labyrinth labs

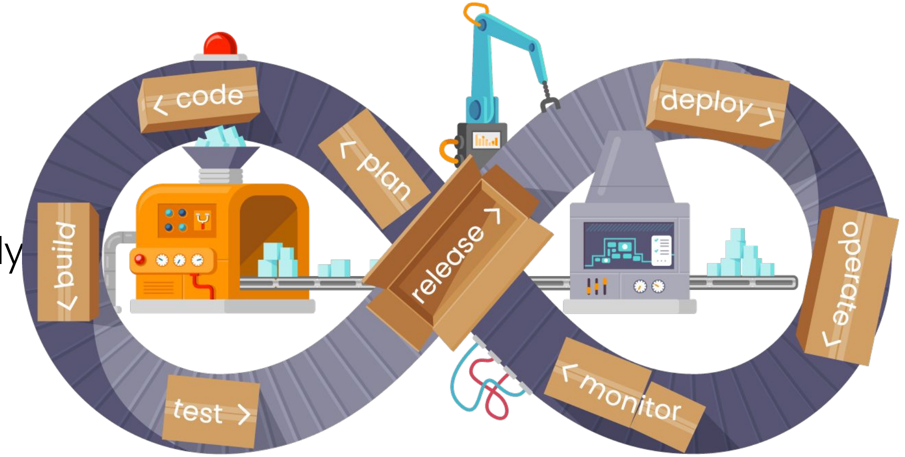
Open Policy Agent Introduction

About us

Labyrinth Labs is a boutique expert shop focusing on **DevOps, Public Cloud** and **Kubernetes** ecosystem.

Our **mission** is to help companies evolve their applications into the **containerized** and **serverless** future.

We are a privately owned company mainly driven by **technical excellence** and our **passion** in modern application **technologies**.



Agenda

- Introduction
- Policy decision model
- Interactions with OPA
- Rego basics
- Performance
- Ecosystem
- Gatekeeper example

Open Policy Agent Introduction

OPA - Overview

- General purpose policy engine
- Decouples policy from application logic
- Offload policy decision-making from your software to enforce policies
- Toolset and framework for policy across the cloud native stack
- Declarative Policy as a Code with REGO

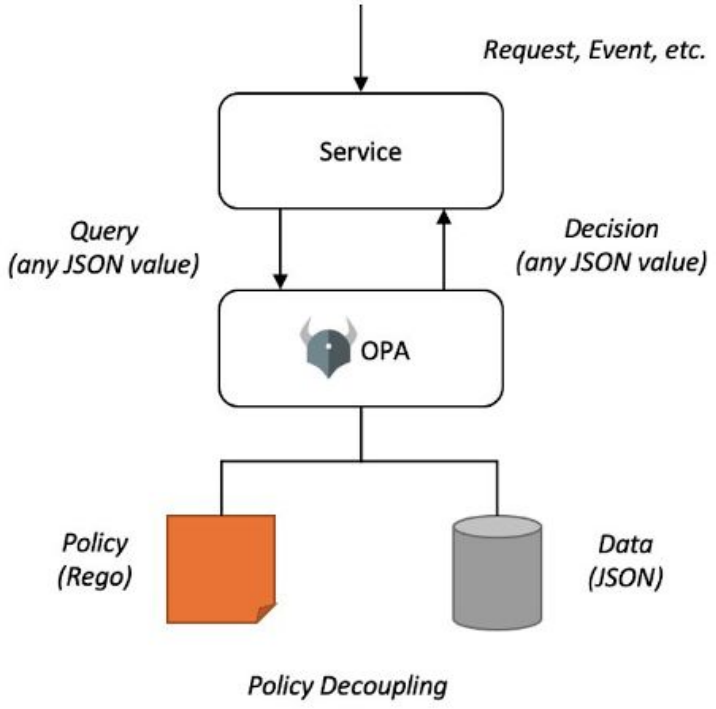
OPA - Community

- CNCF graduated
- 30+ integrations
- Vibrant community
- Good tooling and ecosystems
- Users

Policy Decision Model

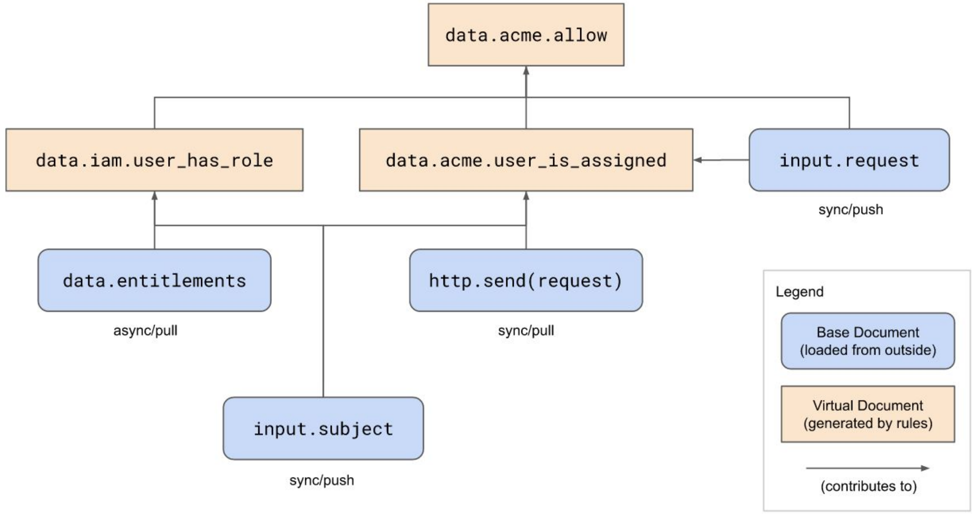
OPA - Operational Model

- Service query OPA with structured data
- Based on policy and data OPA evaluates query
- Returns decision as response with structured data to the service



OPA - Data Model

- Data classification
  - Base documents
  - Virtual documents

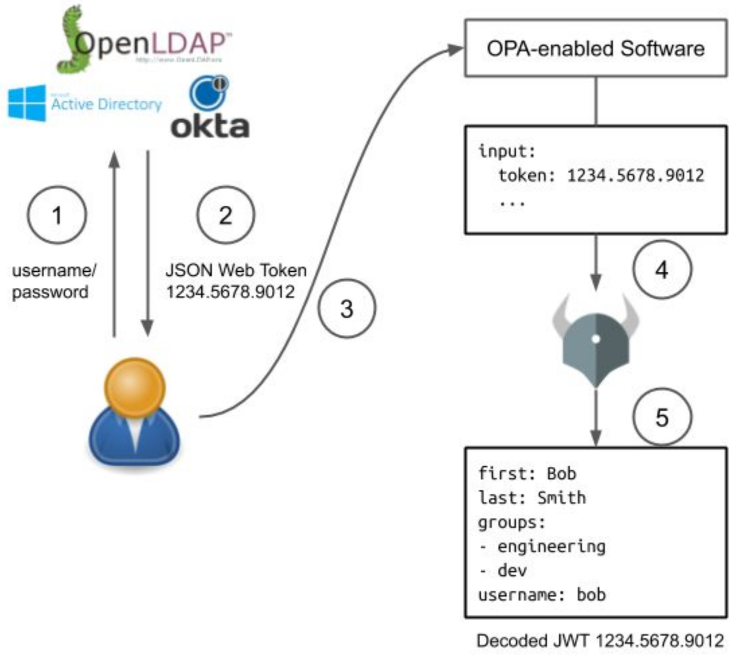


OPA - External Data

- Data size
- Update frequency
- Consistency model
- Loading options:
  - Sync push
  - Sync push overload
  - Periodic pull - Bundle API
  - Async push data
  - *Pull data during evaluation*

OPA - External Data

- *Sync push* - base document
  - updates
  - size
- Usage:
  - User attributes

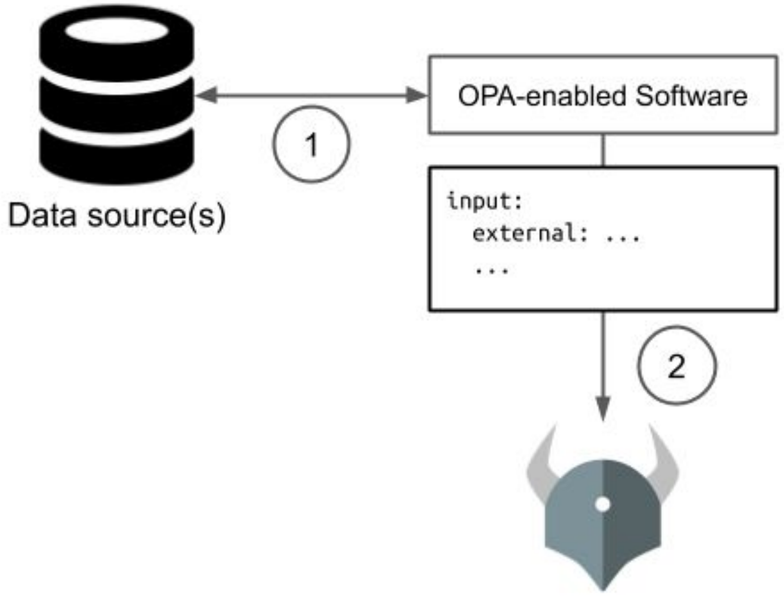


```
input:
  token: 1234.5678.9012
  ...
```

```
first: Bob
last: Smith
groups:
- engineering
- dev
username: bob
```

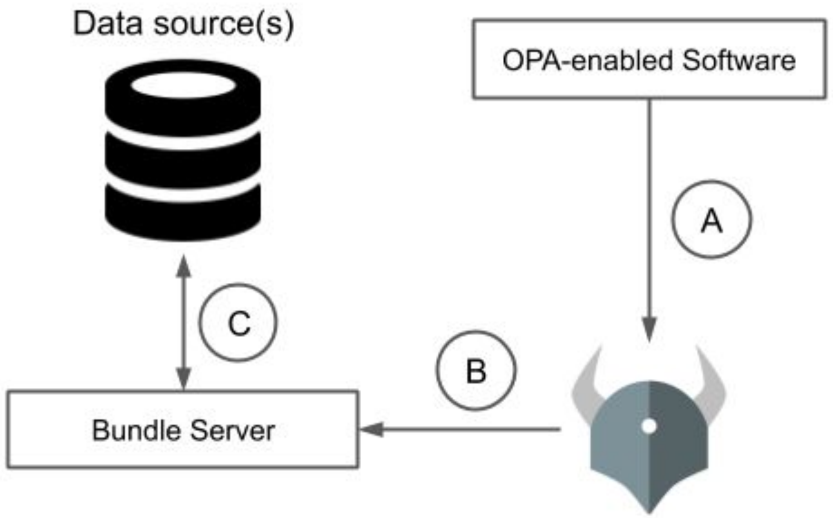
OPA - External Data

- *Sync push overload* - base document
  - updates
  - size
- Usage:
  - Local & Dynamic data



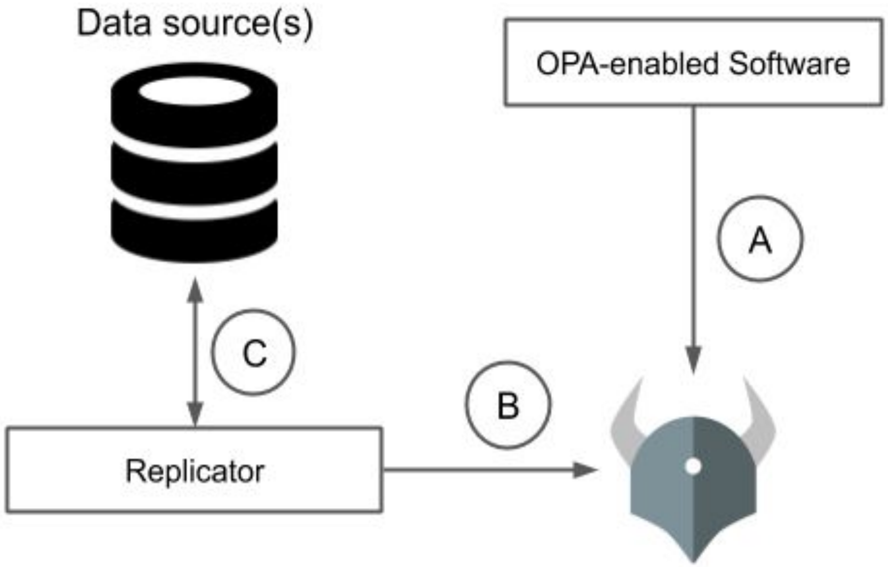
OPA - External Data

- *Bundle API*
  - updates
  - size
  - consistency
  - persistency
- Usage:
  - Static & Medium-sized data



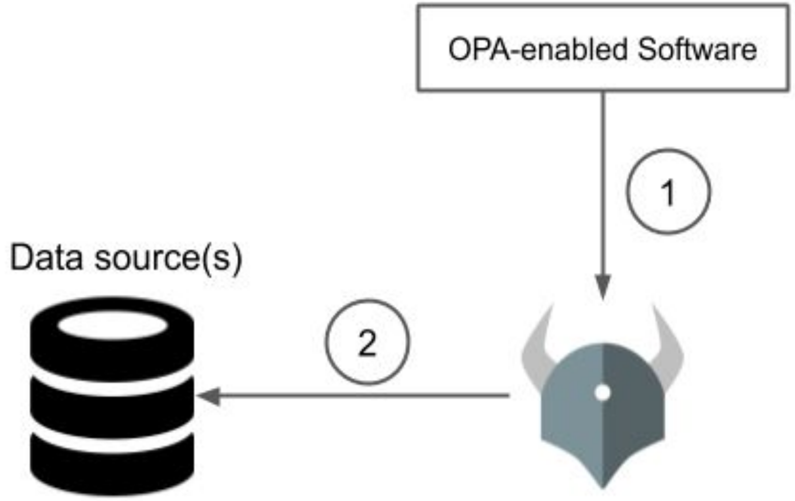
OPA - External Data

- *Push data*
  - updates
  - size
- Usage:
  - Dynamic & Medium-siz data



OPA - External Data

- *Pull data during evaluation*
  - updates
  - size
  - performance
- Usage:
  - Dynamic & Large-sized data



OPA - APIs

- Policy
- Data
- Query
- Compile
- Auth
- Trace
- Metrics
- Config

labyrinth labs

OPA - Policy API

Example Request

```
PUT /v1/policies/example1 HTTP/1.1
Content-Type: text/plain
```

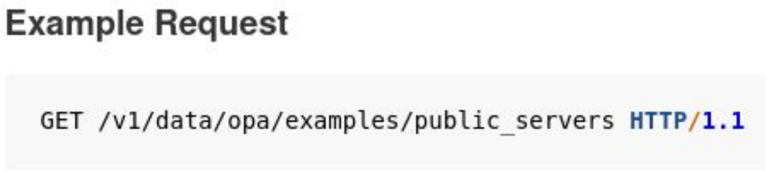
```
package opa.examples

import data.servers
import data.networks
import data.ports

public_servers[server] {
  some k, m
    server := servers[_]
    server.ports[_] == ports[k].id
    ports[k].networks[_] == networks[m].id
    networks[m].public == true
}
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

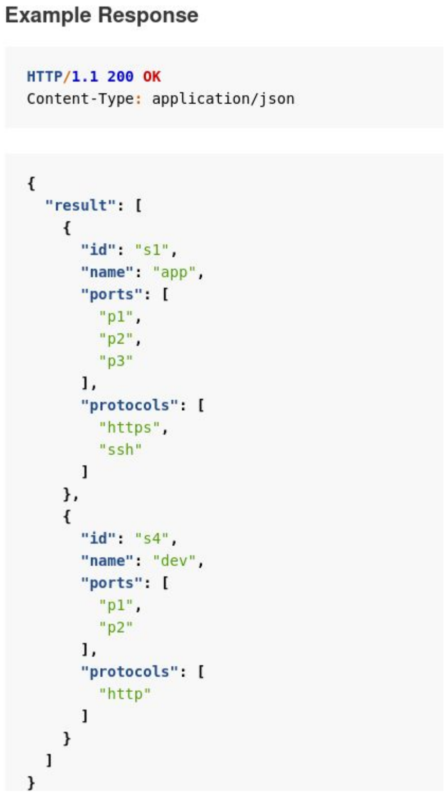
```
{}
```

OPA - Data API



Example Request

```
GET /v1/data/opa/examples/public_servers HTTP/1.1
```



Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "result": [
    {
      "id": "s1",
      "name": "app",
      "ports": [
        "p1",
        "p2",
        "p3"
      ],
      "protocols": [
        "https",
        "ssh"
      ]
    },
    {
      "id": "s4",
      "name": "dev",
      "ports": [
        "p1",
        "p2"
      ],
      "protocols": [
        "http"
      ]
    }
  ]
}
```

OPA - Query API

Policy

```
PUT /v1/policies/example1 HTTP/1.1
Content-Type: text/plain
```

```
package system

main = msg {
  msg := sprintf("hello, %v", input.user)
}
```

Request

```
POST /
Content-Type: application/json
```

```
{
  "user": ["alice"]
}
```

Resposne

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
"hello, alice"
```

POST / == POST /system/main

labyrinth labs

Interactions

OPA - Interactions

- Command line (+interactive mode)
- Library
- Server (HTTP)
    - Sidecar
    - Host-level daemon
    - Distributed service

REGO Language

OPA - REGO Language

- Inspired by Datalog
- Relatively easy for reading / writing
- Declarative

OPA - REGO Example

- Pet Store example with RBAC policy

labyrinth labs

OPA - Rich set of primitives

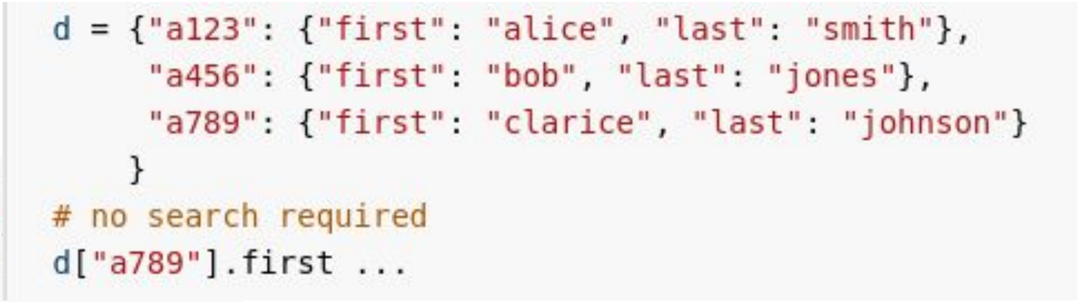
- Arrays / Objects / Sets
  - lookups, iterations
  - concat, …, unions
- Rules, functions
  - conditionals, incremental, else
- Comparisons, Math
- String manipulations, regexp, …
- Many more

labyrinth labs

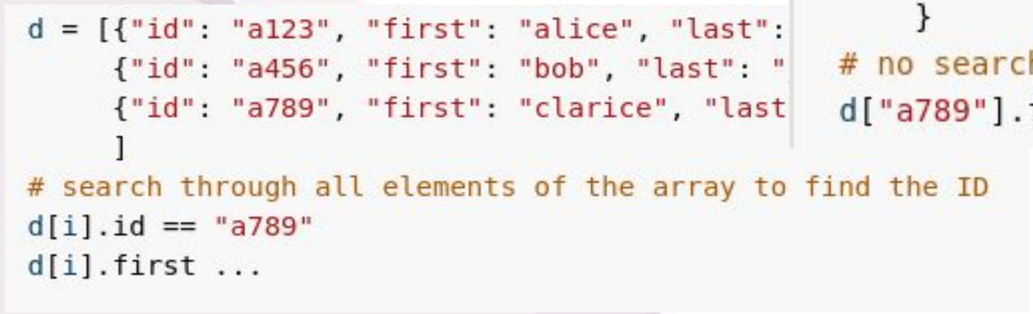
Performance

OPA - Performance

- Evaluation budget order of 1 ms
- Linear fragment
- Use objects over arrays



```
d = {"a123": {"first": "alice", "last": "smith"},
     "a456": {"first": "bob", "last": "jones"},
     "a789": {"first": "clarice", "last": "johnson"}
    }
# no search required
d["a789"].first ...
```



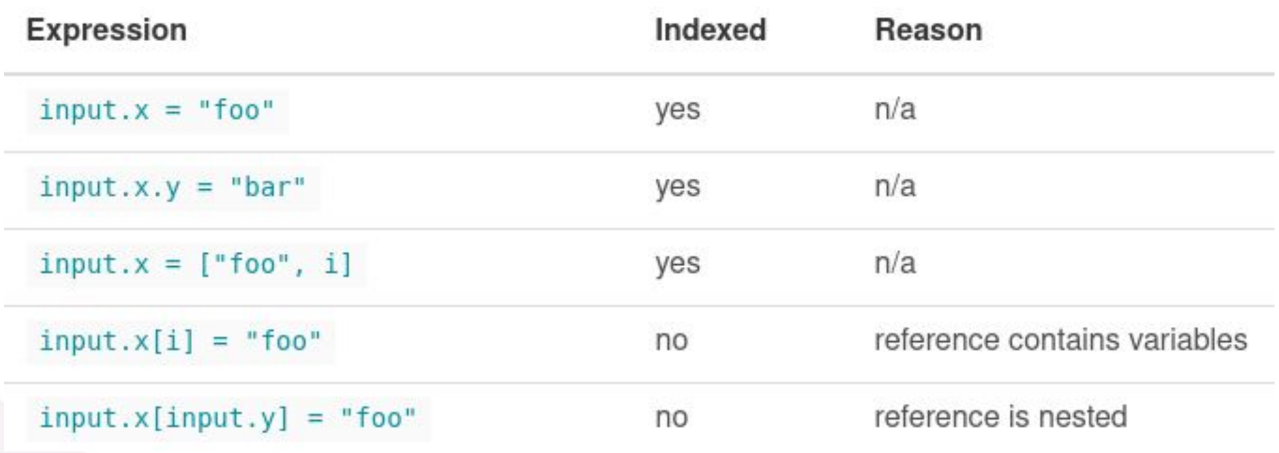
```
d = [{"id": "a123", "first": "alice", "last":
     {"id": "a456", "first": "bob", "last": "
     {"id": "a789", "first": "clarice", "last
     ]
# search through all elements of the array to find the ID
d[i].id == "a789"
d[i].first ...
```

OPA - Performance partial evaluation

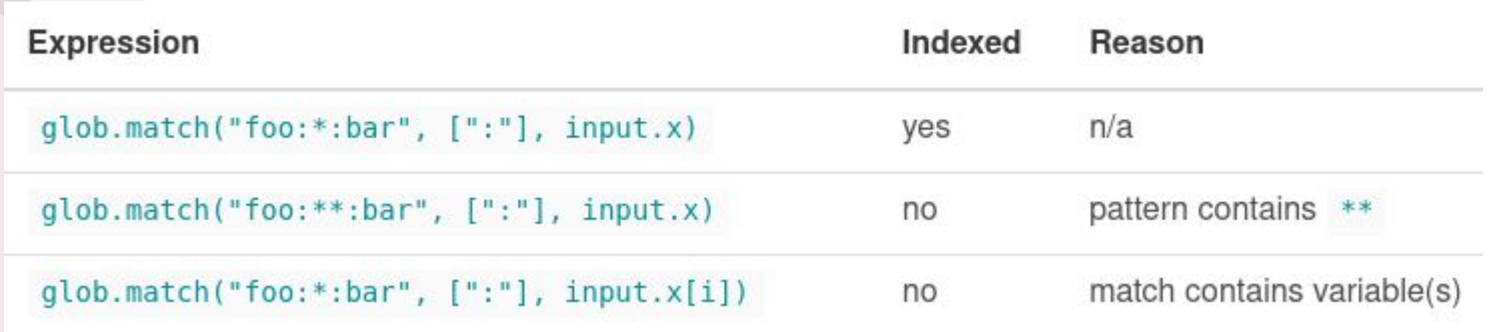
```
allow {
    op = allowed_operations[_]
    input.method = op.method
    input.resource = op.resource
}
```

```
curl localhost:8181/v1/data/smart_home/allow?metrics&partial \
    -d @req.json

{
    “metrics”: {
        “timer_rego_partial_eval_ns”: 10613556,
        “timer_rego_query_compile_ns”: 137427
        “timer_rego_query_eval_ns”: 43921,
        “timer_rego_query_parse_ns”: 200871
    },
    “result”: true
}
```

OPA - Performance indexing statements



Expression	Indexed	Reason
`input.x = "foo"`	yes	n/a
`input.x.y = "bar"`	yes	n/a
`input.x = ["foo", i]`	yes	n/a
`input.x[i] = "foo"`	no	reference contains variables
`input.x[input.y] = "foo"`	no	reference is nested



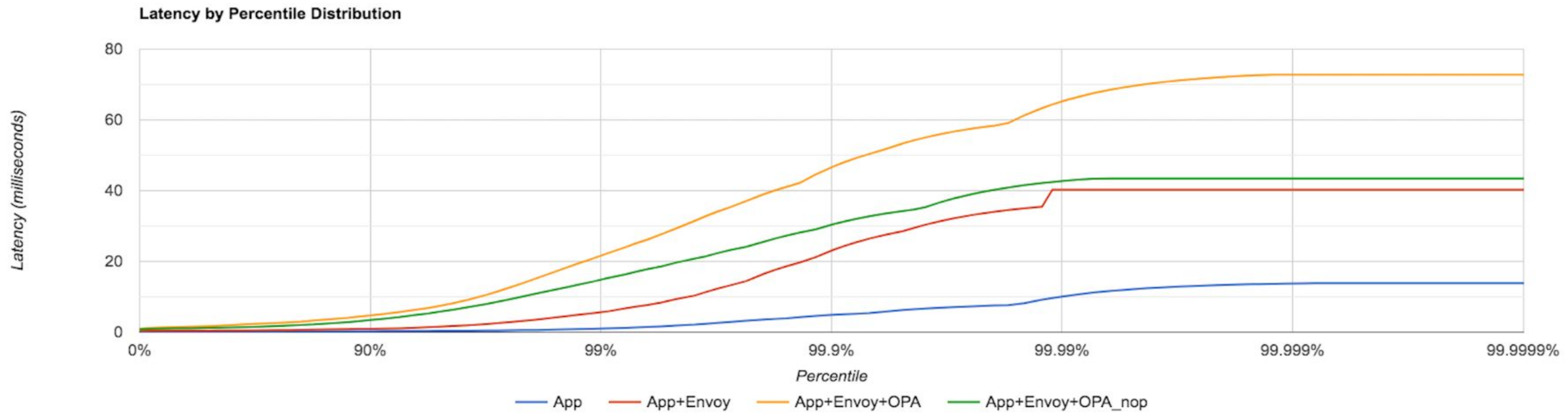
Expression	Indexed	Reason
`glob.match("foo:*:bar", [":"], input.x)`	yes	n/a
`glob.match("foo:**:bar", [":"], input.x)`	no	pattern contains `**`
`glob.match("foo:*:bar", [":"], input.x[i])`	no	match contains variable(s)

OPA - Performance comprehension Indexing

```
some i
intf := input.exposed[i].interface
ports := [port | some j; input.exposed[j].interface == intf; port := input.exposed[j].port]
```

```
deny[msg] {
  some i
  count(exposed_ports_by_interface[i]) > 100
  msg := sprintf("interface '%v' exposes too many ports", [i])
}

exposed_ports_by_interface := {intf: ports |
  some i
  intf := input.exposed[i].interface
  ports := [port |
    some j
    input.exposed[j].interface == intf
    port := input.exposed[j].port
  ]
}
```

OPA - Performance envoy example

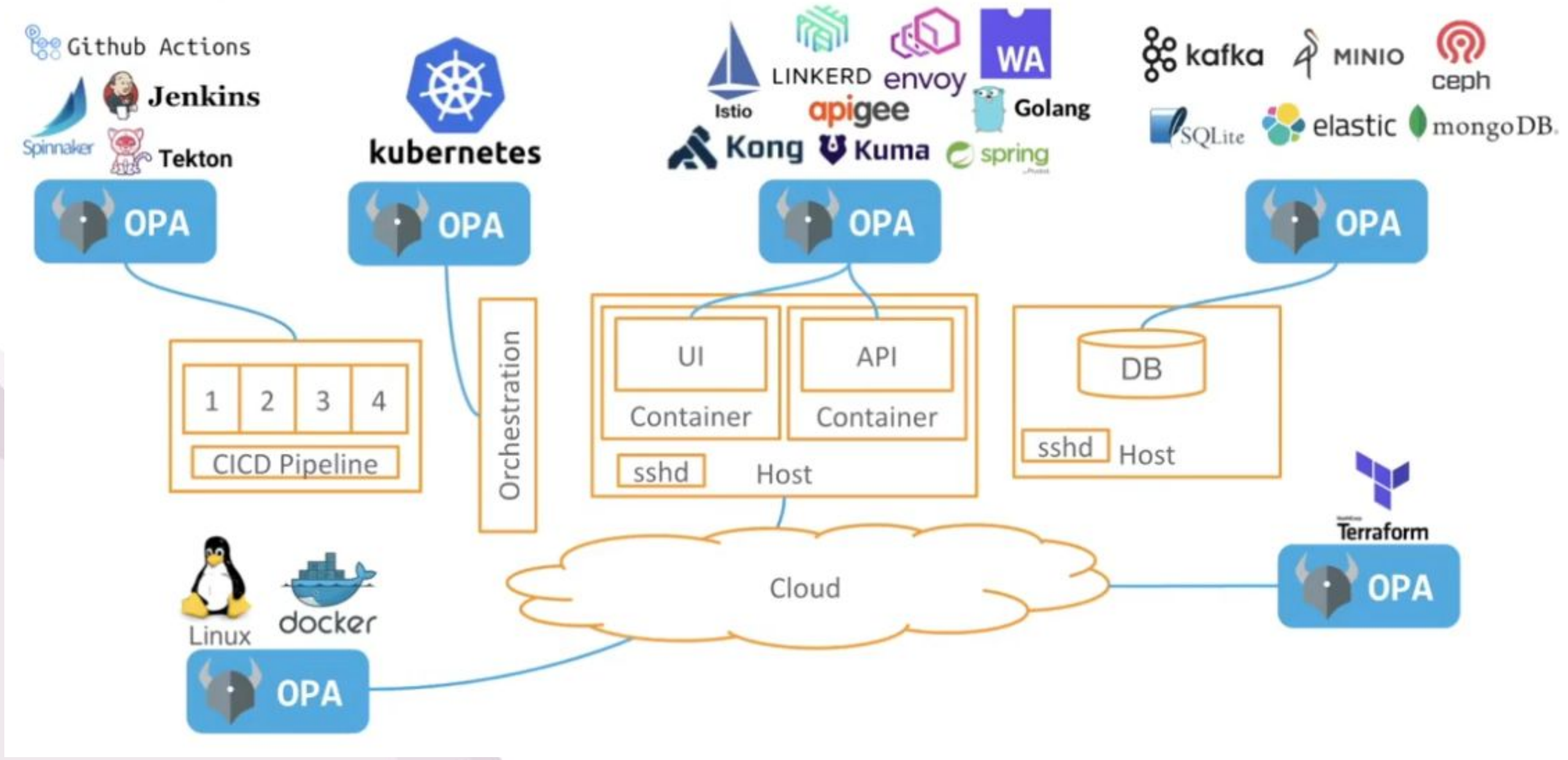


labyrinth labs

Ecosystem

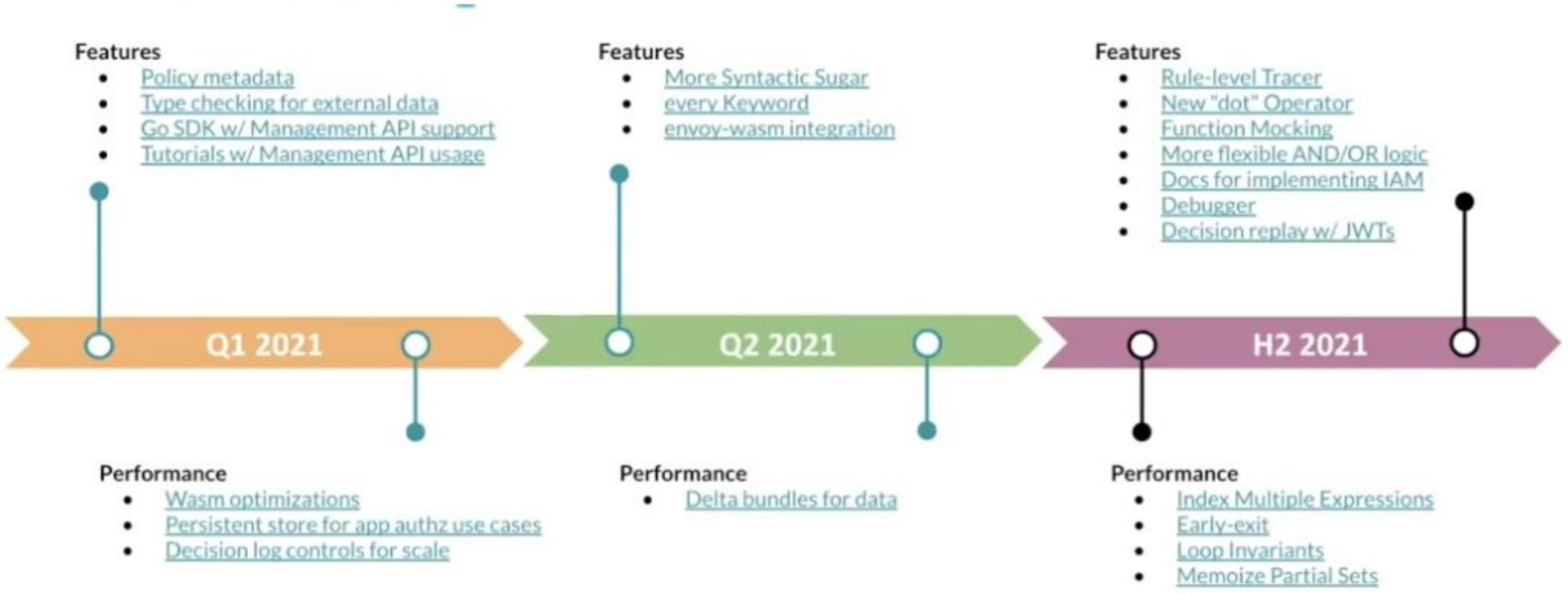
labyrinth labs

OPA - Integrations



labyrinth labs

OPA - Roadmap



Features
- Policy metadata
- Type checking for external data
- Go SDK w/ Management API support
- Tutorials w/ Management API usage

Performance
- Wasm optimizations
- Persistent store for app authz use cases
- Decision log controls for scale

Features
- More Syntactic Sugar
- every Keyword
- envoy-wasm integration

Performance
- Delta bundles for data

Features
- Rule-level Tracer
- New "dot" Operator
- Function Mocking
- More flexible AND/OR logic
- Docs for implementing IAM
- Debugger
- Decision replay w/ JWTs

Performance
- Index Multiple Expressions
- Early-exit
- Loop Invariants
- Memoize Partial Sets

OPA - Tooling

- OPA cli
  - eval
  - bench
  - fmt
  - check
- VS code, IntelliJ plugins
- conftest
- terrascan
- OPAL
- rego playground

OPA - inspiration

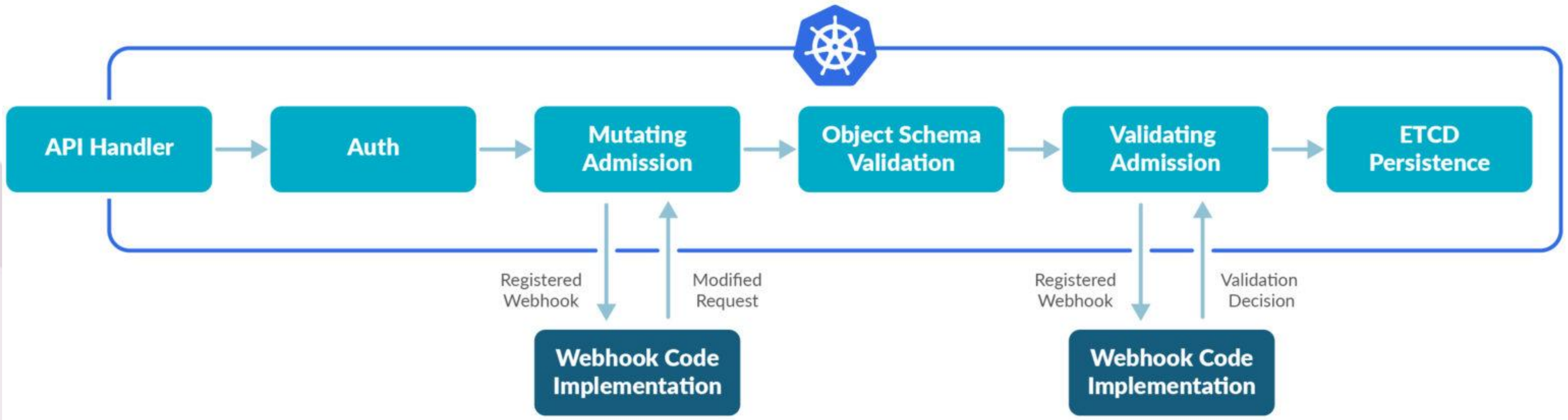
- terrascan rego
- gatekeeper library
- scalr sample policies
- OPA PoC & contributions

Examples

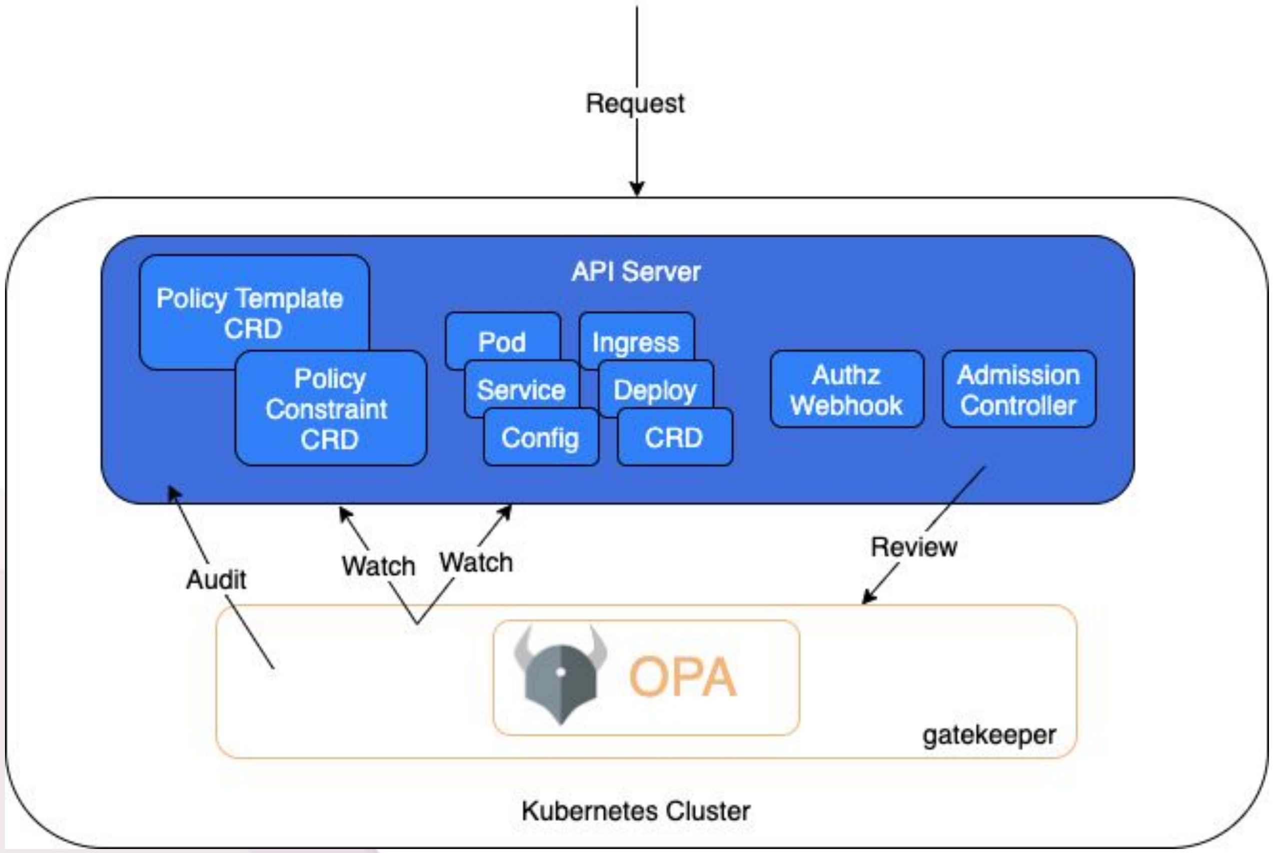
–

Kubernetes Gatekeeper

OPA - Gatekeeper



OPA - Gatekeeper

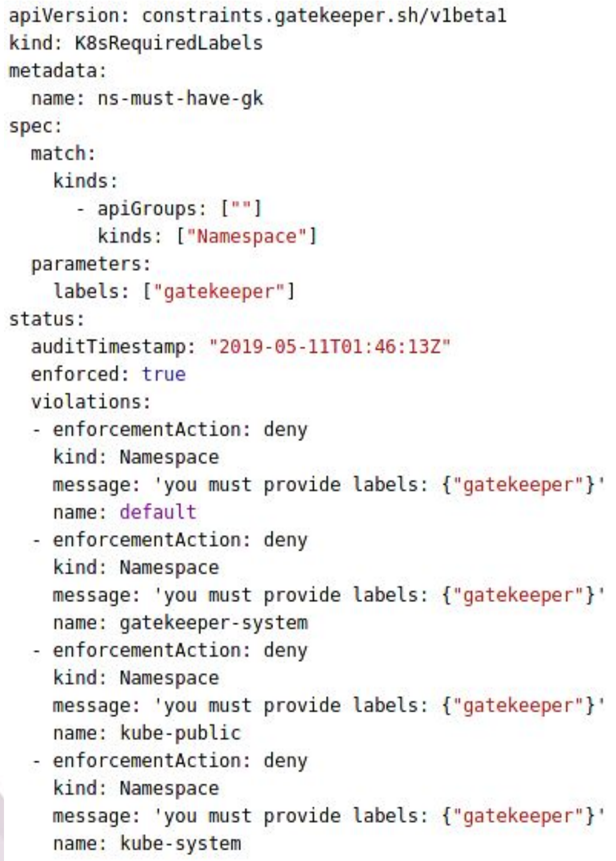


OPA - Gatekeeper policy enforcement

```
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
spec:
  crd:
    spec:
      names:
        kind: K8sRequiredLabels
      validation:
        # Schema for the `parameters` field
        openAPIV3Schema:
          properties:
            labels:
              type: array
              items: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8srequiredlabels

        violation[{"msg": msg, "details": {"missing_labels": missing}}] {
          provided := {label | input.review.object.metadata.labels[label]}
          required := {label | label := input.parameters.labels[_]}
          missing := required - provided
          count(missing) > 0
          msg := sprintf("you must provide labels: %v", [missing])
        }
```

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    labels: ["gatekeeper"]
```

OPA - Gatekeeper audit



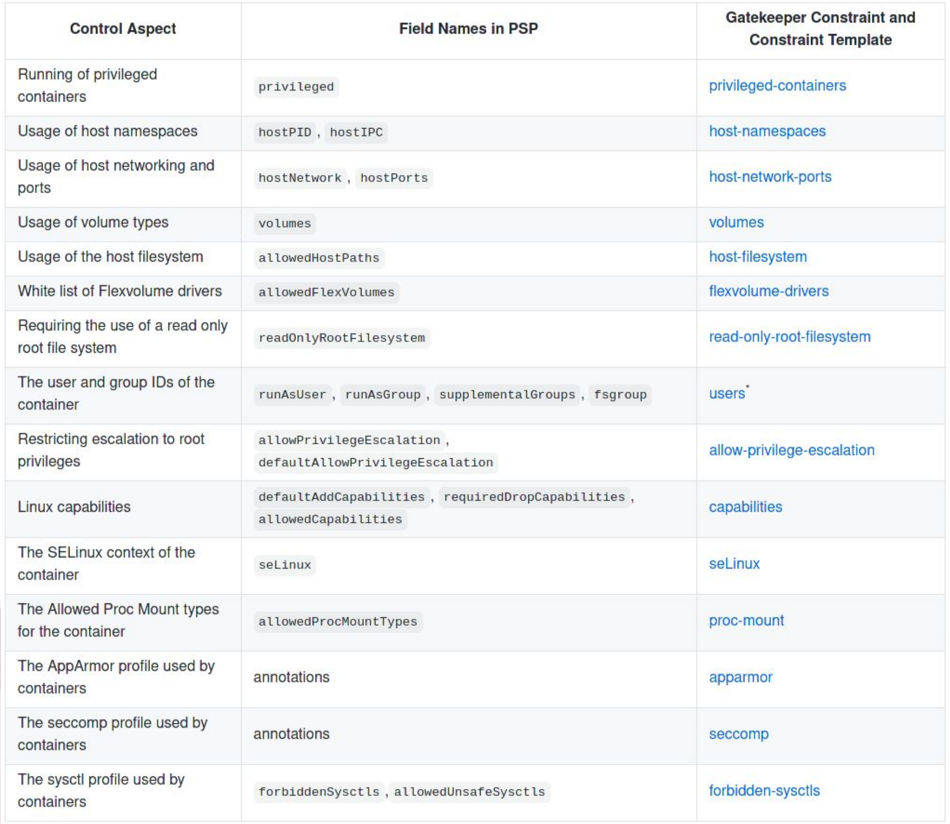
```yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    labels: ["gatekeeper"]
status:
  auditTimestamp: "2019-05-11T01:46:13Z"
  enforced: true
  violations:
  - enforcementAction: deny
    kind: Namespace
    message: 'you must provide labels: {"gatekeeper"}'
    name: default
  - enforcementAction: deny
    kind: Namespace
    message: 'you must provide labels: {"gatekeeper"}'
    name: gatekeeper-system
  - enforcementAction: deny
    kind: Namespace
    message: 'you must provide labels: {"gatekeeper"}'
    name: kube-public
  - enforcementAction: deny
    kind: Namespace
    message: 'you must provide labels: {"gatekeeper"}'
    name: kube-system
```

OPA - Gatekeeper Library General

- K8sAllowedRepos
- K8sBlockNodePort
- K8sContainerLimits
- K8sContainerRatios
- K8sDisallowedTags
- K8sHttpsOnly
- K8sImageDigests
- K8sRequiredLabels

- K8sRequiredProbes
- K8sUniqueIngressHost
- K8sUniqueIngressHost
- K8sExternalIPs

OPA - Gatekeeper PSP replacement



Control Aspect	Field Names in PSP	Gatekeeper Constraint and Constraint Template
Running of privileged containers	`privileged`	privileged-containers
Usage of host namespaces	`hostPID` , `hostIPC`	host-namespaces
Usage of host networking and ports	`hostNetwork` , `hostPorts`	host-network-ports
Usage of volume types	`volumes`	volumes
Usage of the host filesystem	`allowedHostPaths`	host-filesystem
White list of Flexvolume drivers	`allowedFlexVolumes`	flexvolume-drivers
Requiring the use of a read only root file system	`readOnlyRootFilesystem`	read-only-root-filesystem
The user and group IDs of the container	`runAsUser` , `runAsGroup` , `supplementalGroups` , `fsgroup`	users*
Restricting escalation to root privileges	`allowPrivilegeEscalation` , `defaultAllowPrivilegeEscalation`	allow-privilege-escalation
Linux capabilities	`defaultAddCapabilities` , `requiredDropCapabilities` , `allowedCapabilities`	capabilities
The SELinux context of the container	`seLinux`	seLinux
The Allowed Proc Mount types for the container	`allowedProcMountTypes`	proc-mount
The AppArmor profile used by containers	annotations	apparmor
The seccomp profile used by containers	annotations	seccomp
The sysctl profile used by containers	`forbiddenSysctls` , `allowedUnsafeSysctls`	forbidden-sysctls

labyrinth labs

Thank you

Contact:

Martin Dojcak

CTO & Co-founder

martin.dojcak@lablabs.io