



OpenTofu Day

NORTH AMERICA



OpenTofu Day
NORTH AMERICA

Policy-as-Code for Infrastructure-as-Code

Colin Lacy
Software Engineer @ Cisco

“An open source, general-purpose policy engine that unifies policy enforcement across the stack.”

Features:

- Uses a declarative policy language, Rego
- Multiple deployment models – sidecar, daemon, RESTful server, CLI
- Built-in policy testing framework
- Can compile policies to WASM modules for portability



Open Policy Agent

(OPA, pronounced “oh-pa”)

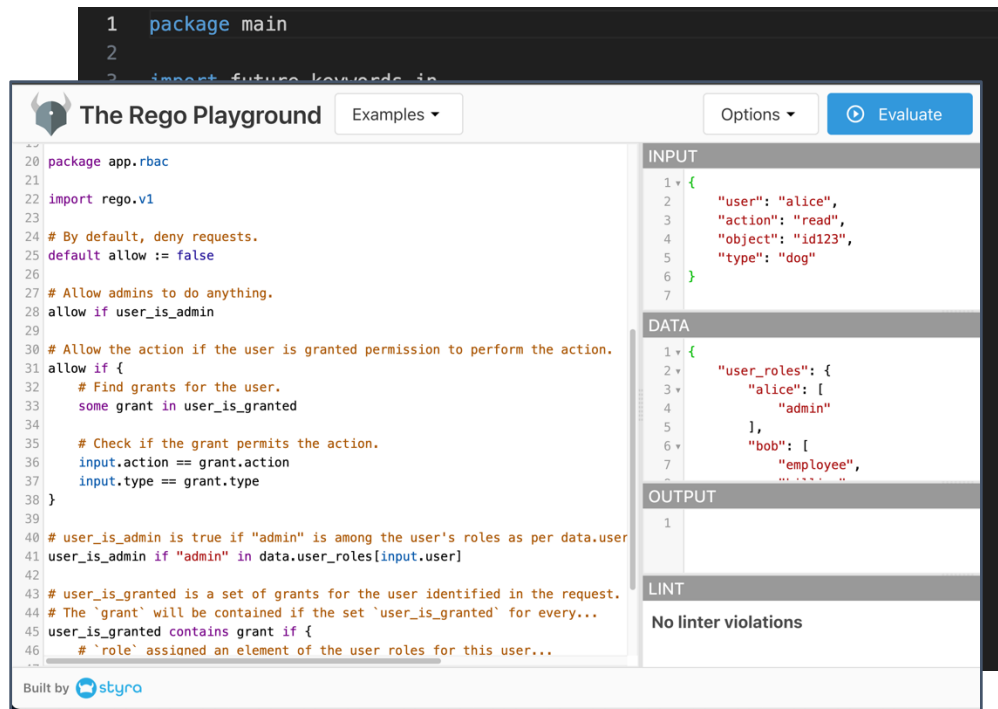
Rego?

A purpose-built language for writing and evaluating **declarative** policy as code.

You write policies that the input under evaluation should match

If at any point there is a mismatch, the data fails the policy

Rego Playground has useful examples



The screenshot displays 'The Rego Playground' interface. The main editor shows a Rego policy for RBAC. The right sidebar shows the evaluation results for the 'INPUT' and 'DATA' sections. The 'OUTPUT' section shows the result of the policy evaluation, and the 'LINT' section shows no linting violations.

```
1 package main
2
3 import future.keywords.in

20 package app.rbac
21
22 import rego.v1
23
24 # By default, deny requests.
25 default allow := false
26
27 # Allow admins to do anything.
28 allow if user_is_admin
29
30 # Allow the action if the user is granted permission to perform the action.
31 allow if {
32   # Find grants for the user.
33   some grant in user_is_granted
34
35   # Check if the grant permits the action.
36   input.action == grant.action
37   input.type == grant.type
38 }
39
40 # user_is_admin is true if "admin" is among the user's roles as per data.user
41 user_is_admin if "admin" in data.user_roles[input.user]
42
43 # user_is_granted is a set of grants for the user identified in the request.
44 # The 'grant' will be contained if the set 'user_is_granted' for every...
45 user_is_granted contains grant if {
46   # 'role' assigned an element of the user roles for this user...
```

INPUT

```
1 {
2   "user": "alice",
3   "action": "read",
4   "object": "id123",
5   "type": "dog"
6 }
```

DATA


```
1 {
2   "user_roles": {
3     "alice": [
4       "admin"
5     ],
6     "bob": [
7       "employee",
8       "manager"
9     ]
10  }
```

OUTPUT

```
1
```

LINT

No linter violations

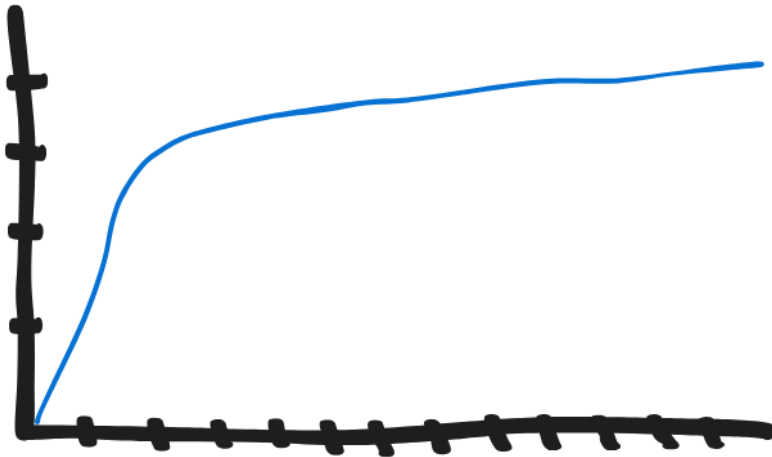
Built by 

Is it **really** worth learning
yet another language?



Yes! Especially when that language fits the purpose

- Difficulty is in the initial core concepts
- Declarative not procedural
- How iterators are handled



```
variable "availability_zones" {  
    description = "A list of availability zones in  
    type = list(string)  
}  
  
provider "aws" {  
    region = var.aws_region  
}  
  
resource "aws_vpc" "main" {  
    # Referencing the base_cidr_block variable all  
    # to be changed without modifying the configu  
    cidr_block = var.base_cidr_block  
}
```

Story time!



OpenGarlic!

My (fictional) open-source library that:

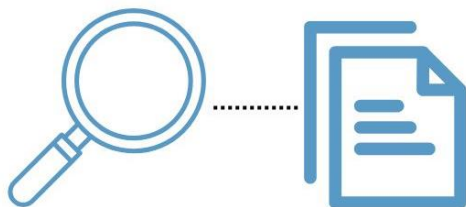
- Solves all of the problems you can think of
- Plus three other things
- Is **not real** at all

I need some GitHub repos...

I'll use OpenTofu
with an automation pipeline!



Stage 1



Inspect OpenTofu config files
using **Conftest**:

- Each repo has a default branch
- Branch protection is enabled
- Each repo has an admin team
- Archive-on-delete is set for each repo



Testing my OpenTofu files

```
$ conftest test main.tf \  
    --policy rego/repos.rego
```

Stage 1



Nothing should be deleted.
EVER!

Stage 2

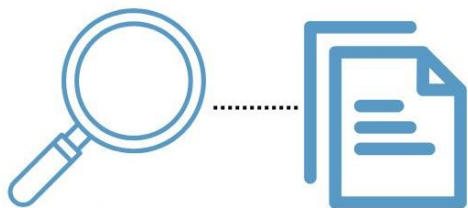




Fetching the plan as JSON

```
$ tofu show -json plan | \  
  opa exec \  
  --stdin-input \  
  --bundle rego \  
  --decision delete_prevention/deny
```

Stage 1



Apply!!!
(and ask for contributors)

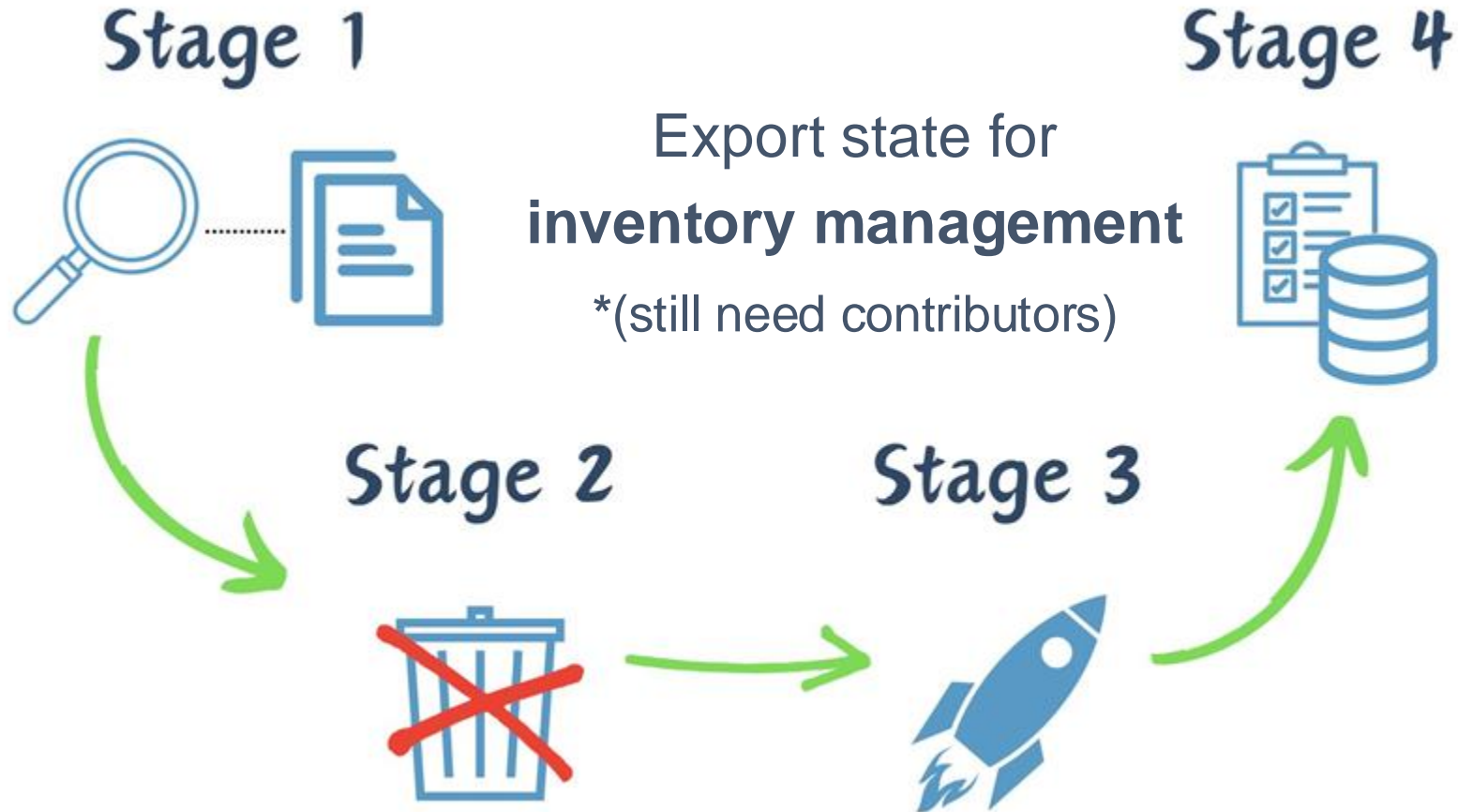
Stage 2



Stage 3



Automation Pipeline





Exporting state via NodeJS script

```
$ tofu show -json | \  
  node state-export
```

But then!



Burnout!

- Issues are piling up
- Contributors are hard to come by
- Constant dependency upgrades/patches
- Still have a day job

I need a break!
Can someone take over?





Ringo
Ziburat



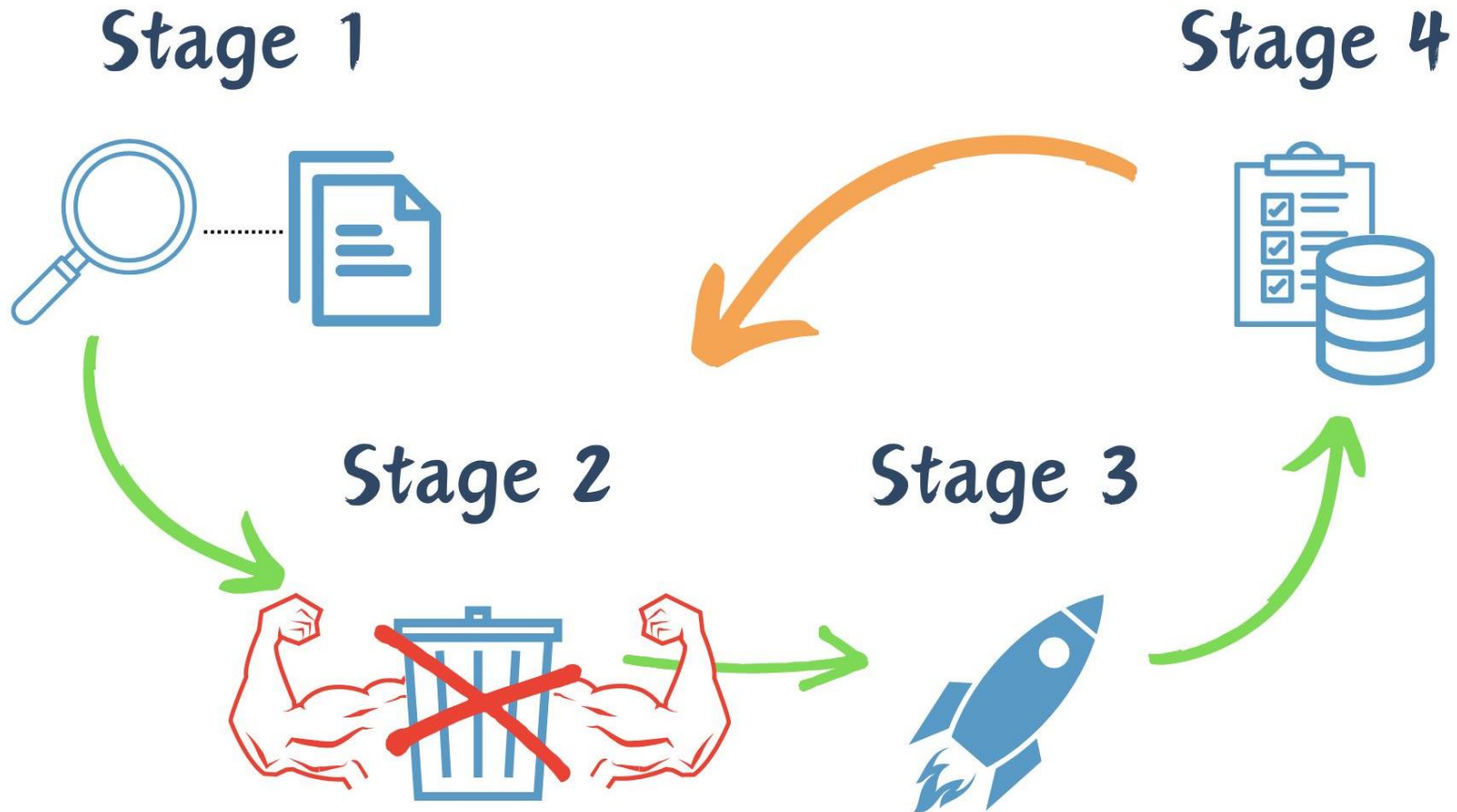
Gunther
Walklein



Arvis
Haversham

The new maintainers!

Automation Pipeline





Fetching external data in my Rego code

```
send_http_request(url) = http.send({  
    "url": url,  
    "method": "GET"  
})  
  
ok_to_delete := addresses if {  
    res := send_http_request("http://foo.bar/resources")  
    # compare planned deletions  
    # to list of resources marked for deletion  
}
```

The End



The code used in
this demo



Let's connect, I'm
friendly!

My videos on code
and cloud



Questions?

