Carnegie Mellon University
Software Engineering Institute

DEVSECOPS DAYS 2021 | LOS ANGELES

DEVSECOPS DAYS

Implementing Policy as Code through Open Policy Agent

SEPTEMBER 15

Marudhamaran Gunasekaran
Principal Consultant @ Practical DevSecOps

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

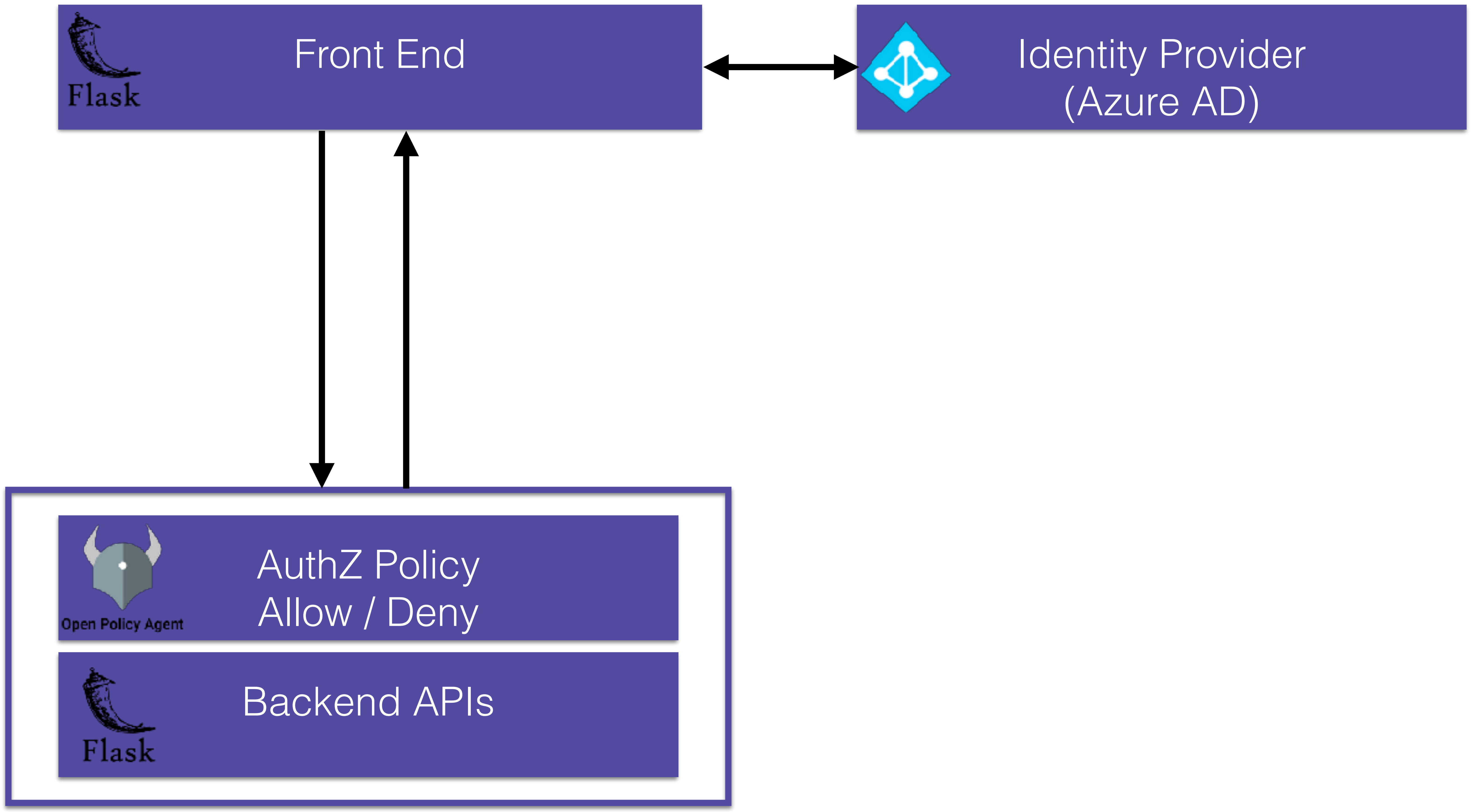
About Me

- ✓ Security Consultant (Practical DevSecOps, and Hysn)
- ✓ Dev, IT, Coach, Compliance, DevSecOps
- ✓ Author @ Pluralsight
- ✓ Creator, Maintainer @ OWASP ZAP Dot Net API
- ✓ MCT, CDP, CEH, ISO 27001 LA, PSM I, II, III, and more

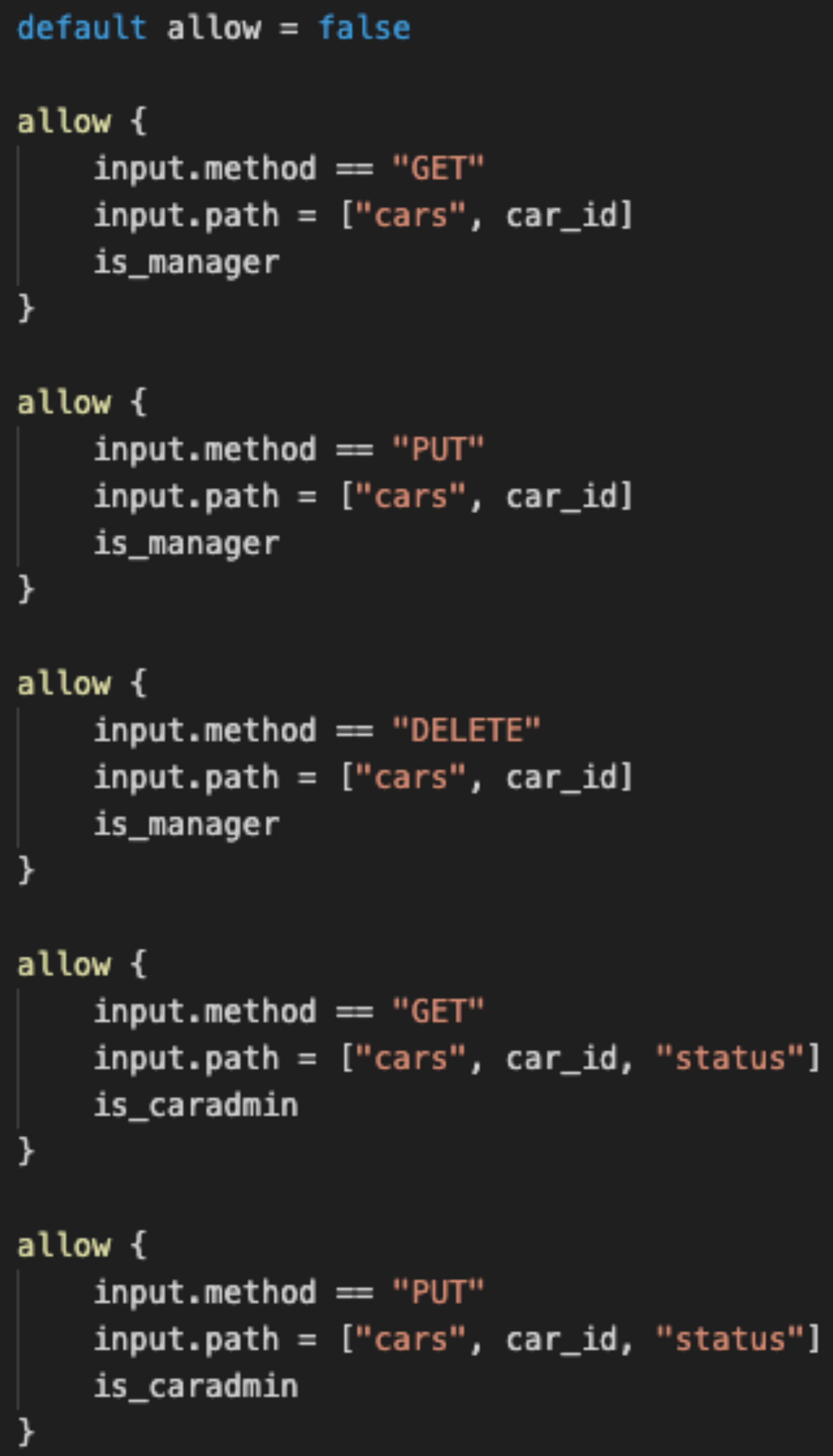
Agenda

- ✓ Exploring Policy as Code through an example
- ✓ Why you should consider Policy as Code
- ✓ Integrating apps or tools with Policy as Code
- ✓ Writing policies in Rego
- ✓ Resources to help you get started

How about a demonstration first?



Traditionally Policies are Written as Documents



```
default allow = false

allow {
    input.method == "GET"
    input.path = ["cars", car_id]
    is_manager
}

allow {
    input.method == "PUT"
    input.path = ["cars", car_id]
    is_manager
}

allow {
    input.method == "DELETE"
    input.path = ["cars", car_id]
    is_manager
}

allow {
    input.method == "GET"
    input.path = ["cars", car_id, "status"]
    is_caradmin
}

allow {
    input.method == "PUT"
    input.path = ["cars", car_id, "status"]
    is_caradmin
}
```

Why consider Policy as Code?

Documentation

Automation

Versioning

"Treat policy as a separate concern....
just like DB, messaging, monitoring,
logging, orchestration, CI/CD ..."

- Torin Sandall, Co-Creator, OPA

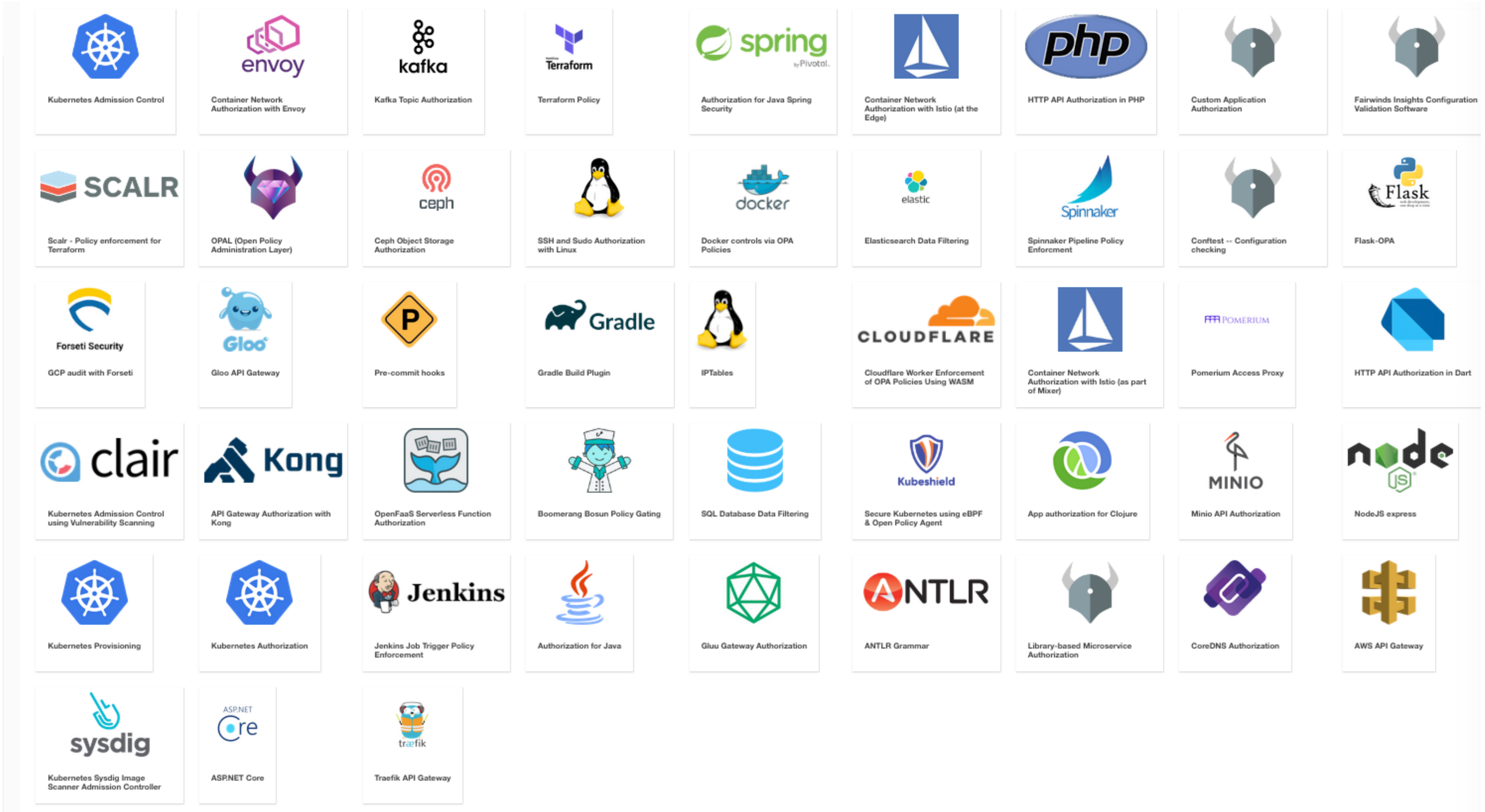
Open Policy Agent

www.openpolicyagent.org

Policy-based control for cloud native environments
Flexible, fine-grained control for administrators across the stack

Container
API
UI
Host
Kubernetes
SSH
UI
Data
Servicemesh
CICD
API
Ingress
Cloud

OPA Ecosystem



- Kubernetes Admission Control
- Container Network Authorization with Envoy
- Kafka Topic Authorization
- Terraform Policy
- Authorization for Java Spring Security
- Container Network Authorization with Istio (at the Edge)
- HTTP API Authorization in PHP
- Custom Application Authorization
- Fairwinds Insights Configuration Validation Software
- Scalr - Policy enforcement for Terraform
- OPAL (Open Policy Administration Layer)
- Ceph Object Storage Authorization
- SSH and Sudo Authorization with Linux
- Docker controls via OPA Policies
- Elasticsearch Data Filtering
- Spinnaker Pipeline Policy Enforcement
- Conftest -- Configuration checking
- Flask-OPA
- GCP audit with Forseti
- Gloo API Gateway
- Pre-commit hooks
- Gradle Build Plugin
- IPTables
- Cloudflare Worker Enforcement of OPA Policies Using WASM
- Container Network Authorization with Istio (as part of Mixer)
- Pomerium Access Proxy
- HTTP API Authorization in Dart
- Kubernetes Admission Control using Vulnerability Scanning
- API Gateway Authorization with Kong
- OpenFaaS Serverless Function Authorization
- Boomerang Bosun Policy Gating
- SQL Database Data Filtering
- Secure Kubernetes using eBPF & Open Policy Agent
- App authorization for Clojure
- Minio API Authorization
- NodeJS express
- Kubernetes Provisioning
- Kubernetes Authorization
- Jenkins Job Trigger Policy Enforcement
- Authorization for Java
- Gluu Gateway Authorization
- ANTLR Grammar
- Library-based Microservice Authorization
- CoreDNS Authorization
- AWS API Gateway
- Kubernetes Sysdig Image Scanner Admission Controller
- ASP.NET Core
- Traefik API Gateway

OPA does not know whether the service is Kubernetes, or python flask api, or ssh, or docker.

OPA does not care what your service is, as long as the *input is JSON.*

OPA makes policy decisions based on the input JSON, and *responds with a JSON output.*

Integrating OPA with your app/service/tool/…

Create input for OPA

Query OPA

Check allow / deny

Integrating OPA with your app/service/tool/...

Create input for OPA
Query OPA
Check allow / deny

```python
@app.before_request
def check_authorization():
    try:
        input = json.dumps({
            "method": request.method,
            "path": request.path.strip().split("/")[1:],
            "access_token": get_access_token(request),
            "id_token": get_id_token(request),
        }, indent=2)
        url = os.environ.get("OPA_URL", "http://localhost:8000")
        app.logger.debug("OPA query: %s. Body: %s", url, input)
        response = requests.post(url, data=input)
    except Exception as e:
        app.logger.exception("Unexpected error querying OPA.")
        abort(500)

    if response.status_code != 200:
        app.logger.error("OPA status code: %s. Body: %s",
                         response.status_code, response.json())
        abort(500)

    allowed = response.json()
    app.logger.debug("OPA result: %s", allowed)
    if not allowed:
        abort(403)
```

OPA makes the policy decision.

Service enforces the policy decision.

Writing Policies in Rego

- ✓ OPA REPL (Read, Execute, Print, Loop)
- ✓ Visual Studio Code
- ✓ play.openpolicyagent.org

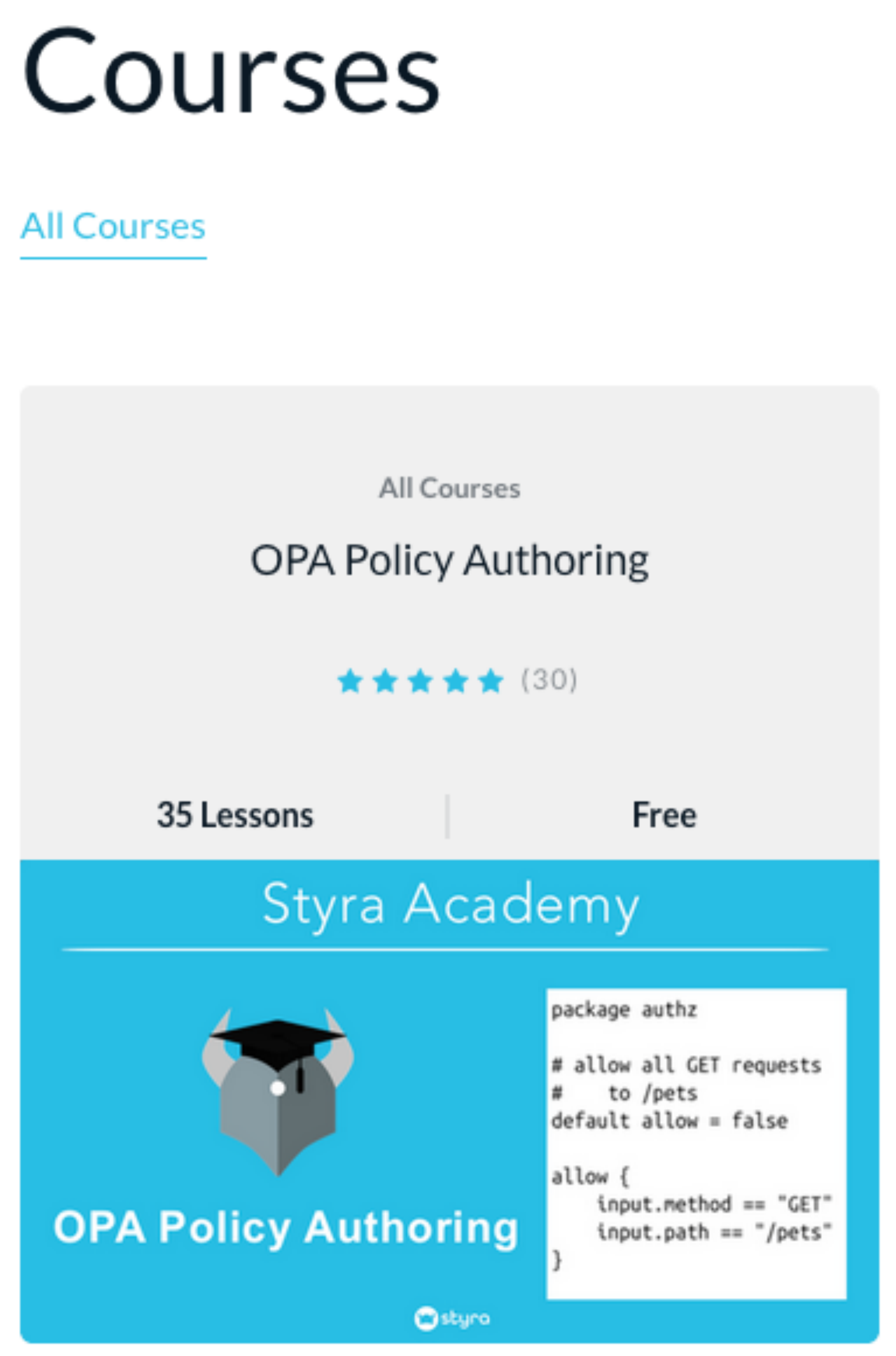
awesome-policy-as-code

https://github.com/hysnsec/awesome-policy-as-code

Blogs, videos, tools, and other resources

Free course about OPA and Rego

https://academy.styra.com/collections



Courses

All Courses

All Courses

OPA Policy Authoring

★★★★★ (30)

35 Lessons | Free

Styra Academy

OPA Policy Authoring

```
package authz

# allow all GET requests
#    to /pets
default allow = false

allow {
    input.method == "GET"
    input.path == "/pets"
}
```

More Questions?

maran@practical-devsecops.com

Practical DevSecOps