Checking the Blueprints with Regula

Matt Everson, Senior Engineer at Snyk

With thousands of resources deployed in the cloud, how can we test changes before they go live?

Define Infrastructure as Code

(test against rules with Regula)

Infrastructure as Code

- Definitions of infrastructure configuration, typically committed to source control
- Common components: Terraform, CloudFormation, Kubernetes, Docker
- Security opportunity to assess attack surface before deployment to any environment

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }

  required_version = ">= 0.14.9"
}

provider "aws" {
  profile = "default"
  region  = "us-west-2"
}

resource "aws_instance" "app_server" {
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInstance"
  }
}
```

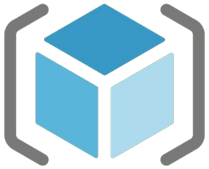

ONE PULL REQUEST
TO CRIPPLE THEM ALL

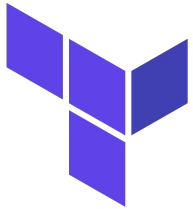
A Simple Example

https://learn.hashicorp.com/tutorials/terraform/aws-build

What is Regula?

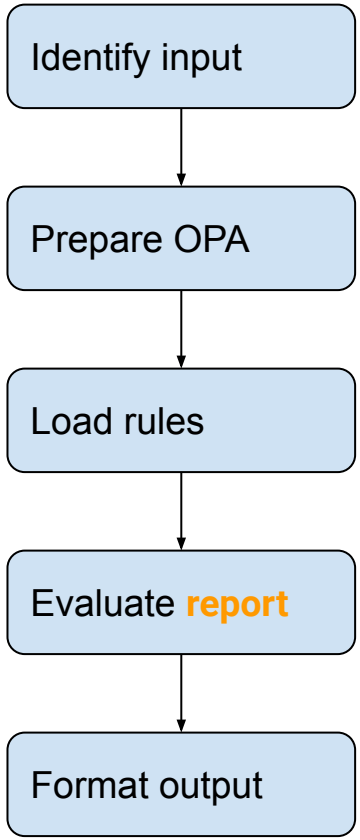
- Prebuilt compliance rules
- Library of rego code to simplify custom rules
- Produces report including remediation steps
- Custom waivers
- Can be integrated into CI/CD





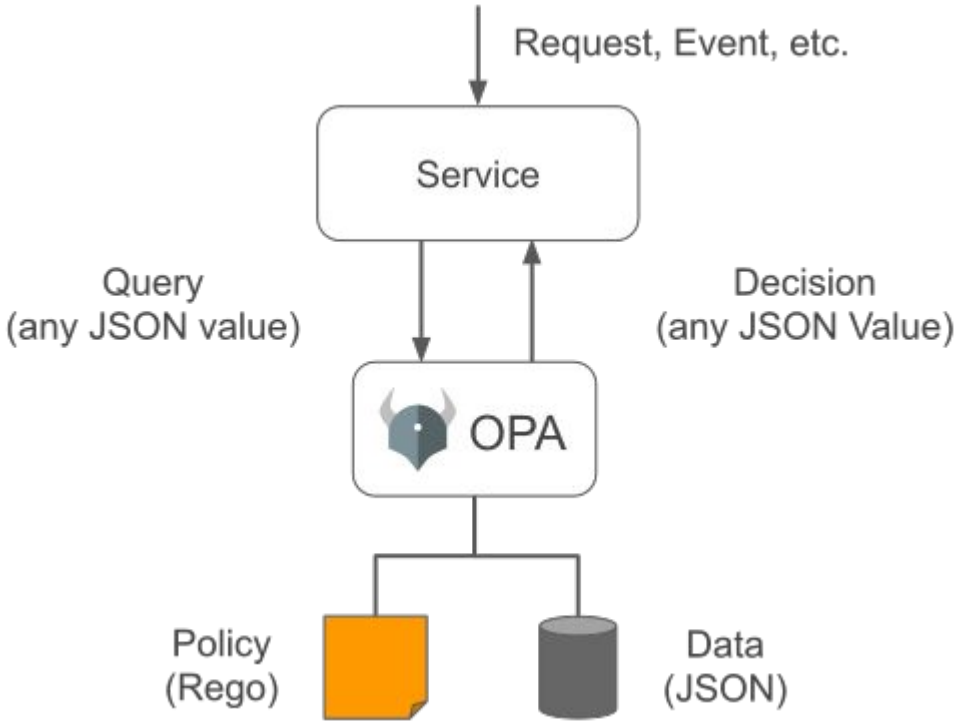
Regula Internals

- Rego functions
  - Adjust inconsistencies in input
  - Execute rules
  - Get resources
  - Get metadata



What is Open Policy Agent (OPA)?

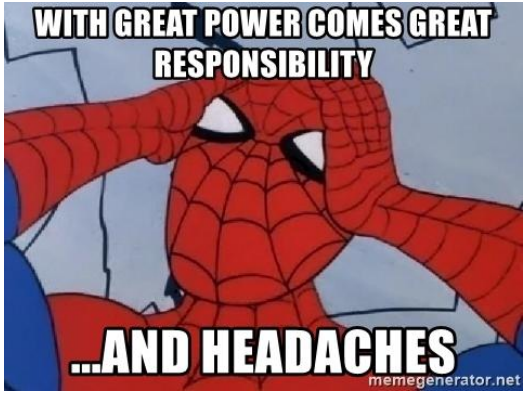
- General-purpose policy engine
- High-level query language
- Compatible with JSON
- Can traverse multiple layers of hierarchy
- Can leverage external information
- Wide adoption for Kubernetes and other use cases
- Cloud Native Computing Foundation (CNCF) graduated project, used by companies like NetFlix



Rego Language

- Inspired by Datalog
- Declarative, not imperative
- Concise queries that can be optimized through improvements to the engine

```
33  resource_type := "azurerm_postgresql_server"
34
35  default allow = false
36
37  allow {
38    input.ssl_enforcement == "Enabled"
39  } {
40    # New version of the provider.
41    input.ssl_enforcement_enabled == true
42  }
```



https://www.openpolicyagent.org/docs/latest/policy-language/

Rego Language

- Every document is an if statement

```
if ( input.method == “GET” ) {
  console.println(true)
}
```

```
# An API call is allowed if the method is a GET
allow {
   input.method == “GET”
}
```

Rego Language

- Documents with the same name are combined with OR
- Statements within each block are combined with AND

```
function allow() {
    return allow1() || allow2()
}


function allow1() {
    return input.method == “GET”
}


function allow2() {
    return input.method == “POST” &&
           input.is_admin == true
}
```

```
# An API call is allowed if the method is a GET
allow {
    input.method == “GET”
}


# An API call is allowed if the method is POST and
#   the user is an admin
allow {
    # Only admins can create new objects
    input.method == “POST”
    input.user_is_admin == true
}
```

Simple Rule

- Package name
- Metadata
- Resource type to target
- Default behavior
- Rule logic based on attributes and conditions

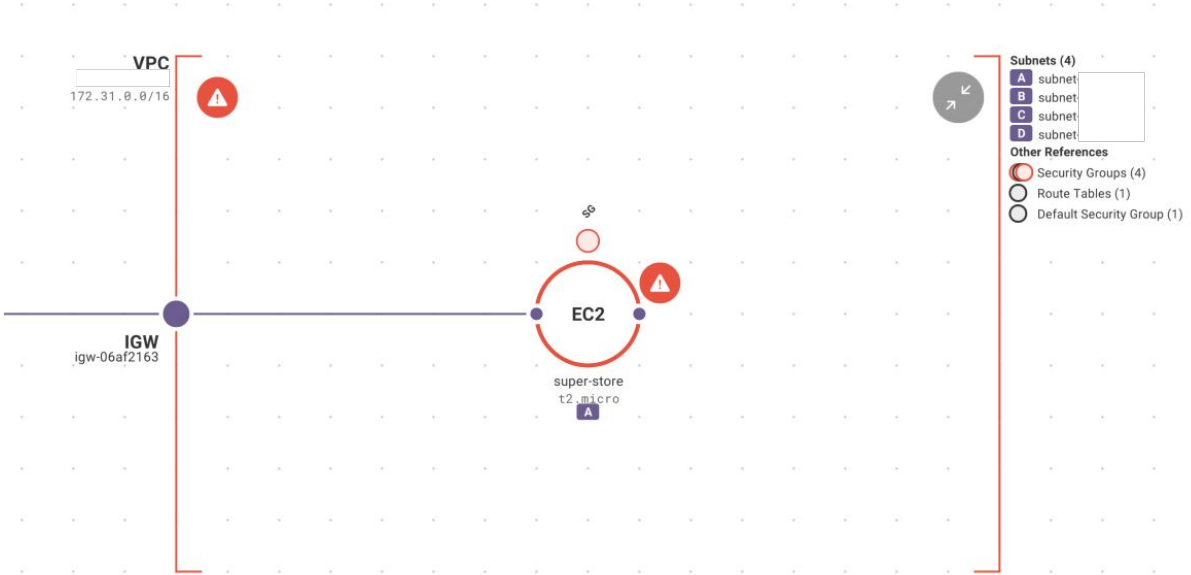
```
package rules.iam_long_description

__rego__metadoc__ := {
  "id": "CUSTOM_0001",
  "title": "IAM policies must have a description of at least 25
characters",
  "description": "Per company policy, it is required for all IAM
policies to have a description of at least 25 characters.",
  "custom": {
    "controls": {
      "CORPORATE-POLICY": [
        "CORPORATE-POLICY_1.1"
      ]
    },
    "severity": "Low"
  }
}

resource_type = "aws_iam_policy"

default allow = false

allow {
  count(input.description) >= 25
}
```

Advanced Rules

- Advanced rules can target multiple types of resources, but require a policy document that contains a set of judgements.
- Custom messages
- Complex logic

Regula at Fugue

- Upload results
- Sync custom rules
- Reporting and visualization



Future

- Migrate helper functions from rego to Go
- Support additional input types

regula.dev