Jess Males
SysAdmin for 20+ years

- give me a terminal
- bash
- give me an editor
- vim
- video game?
- doom-ascii
AKA - a greybeard
Grey Beard

Stereotype:
- Knows their tools
- Get off my lawn!
- Stuck in the past

Sources: True Grit, Paramount, 2010; Kill Bill, Miramax, 2003
So as not to be stuck in the past, and to really know the tool, I took a dive into the OpenTofu code base.
The OpenTofu source is very accessible.

If I can get it, you can too.
What it does
Infrastructure as code

```
$ cat main.tf
variable "revision" {
  default = 1
}

resource "terraform_data" "replacement" {
  input = var.revision
}

resource "terraform_data" "test" {
  lifecycle {
    replace_triggered_by = [terraform_data.replacement]
  }

  provisioner "local-exec" {
    command = "touch foo"
  }
}
```

```
$ tofu plan -out=plan.out
...

# terraform_data.replacement will be created
+ resource "terraform_data" "replacement" {
  + id = (known after apply)
  + input = 1
  + output = (known after apply)
}

# terraform_data.test will be created
+ resource "terraform_data" "test" {
  + id = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.
$ tofu apply plan.out
...

terraform_data.test: Creation complete after 0s [id=8a078c17-5400-32f9-82f5-b686bea13e99]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```
OpenTofu is a fork of Terraform that is open-source, community-driven and managed by the Linux Foundation.
Why?

August 10, 2023

HashiCorp announced that, from version 1.6 onwards, Terraform will no longer be released under a Mozilla Public License (MPL v2). Instead, it would use the non-open source BSL v1.1. license.

At this point, Terraform had been developed openly for more than nine years. There was no outreach to the community regarding this change.
Reaction

- "HashiCorp adopts Business Source License" - 632 points, 731 comments
  - https://news.ycombinator.com/item?id=37081306
- "OpenTF announces fork of Terraform" - 1711 points, 486 comments
  - https://news.ycombinator.com/item?id=37262440

context https://news.ycombinator.com/best on Mar 14, 2024, roughly 9:00 pm in the evening

average of 515 points and 262 comments per story
Competition is Healthy

- OpenTofu, as an upstart, must maintain compatibility to allow for adoption. But, they must offer compelling features to persuade users to switch.

- Terraform, as the incumbent, should innovate, in order to maintain their position.
Providers
AWS, GCP, AzureRM, DataDog, &c

- Did not get relicensed
- These should serve as a unifying factor, limiting drift.
<table>
<thead>
<tr>
<th>Terraform Release Date</th>
<th>Version</th>
<th>Time Between Releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/6/2024</td>
<td>v1.8-beta1</td>
<td>49 days, 7 weeks</td>
</tr>
<tr>
<td>1/17/2024</td>
<td>v1.7</td>
<td>105 days, 15 weeks</td>
</tr>
<tr>
<td>10/4/2023</td>
<td>v1.6</td>
<td>114 days, 16.3 weeks</td>
</tr>
<tr>
<td>6/12/2023</td>
<td>v1.5</td>
<td>96 days, 13.7 weeks</td>
</tr>
<tr>
<td>3/8/2023</td>
<td>v1.4</td>
<td>168 days, 24 weeks</td>
</tr>
<tr>
<td>9/21/2022</td>
<td>v1.3</td>
<td>126 days, 18 weeks</td>
</tr>
<tr>
<td>5/18/2022</td>
<td>v1.2</td>
<td>161 days, 23 weeks</td>
</tr>
<tr>
<td>12/8/2021</td>
<td>v1.1</td>
<td>183 days, 26.1 weeks</td>
</tr>
<tr>
<td>6/8/2021</td>
<td>v1.0</td>
<td>55 days, 7.9 weeks</td>
</tr>
<tr>
<td>4/14/2021</td>
<td>0.15</td>
<td>133 days, 19 weeks</td>
</tr>
<tr>
<td>12/2/2020</td>
<td>0.14</td>
<td>114 days, 16.3 weeks</td>
</tr>
<tr>
<td>8/10/2020</td>
<td>0.13</td>
<td>446 days, 63.7 weeks</td>
</tr>
<tr>
<td>5/22/2019</td>
<td>0.12</td>
<td>552 days, 78.9 weeks</td>
</tr>
<tr>
<td>11/16/2017</td>
<td>0.11</td>
<td></td>
</tr>
<tr>
<td>Average:</td>
<td></td>
<td>177 days, 25 weeks</td>
</tr>
</tbody>
</table>

*Take this with a grain of salt, semantic versioning makes no guarantees on number of releases, or amount of improvement per release.
Code Changes, Over Time

Comparing versions:

```bash
$ git diff terraform-v1.4.0 opentofu-v1.6.0
```
Crossing the Streams

$ git clone https://github.com/opentofu/opentofu
$ cd opentofu/
$ git remote rename origin opentofu
$ git branch -m opentofu-main

$ git remote add terraform https://github.com/hashicorp/terraform
$ git fetch terraform
$ git checkout -b terraform-main terraform/main

For educational purposes and study only. **DO NOT** copy code.
Crossing the Streams, part 2
Tags - Complete

$ # one time -- do not drop tags that are not present upstream
$ git config --local fetch.pruneTags false
$ # clear un-attributed tags
$ for tag in $(git tag --list | grep -v -E '^(opentofu|terraform)');
do git tag -d ${tag}; done

$ # for each source, fetch and rename tags
$ git fetch opentofu && for tag in $(git tag --list | grep -v -E
  '^(opentofu|terraform)'); do git tag opentofu-${tag} ${tag}; git tag
 -d ${tag} >/dev/null 2>/dev/null; done
$ git fetch terraform && for tag in $(git tag --list | grep -v -E
  '^(opentofu|terraform)'); do git tag terraform-${tag} ${tag}; git tag
 -d ${tag} >/dev/null 2>/dev/null; done

For educational purposes and study only. DO NOT copy code.
Crossing the Streams, part 3

Tags - Historical

```
$ # for opentofu, fetch and rename tags as previously
$ # for each source, fetch and rename tags
$ git fetch terraform --tags \\n  '^refs/tags/v1.6*' '^refs/tags/v1.7*' \\n  '^refs/tags/v1.8*' '^refs/tags/v1.9'*
  
  # update exclusions as necessary
$ for tag in $(git tag --list | grep -v -E '^((opentofu|terraform))');
do git tag terraform-{$tag} ${tag}; git tag -d ${tag} >/dev/null 2>/dev/null; done
```

All code should be MPLv2 code; it was already in the repo, just untagged. Still, remember as a contributor:

Pay attention to copyright: please read the Developer Certificate of Origin [DCO], write the code yourself, avoid copy/paste. Disable or limit your AI coding assistant.

https://developercertificate.org/
Mental Model

Infrastructue: Compute Instance, Cloud DB

DAG: A, B, C, D, E, F, G

Code: Resource α, Data β, Resource γ
Looking from the Outside, In
$ TF_LOG=TRACE tofu plan -out=plan.out

trace options: TRACE, DEBUG, INFO, WARN or ERROR

increasing verbosity

$ TF_LOG_CORE=TRACE tofu plan -out=plan.out # just core/cli logs
$ TF_LOG_PROVIDER=TRACE tofu plan -out=plan.out # just provider logs

$ export TF_LOG_PATH=path/to/file
$ # logs appended to file, rather than written to stderr
Log Output

$ TF_LOG=TRACE tofu plan -out=plan.out 1> traces/out01_plan.txt 2> traces/trace01_plan.txt
$ TF_LOG=TRACE tofu apply plan.out 1> traces/out02_apply.txt 2> traces/trace02_apply.txt
Looking from the Inside, Out
Architecture

1. User commands
   - CLI (command package)
   - Backend.Operation

2. Configuration Loader
   - (configload.Loader)
   - configs.Config
   - Backend.Operation
   - Operation Arguments
     - (input variables, etc.)
     - opentf.Context

3. Graph Builder
   - (opentf.GraphBuilder.impl.)
   - opentf.Graph
   - Graph Walker
     - (ContextGraphWalker)
     - (for each graph vertex, in parallel)
     - opentf.GraphNodeEvalable.impl.
     - Vertex Evaluation
       - (details below)
CLI

- internal/command package
- mostly scaffolding
  - parses arguments
  - starts encryption
  - starts backend
  - create context
    - with a defer on a final unlock on State
Configuration Loader

- **tofu init processing**
- installs child modules into local environment
- while running, recursively loads all child modules to get fully interpolated `configs.Config` object*

* ok, mostly interpolated, as some objects are only known after a graph walk (e.g. `count=1`). These components are left as chunks of `hcl.body` or `hcl.expression`. 
Backend

- Workspaces are part of the Backend interface - you're always in one, even if it's default.

- An interface of Enhanced that provides plan/apply/refresh, but only Local, Remote, and Cloud go this far. Most backends are actually fronted by the `local.Local` backend, which performs this functionality on their behalf.
State Manager

- called by Backend
- `statemgr.Filesystem` (local)
- returns `tofu.Context`
Context

- internal/tofu/context.go
- points to root of DAG, and kicks off GraphWalking
- internal/tofu/hook_stop.go - stophooks
  - a tin-can and string into all subsequent graph-walking in order to signal ctrl-c, allowing some graceful save of state before a second hard kill.
GraphBuilder

- each resource gets a vertex in the graph
- 'vertex' and 'node' used interchangeably in the code, but 'vertex' preferred.
GraphWalker

- tofu.ContextGraphWalker
- EnterPath is called once for each module tofu.EvalContext
  - is per-module Context
  - keeps namespaces separate
Vertex

- processed concurrently (where dependencies allow)
- as such, there's lock (mutex) management for state.State
  - helper states.SyncState
- processing proceeds as a series of transforms (implementations of `tofu.GraphTransformer`) that mutate the graph
Transformers

look in `internal/tofu` for code

◆ **ConfigTransformer**: populates config from code, turning them into **Resources**

◆ **StateTransformer**: populates config from known state

◆ **ReferenceTransformer**: resolves dependencies adding edges between vertices

◆ **ProviderTransformer**: associates each **resource** with appropriate provider, as well as creating a dependency on the **provider**.
Write Your Own Provider
Version


- state data is strongly typed

Protocol version 5: https://github.com/hashicorp/terraform-provider-scaffolding

- state data is not typed, rather each attribute is an interface{} and types are asserted at runtime
Structure of a Provider

- Metadata method - associates each data source and resource to the provider
- Schema method - provider level configuration
- Configure method - configure shared clients
- DataSources method - enumerate the provider's data sources
- Resources method - enumerate the provider's resources
- + CRUD methods for the resources

OpenTofu Quirks

Acceptance tests need some overrides; the defaults assume a particular affiliation.

```
TF_ACC_TERRAFORM_PATH="/path/to/opentofu"
TF_ACC_PROVIDER_NAMESPACE="hashicorp"
TF_ACC_PROVIDER_HOST="registry.opentofu.org"
```

https://github.com/orgs/opentofu/discussions/975
Advice

Keep the provider as _____ as possible.

It should be a simple mapping between _____ and your ____.

Keep the provider thin by keeping __________ in the api.
Advice

UI
- Dependency Resolution
- State Tracking
- OpenTofu

Provider
- Thin Mapping

API
- Error Handling
- Resource Management
- Account/Identity Management
Follow the Leader

- [https://github.com/grafana/terraform-provider-grafana](https://github.com/grafana/terraform-provider-grafana)
  - updated, framework implementation, with remnants
  - repo size: 3 Mb, 180 go files, ~30k loc*

- [https://github.com/hashicorp/terraform-provider-aws](https://github.com/hashicorp/terraform-provider-aws)
  - solid, but large. prefers framework implementation, and solid tests
  - repo size: 200 mb, ~7800 go files, ~2M loc*

* cloc utility
Advice, when building

Straight out of the docs:

```
$ GOOS=linux GOARCH=amd64 CGO_ENABLED=0 go build -o tofu -v -buildvcs=false ./cmd/tofu
```

For the love of FSM, and all that is tomato sauced, no!

```
$ GOOS=linux GOARCH=amd64 CGO_ENABLED=0 go build -o tofu-dev -v -buildvcs=false ./cmd/tofu
```

Realizing that you're using your dev binary for production work will cost 3 hours and 38 minutes of troubleshooting that you'll never get back. Not that this happened to me, it was a coworker; yeah, that's the ticket.
Advice, for the lazy

Save yourself some cognitive hiccups tracking whether `init` needs to be run.

```bash
$ type ot
ot is a function
ot ()
{
    if (( $# == 0 )); then
        if ! [[ -d .terraform ]]; then
            tofu init;
        else
            tofu plan -out=plan.out;
        fi;
    else
        tofu "$@";
    fi
}
```
Next Steps

Source: https://opentofu.org/blog/
Golang

Chapter 3: Go Language Foundations is a great overview

2nd edition expected for Oct '24
OpenTofu Community

- slack - https://opentofu.org/slack/
- Issues Board - https://github.com/opentofu/opentofu/issues
Staying Informed

- Weekly Updates are kept in the repo
- Steering Committee
- Release Notes
  - https://github.com/opentofu/opentofu/releases
OpenTofu - Work in Progress

- state encryption
  - Encrypt state; plans too can be protected
  - multiple keys supported (so that different teams can maintain the same state
  - specify key directly, or through provider (e.g. AWS KMS)
- https://github.com/opentofu/opentofu/blob/main/docs/state_encryption.md
Contributing

- [https://github.com/opentofu/opentofu/issues/new/choose](https://github.com/opentofu/opentofu/issues/new/choose)
  - bugs
  - feature request
  - feature idea, and can describe it in detail? write an RFC
- minor bug fix? Submit a PR
- major refactor? Create an issue to discuss the larger implications and allow coordination
Questions?

Source: https://opentofu.org/blog/