldman Sachs ...and more.

Recent End-user Presentations

- ABN AMRO
- Adobe
- Microsoft
- Goldman Sachs
- Google
- Yelp
- and more

Sessions at KubeCon US 2019

- Yelp - How Yelp moved security from the app to the mesh
- Google - Enforcing service mesh structure using OPA
- Goldman Sachs - K8s policy enforcement using OPA at Goldman Sachs
- Snyk - Applying policy throughout the app lifecycle with OPA
- Reddit - Kubernetes at Reddit: Tales from Production
- Adobe - What Makes A Good Multi Tenant Kubernetes Solution
- Giant Swarm - Using OPA for complex CRD Validation and Defaulting

OPA Summit at KubeCon US 2019

- Capital One - Open Policy Agent for Policy-enabled Kubernetes and CICD
- Chef - Open Policy Agent in Practice: From Angular to OPA in Chef Automate
- Pinterest - Open Policy Agent at Scale: How Pinterest Manages Policy Distribution
- Tripadvisor - Building a Testing Framework for Integrating Open Policy Agent into Kubernetes
- Atlassian - Deploying Open Policy Agent at Atlassian

Sessions at Virtual KubeCon EU 2020

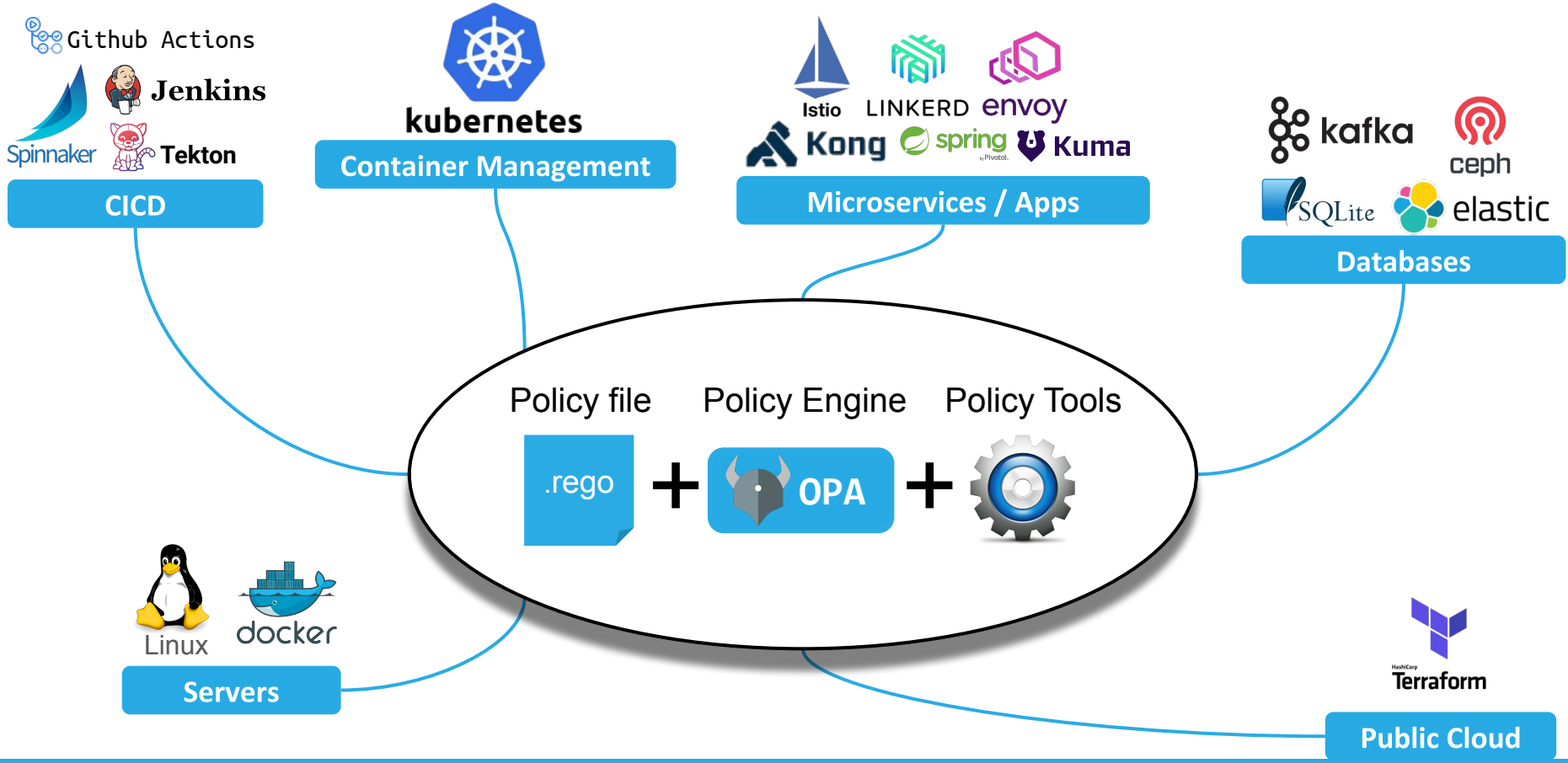
- AquaSecurity: Handling Container Vulnerabilities with Open Policy Agent
- ABN AMRO: How ABN AMRO Switched Cloud Providers Without Anyone Noticing
- Medudoc: Securing Your Healthcare Data with OPA

Other events or public confirmation of using OPA: Bank of New York Mellon, AWS, Synemedia, Pure Storage, VMware, Netflix, Daimler, T-Mobile, Salesforce

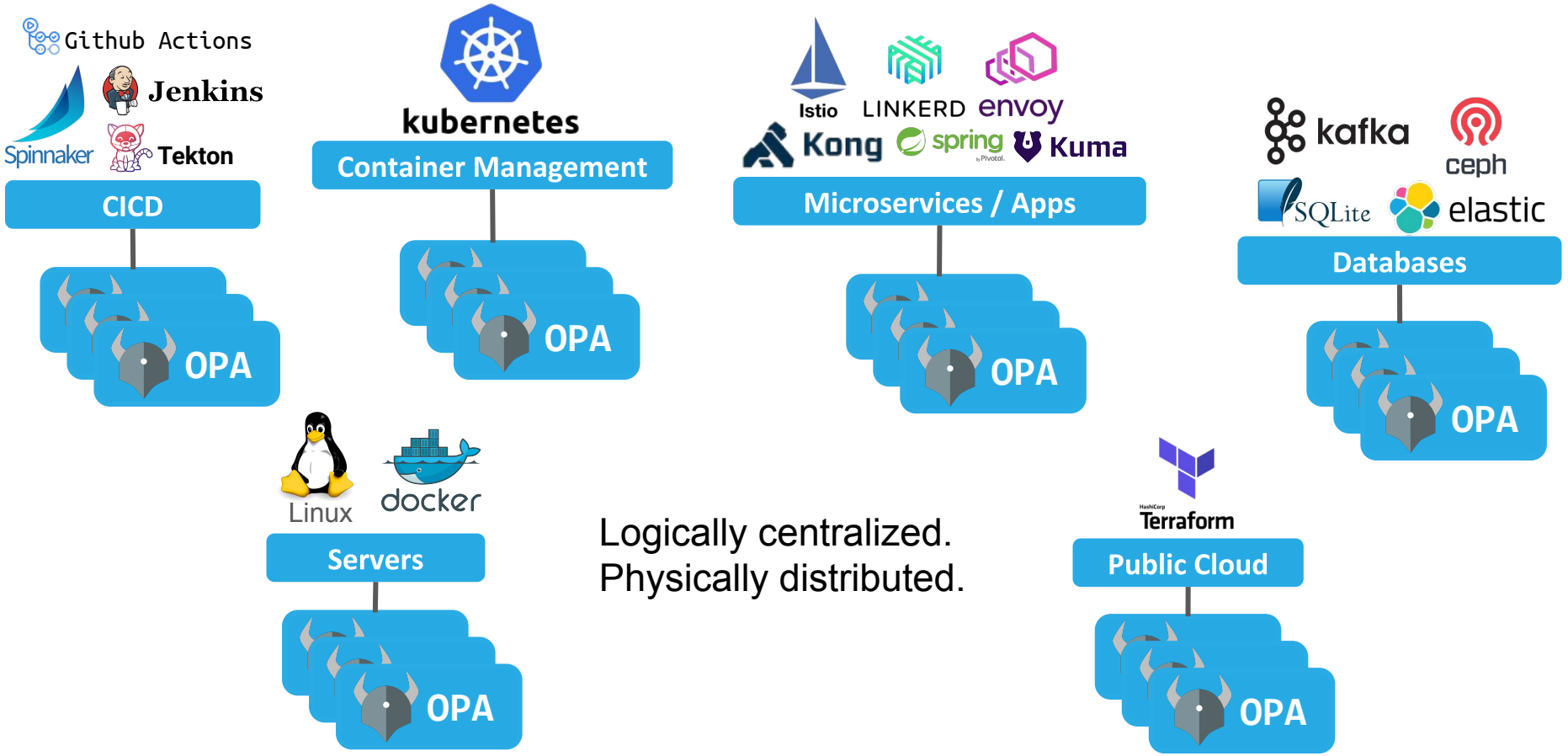
- Atlassian
- TripAdvisor
- Pinterest
- Chef
- Capital One
- and more

Architecture

Open Policy Agent: Unified Authorization for Cloud-Native



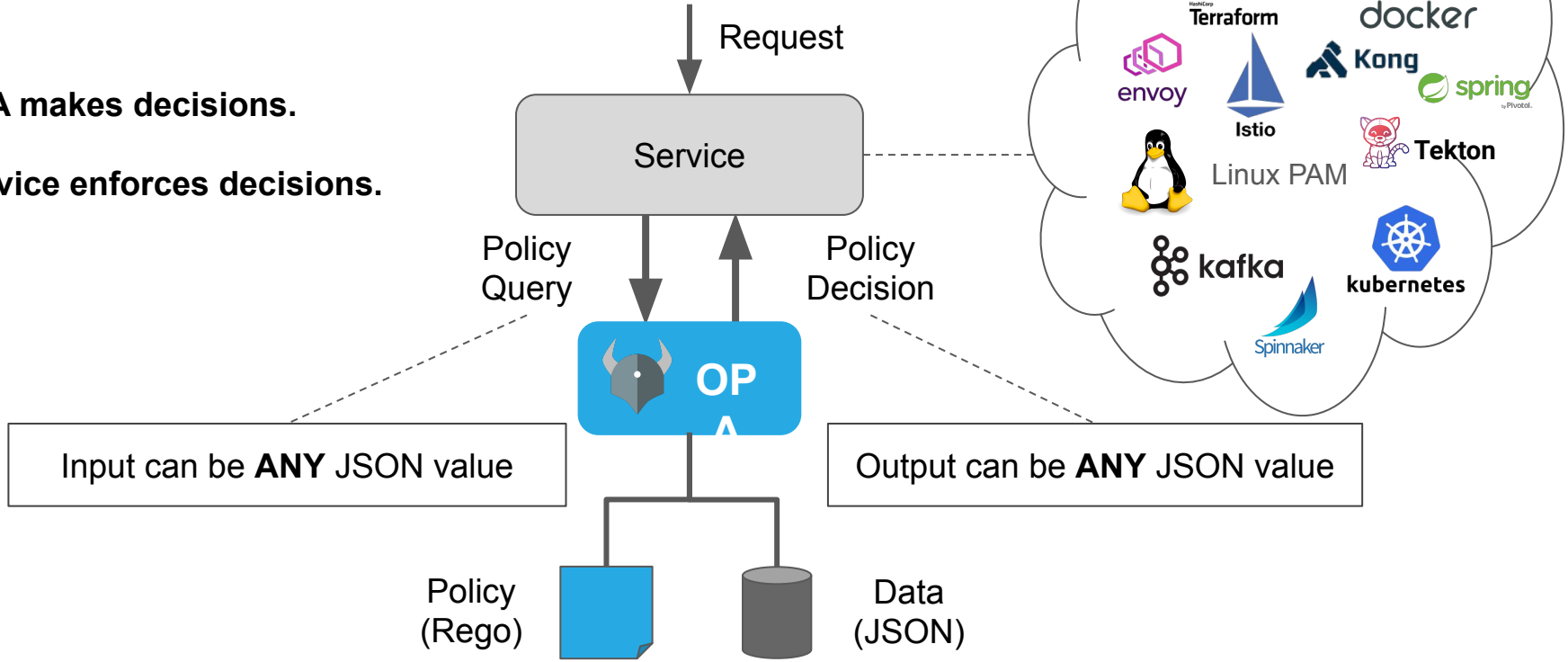
Open Policy Agent: Distributed Authorization for Availability and Perf



OPA: General-purpose Policy Engine

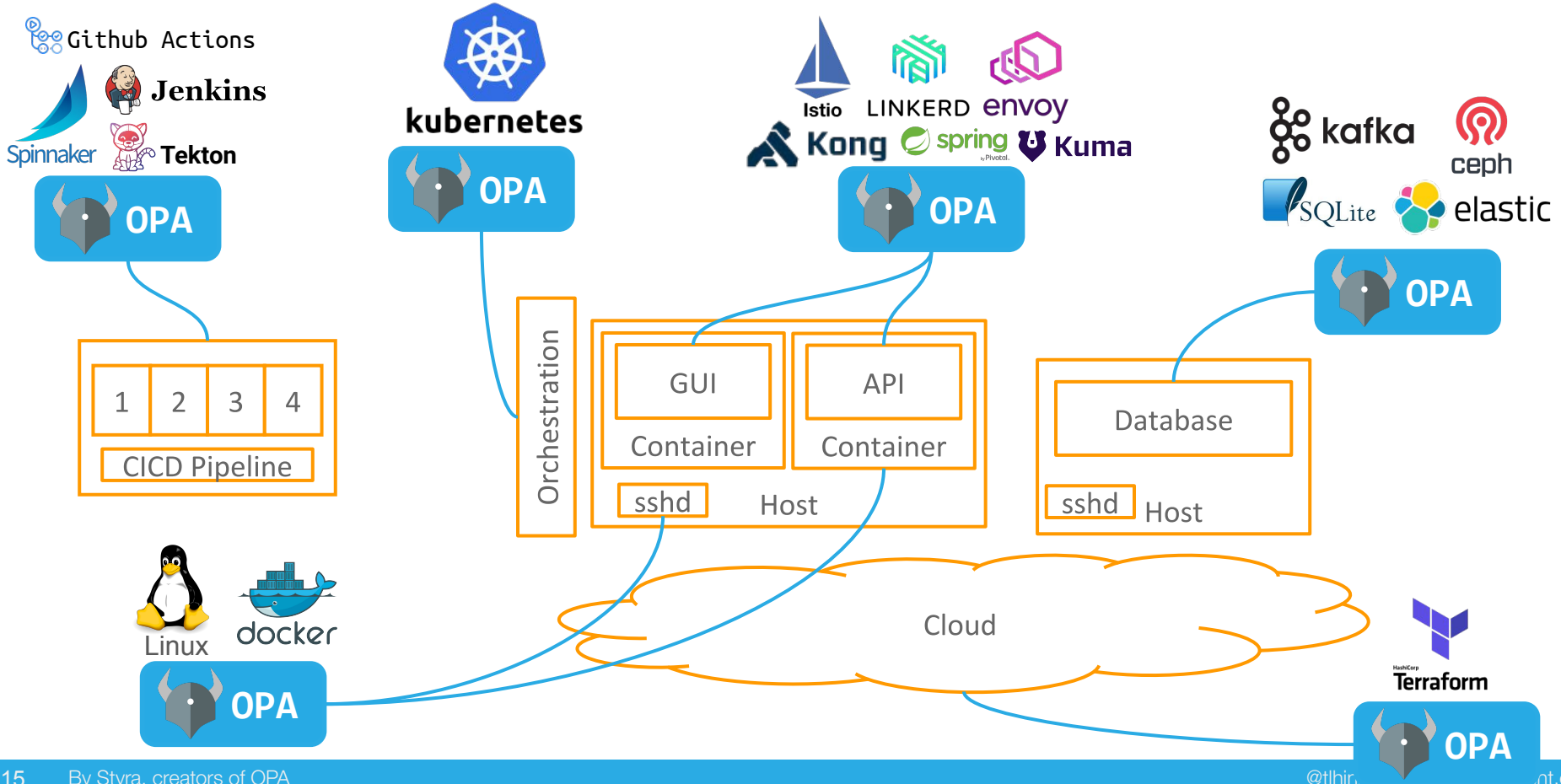
OPA makes decisions.

Service enforces decisions.

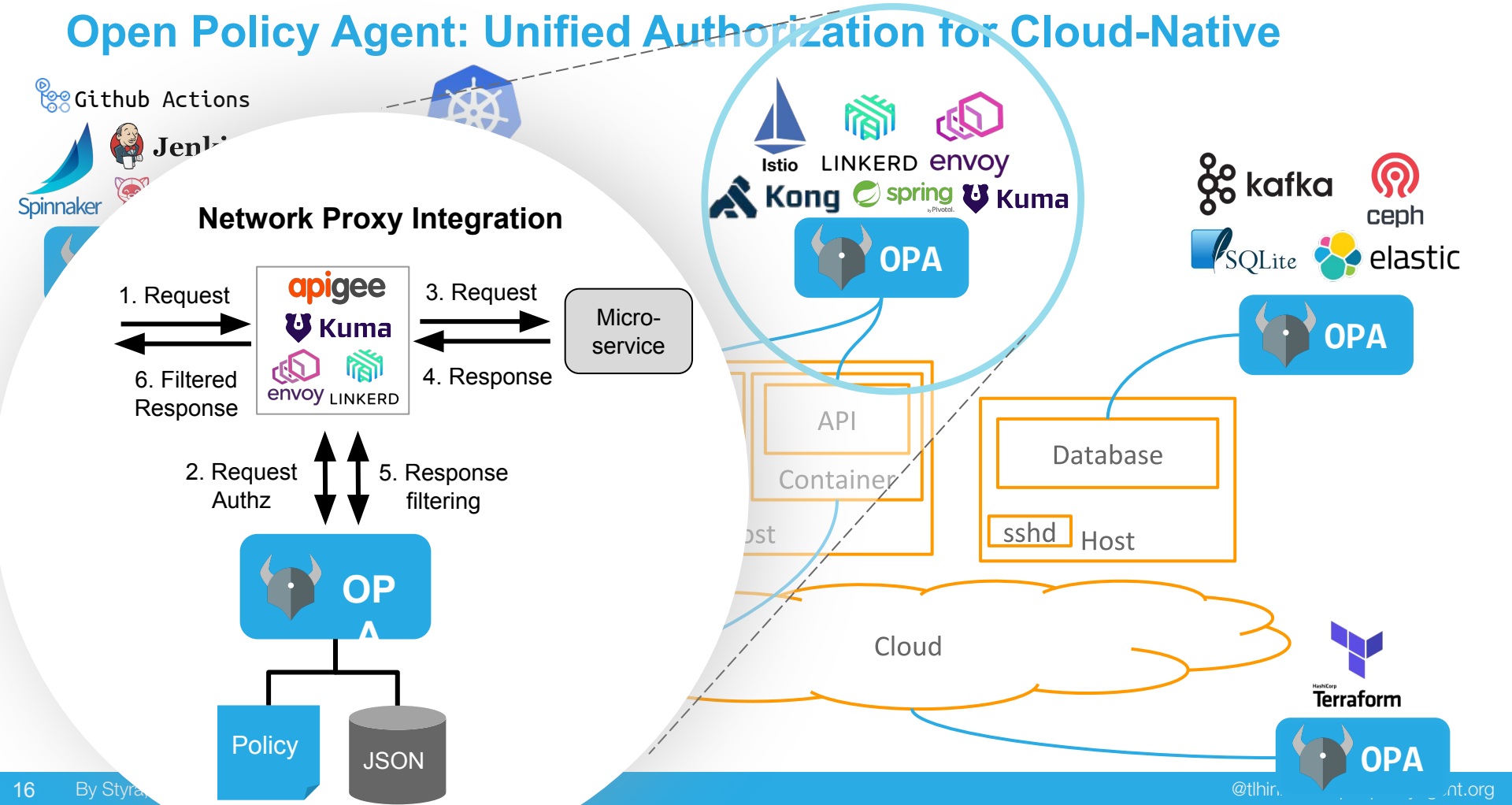


OPA runs as: Library (Go), WASM, Sidecar, Daemon, or Service

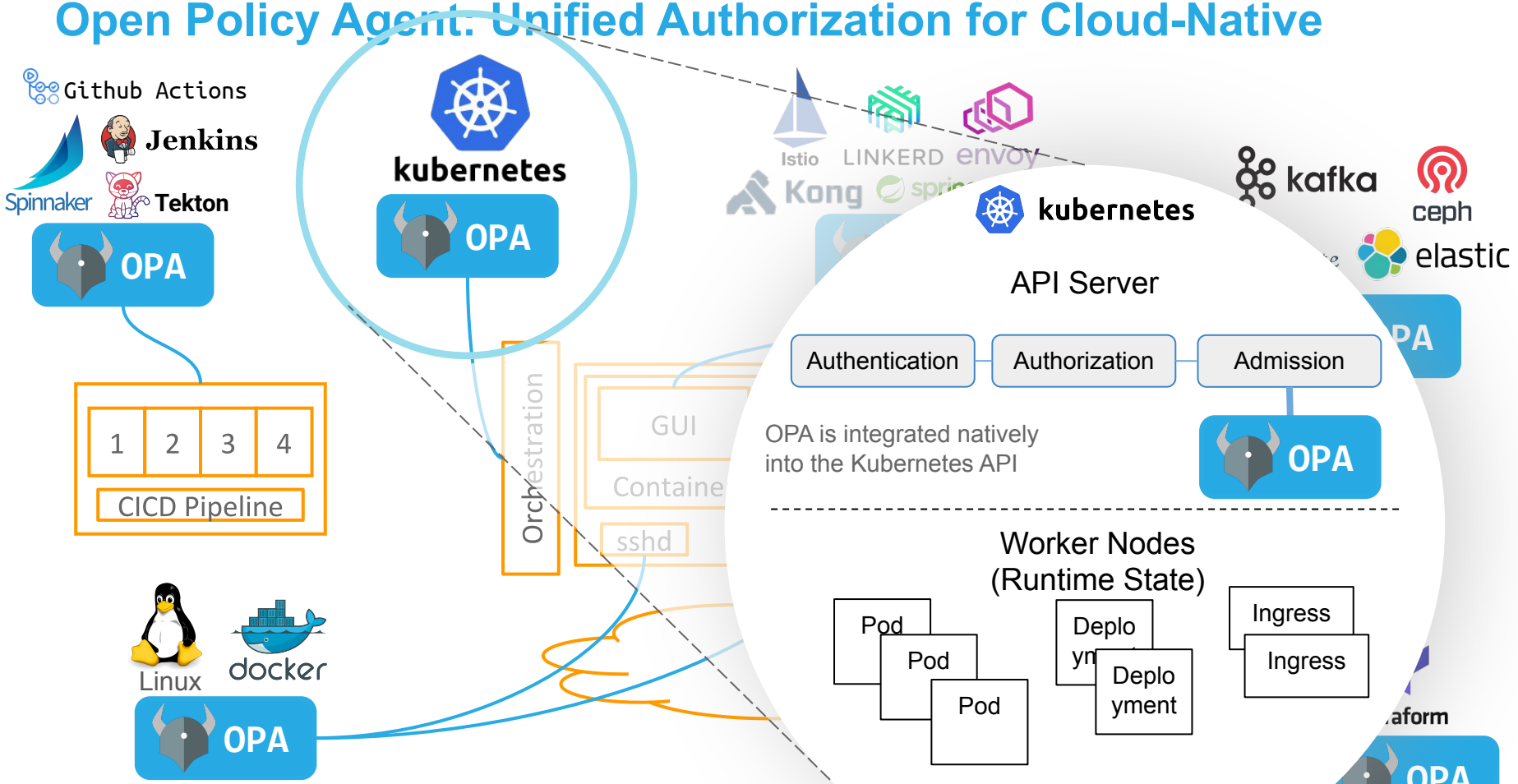
Open Policy Agent: Unified Authorization for Cloud-Native



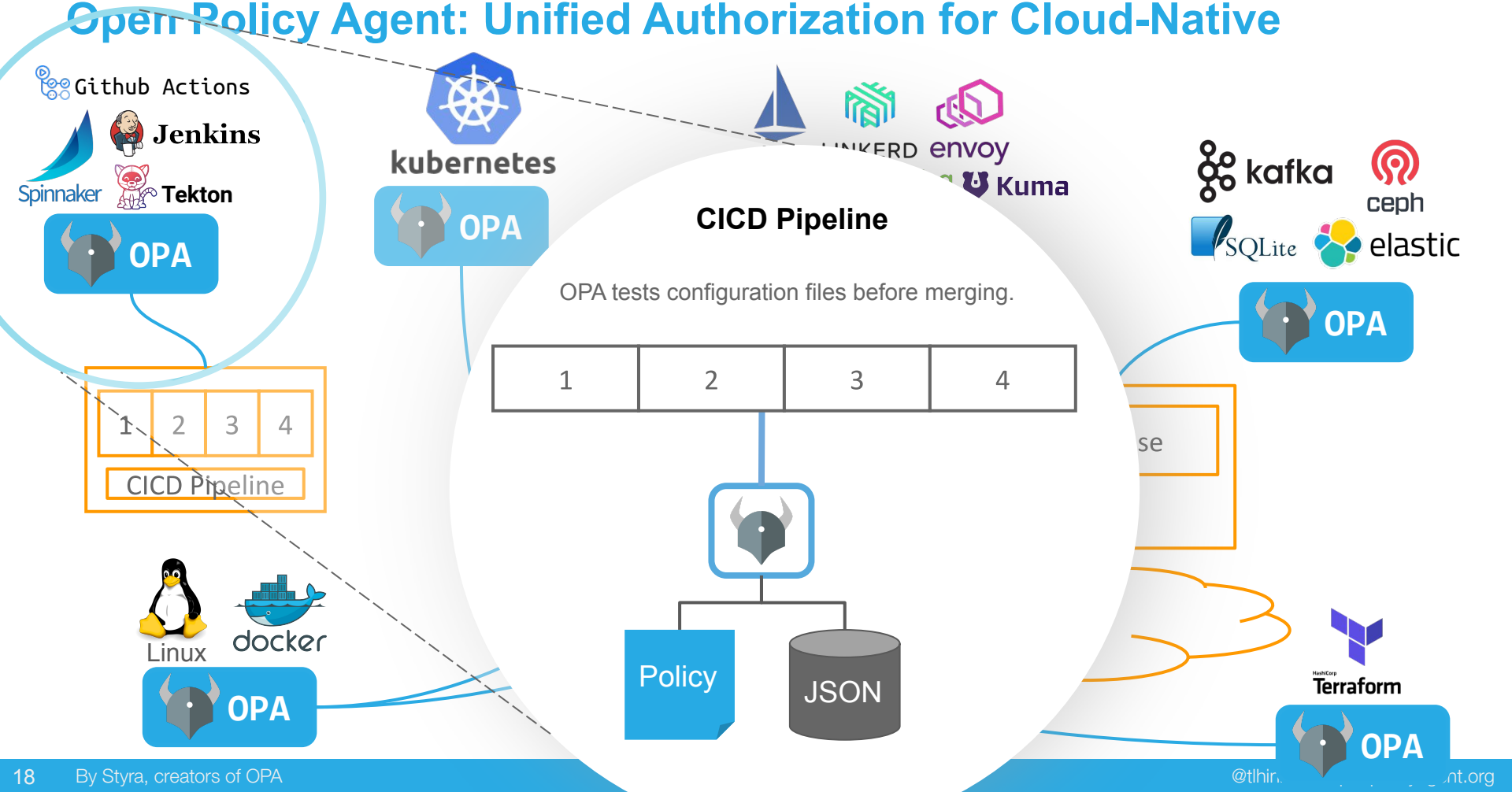
Open Policy Agent: Unified Authorization for Cloud-Native



Open Policy Agent: Unified Authorization for Cloud-Native

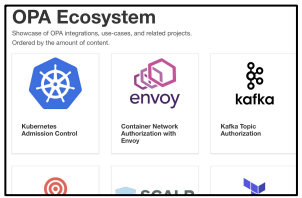


Open Policy Agent: Unified Authorization for Cloud-Native



Policy and Tooling

Policy Development Life Cycle and OPA Tooling

	Requirements	Author Policy	CI	Deploy	Monitor	Log
Stages	Choose domain Assemble real-world policy (spreadsheet) Understand data dependencies Choose enforcement points (PEPs)	Write Rego Decide/learn input and decision schemas Modularize policy and delegate for collaboration	Assemble policy Test policy Create policy build artifacts	Deploy policy to OPA Deploy / refresh data to OPA	OPA health	Record decisions for audit
OPA Tooling	 40 integrations	IDEs REPL Playground VScode IntelliJ	Tools Unit test Profile Benchmark Build	APIs Bundle API Push API Pull builtins JWT builtins	APIs Status API Prometheus	APIs Log API

Design Principles of Rego, OPA's Policy Language

Syntax

Mirror declarative real-world policies

99% of Rego statements are IF statements, like those found in PDF/email policies

```
allow { user == "alice" }    # allow if user is alice
```

Semantics

Embrace hierarchical data

Rego provides first-class support for deeply nested data and 130+ common builtins

```
input.token.claims[i].id
```

Algorithms

Optimize Performance Automatically

Policy author is responsible for correctness. OPA is responsible for performance.

Rego Overview

When writing Rego you do two things:

1. Write Rules that make policy decisions. A Rule is a conditional assignment.

Assignment	IF	Conditions
allow is true	IF	user is alice and action is read

```
allow = true        { user == "alice"; action == "read" }
```

IF

- Value assigned to a variable
- Element assigned to a set
- Key assigned to a value
- Function call assigned to a result

- Variable assignment
- Reference, e.g. input.user
- Equality or inequality
- Function call, iteration, ...

2. Organize Rules into Policies (*packages*). A Policy is a set of Rules with a hierarchical name.

```
package foo

rule1
rule2
rule3
```

```
package foo.bar

rule1
rule2
```

```
package baz

rule1
rule2
rule3
```

Envoy Policy Example

OPA Playground

JSON/YAML from Envoy

```
parsed_path: [“api”, “v1”, “products”]
attributes:
 source:
  address:
   Address:
    SocketAddress:
     address: "172.17.0.10"
     PortSpecifier:
      PortValue: 36472
 destination:
  address:
   Address:
    SocketAddress:
     address: "172.17.0.17"
     PortSpecifier:
      PortValue: 9080
 request:
  http:
   id: 13359530607844510314
   method: GET
   headers: ...
   path: "/api/v1/products"
   host: "192.168.99.100:31380"
   protocol: "HTTP/1.1"
```

OPA Policy: Allow all GET and some PUT

```
package envoy.authz

# everyone can GET /
allow {
    input.attributes.request.http.method == "GET"
    input.parsed_path = ["/"]
}

# updates to /v1/admin/{id} dependent on source IP
allow {
    input.attributes.request.http.method == "PUT"
    input.parsed_path = ["v1", "admin", id]
    user_is_admin
    src := input.attributes.source.address.Address.SocketAddress.address
    net.cidr_contains("172.28.0.0/16", src)
}

user_is_admin { ... }
```

Kubernetes Policy Example

[OPA Playground]

JSON/YAML from Kubernetes

```
apiVersion: admission.k8s.io/v1beta1
kind: AdmissionReview
request:
  kind:
    group: ''
    kind: Pod
    version: v1
  namespace: opa
  object:
    metadata:
      labels:
        app: nginx
      name: nginx
      namespace: opa
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        volumeMounts:
        - mountPath: "/var/run/serviceaccount"
          name: default-token-tm9v8
          readOnly: true
  operation: CREATE
```

OPA Policy: All images come from a trusted registry

```
package kubernetes.admission

deny[msg] {
    input.request.kind.kind == "Pod"
    some i
    image := input.request.object.spec.containers[i].image
    not startswith(image, "hooli.com/")
    msg := sprintf("image comes from bad registry: %v", [image])
}
```

Join Us!

Tim Hinrichs
@tlhinrichs

Styra
styra.com
@styrainc

Open Policy Agent
openpolicyagent.org
@openpolicyagent