OPA: Community

- Project started at Styra, Inc. in early 2016
- Case Studies
  - **Netflix**: Distributed authorization for internal resources
    - HTTP and gRPC API authorization for internal microservices
    - Kafka protocol-level authorization for topics
    - Used across several teams
  - **Medallia**: Enforce & audit infrastructure invariants
    - Terraform risk management
    - AWS network security group auditing
  - **Chef**: End-user authorization service
    - gRPC API authorization
    - User permission auditing
- Many other companies using OPA including Cloudflare, Intuit, Pinterest, State Street, Nefeli Networks, SolarWinds, Cisco, and more.

- 36 contributors
- 400 slack users
- 25K image pulls/week
- 20+ integrations
What Is OPA?
OPA: general-purpose policy engine
OPA: general-purpose policy engine

GET /salary/bob HTTP/1.1
Authorization: alice
OPA: general-purpose policy engine

---

GET /salary/bob HTTP/1.1
Authorization: alice

true or false

---

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

---

OPA

Policy
(Rego)

Data
(JSON)
OPA: general-purpose policy engine

"Service" refers to any one of:
- Custom service
- API gateway
- Message broker
- Kubernetes API server
- CI/CD pipeline script
Input can be any JSON value:

- "alice"
- ["v1", "users", "bob"]
- {"kind": "Pod", "spec": ...}

Output can be any JSON value:

- true
- "request rejected"
- {"servers": ["web1", "web2"]}

"Service" refers to any one of:
- Custom service
- API gateway
- Message broker
- Kubernetes API server
- CI/CD pipeline script
OPA: general-purpose policy engine

- Declarative Policy Language (Rego)
  - Can user X do operation Y on resource Z?
  - What invariants does workload W violate?
  - Which users can SSH into production hosts?

openpolicyagent.org
OPA: general-purpose policy engine

- **Declarative Policy Language (Rego)**
  - Can user X do operation Y on resource Z?
  - What invariants does workload W violate?
  - Which users can SSH into production hosts?

- **Library, sidecar, host-level daemon**
  - Policy and data are kept in-memory
  - Zero decision-time dependencies
OPA: general-purpose policy engine

- Declarative Policy Language (Rego)
  - Can user X do operation Y on resource Z?
  - What invariants does workload W violate?
  - Which users can SSH into production hosts?

- Library, sidecar, host-level daemon
  - Policy and data are kept in-memory
  - Zero decision-time dependencies

- Management APIs for control & observability
  - Bundle service API for sending policy & data to OPA
  - Status service API for receiving status from OPA
  - Log service API for receiving audit log from OPA
**OPA: general-purpose policy engine**

- **Declarative Policy Language (Rego)**
  - Can user X do operation Y on resource Z?
  - What invariants does workload W violate?
  - Which users can SSH into production hosts?

- **Library, sidecar, host-level daemon**
  - Policy and data are kept in-memory
  - Zero decision-time dependencies

- **Management APIs for control & observability**
  - Bundle service API for sending policy & data to OPA
  - Status service API for receiving status from OPA
  - Log service API for receiving audit log from OPA

- **Tooling to build, test, and debug policy**
  - opa run, opa test, opa fmt, opa deps, opa check, etc.
  - VS Code plugin, Tracing, Profiling, etc.
# OPA: general-purpose policy engine

<table>
<thead>
<tr>
<th>Component</th>
<th>Policy Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admission Control</td>
<td>“Restrict ingress hostnames for payments team.” “Ensure container images come from corporate repo.”</td>
</tr>
<tr>
<td>API Authorization</td>
<td>“Deny test scripts access to production services.” “Allow analysts to access APIs serving anonymized data.”</td>
</tr>
<tr>
<td>SSH &amp; sudo</td>
<td>“Only allow on-call engineers to SSH into production servers.”</td>
</tr>
<tr>
<td>Data Protection</td>
<td>&quot;Trades exceeding $10M must be executed between 9AM and 5PM and require MFA.&quot;</td>
</tr>
<tr>
<td>Data Filtering</td>
<td>&quot;Users can access files for past 6 months related to the region they licensed.&quot;</td>
</tr>
</tbody>
</table>
How does OPA work?
How does OPA work?

Example policy

"Employees can read their own salary and the salary of anyone they manage."
OPA: Declarative Language (Rego)

**Example policy**

Employees can read their own salary and the salary of anyone they manage.
OPA: Declarative Language (Rego)

**Example policy**

Employees can read their own salary and the salary of anyone they manage.

**Input Data**

- method: "GET"
- path: ["salary", "bob"]
- user: "bob"
Example policy

Employees can read their own salary and the salary of anyone they manage.

Input Data

method: "GET"
path: ["salary", "bob"]
user: "bob"
OPA: Declarative Language (Rego)

**Example policy**

Employees can read their own salary and the salary of anyone they manage.

**Input Data**

- method: "GET"
- path: ["salary", "bob"]
- user: "bob"

```rego
allow = true {
    input.method = "GET"
    input.path = ["salary", "bob"]
    input.user = "bob"
}
```
Example policy

Employees can read their own salary and the salary of anyone they manage.

Input Data

method: "GET"
path: ["salary", "bob"]
user: "alice"

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

Different user now!
OPA: Declarative Language (Rego)

Example policy

Employees can read their own salary and the salary of anyone they manage.

Input Data

method: "GET"
path: ["salary", "bob"]
user: "alice"

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

This statement will "FAIL"

Different user now!
OPA: Declarative Language (Rego)

Example policy

Employees can read their own salary and the salary of anyone they manage.

Input Data

method: "GET"
path: ["salary", "bob"]
user: "alice"

Context Data

{
  "managers": {
    "bob": ["alice", "fred"]
    "alice": ["fred"]
  }
}
OPA: Declarative Language (Rego)

Example policy

Employees can read their own salary and the salary of anyone they manage.

Input Data

method: "GET"
path: ["salary", "bob"]
user: "alice"

Context Data

{
  "managers": {
    "bob": ["alice", "fred"]
    "alice": ["fred"]
  }
}
Example policy

Employees can read their own salary and the salary of anyone they manage.

Input Data

method: "GET"
path: ["salary", "bob"]
user: "alice"

Context Data

{
  "managers": {
    "bob": ["alice", "fred"]
    "alice": ["fred"]
  }
}
OPA: Declarative Language (Rego)

Example policy

Employees can read their own salary and the salary of anyone they manage.

Input Data

method: "GET"
path: ["salary", "bob"]
user: "alice"

Context Data

{
  "managers": {
    "bob": ["alice", "fred"]
    "alice": ["fred"]
  }
}
import data.managers

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

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

More information at openpolicyagent.org

See **How Do I Write Policies?**
- Explains language constructs

See **Language Reference**
- Documents built-in functions: glob, regex, JWTs, x509, etc.

See **Tutorials** section
- HTTP APIs, Kubernetes, Docker, Terraform, Kafka, SSH, etc.
Integrating with OPA
Integrating OPA: Overview

Command Line

- Useful for CI/CD pipelines
- Run query and check output
- Use --fail to exit on error/undefined

HTTP

- Any language, any framework
- Run OPA sidecar/host daemon
- Security: UNIX domain or TLS+Auth/n+Auth/z

Golang library

- Lightweight, simple API
- No extra deployment steps

```bash
$ opa eval
  --input=input.json
  --format=json
  --data=cicd.rego
  'data.cicd.violations[x]' 
```

```json
POST /v1/data/authz/allow

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

```go
ctx := context.Background()
r := rego.New(...)
rs, err := r.Eval(ctx)
// handle error(s)
// handle result set
```
import "github.com/open-policy-agent/opa/rego"

func main() {

    ctx := context.Background()
    r := rego.New(rego.Query("a = 1"))
    rs, err := r.Eval(ctx)

    if err != nil {
        log.Fatal(err)
    }

    log.Println(rs)

}
Integrating OPA: Rego Parameters

```go
r := rego.New(
    rego.Query(`
        input.method = "GET";
        input.path = ["salary", employee_id];
        input.user = employee_id
    `),
    rego.Input(map[string]interface{}{
        "method": "GET",
        "path": []string{"salary", "bob"},
        "user": "bob",
    })),
)

rs, _ := r.Eval(ctx)

log.Println(rs)    // Output: [{[true true true] {employee_id: bob}}]
```
Integrating OPA: Rego Parameters

```go
r := rego.New(
    rego.Query(`
        input.method = "GET";
        input.path = ["salary", employee_id];
        input.user = employee_id
    `),
    rego.Input(map[string]interface{}{
        "method": "GET",
        "path": []string{"salary", "bob"},
        "user": "alice",
    })),
)

rs, _ := r.Eval(ctx)

log.Println(rs) // Output: []
```

Different user
## Integrating OPA: Rego Parameters

```rego
module := `package example
import data.managers

default allow = false

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

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

```go
store := inmem.NewFromObject(map[string]interface{}{
  "managers": map[string]interface{}{
    "bob": []string{"alice", "fred"},
    "alice": []string{"fred"},
  }})

query := "data.example.allow = true"

r := rego.New(
  rego.Module("example.rego", module),
  rego.Store(store),
  rego.Query(query),
  rego.Input(input),
)

rs, _ := r.Eval(ctx)

log.Println(rs) // Output: [{[true] {}}]
```
Integrating OPA: Rego Parameters

module := `package example
import data.managers
default allow = false
allow = true {
  input.method = "GET"
  input.path = ["salary", employee_id]
  input.user = employee_id
}
allow = true {
  input.method = "GET"
  input.path = ["salary", employee_id]
  input.user = managers[employee_id][_]}

store := inmem.NewFromObject(
  map[string]interface{}
  "managers": map[string]interface{}
  "bob": []string{"alice", "fred"},
  "alice": []string{"fred"},
})
query := "data.example.allow = true"
r := rego.New(
  rego.Module("example.rego", module),
  rego.Store(store),
  rego.Query(query),
  rego.Input(input),
)
rs, _ := r.Eval(ctx)
log.Println(rs) // Output: {{true}{}]
if len(rs) == 0 ⇒ Deny
else ⇒ Allow
func main() {
    router := mux.NewRouter()
    router.Use(authorize)
    http.ListenAndServe(":8000", router)
}

func salaryGet(w http.ResponseWriter, r *http.Request) {
    id := mux.Vars(r)["id"]
    salary, ok := salaries[id]
    if !ok {
        w.WriteHeader(http.StatusNotFound)
        return
    }
    bs, _ := json.Marshal(salary)
    w.Header().Set("Content-Type", "application/json")
    w.WriteHeader(http.StatusOK)
    w.Write(bs)
}
Integrating OPA: gorilla/mux example

```go
func main() {
    router := mux.NewRouter()
    router.Use(authorize)
    http.ListenAndServe(":8000", router)
}

func authorize(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        // 1. Authorize request
        // 2. On error return 500
        // 3. On failure return 403
        // 4. On success call next(w, r)
    })
}
```
func authorize(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        eval := rego.New(
            rego.Module("example.rego", module),
            rego.Store(store),
            rego.Input(makeInput(r)),
            rego.Query("data.example.allow = true"))

        rs, err := eval.Eval(r.Context())

        if err != nil {
            w.WriteHeader(http.StatusInternalServerError)
        } else if len(rs) == 0 {
            w.WriteHeader(http.StatusForbidden)
        } else {
            next.ServeHTTP(w, r)
        }
    })
}

func makeInput(r *http.Request) interface{} {
    return map[string]interface{}{
        "method": r.Method,
        "path": strings.Split(strings.Trim(r.URL.Path, "/"), "/"),
        "user": r.Header.Get("Authorization"),
    }
}
Integrating OPA: Library References

- Example from this talk
  - https://github.com/tsandall/kubecon-shanghai-2018

- GoDoc examples
  - https://godoc.org/github.com/open-policy-agent/opa/rego

- Full example for HTTP API authorization
  - https://github.com/open-policy-agent/example-api-authz-go
  - Includes instantiation of bundle, status, and decision log plugins
Thank You! Questions?

slack.openpolicyagent.org

open-policy-agent/opa

See [low-hanging-fruit](#) and [help-wanted](#) labels if you want to contribute.