HashiCorp Terraform

THE LINUX FOUNDATION OPEN SOURCE SUMMIT EUROPE

Deep dive with no fear

Viktor Turbinskii
@devopsv

THE LINUX FOUNDATION

Who Am I?

Victor Turbinsky

DevOps Engineer

at

Texuna

- Russian Engineer living in Cork, Ireland
- Started IT career in 2005, as System Administrator, same time studying in university.
- Networking/*BSD systems background
- Now - Systems engineer with common DevOps duties:
  - Infrastructure as a code using mostly Terraform, Ansible, Bash, Python
  - CI/CD with Jenkins
  - Docker, Kubernetes
  - Security and disaster recovery

Texuna

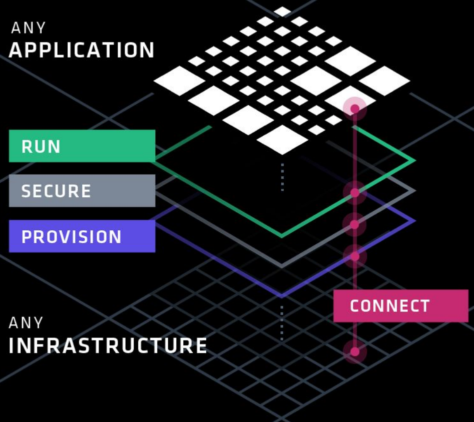
texuna.com

- Data management solutions
- Vendor neutral systems Integrator
- Data Warehouses
- Founded in London in 2000
- Some of our larger clients include:
  - The Department for Education
  - National College for Teaching and Leadership
  - Jisc (digital solutions for UK education and research)
  - Higher Education Funding Council for England.

Outline of the presentation

- Terraform overview
- Quick comparison with other tools
- Terraform Architecture overview
- From simple cases to complex adoption scenarios
- Providers, Provisioners, Modules and Terraform Modules Registry
- Debug, security options, sensitive managing, gotchas
- HCL v2 - HashiCorp configuration language
- Power of community and Tools around
- Learning material, How and where to dive deeper ?
- Summary, Q & A

HashiCorp Suite



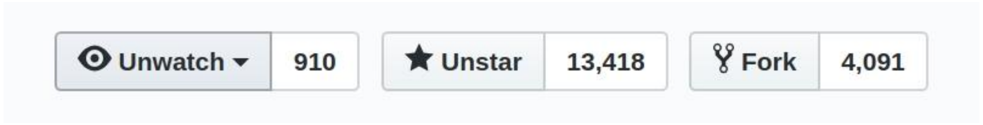
OPEN SOURCE

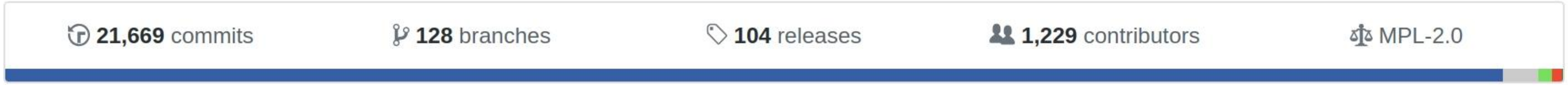
- HashiCorp Nomad
- HashiCorp Vault
- HashiCorp Terraform
- HashiCorp Packer
- HashiCorp Vagrant
- HashiCorp Consul

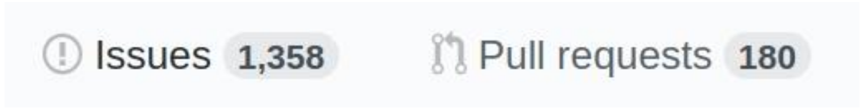
Terraform

- A provisioning declarative tool that based on Infrastructure as a Code paradigm
- Uses own syntax - HCL (Hashicorp Configuration Language)
- Written in Golang.
- Helps to evolve you infrastructure, safely and predictably
- Applies Graph Theory to IaaC
- Terraform is a multipurpose composition tool:
  - Composes multiple tiers (SaaS/PaaS/IaaS)
  - A plugin-based architecture model
- Open source. Backed by Hashicorp company and Hashicorp Tao (Guide/Principles/Design)

Terraform: GitHub Stats


Unwatch 910
Unstar 13,418
Fork 4,091


21,669 commits
128 branches
104 releases
1,229 contributors
MPL-2.0


Issues 1,358
Pull requests 180

Source: GitHub
https://github.com/hashicorp/terraform

Other tools

- Cloudformation, Heat, etc.
- Ansible, Chef, Puppet, etc.
- Boto, fog, apache-libcloud, etc.
- Custom tooling and scripting

AWS Cloudformation / OpenStack Orchestration (Heat)

- AWS Locked-in
- Initial release in 2011
- Sources hidden behind a scene
- AWS Managed Service / Free
- Cloudformation Designer
  - Drag-and-drop interface.
- Json, Yaml (since 2016)
- Rollback actions for stack updates
- Change sets (since 2016)

- Open source
- Initial release around 2012
- Heat provides CloudFormation-compatible Query API for Openstack
- UI: Heat Dashboard
- Yaml

Ansible, Chef, Puppet, etc

- Created for the purpose to be a configuration management tool.
- Suggestion: don’t try to mix configuration management and resource orchestration.
- Different approaches:
  - Declarative: Puppet, Salt
  - Imperative: Ansible, Chef
- The steep learning curve if you want to use orchestration capabilities of some of these tools.
- Different languages and approaches:
  - Chef - Ruby
  - Puppet - Json-like syntax / Ruby
  - Ansible - Yaml

Boto, fog, apache-libcloud, etc.

- low-level access to APIs
- Some libs focused on specific cloud providers, others provide common interface for few different clouds
- Inspires to create custom tooling

Custom tooling and scripting

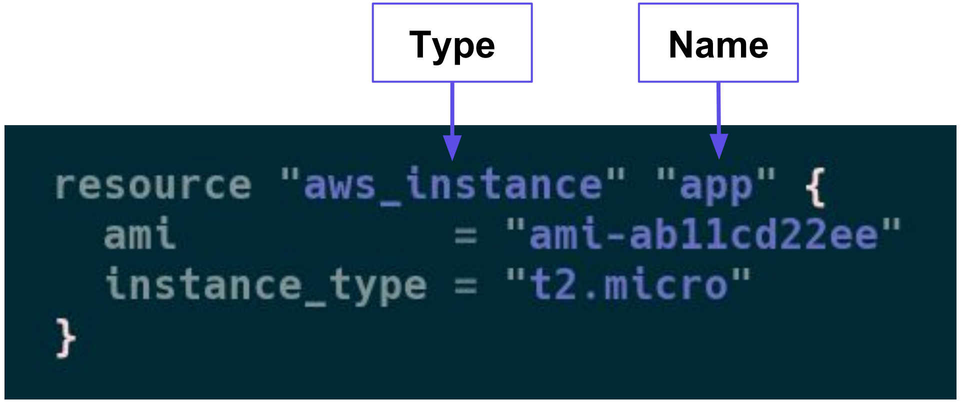
- Error-prone and tedious
- Requires many human-hours
- The minimum viable features
- Slowness or impossibility to evolve, adopt to quickly changing environments

Terraform is not a cloud agnostic tool

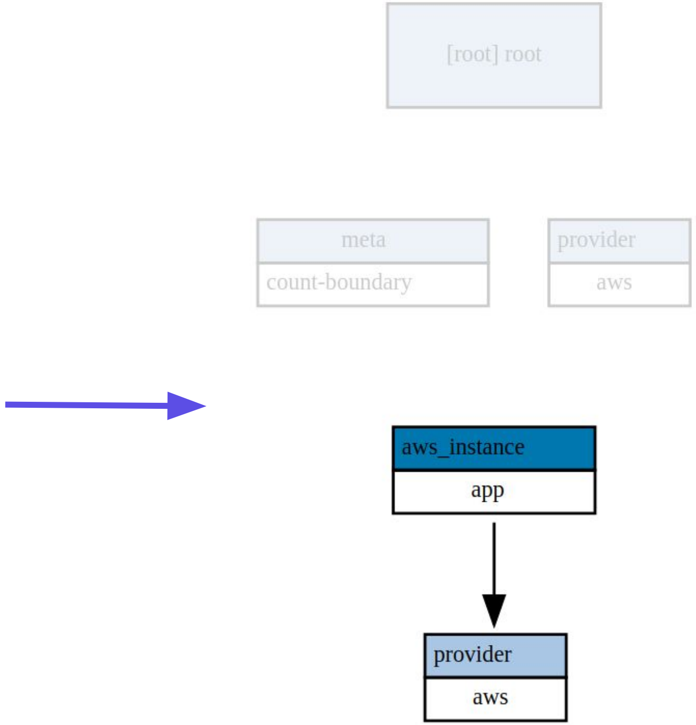
It’s not a magic wand that gives you power over all clouds and systems.

It embraces all major Cloud Providers and provides common language to orchestrate your infrastructure resources.

Terraform: Example (Simple resource)

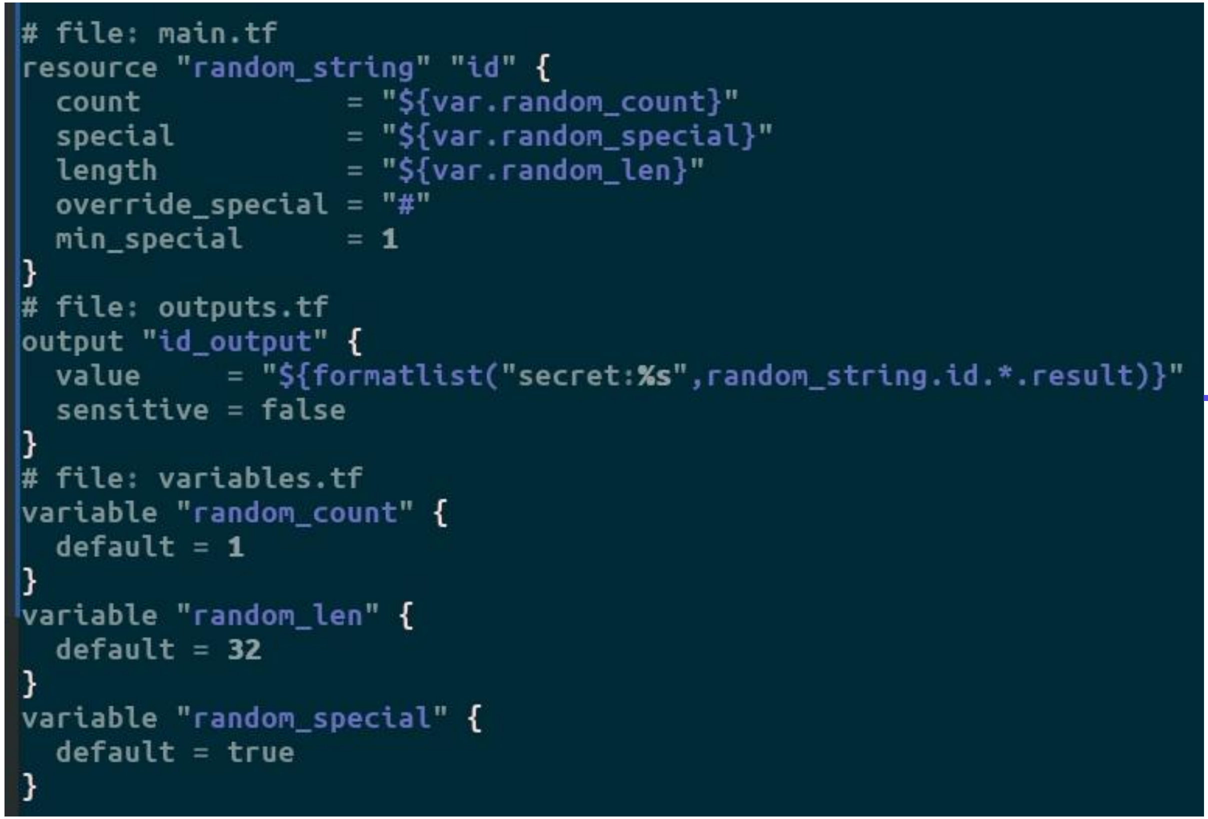


```
resource "aws_instance" "app" {
  ami           = "ami-ab11cd22ee"
  instance_type = "t2.micro"
}
```

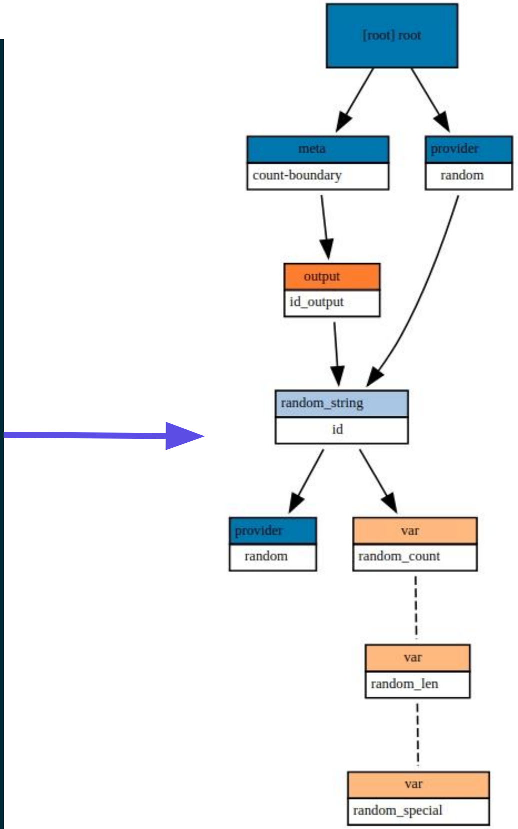


example

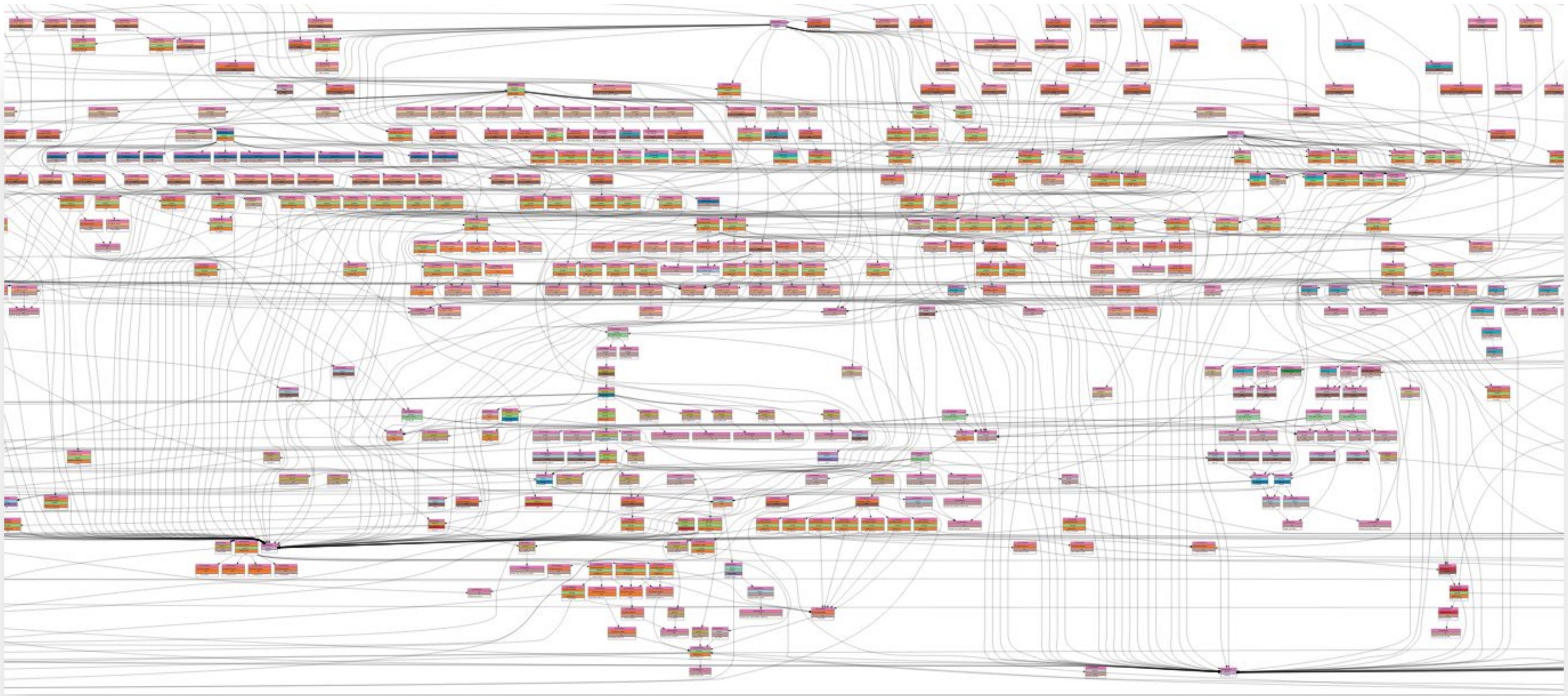
Terraform: Example (Simple local resource)



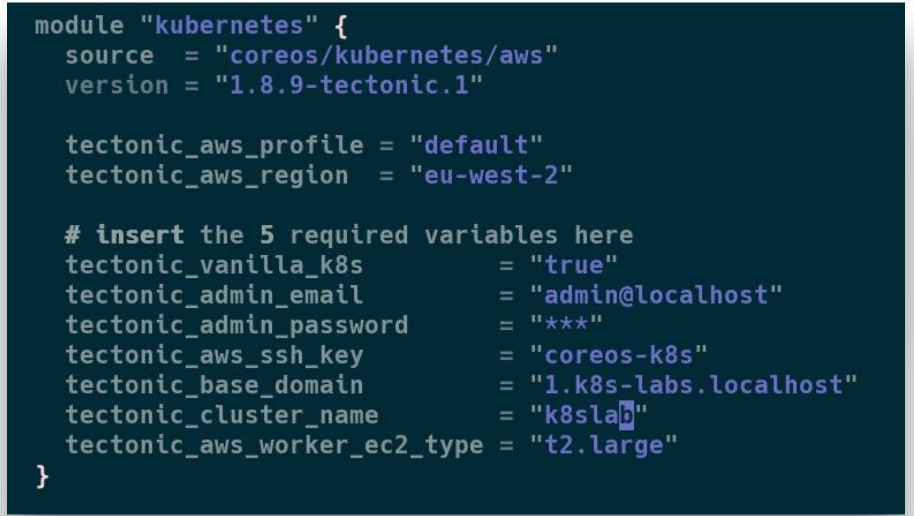
```
# file: main.tf
resource "random_string" "id" {
  count              = "${var.random_count}"
  special            = "${var.random_special}"
  length             = "${var.random_len}"
  override_special = "#"
  min_special        = 1
}
# file: outputs.tf
output "id_output" {
  value     = "${formatlist("secret:%s",random_string.id.*.result)}"
  sensitive = false
}
# file: variables.tf
variable "random_count" {
  default = 1
}
variable "random_len" {
  default = 32
}
variable "random_special" {
  default = true
}
```



“There ain't nothing up there
but pain and suffering
on a scale you can't even imagine.”



It's just a single Module!



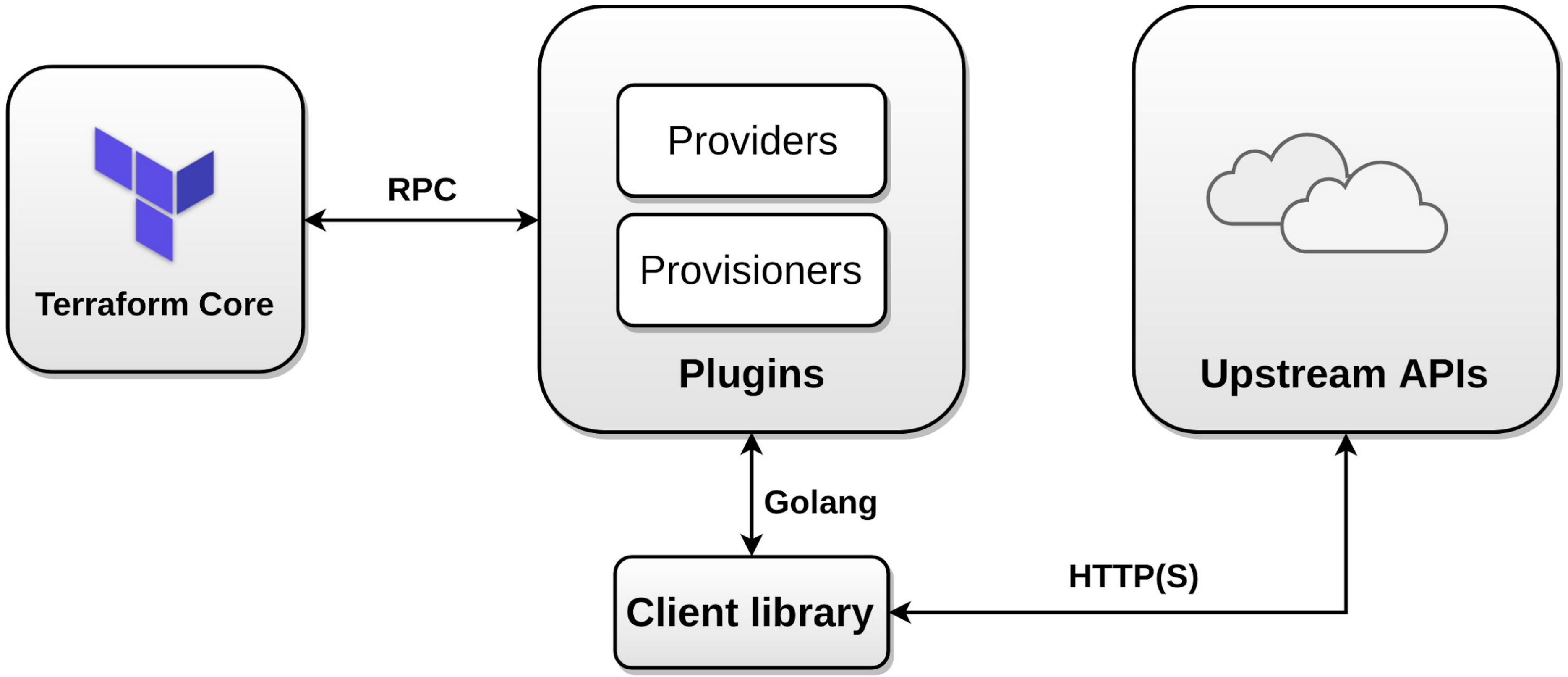
```
module "kubernetes" {
  source  = "coreos/kubernetes/aws"
  version = "1.8.9-tectonic.1"

  tectonic_aws_profile = "default"
  tectonic_aws_region  = "eu-west-2"

  # insert the 5 required variables here
  tectonic_vanilla_k8s         = "true"
  tectonic_admin_email         = "admin@localhost"
  tectonic_admin_password      = "***"
  tectonic_aws_ssh_key         = "coreos-k8s"
  tectonic_base_domain         = "1.k8s-labs.localhost"
  tectonic_cluster_name        = "k8slab"
  tectonic_aws_worker_ec2_type = "t2.large"
}
```

Terraform Module Registry

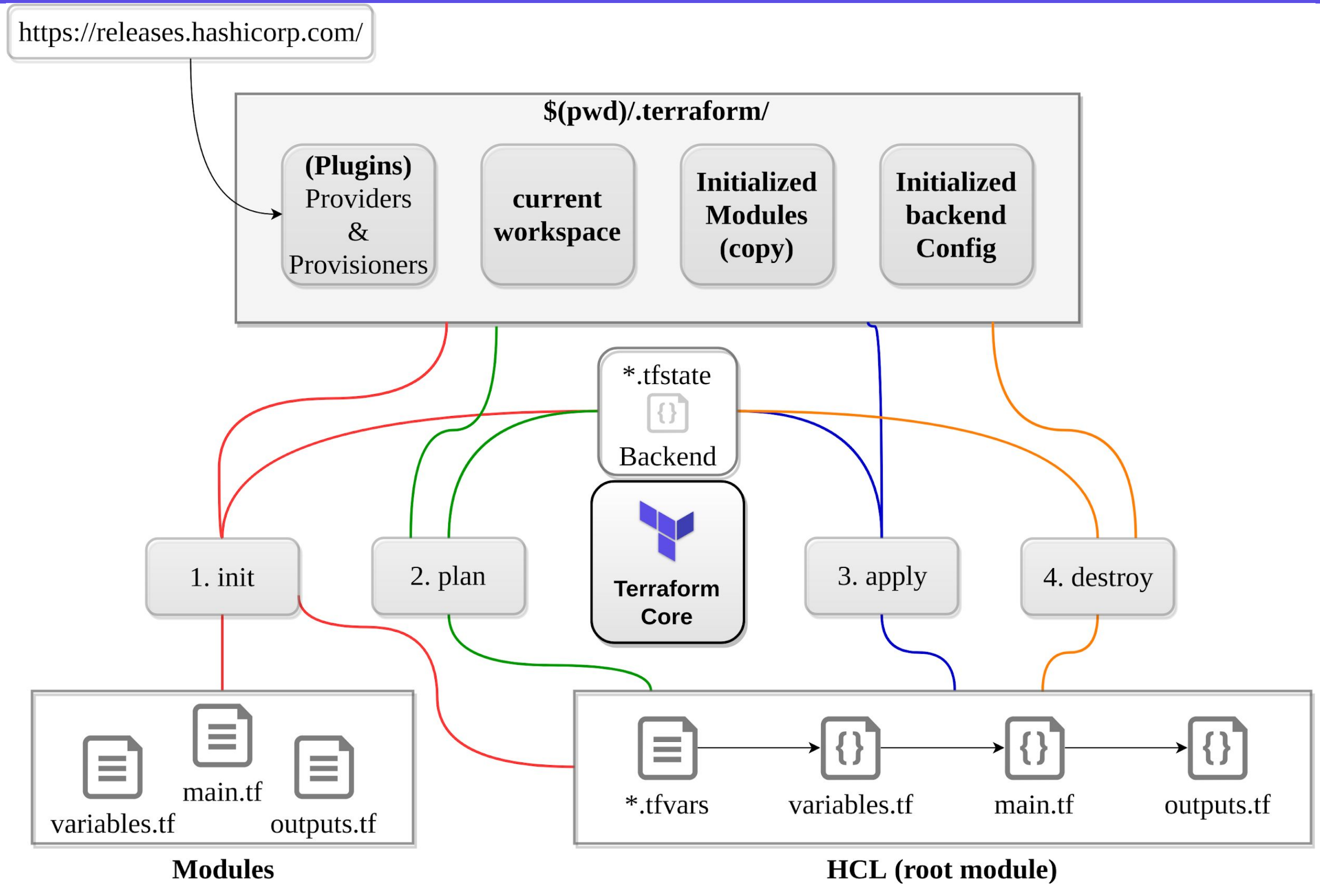
Discover modules for common infrastructure configurations for any provider

https://github.com/coreos/terraform-aws-kubernetes

Architecture



Simple workflow



example

Terraform Core: Init

1. This command will never delete your existing configuration or state.
2. Checkpoint → https://checkpoint.hashicorp.com/
3. .terraformrc → enable plugin_cache_dir, disable checkpoint
4. Parsing configurations, syntax check
5. Checking for provisioners/providers (by precedence, only once)→
   “.”, terraform_bin_dir, terraform.d/plugins/linux_amd64
   .terraform/plugins/linux_amd64
6. File lock.json contains sha-512 plugin hashes (.terraform)
7. Loading backend config (if it’s available, local instead)
   Backend Initialization: Storage for terraform state file.

Terraform Core: Plan + Apply

1. Starting Plugins: Provisioners/Providers
2. Building graph
   a. Terraform core traverses each vertex and requests each provider using parallelism
3. Providers syntax check: resource validation
4. If backend == <nil>, use local
5. If “-out file.plan” provided - save to file - the file is not encrypted
6. Terraform Core calculates the difference between the last-known state and the current state
7. Presents this difference as the output of the terraform plan operation to user in their terminal

Terraform Core: Destroy

1. Measure twice, cut once
2. Consider -target flag
3. Avoid run on production
4. No “Retain” flag - Remove resource from state file instead
5. terraform destroy tries to evaluate outputs that can refer to non existing resources #18026
6. prevent_destroy should let you succeed #3874
7. You can’t destroy a single resource with count in the list

Terraform Backends

- Locking
- Workspaces (former known as environments)
- Encryption at rest
- Versioning
- Note: Backend configuration doesn’t support interpolations.

HashiCorp Consul

etcd

Azure Blob

AWS S3

Terraform ENTERPRISE

Google Cloud Storage

Terraform: Providers (Plugins)

125+ infrastructure providers

Major Cloud Partners

aws

Google Cloud Platform

Azure

ORACLE

Alibaba Cloud

vmware

Terraform: Providers (Plugins)

- Abstraction above the upstream API
- Invoke only upstream APIs for the basic CRUD operations
- Providers are unaware of anything related to configuration loading, graph theory, etc.
- Consume an external client library (don’t try to implement client library itself, modularize)
  - aws-sdk-go
  - azure-sdk-for-go
  - k8s.io/apimachinery
  - k8s.io/client-go
  - ...

Terraform: Providers (Plugins)

Can be integrated with any API using providers framework

- Note: Terraform Docs → Extending Terraform → Writing Custom Providers

- GitLab
- GitHub
- BitBucket

- OpenFaaS
- **OpenAPI**
- Generic Rest API
- Stateful

- DNS
- Palo Alto Networks
- F5 BIG-IP

- Docker
- Kubernetes
- Nomad
- Consul
- Vault
- Terraform :)

- Template
- Random
- Null
- External (escape hatch)
- Archive

- NewRelic
- Datadog
- PagerDuty

- Digital Ocean
- Fastly
- OpenStack
- Heroku

Terraform Provisioners

- Run code locally or remotely on resource creation
- Resource is tainted if provisioning failed. (next apply it will be re-created
- You can run code on deletion. If it fails - resources are not removed

CHEF habitat

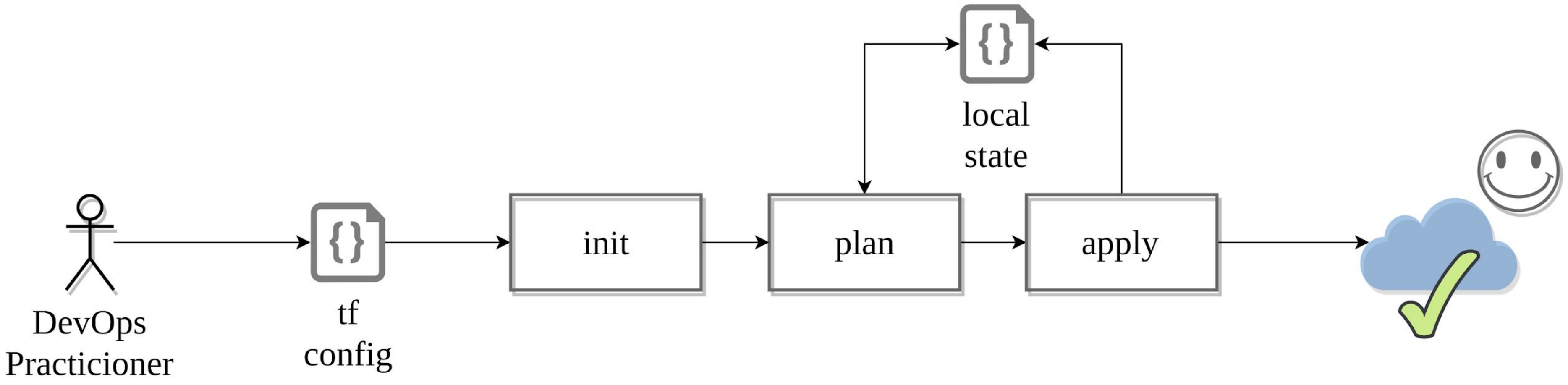
SALTSTACK masterless

ANSIBLE

null_resource +
- file
- local-exec
- remote-exec

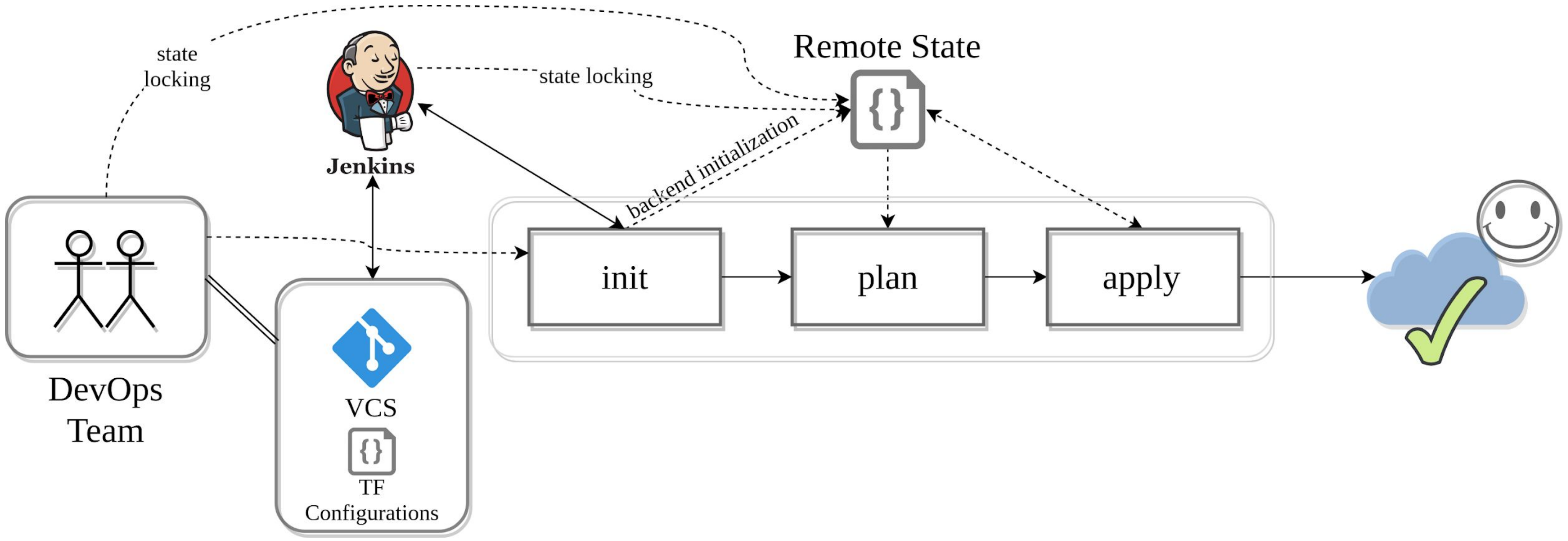
Workflow: Adoption stages

Single
contributor



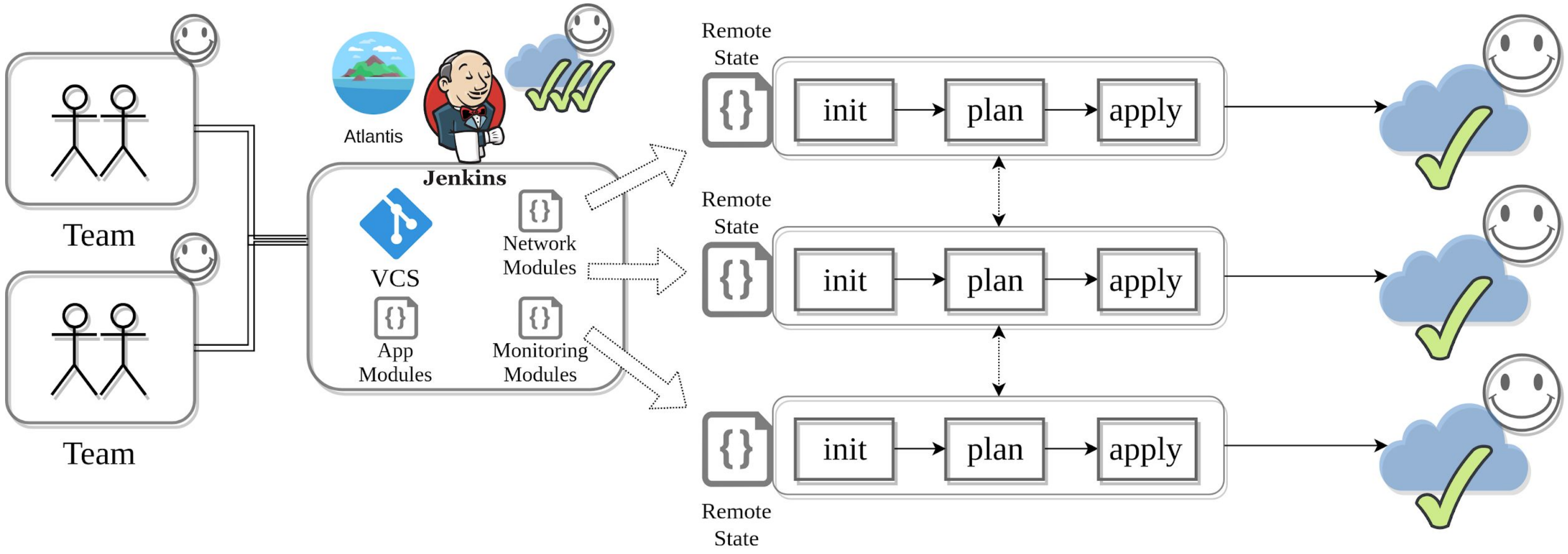
Workflow: Adoption stages

Team Collaboration



Workflow: Adoption stages

Multiple Teams



Workflow: Modules

- Code reuse
- Apply versioning
- Use version constraints
- Store code remotely
- Easier testing
- Encapsulation
- Use and contribute to Module Registry

Terraform Module Registry

Discover modules for common infrastructure configurations for any provider

Workflow: Modules

What is the good terraform module?

- Clean and flexible code
- Well presented default values
- Covered with tests
- Examples
- Documentation
- Changelog
- Secure

☺ Do not overload modules with features:
Terraform does cloning everything

Debug

- TF_FORK=0

  add an environment variable to prevent forking #8795
- TF_LOG → TRACE, DEBUG, INFO, WARN or ERROR
- TF_LOG_PATH
- Found a bug? Report to the right place:
  - Check GitHub, it’s highly likely a known bug/”feature”
  - https://github.com/hashicorp/terraform - Core Issues
  - https://github.com/terraform-providers - Provider Plugins
  - Check Golang SDK’s bugs
  - Check Cloud provider documentation
- Don’t forget to obfuscate your crash log :)
- Use delve debugger to learn how Terraform core works!

Terraform state file

1. Backup your state files + use Versioning and Encryption
2. Do Not edit manually!
3. Main Keys: cat terraform.tfstate.backup | jq 'keys'
    a. "lineage" - Unique ID, persists after initialization
    b. "modules" - Main section
    c. "serial" - Increment number
    d. "terraform_version" - Implicit constraint
    e. "version" - state format version
4. Use “terraform state” command
    a. mv - to move/rename modules
    b. rm - to safely remove resource from the state. (destroy/retain like)
    c. pull - to observe current remote state
    d. list & show - to write/debug modules

Terraform tips

1. Use terraform console
   a. echo "random_string.new.result" | terraform console
2. Use workspaces for simple scenarios
3. Isolate state files and don’t use workspaces :)
4. To review output from terraform modules:

   terraform output -module=mymodule
5. My state is changed every time when I’m running terraform with different users! (binary files/lambda functions)
   a. substr("${path.module}"/, length(path.cwd) + 1, -1)
   b. ignore_changes = ["filename"]

Terraform: common workflow issues

1. Mess up with workspaces
2. Hard-coded values
3. Not following naming convention (tags)
4. TF can’t detect changes to computed values
5. Renaming modules, resources
6. Double references
7. Syntax problems
8. Variable “somevar” should be type map, got list
9. Timeouts
10. Permissions

Terraform: Sensitive information

1. terraform plan "-out plan-latest" - is not secured
2. terraform state - not secured.
   a. Encryption on backend at rest
   b. terraform pull - exposes sensitive
   c. use data sources - grant only what you need
3. Data remote state - Not possible to expose just single or few outputs
4. Sensitive output
5. Encrypt tfvars
6. terraform output sensitive = true
   a. seems ok? remote_secured = <sensitive>
   b. terraform refresh → exposed, remote_secured = 79e6

Terraform: Sensitive information

1. How to handle secrets in state file?
   a. Terrahelp - https://github.com/opencredo/terrahelp
   b. Don’t store secrets :) Use:
      i. AWS/Google Cloud/Azure Key Vault/ etc. KMS -like + user-data mechanisms
      ii. AWS System Manager Parameter store
      iii. AWS Secrets manager
      iv. Use resource Roles
      v. If set master-password for DB service - change it after creation.
2. Secure state at rest using backend built-in encryption
3. Secure tfvars and other project/module specific information with:
   a. pass - The password store - https://www.passwordstore.org/
   b. git-crypt - https://github.com/AGWA/git-crypt

Terraform: Gotchas?

1. output and values don’t support count
2. How to output resource with count 0 (ugliest hell)
    a. https://github.com/hashicorp/terraform/issues/16726
    b. https://github.com/hashicorp/terraform/issues/17425
3. Setup caching and disable checkpoint in terraformrc
4. Use autocomplete and zsh
5. Use Constraints: it is recommended to constrain the acceptable provider versions via configuration, to ensure that new versions with breaking changes will not be automatically installed by terraform init in future
    a. Use constraints for everything !

Terraform: Gotchas?

1. Do not overuse “Depends_on”
   a. Use implicit dependencies via interpolation expressions
   b. Explicit dependencies are required rarely, often with null-resources/custom providers/etc.
2. Do not overuse terraform import (at least for now - 0.11/0.12)
   a. better is to create resources from scratch
   b. it doesn’t generate code for you
3. Share some parts of infrastructure using “data terraform_remote_state"
   a. Consider to use data resources for the same purpose
4. Do not overuse “workspaces” (former environments)
   a. they don’t have straightforward workflow
   b. you can’t use different backends
5. Enjoy clean code! Automate it: terraform fmt, pre-commit-terraform
   https://github.com/antonbabenko/pre-commit-terraform

Terraform: Gotchas?

1. Data sources can lock tfstate
2. Overrides: *override.tf - can be used to temporarily adjust your infra in CI
3. Modules don’t have count, but you can have variable and count internally in the module.
4. Terraform extensively uses TempFile() to store temp data during the run-time:
   /tmp/terraform-log##########
   /tmp/state-*
5. Prevent_Destroy behaviour:
   set prevent_destroy - works? Try to remove resource… (don’t…)
   Use “terraform state rm” type.resource.name

Terraform challenges and how can you help?

- More Providers: Networking, New Cloud Service Providers, On-premise systems
- Improving quality of current providers, new features, testing.
- Modules
- Importing resources
- New provisioners support
- Storing sensitive information
- Splitting monoliths → More tools

Long winding road to 0.12

0.11.x → 0.12.x

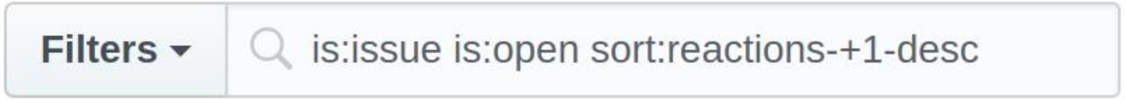
- Modules attributes:
  - "providers" - provider inheritance for modules
  - "version"   - constraints
- Interactive “terraform apply”
- Interpolation improvements, new features:
  - timeadd
  - rsadecrypt
- Myriad bug fixes
- Improvements:
  - Backends
  - CLI
  - Provisioners
- Many issues are on-hold till 0.12 version is released

Terraform: 0.12 New Features

- Better error messages
- Reliable JSON Syntax - 1:1 mapping to Json
  - Comments in JSON
- Template Syntax Improvements
- Rich and Complex Value Types
  - Return Module resources as Object values
  - Maps of Maps? It’s possible!
- Conditionally Omitted Arguments
- Conditional Operator Improvements
- Splat Operator
- For and For-Each - Finally! - For nested blocks!

Terraform issues:

- Support count in resource fields #7034 (For / For-Each → 0.12)
- depends_on cannot be used in a module #10462
  - Proposal: Module-aware explicit dependencies #17101 (freezed, ~ 0.12+)
- Allow using lists and maps with conditionals #12453 (Conditionals → 0.12)
- Support the count parameter for modules #953 (Breaking change,freezed)
- Support for nested maps in variables #2114 (Complex types → 0.12)
- Data sources should allow empty results without failing #16380
- allow `-target` to accept globs #2182 (Thumb up!)
- Storing sensitive values in state files #516 (Vault integration ? Thumbs up !)



```
is:issue is:open sort:reactions-+1-desc
```

👍

Helper Tools around:

- Reformat the output of terraform plan to be easier to read and understand.
  https://github.com/coinbase/terraform-landscape
- Export existing AWS resources to Terraform style (tf, tfstate)
  https://github.com/dtan4/terraforming
- Terraform version manager
  https://github.com/Zordrak/tfenv
- Generate documentation from Terraform modules
  https://github.com/segmentio/terraform-docs
- Detect errors that can not be detected by terraform plan
  https://github.com/wata727/tflint

Terraform: Interactive Graph visualizations

Blast Radius

https://28mm.github.io/blast-radius-docs/

https://github.com/28mm/blast-radius

Author	: Patrick McMurchie
Licence	: MIT

Terraform wrappers

gruntwork.io

Terragrunt

https://github.com/gruntwork-io/terragrunt

- **Makefiles**
- **Bash wrappers**
- **Python wrappers**
- **???**

Test your code

gruntwork.io

Terratest

https://github.com/gruntwork-io/terratest

terraform-compliance

https://github.com/eerkunt/terraform-compliance

Kitchen Terraform

https://github.com/newcontext-oss/kitchen-terraform

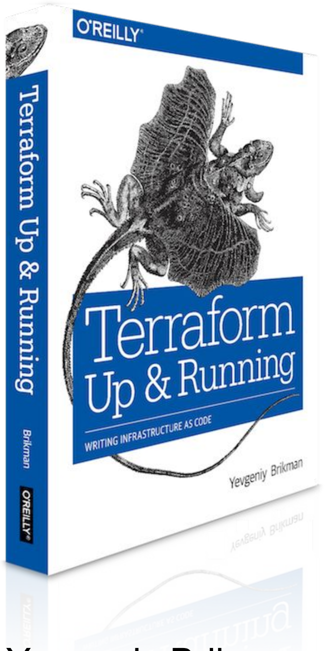
Automation and Safety!

Atlantis

https://www.runatlantis.io/
https://github.com/runatlantis/atlantis

- Empower your Developers collaborate on IaaC
  and be the part of DevOps
- Avoid mistakes
- Audit Logs
- Compliance
- Doesn’t break you workflow
- Golang and webhooks under the hood

Source of Knowledge



Yevgeniy Brikman
https://blog.gruntwork.io

awesome
https://github.com/shuaibiyy/awesome-terraform

HashiCorp

https://www.hashicorp.com/blog
https://www.terraform.io/docs/index.html

https://github.com/hashicorp/terraform

Source of Knowledge

Linux Academy

https://linuxacademy.com

- Managing Applications and Infrastructure with Terraform
- Deploying to AWS with Ansible and Terraform

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

Happy Terraforming!
Thank you!

Turbinskii Viktor

Texuna

twitter: @devopsv
github: @Victorion
email : sys.viktor@gmail.com

THE LINUX FOUNDATION