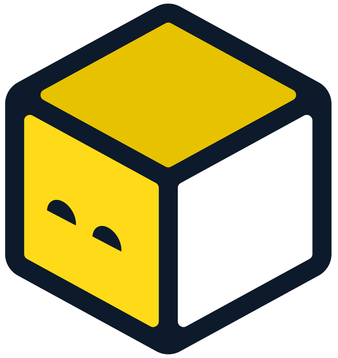
Secure and effective GitOps and Infrastructure-as-Code with OpenTofu



Wojciech Barczynski | VPE@Spacelift | TSC@OpenTofu

Whoami

- VP of Engineering at Spacelift
- Member of OpenTofu TSC
- Before:
  Director of Infra/Platform
  & Head of Engineering

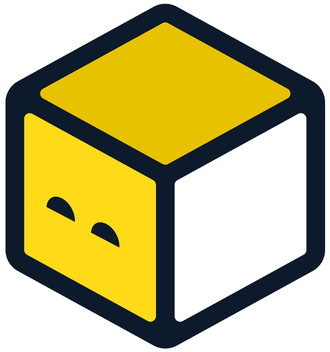
Whoami

- Infrastructure orchestrator
- IaC GitOps best practices
- Founding partner and top contributor to OpenTofu

spacelift

What is OpenTofu?

- Community-driven Open-source Terraform fork
- The Linux Foundation project
- Drop-in replacement
- Since September 2023



What is OpenTofu?

Declarative infrastructure-as-code (in Git):

```
variable "repository_name" {
  type    = string
  default = "my_app"
}

resource "github_repository" "my_repo" {
  name        = var.repository_name
  description = "My App repository created by OpenTofu!"
  visibility = "public"
}
```

What is OpenTofu?

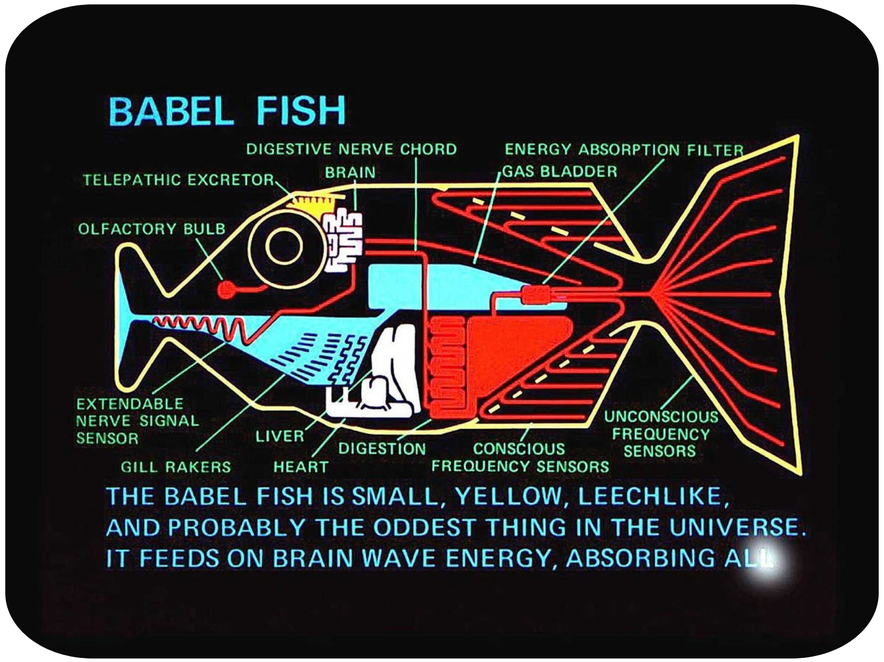
1. `tofu plan` – tell me what will happen (+, -, ~),
2. `tofu apply` – 🚀,
3. State file – snapshot of your infra.

What is OpenTofu?

- Plan – safety and shift left X,
- State – drift detection 🎉.

What is OpenTofu?

- Common language*
- For your cloud, tools, to your software
- Easy to extend (modules, providers, and functions)



Infrastructure-as-code projects

Use cases:

1. Security & control
2. Increase the velocity
3. Reduce cost
4. Collaboration

Infrastructure-as-code projects

Use cases:

5. Scaling up
6. Standarization
7. Opening up IaC* and democratization
8. Internal platforms

Pitfalls ⚠️

- (Remote) state management
- Large stacks & rigid infrastructure
- Running OpenTofu from your machine (Yolo)
- Reusable modules

Pitfalls ⚠️

- Not addressing the drift
- Reviewing every single chance
- Inconsistencies across stacks
- Outgrowing your tooling

1. State management

Basics:

- Remote with locks;
- (Encrypted) objects storage: AWS, GCP, and Azure;
- O(T)ACOS, for example, Spacelift, Env0, or Scalr.

1. State management

Client-side encryption (OpenTofu docs):

```
terraform{
  state_encryption {
    statefile {
      key_provider {

      }
      method {

      }
      enforced = true
    }
}
```

1. State management

Client-side encryption (OpenTofu docs):

- encryption passphrase,
- key management system support such as AWS KMS, GCP KMS, or OpenBao.

2. Large state

- Slow plan,
- Slow apply,
- Unhappy users,
- Leads to the rigid infrastructure 💀.

2. Rigid infrastructure

- Hard dependencies between workspaces/stacks;
- You need to be a super admin;
- Circular dependencies;
- Too opinioned modules from the start.

2. Rigid infrastructure

- `terragrunt run-all`

2. Rigid infrastructure

- makes it share the ownership,
- halts democratization,
- increases the risk of change.

2. Panacea

- Break your stack into smaller ones
- and connect them with...

2. Panacea

	Description	+	-
`data`	read from resources	Easy to implement	Slow
key/value;	my default	Fast	C. Deps.
O(T)ACOS env vars	k/v overlay Context	Reusablity	Tracing cycles
Graph /workflow	Spacelift Dependencies	Visible dependencies	Not standard

3. Running IaC from your laptop

https://twitter.com/mtcderek

- Uff, luckily it was my test env
- ...
- ...
- 🚨

3. Running IaC from your laptop

Options depending on the IaC maturity:

- Generic CI/CD, e.g., OpenTofu GitHub action;
- Atlantis (open source);
- O(T)ACOS.

3. Shift-left X

1. `tofu plan` - you can validate the changes before apply.
2. shift-left X, where X = best practises, security, and policies.

3. Shift-left X

In your CI/CD or git .pre-commit:

- Linters: `tofu fmt` & `tflint`;
- Security: `tfsec`, `kics`, or `checkov`;
- Cost estimation: `tf-cost-estimation` & `infracost`.

4. Reusable Modules

1. Similar to internal application libraries,
2. We want to help...,
3. Too opinioned modules too early,
4. Lack of module test cases.

5. Reusable Providers

Provider-defined functions (go):

```
terraform {
  required_providers {
    myhelper = {
      # yantrio/helpers registers a function named "echo"
      source = "yantrio/helpers"
    }
  }
}

locals {
  myval = provider::myhelper::echo("Hello Functions!")
}
```

5. Reusable Providers

Two experimentals providers available:

- terraform-provider-lua
- terraform-provider-go

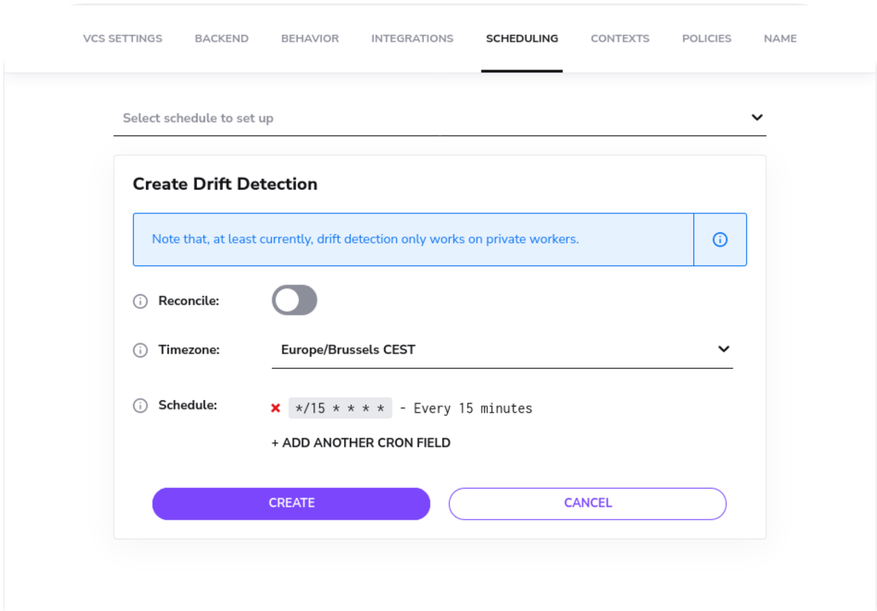
A powerful that lets you use the general purpose languages.

5. Not addressing the drift

- Thread to your control and security,
- Not only clickups,
- Only matter of time.

6. Drift detection

- Catch ClickOps;
- Overlapping IaC configurations;
- Must for the security and control.



6. Drift detection

- Ensure your rarely touched project, still works;
- Clean up after incidents;
- Track a migration from ClickOps to IaC.

7. Reviewing every single chance

- Bringing your best engineers to review every change is not effective;
- You just move a bottleneck somethere else.

7. Policies and Guardrails

Policies with conftest and Rego:

- Warning on deletion or recreation
- Enforce tagging
- Reduce blast radius
- Based on 3rd party tools, e.g., tfsec or checkov.

Open Policy Agent

7. Decision automation

Approval policies (from library):

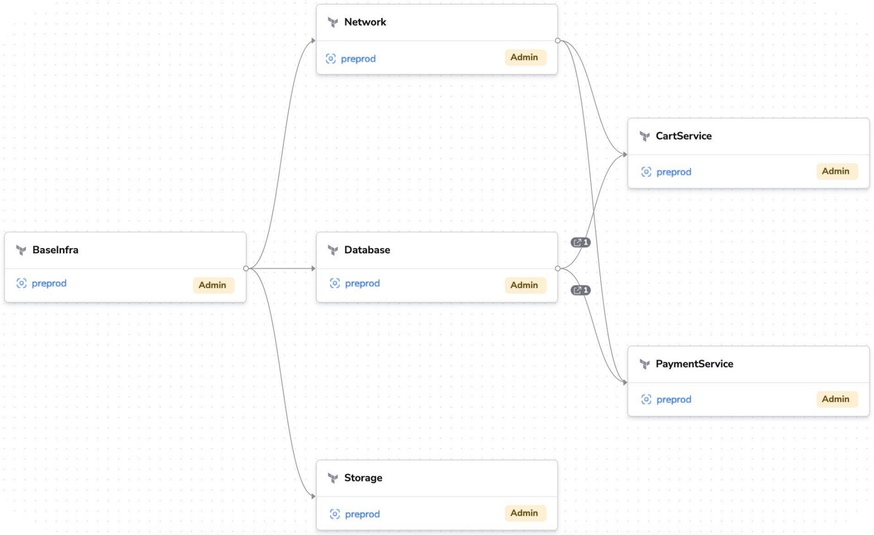
```
approval_list := [
  "aws_iam_access_key",
  "aws_security_group"]

...

approve {
  security_approval
  count(input.reviews.current.approvals) > 0
}
```

8. Inconsistencies across stacks

- Model the deps in your pipeline;
- Drift detection;
- terragrunt;
- TACOS.



9. Outgrowing your tooling

1. Local execution;
2. Basic automation with generic CI/CD;
3. GitOps;
4. Decision automation;
5. Platforms and API-based workflows.

9. Outgrowing your tooling

Common pattern:

- Take a simple CI/CD tool;
- Extend it to bring advanced capabilities;
- Keep investing even if bringing new features takes ages.

Security - Bonus

Do you know that nothing prevents `tofu plan` to apply changes?

Security - Bonus

Do you know that nothing prevents `tofu plan` to apply changes?

Solution: read-only role for plans and write role for applies.

Summary

- Well executed IaC projects benefits both engineering and business;
- Assume you will open your OpenTofu IaC to others;
- Leverage the rich OpenTofu and Terraform ecosystem.

Summary

OpenTofu:

- State encryption;
- Provider-defined functions in Lua and Golang;
- Coming in 1.8.x: variables evaluation prior to module configuration.

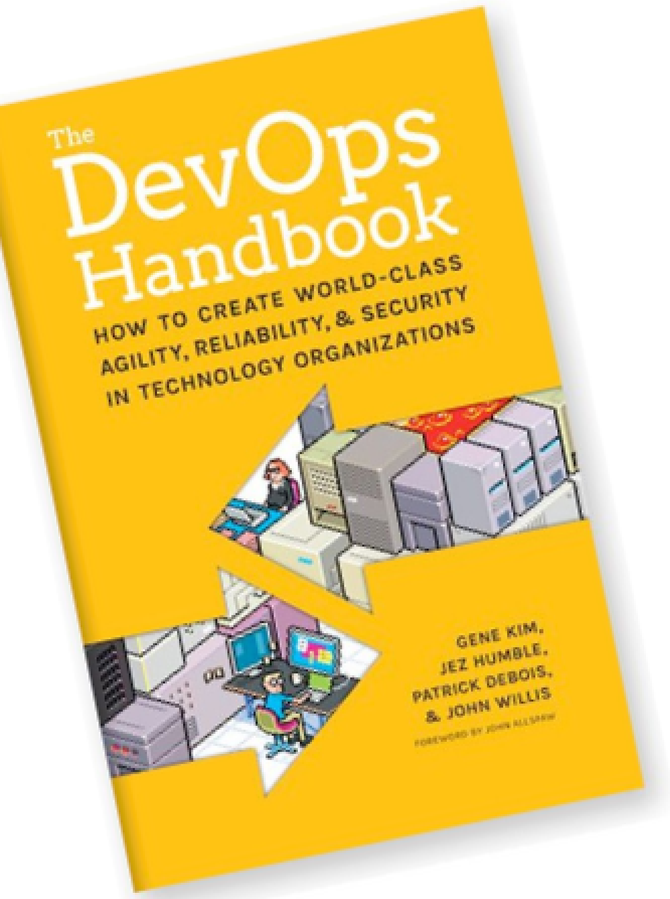
It is a community-driven project, join the slack, give feedback and share your ideas.

Questions?

github.com/wojciech12/talks

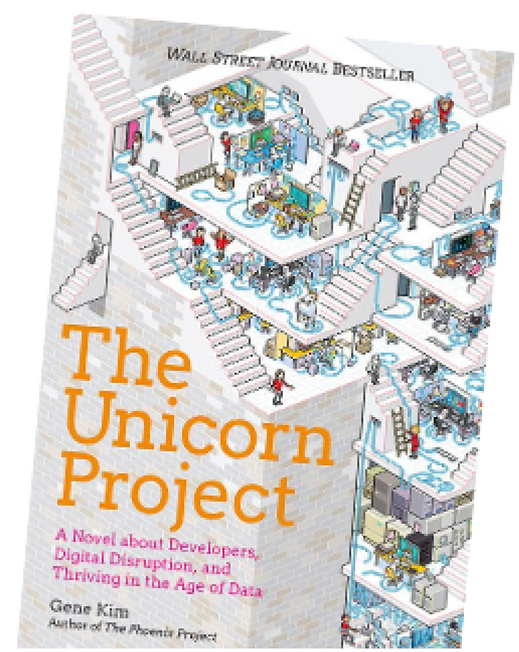
Backup Slides

Recommended read



1 Concrete, do A, do B,
because C
(DevOps here as culture ☺)

2 The first 4 chapters,
the rest if you have time.



3

A story,
not as straight forward
as (1)

Rabbit holes everywhere...

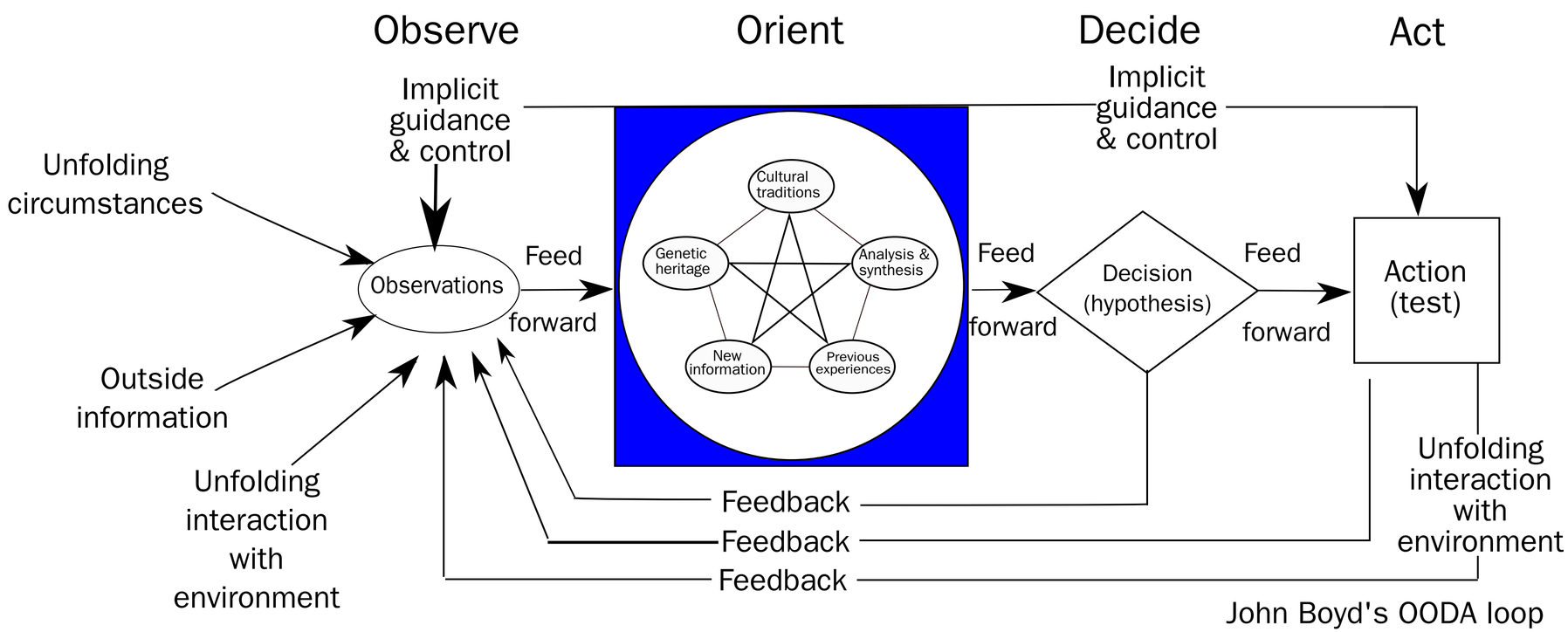
Approach:

- The iteration, decision, and deliver;
- As soon as possible to get into the cycle Patch Patch Patch.

Alternative take:

- Tracer bullet development;
- Lean v1/v2.

OODA



John Boyd's OODA loop