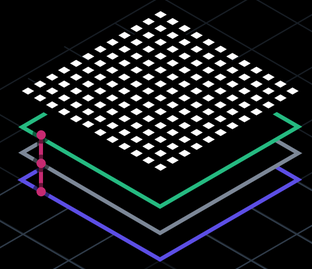
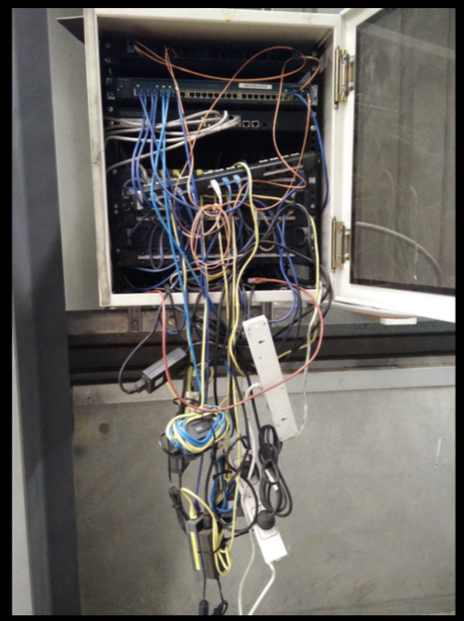
HashiCorp

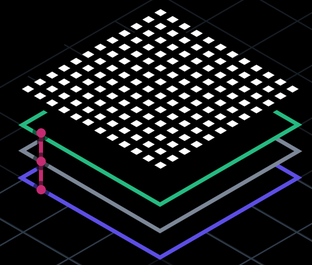
CLOUD INFRASTRUCTURE AUTOMATION



Carl Javier

Managing Consultant - Hashicorp Servian Lead
Loves snow and snowboarding
Former Network/Data Centre Engineer





HashiCorp

Everyone give a quick intro

- Name
- Tech Journey
- What do you want to learn today
- Fun Fact

HashiCorp

HashiCorp

Terraform

Agenda

Evolution of Infrastructure

Basic Configuration

Outputs & Variables

Configuration Format and Auto-Formatting

Modules, Provisioners, and Providers

Graph Theory

Meta Parameters

Terraform Enterprise: Remote State

Terraform Enterprise: GitHub Deploys

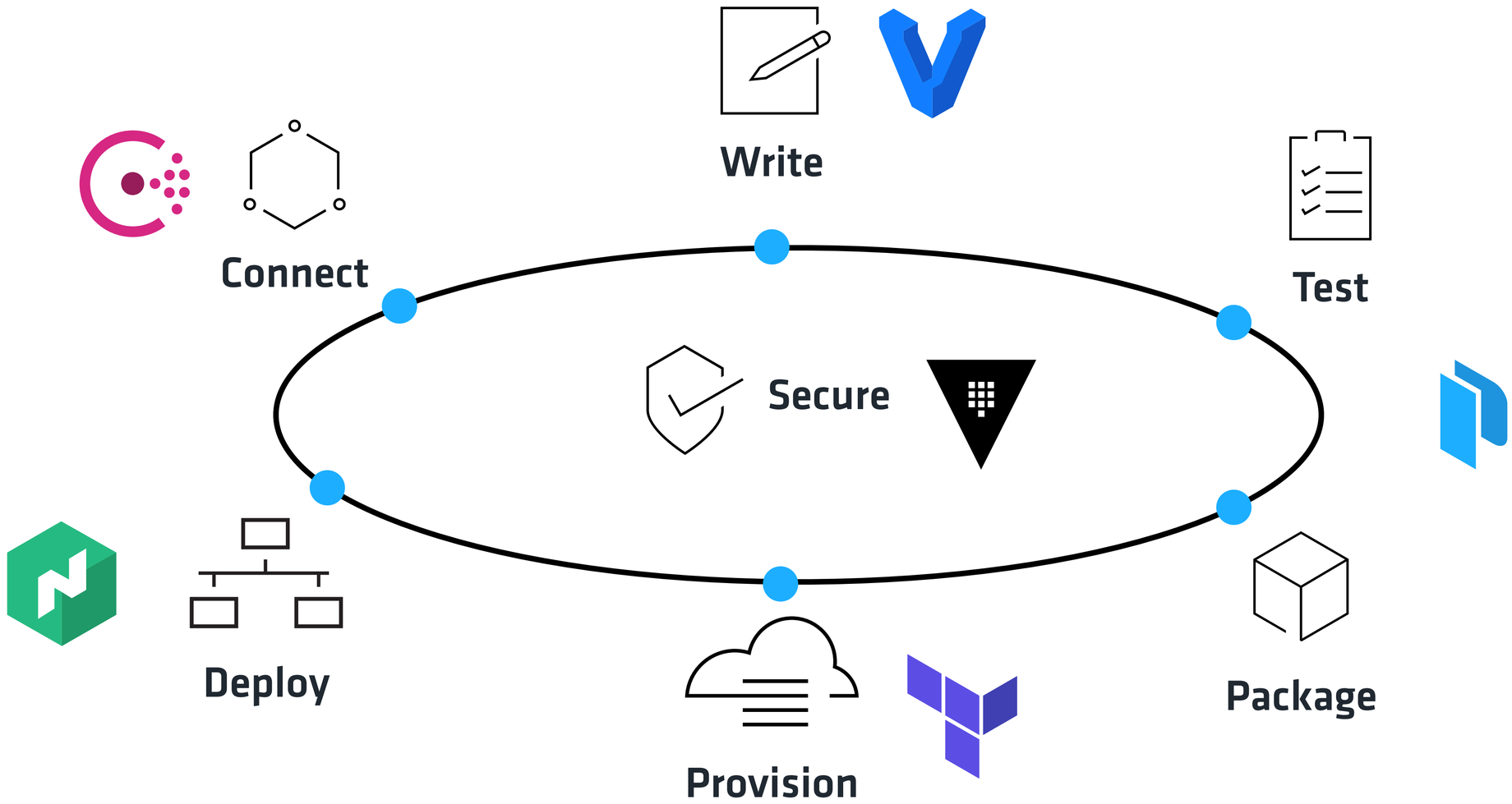
Destroy

HashiCorp

Introduction

HashiCorp

Application Delivery Lifecycle



https://www.hashicorp.com/resources/application-delivery-hashicorp

Terraform's Goals

Unify the view of resources using infrastructure as code

Support the modern data center (IaaS, PaaS, SaaS)

Expose a way for individuals and teams to safely and predictably change infrastructure

Provide a workflow that is technology agnostic

Manage anything with an API

HashiCorp

Terraform vs. Other Tools

Provides a high-level abstraction of infrastructure (IaC)

Allows for composition and combination

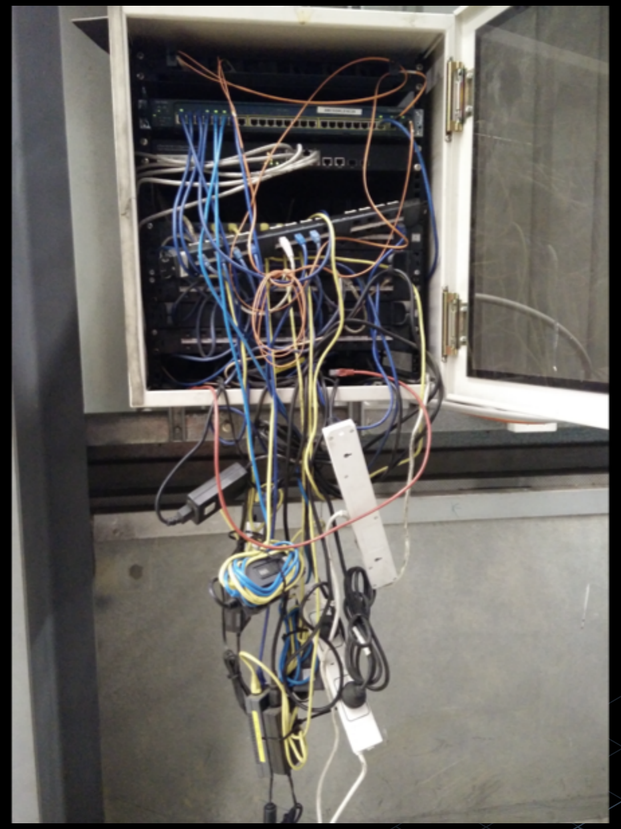
Supports parallel management of resources (graph, fast)

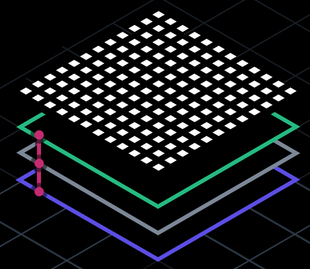
Separates planning from execution (dry-run)

HashiCorp

Progression of Infrastructure Management

HashiCorp

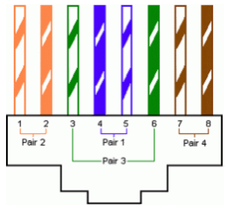




HashiCorp

Maturity

HashiCorp



Manual Everything

HashiCorp

```
#!/bin/bash
```

Basic Automation

HashiCorp

Machine Virtualization

HashiCorp

*aaS

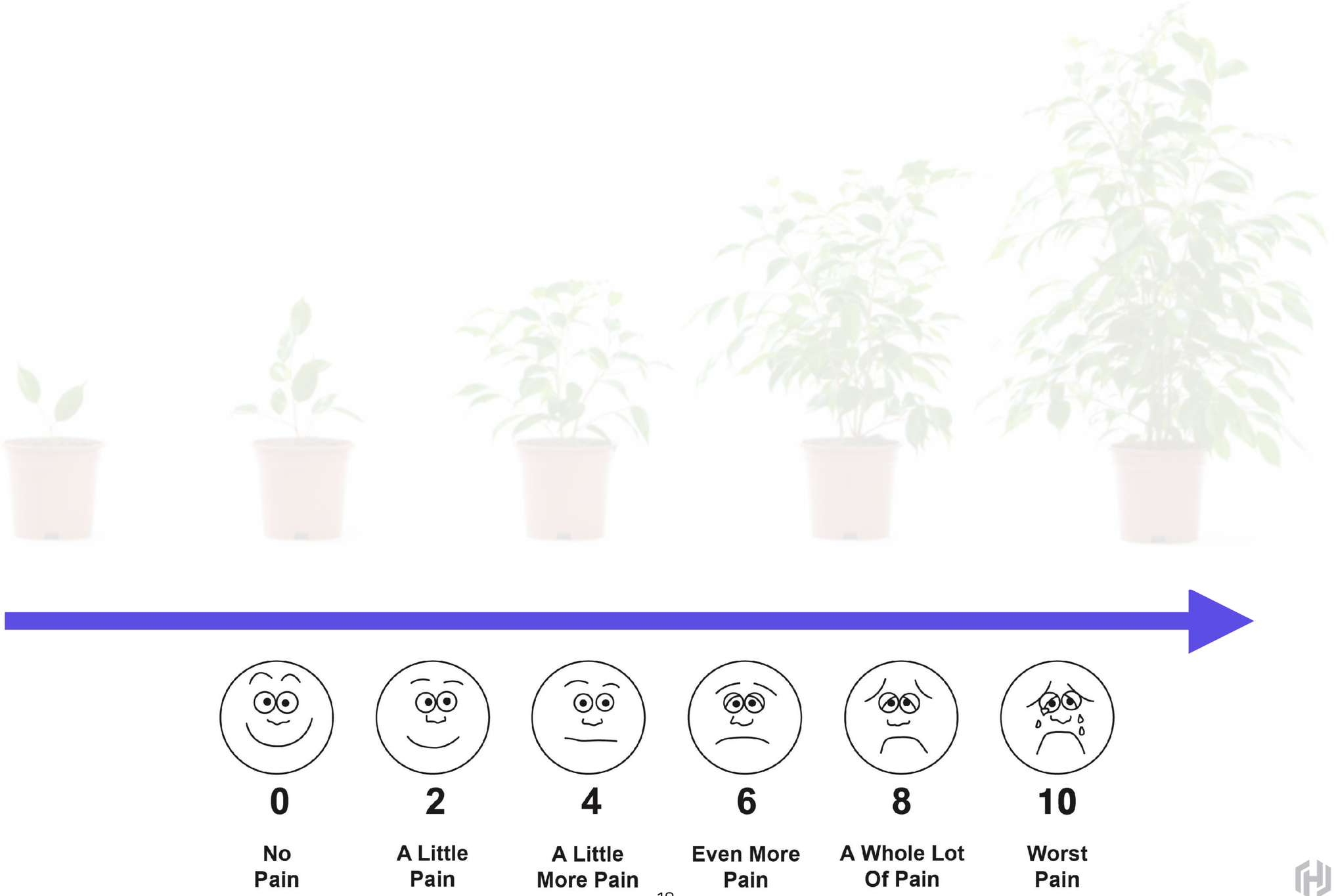
HashiCorp

Datacenter-as-Computer

HashiCorp

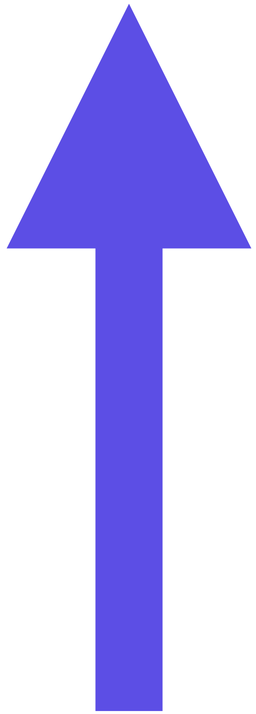
Maturity

HashiCorp


0
No Pain
2
A Little Pain
4
A Little More Pain
6
Even More Pain
8
A Whole Lot Of Pain
10
Worst Pain

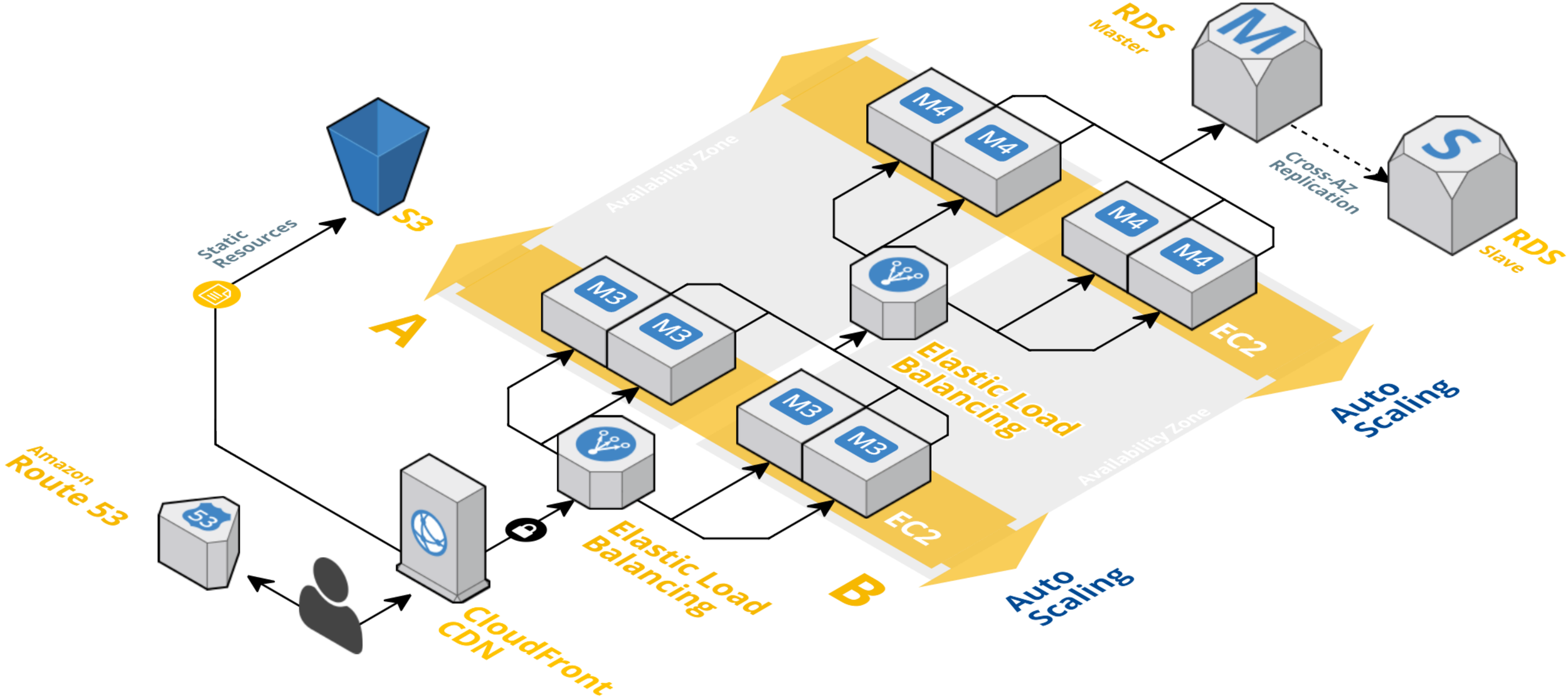
HashiCorp

Capability

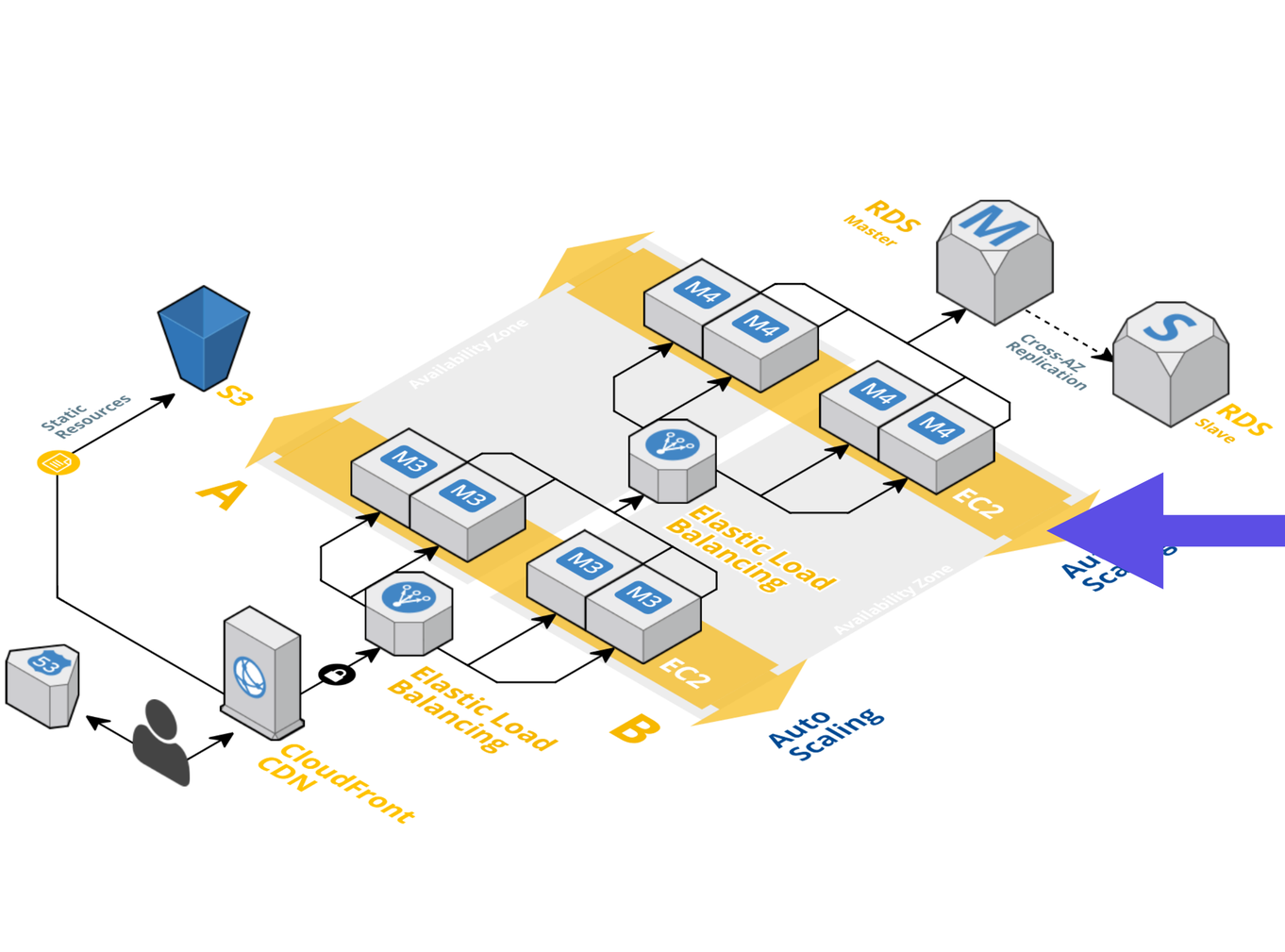


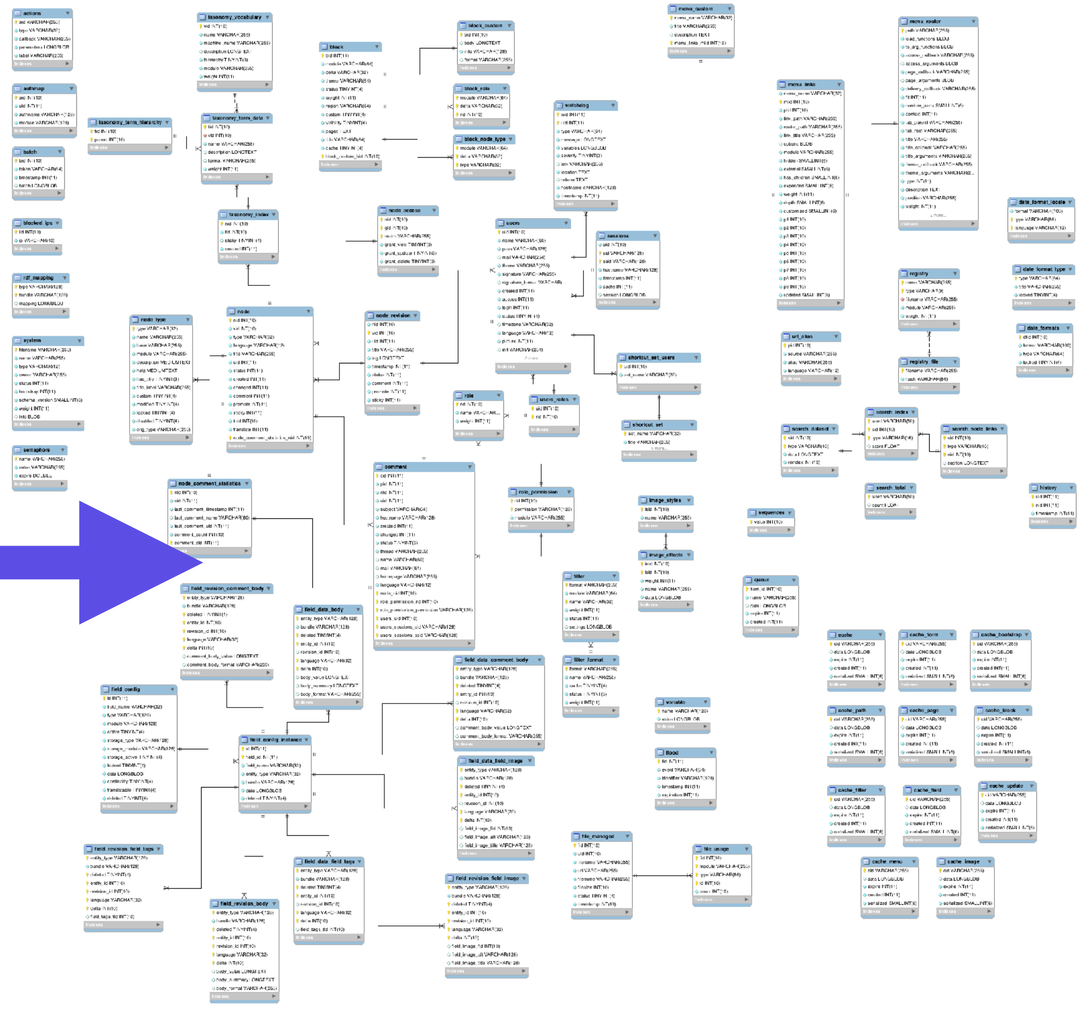
Complexity

HashiCorp

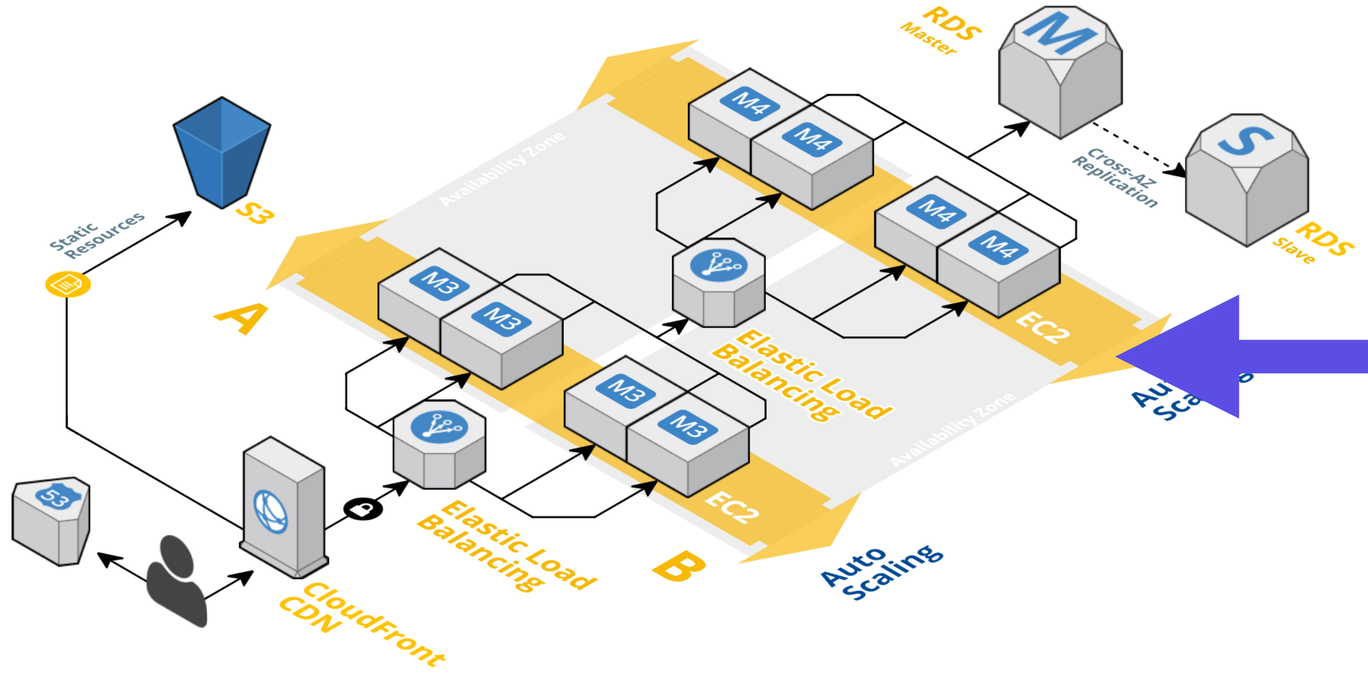

RDS
Master
M
S
Cross-AZ
Replication
RDS
Slave
S3
Static
Resources
Availability Zone
M4
M4
M4
M4
M3
M3
M3
M3
A
B
Elastic Load
Balancing
Elastic Load
Balancing
EC2
EC2
Auto
Scaling
Auto
Scaling
Availability Zone
Amazon
Route 53
53
CloudFront
CDN

HashiCorp

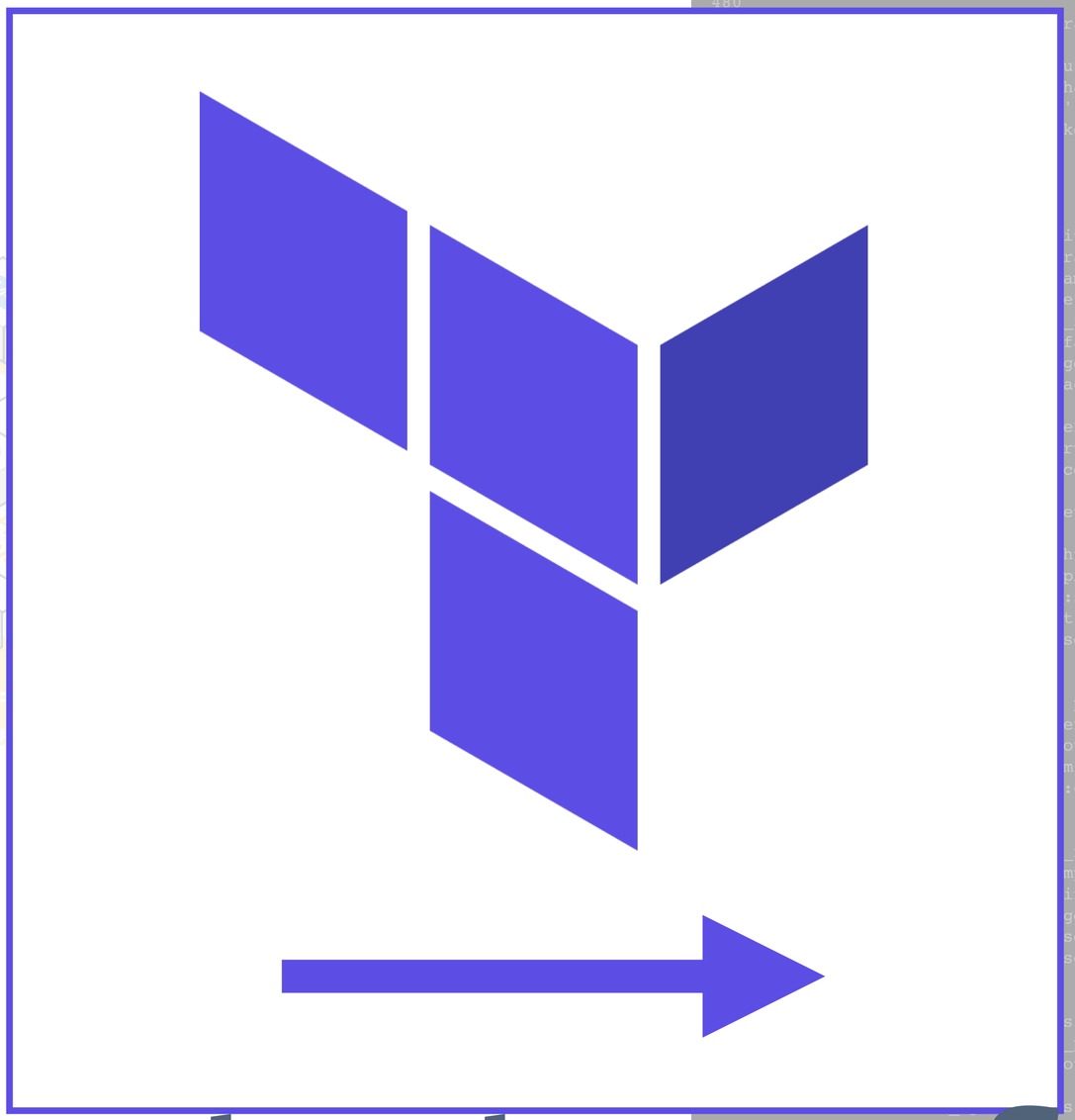

RDS
Master
RDS
Slave
Cross-AZ
Replication
Availability Zone
Availability Zone
EC2
EC2
Elastic Load
Balancing
Elastic Load
Balancing
Auto
Scaling
Static
Resources
S3
S3
CloudFront
CDN
A
B



HashiCorp




```ruby
def update
  return update_api if api_request?

  if authorized_action(@account, @current_user, :manage_account_settings)
    respond_to do |format|

      custom_help_links = params[:account].delete :custom_help_links
      if custom_help_links
        @account.settings[:custom_help_links] = custom_help_links.select{|k, h| h['state'] != 'delet
          hash = index_with_hash[1]
          hash.delete('state')
          hash.assert_valid_keys ["text", "subtext", "url", "available_to"]
          hash
        end
      end

      params[:account][:turnitin_host] = validated_turnitin_host(params[:account][:turnitin_host])
      enable_user_notes = params[:account].delete :enable_user_notes
      allow_sis_import = params[:account].delete :allow_sis_import
      params[:account].delete :default_user_storage_quota_mb unless @account.root_account? && !@acco
      unless @account.grants_right? @current_user, :manage_storage_quotas
        [:storage_quota, :default_storage_quota, :default_storage_quota_mb,
         :default_user_storage_quota, :default_user_storage_quota_mb,
         :default_group_storage_quota, :default_group_storage_quota_mb].each { |key| params[:account
      end
      if params[:account][:services]
        params[:account][:services].slice(*Account.services_exposed_to_ui_hash(nil, @current_user, @
          @account.set_service_availability(key, value == '1')
        end
        params[:account].delete :services
      end
      if @account.grants_right?(@current_user, :manage_site_settings)
        # If the setting is present (update is called from 2 different settings forms, one for notif
        if params[:account][:settings] && params[:account][:settings][:outgoing_email_default_name_o
          # If set to default, remove the custom name so it doesn't get saved
          params[:account][:settings][:outgoing_email_default_name] = '' if params[:account][:settin
        end

        google_docs_domain = params[:account][:settings].try(:delete, :google_docs_domain)
        if @account.feature_enabled?(:google_docs_domain_restriction) &&
           @account.root_account? &&
           !@account.site_admin?
          @account.settings[:google_docs_domain] = google_docs_domain.present? ? google_docs_domain
        end

        @account.enable_user_notes = enable_user_notes if enable_user_notes
        @account.allow_sis_import = allow_sis_import if allow_sis_import && @account.root_account?
        if @account.site_admin? && params[:account][:settings]
          # these shouldn't get set for the site admin account
          params[:account][:settings].delete(:enable_alerts)
          params[:account][:settings].delete(:enable_eportfolios)
        end
      else
        # must have :manage_site_settings to update these
        [ :admins_can_change_passwords,
          :admins_can_view_notifications,
          :enable_alerts,
          :enable_eportfolios,
          :enable_profiles,
          :show_scheduler,
          :global_includes,
          :gmail_domain
        ].each do |key|
          params[:account][:settings].try(:delete, key)
```

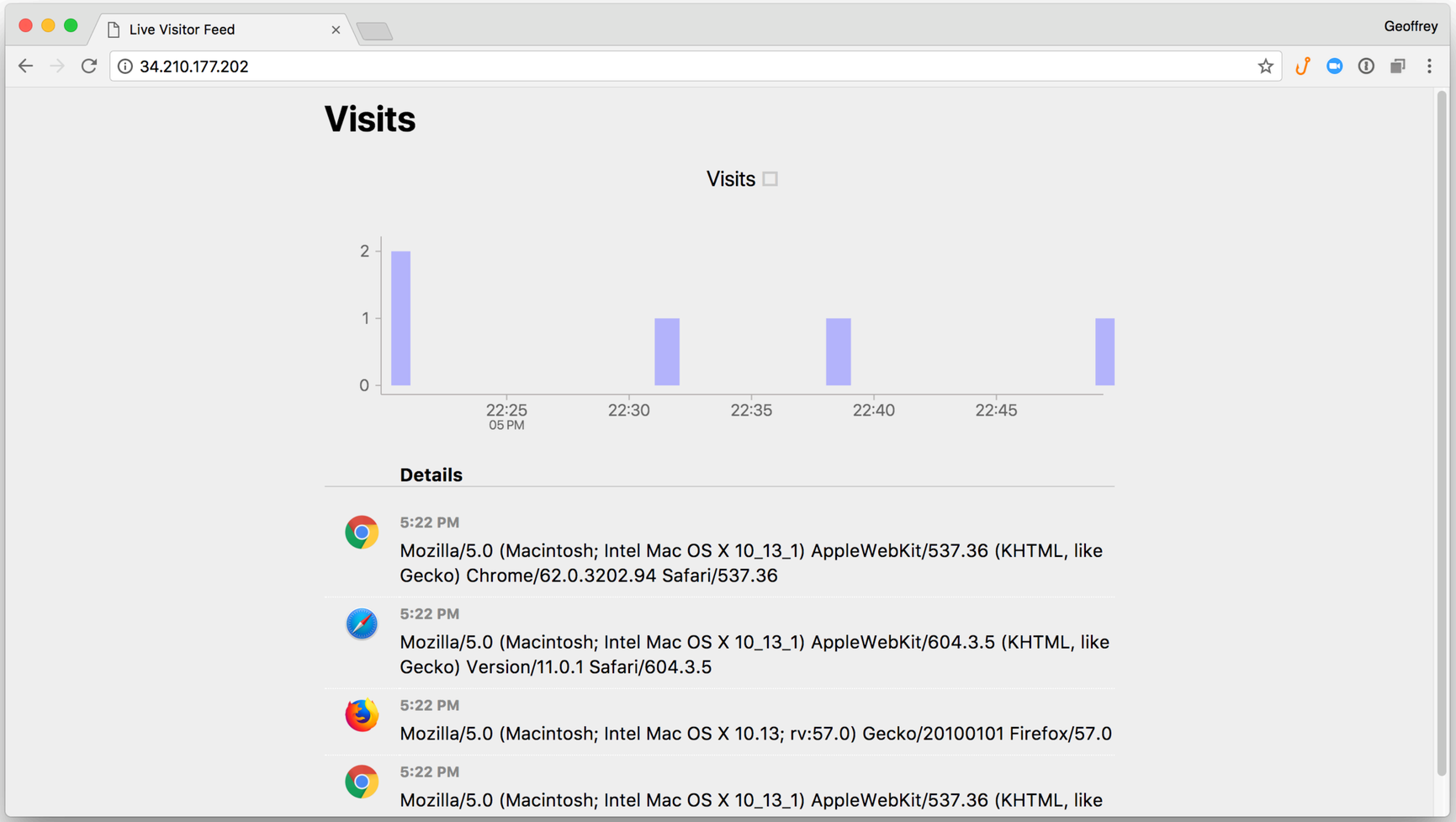


Infrastructure as Code

HashiCorp

What You'll Build

HashiCorp


Live Visitor Feed
Geoffrey
34.210.177.202
Visits
Visits
2
1
0
22:25
05 PM
22:30
22:35
22:40
22:45
Details
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/604.3.5 (KHTML, like Gecko) Version/11.0.1 Safari/604.3.5
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:57.0) Gecko/20100101 Firefox/57.0
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like

EXERCISE

Instructor Demo

Build a server on AWS that runs a single binary web application written in Go.

KNOW IT WORKED IF: You visit the IP address in the web browser and can see a chart with a list of visitors.

HashiCorp

EXERCISE

Lab 0: Choose your editor

Duration: 15 minutes

Goal: Make sure you can login to your workstation to run commands and edit files.

Basic Configuration

HashiCorp

What You'll Learn

- Login to your workstation
- Create a basic Terraform configuration
- Initialize and apply the configuration (create infrastructure)
- Change and re-apply the configuration

HashiCorp

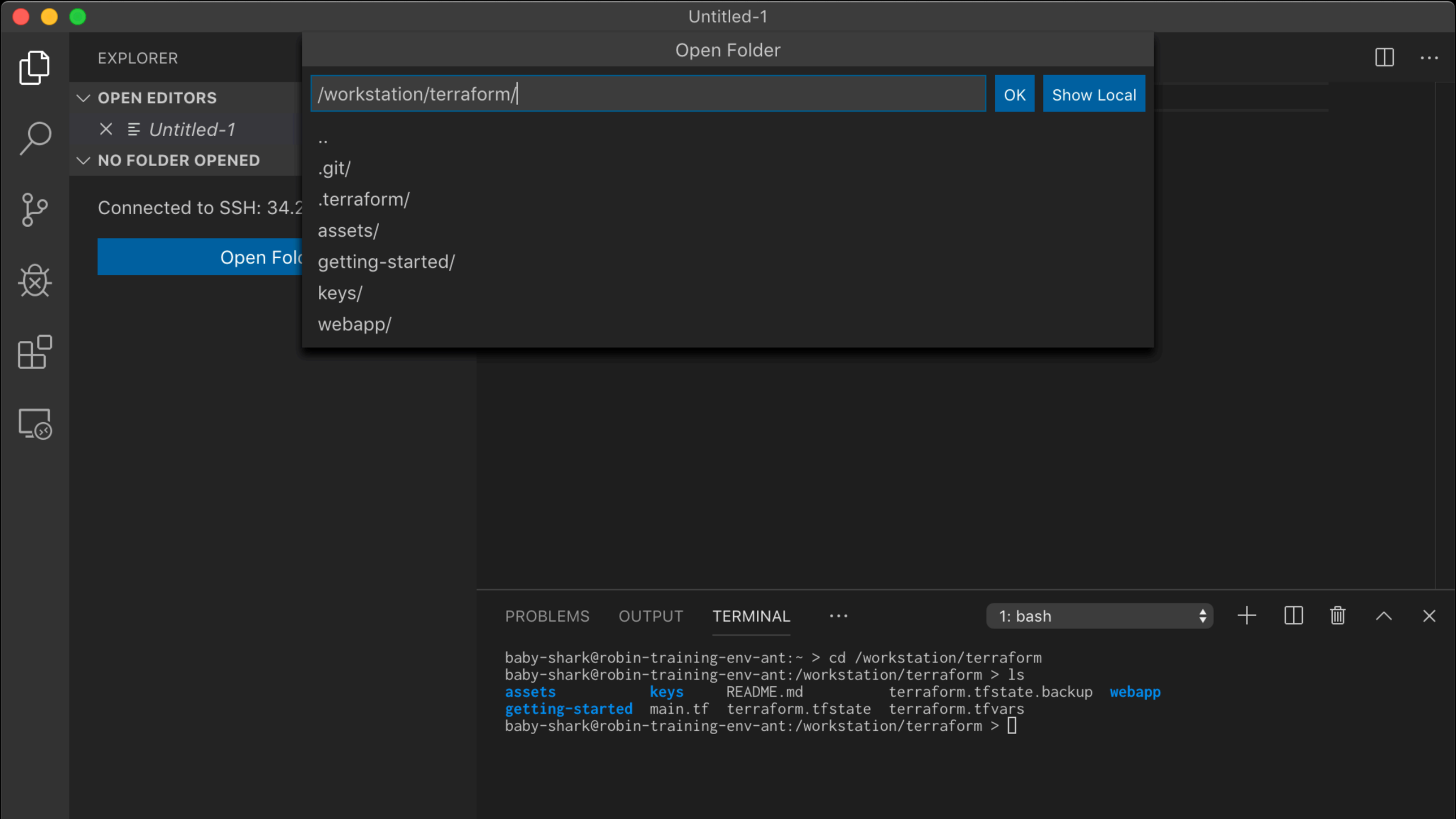
EXERCISE

Explore Workstation

Use VSCode, SSH, or the web console SSH tool to login to your workstation.

Change directory to `/workstation/terraform` and look inside `main.tf`.

HashiCorp


Untitled-1
EXPLORER
OPEN EDITORS
Untitled-1
NO FOLDER OPENED
Connected to SSH: 34.2
Open Fol
Open Folder
/workstation/terraform/
OK
Show Local
..
.git/
.terraform/
assets/
getting-started/
keys/
webapp/
PROBLEMS
OUTPUT
TERMINAL
1: bash
baby-shark@robin-training-env-ant:~ > cd /workstation/terraform
baby-shark@robin-training-env-ant:/workstation/terraform > ls
assets
keys
README.md
terraform.tfstate.backup
webapp
getting-started
main.tf
terraform.tfstate
terraform.tfvars
baby-shark@robin-training-env-ant:/workstation/terraform >

TERMINAL

```
> cd /workstation/terraform

> nano main.tf # Or use `vim main.tf` if you prefer
```

MAIN.TF

```
#
# Your security group ID is:
#
#     abcd1234
#
# ...
#

provider "aws" {
  access_key = "AKIAIZT7VATBGPC3AP3Q"
  secret_key = "W/txHyHA8SsevaENCsUIS9/KGoIScy2yoqrfodKz"
  region     = "us-west-2"
}

resource "aws_instance" "web" {
  # ...
}
```

MAIN.TF

```
provider "aws" {
  access_key = "AKIAIZT7VATBGPC3AP3Q"
  secret_key = "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"
  region     = "us-west-2"
}

resource "aws_instance" "web" {
  ami             = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"

  subnet_id                   = "subnet-0c5f00e7c2fe92579"
  vpc_security_group_ids = ["sg-0e8be760e755de7cc"]

  tags = {
    "Identity" = "<user>-training-env-<animal>"
  }
}
```

Command Line Interface

All interactions with Terraform occur via the CLI

Terraform is a local tool (runs on the current machine)

The terraform ecosystem also includes providers for many cloud services, and a module repository

Hashicorp also has products to help teams manage Terraform: Terraform Cloud and Terraform Enterprise

HashiCorp

TERMINAL

```
> terraform help
Common commands:
    apply               Builds or changes infrastructure
    console             Interactive console for Terraform interpolations
    destroy             Destroy Terraform-managed infrastructure
    env                 Workspace management
    fmt                 Rewrites config files to canonical format
    get                 Download and install modules for the configuration
    graph               Create a visual graph of Terraform resources
    import              Import existing infrastructure into Terraform
    init                Initialize a Terraform working directory
    output              Read an output from a state file
    plan                Generate and show an execution plan
    providers           Prints a tree of the providers used in the config
    push                Upload this Terraform module to Atlas to run
    refresh             Update local state file against real resources
    show                Inspect Terraform state or plan
    taint               Manually mark a resource for recreation
```

Terraform Init

You must run `terraform init` to initialize a new Terraform working directory, and after changing provider configuration

You **do not** need to run `terraform init` before each command

Similar to `git init`

HashiCorp

TERMINAL

```
> terraform init
Initializing provider plugins...
- Checking for available provider plugins on https://releases...
- Downloading plugin for provider "aws" (2.16)...

The following providers do not have any version constraints in
configuration, so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain
breaking changes, it is recommended to add version = "..." constraints to
the corresponding provider blocks in configuration, with the constraint
strings suggested below.

* provider.aws: version = "~> 2.16"

Terraform has been successfully initialized!
```

Command: `terraform plan`

The plan shows you what will happen

You can save plans to guarantee what will happen

Plans show reasons for certain actions (such as re-create)

Prior to Terraform, users had to guess change ordering, parallelization, and rollout effect

HashiCorp

Command: `terraform plan`

`+` resource will be created

`-` resource will be destroyed

`~` resource will be updated in-place

`-/+` resources will be destroyed and re-created

HashiCorp

TERMINAL

```
> terraform plan
  + resource "aws_instance" "web" {
      + ami                          = "ami-0e63f50857fdc1f9f"
      + arn                          = (known after apply)
      + availability_zone            = (known after apply)
      + cpu_core_count               = (known after apply)
      + get_password_data            = false
      + instance_type                = "t2.micro"
      + placement_group              = (known after apply)
      + private_ip                   = (known after apply)
      + public_dns                   = (known after apply)
      + public_ip                    = (known after apply)
      + security_groups              = (known after apply)
# ...
Plan: 1 to add, 0 to change, 0 to destroy.
```

Command: `terraform apply`

Executes changes in order based on the resource graph

Parallelizes changes when possible

Handles and recovers transient errors

Runs `plan` first unless given a serialized plan file

HashiCorp

Command: `terraform apply`

Updates existing resources when updates are allowed

Re-creates existing resources when updates are not allowed

HashiCorp

TERMINAL

```
> terraform apply

# ...

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.web: Creating...
aws_instance.web: Still creating... [10s elapsed]
aws_instance.web: Still creating... [20s elapsed]
aws_instance.web: Still creating... [30s elapsed]
aws_instance.web: Creation complete after 31s [id=i-0b8bf14603ebdc264]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Command: `terraform show`

Displays human-friendly state

Can be used to get on-the-fly information

HashiCorp

TERMINAL

```
> terraform show
# aws_instance.web:
resource "aws_instance" "web" {
    ami                          = "ami-0e63f50857fdc1f9f"
    arn                          = "..."
    associate_public_ip_address = true
    availability_zone            = "us-west-2a"
    cpu_core_count               = 1
    cpu_threads_per_core         = 1
    disable_api_termination      = false
    ebs_optimized                = false
    get_password_data            = false
    id                           = "i-09c617e1c6d3514a8"
# ...
    public_ip                    = "34.214.210.211"
```

TERMINAL

```
$ ping 34.214.210.211
PING 34.214.210.211 (34.214.210.211) 56(84) bytes of
data.
64 bytes from 34.214.210.211: icmp_seq=1 ttl=63
time=0.807 ms
64 bytes from 34.214.210.211: icmp_seq=2 ttl=63
time=0.770 ms
<Ctrl-C>
```

```
resource "aws_instance" "web" {
  ami             = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"

  subnet_id                 = "subnet-0c5f00e7c2fe92579"
  vpc_security_group_ids = ["sg-0e8be760e755de7cc"]

  tags = {
    "Identity" = "..."
    "Name"      = "Moose”
    "Zip"       = "zap"
  }
}
```

TERMINAL

```
> terraform plan
aws_instance.web: Refreshing state...
[id=i-0b8bf14603ebdc264]
# ...
  # aws_instance.web will be updated in-place
  ~ resource "aws_instance" "web" {
# ...
      ~ tags      = {
          "Identity" = “...”
        + "Name"     = “Moose”
        + "Zip"      = “zap”
        }
# ...
Plan: 0 to add, 1 to change, 0 to destroy.
```

TERMINAL

```
> terraform apply
aws_instance.web: Refreshing state...
[id=i-0b8bf14603ebdc264]
# ...
  # aws_instance.web will be updated in-place
  ~ resource "aws_instance" "web" {
# ...
      ~ tags      = {
          "Identity" = “...”
        + "Name"     = “Moose”
        + "Zip"      = “zap”
        }
# ...
Apply complete! Resources: 0 added, 1 changed, 0
destroyed.
```

Terraform 0.12

Released in May of 2019

Better expression syntax, type support, iteration constructs, and many other improvements

The command `terraform 0.12upgrade` will upgrade configuration to the new syntax

HashiCorp

TIP

Use Version Control Tools For Rollback

Terraform only knows about your configuration and the state of the infrastructure it has built.

If you need to rollback, use the capabilities of your version control software to revert to an earlier version of `main.tf`, then run `terraform apply` on it.

HashiCorp

What You Learned

- Login to your workstation
- Create a basic Terraform configuration
- Initialize and apply the configuration (create infrastructure)
- Change and re-apply the configuration

HashiCorp

EXERCISE

Lab 1: Lab Setup

Duration: 20 minutes

Goal: Familiarize yourself with your workstation and the Terraform CLI

Outputs

HashiCorp

What You'll Learn

- Print dynamic data with an output
- Query specific values from the output

HashiCorp

Outputs

Outputs define useful values that will be highlighted to the user when Terraform applies: IP addresses, usernames, generated keys

Outputs can be queried using the `terraform output` command

Outputs are a way to easily extract and query information from all of Terraform's collected data

HashiCorp

MAIN.TF

```
resource "aws_instance" "web" {
# ...
}

output "public_ip" {
  value = aws_instance.web.public_ip
}

# Prior to Terraform 0.12 syntax:

output "public_ip" {
  value = "${aws_instance.web.public_ip}"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
# ...
}

output "public_ip" {
  value = aws_instance.web.public_ip
}

output "public_dns" {
  value = aws_instance.web.public_dns
}
```

TERMINAL

```
> terraform refresh
aws_instance.web: Refreshing state... [id=i-0b8bf14603ebdc264]

Outputs:

public_dns = ec2-34-222-156-11.us-west-2.compute.amazonaws.com
public_ip = 34.222.156.11
```

TERMINAL

```
> terraform output
public_dns = ec2-34-222-156-11.us-west-2.compute.amazonaws.com
public_ip = 34.222.156.11
```

TERMINAL

```
> terraform output public_dns
ec2-34-222-156-11.us-west-2.compute.amazonaws.com
```

TERMINAL

```
> terraform output public_dns
ec2-34-222-156-11.us-west-2.compute.amazonaws.com
> ping $(terraform output public_dns)
PING ec2-34-222-156-11.us-west-2.compute.amazonaws.com
(34.222.156.11) 56(84) bytes of data.
64 bytes from ec2-34-222-156-11.us-
west-2.compute.amazonaws.com (34.222.156.11): icmp_seq=1
ttl=63 time=0.440 ms
64 bytes from ec2-34-222-156-11.us-
west-2.compute.amazonaws.com (34.222.156.11): icmp_seq=2
ttl=63 time=0.570 ms
<Ctrl-C>
```

What You Learned

- Print dynamic data with an output
- Query specific values from the output

HashiCorp

EXERCISE

Lab 2: Outputs

Duration: 10 minutes

Goal: Configure specific outputs and learn to query your configuration for those.

Console

HashiCorp

What You'll Learn

- Interact with live data in the console
- Use the console from the command line

HashiCorp

REPL

A REPL is a Read-Evaluate-Print-Loop

The Terraform console provides an interactive REPL for writing, executing, and querying Terraform configuration

HashiCorp

Command: `terraform console`

The `terraform console` command creates an interactive console for interacting with Terraform

This is useful for experimenting with interpolations and interacting with Terraform state

HashiCorp

TERMINAL

```
> terraform console
> aws_instance.web.ami
ami-0e63f50857fdc1f9f
> aws_instance.web.public_ip
54.174.40.55
> aws_instance.web.tags
{
  "Identity" = "robin-training-env-ant"
  "Name" = "Moose"
  "Zip" = "zap"
}
> <Ctrl-D>
```

TERMINAL

```
> terraform console
> exit
```

TERMINAL

```
> echo "aws_instance.web.ami" | terraform console
ami-0e63f50857fdc1f9f
```

EXERCISE

Lab 3: REPL

Duration: 10 minutes

Goal: The Read-Evaluate-Print-Loop is useful for understanding interpolation and finding specific information.

Auto-Formatting

HashiCorp

Auto-Formatting

Terraform has built-in support for auto-formatting configuration files to match the HCL specification.

There is no need to manually align things.

This helps keep configurations VCS-friendly and reduces bike-shed arguments over formatting styles.

Plugins exist for most major editors.

HashiCorp

Command: `terraform fmt`

The `terraform fmt` command formats configuration files.

It does not prompt for confirmation or

It also reports on syntax errors, if they exist.

HashiCorp

EXERCISE

Format Your Code

Run the `terraform fmt` command with no arguments. By default it will choose all Terraform configurations in the current working directory non-recursively.

Open the `main.tf` file and see that it has been formatted.

Command: `terraform fmt`

Please feel free to use the `terraform fmt` command through this training and at your leisure, but we will not explicitly discuss it anymore.

HashiCorp

Variables

HashiCorp

What You'll Learn

- Use variables to pass values to your configuration
- Refactor existing configuration to use variables
- Keep sensitive data out of source code control
- Pass variables to Terraform in several ways

HashiCorp

Variables

Work a lot like variables from programming languages

Allow you to remove hard coded values and pass parameters to your configuration

Can help make configuration easier to understand and re-use

Must always have a value. Variables are never optional, but they can have a default value

HashiCorp

EXAMPLE.TF

```
variable "animal" {}
```

EXAMPLE.TF

```
variable "animal" {
  type        = "string"
  default     = "moose"
  description = "An animal, such as whale, elephant, or
bear"
}
```

Variable Types

The most common variable types are `string`, `number`, `list`, and `map`

Other support types include `bool` (true/false), `set`, `object`, and `tuple`

If omitted, the type is inferred from the default value

If neither type nor default is provided, the type is assumed to be `string`

HashiCorp

```
variable "access_key" {}
variable "secret_key" {}
variable "region" {
  default = "us-west-2"
}
```

VARIABLES.TF

```
variable "access_key" {
  type = "string"
}

variable "secret_key" {
  type = "string"
}

variable "region" {
  type    = "string"
  default = "us-west-2"
}
```

VARIABLES.TF

```
variable "access_key" {
  type        = "string"
  description = "AWS access key"
}

variable "secret_key" {
  type        = "string"
  description = "AWS secret access key"
}

variable "region" {
  type        = "string"
  default     = "us-west-2"
  description = "AWS region"
}
```

Using Variables

Variables can be referred to by using the syntax ***var.<name>***

Variables' values can also be interpolated into a string using the syntax ***"${var.<name>}"***

HashiCorp

MAIN.TF

```
provider "aws" {
  access_key = var.access_key
  secret_key = var.secret_key
  region     = var.region
}

# ...
```

TERMINAL

```
> terraform plan
var.access_key
  Enter a value:

<Ctrl + C>
```

TERMINAL

```
> export TF_VAR_access_key=<access_key>

> terraform plan
var.secret_key
  Enter a value:

<Ctrl + C>
```

```
access_key = “AKIAIZT7VATBGPC3AP3Q"
secret_key = "W/txHyHeNCsUIS9/KGoIScy2yoqrfodKz"
# ami                    = "ami-0e63f50857fdc1f9f"
# subnet_id              = "subnet-0c5f00e7c2fe92579"
# identity               = "robin-training-env-ant"
# region                 = "us-west-2"
# vpc_security_group_ids = ["sg-0e8be760e755de7cc"]
```

TERMINAL

```
> terraform plan -var secret_key=<secret_key>
aws_instance.web: Refreshing state... (ID:
i-02f5717f1a84502ed)

# ...

No changes. Infrastructure is up-to-date.

# ...
```

TERMINAL

```
> terraform plan -var-file secrets.tfvars
```

Specifying Variable Values

1. Default values
2. Environment variables
3. Saved in variable definition files (`terraform.tfvars`)
4. Using the ***`-var`*** or ***`-var-file`*** command line option
5. Configured in a Terraform Enterprise or Terraform Cloud workspace
6. Entered via CLI prompts

HashiCorp

```
variable "num_webservers" {
  type    = "number"
  default = 5
}

variable "zones" {
  type    = "list"
  default = ["us-east-1a", "us-east-1b"]
}

variable "image_ids" {
  type    = "map"
  default = {
    us-east-1 = "image-1234"
    us-west-2 = "image-4567"
  }
}
```

WARNING

Variables & Security

Never put secret values, like passwords or access tokens, in `.tf` files or other files that are checked into source control.

HashiCorp

TIP

Use git secrets Plugin

Install on each repository.

It will warn you if you try to commit code that looks like an AWS secret. (Configurable for others, too.)

https://github.com/awslabs/git-secrets

HashiCorp

What You Learned

- Use variables to store sensitive values
- Refactor existing configuration to use variables
- Keep sensitive data out of source code control systems
- Populate variables to Terraform in several ways

HashiCorp

EXERCISE

Lab 4: Variables

Duration: 15 minutes

Goal: Remove all hardcoded values in your current configuration and use variables instead.

Configuration Format

HashiCorp

Configuration Syntax

Configurations are written in HashiCorp Configuration Language (HCL)

HCL is designed to strike a balance between human-readable and machine-parsable

Terraform can also read configuration in JSON, but we recommend using HCL, except when the configuration is machine-generated

HashiCorp

Load Order & Semantics

Configuration can be in a single file or split across multiple files. Terraform will process all files in the current working directory which end in `.tf` or `.tf.json`

Sub-folders are not included (non-recursive)

Files are processed in lexicographical (dictionary) order

Any files with a different extension are ignored (e.g. `.jpg` or `.tf.old`)

Generally, the order in which things are defined doesn't matter.

HashiCorp

Load Order & Semantics

Parsed configurations are *appended* to each other, not merged

Resources with the same name are **not** merged (this will produce an error)

Configuration syntax is declarative, so references to other resources do not depend on the order they are defined

HashiCorp

EXERCISE

Which Files Does Terraform Use?

Which of the following files would be parsed by Terraform when a command is issued?

```
> tree
.
├── README.md
├── main.tf
├── other.tf
├── gravatar.jpg
├── alphabet
│   ├── README.md
│   ├── main.tf
```

HashiCorp

TERMINAL

```
> tree
.
├── README.md
├── main.tf
├── other.tf
├── gravatar.jpg
├── alphabet
│   ├── README.md
│   ├── main.tf
```

Configuration Organization Patterns

Teams choose to organize their configurations in many different patterns.

HashiCorp

TERMINAL

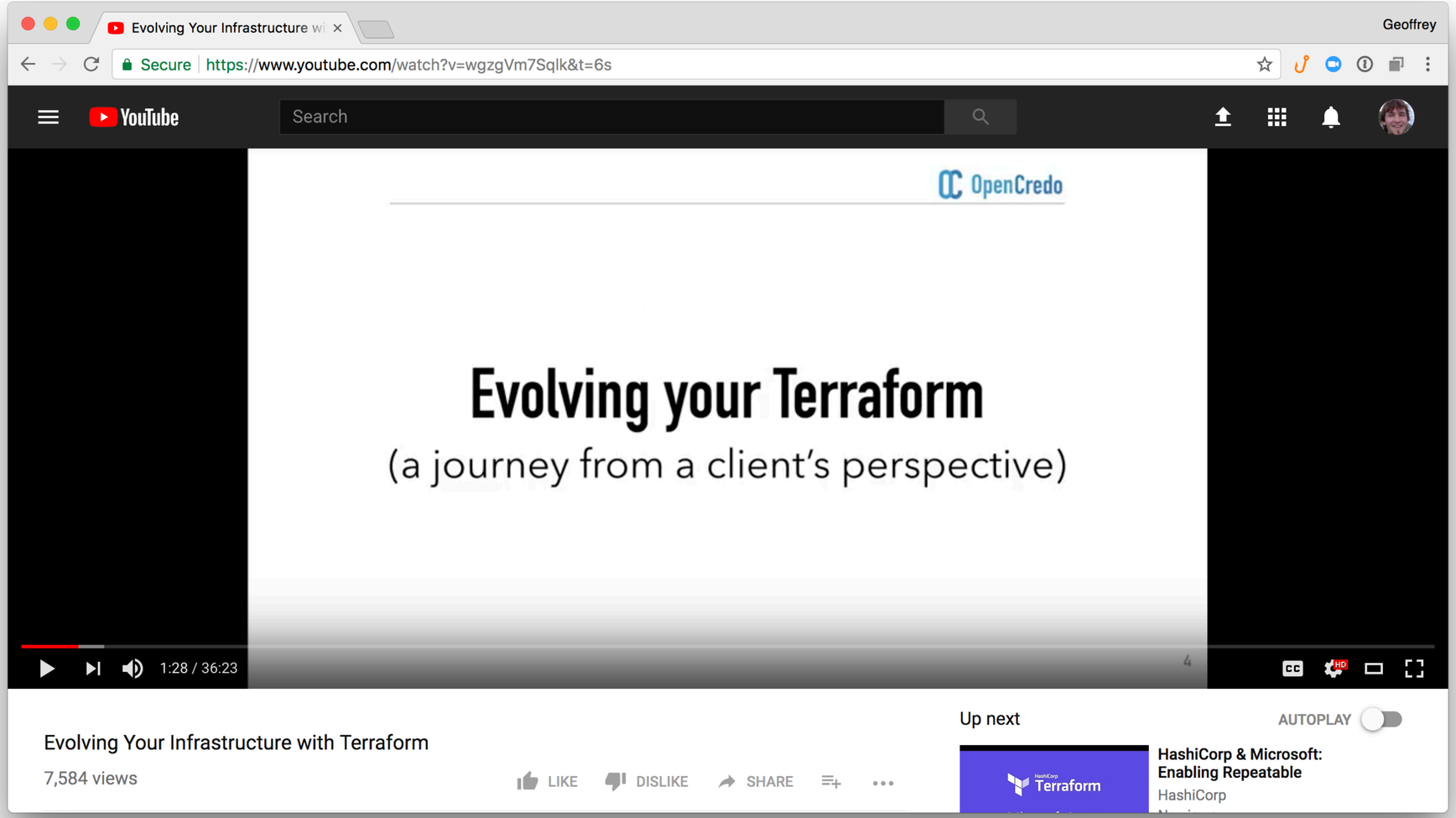
```
> ls
main.tf
outputs.tf
variables.tf
```

TERMINAL

```
> ls
instances.tf
load-balancers.tf
shared.tf
```

TERMINAL

```
> ls
everything.tf
```


Evolving Your Infrastructure wi
Geoffrey
Secure https://www.youtube.com/watch?v=wgzgVm7Sqlk&t=6s
YouTube
Search
OpenCredo
Evolving your Terraform
(a journey from a client's perspective)
1:28 / 36:23
Evolving Your Infrastructure with Terraform
7,584 views
LIKE
DISLIKE
SHARE
Up next
AUTOPLAY
HashiCorp & Microsoft: Enabling Repeatable
HashiCorp

HashiCorp

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}
```

Top Level Keywords

```
provider        module
variable        data
resource        terraform
output          locals
```


HashiCorp

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}

resource "aws_instance" "web" {
  ami           = "ami-07669fc90e6e6cc47"
  instance_type = "t2.micro"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}

resource "aws_instance" "web-2" {
  ami           = "ami-07669fc90e6e6cc47"
  instance_type = "t2.micro"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}

resource "digitalocean_droplet" "web" {
  image  = "ubuntu-18-04-x64"
  name   = "web"
  size   = "s-1vcpu-1gb"
  region = "nyc2"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}
```

MAIN.TF

```
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}

resource "dnsimple_record" "web" {
  domain = "hashicorp.com"
  name   = "web"
  ttl    = "3600"
  type   = "A"
  value  = aws_instance.web.public_ip
# Terraform pre-0.12 syntax:
#  value  = "${aws_instance.web.public_ip}"
}
```

MAIN.TF

```
# This is a comment
resource "aws_instance" "web" {
  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "t2.micro"
}

/* This is a multiline
comment */
resource "dnsimple_record" "web" {
  # ...
}
```

MAIN.TF

```
variable "thing" {
  description = <<EOF
This is a
multiline
string
EOF
}
```

Syntax Highlighting

Plugins for Terraform/HCL exist for most major editors, but `ruby` tends to work best if one does not exist.

HashiCorp

Modules

HashiCorp

What You'll Learn

- The Terraform Public Module Registry
- Use a module from `main.tf`
- Understand how modules are stored on disk

HashiCorp

Modules

Portable Terraform configurations (packages)

Allow separation of concerns and responsibilities among teams

Parallels: Chef Cookbook, Puppet Module, Ruby gem

HashiCorp

Modules

Modules are just Terraform configurations inside a folder - there's nothing special about them.

Complex configurations, team projects, and multi-repository codebases will benefit from modules. Get into the habit of using them wherever it makes sense.

HashiCorp

TERMINAL

```
> tree my-module
my-module
└── main.tf
```

```
variable "ami_id" {}
variable "instance_type" {}

resource "aws_instance" "db" {
  # ...
}

output "address" {
  value = aws_instance.db.public_dns
}
```

MAIN.TF

```
module "my-module" {
  # Source can be any URL or file path
  source = "../../my-module"

}
```

MAIN.TF

```
module "my-module" {
  # Source can be any URL or file path
  source = "../../my-module"

  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "m5.large"
}
```

MAIN.TF

```
module "my-module" {
  # Source can be any URL or file path
  source = "../../my-module"

  ami           = "ami-0e63f50857fdc1f9f"
  instance_type = "m5.large"
}

output "db-address" {
  value = module.my-module.address
}
```

Modules

Variables defined in a module become arguments to module block

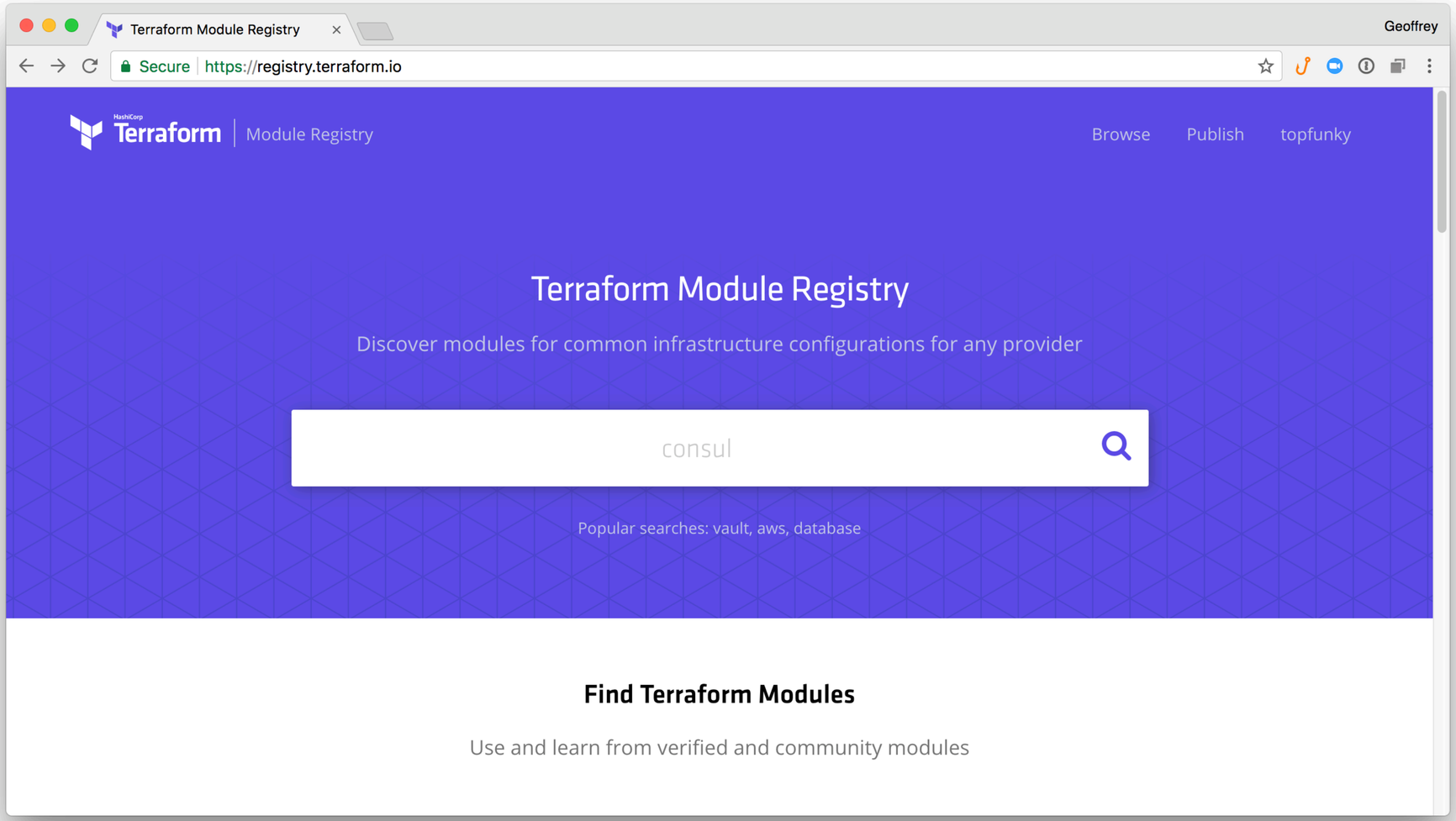
Outputs defined in module become attributes from module definition

Configuration inside a module is a "black box" from the outside

Individual resources are not accessible outside the module

HashiCorp

```
module "my-module" {
  # Source can be any URL or file path
  source = "../../my-module"

  argument_1 = "value"
  argument_2 = "value"
}

output "example" {
  value = module.my-module.aws_instance.db.public_ip
}
```


Terraform Module Registry
Geoffrey
Secure
https://registry.terraform.io
HashiCorp
Terraform
Module Registry
Browse
Publish
topfunky
Terraform Module Registry
Discover modules for common infrastructure configurations for any provider
consul
Popular searches: vault, aws, database
Find Terraform Modules
Use and learn from verified and community modules

HashiCorp Terraform | Module Registry

Search Browse Publish Sign-in

dynamic-keys

AWS

Version 1.0.0

Terraform module that dynamically generates a public/private keypair.

Published April 6, 2018 by mitchellh

Total provisions: 127

Source: github.com/mitchellh/terraform-aws-dynamic-keys (report an issue)

Provision Instructions

Copy and paste into your Terraform configuration, insert the variables, and run `terraform init`:

```
module "dynamic-keys" {
  source  = "mitchellh/dynamic-keys/aws"
  version = "1.0.0"

  # insert the 1 required variable here
}
```


HashiCorp

```
variable "identity" {}

module "keypair" {
  source = "mitchellh/dynamic-keys/aws"
  path   = "${path.root}/keys"
  name   = "${var.identity}-key"
}
```

```
output "key_name" {
  value = aws_key_pair.generated.key_name
}

output "public_key_openssh" {
  value = tls_private_key.generated.public_key_openssh
}

output "private_key_pem" {
  value = tls_private_key.generated.private_key_pem
}

output "public_key_filepath" {
  value = local.public_key_filename
}

output "private_key_filepath" {
  value = local.public_key_filename
}
```

TERMINAL

```
> terraform plan
Failed to load root config module: Error loading modules:
module example: not found, may need to be downloaded
using 'terraform get'
```

TERMINAL

```
> terraform get
Downloading mitchellh/dynamic-keys/aws 1.0.0 for
keypair...
- keypair in .terraform/modules/keypair/mitchellh-
terraform-aws-dynamic-keys-37d94b3
```

TERMINAL

```
> terraform plan

Error: Could not satisfy plugin requirements

# ...

Error: provider.tls: no suitable version installed
  version requirements: "(any version)"
  versions installed: none


Error: provider.local: no suitable version installed
  version requirements: "(any version)"
  versions installed: none
```

TERMINAL

```
> terraform init
Initializing modules...

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "null" (terraform-
providers/null) 2.1.2...
- Downloading plugin for provider "tls" (terraform-
providers/tls) 2.0.1...
- Downloading plugin for provider "local" (terraform-
providers/local) 1.3.0...

# ...
```

TIP

Module References Are Stored in `.terraform`

A hidden directory is built at the root of your project

You can use the `tree` command or `ls -l` to see the contents

Note that local modules are symlinked. This explains why changes to module code are available immediately

HashiCorp

TERMINAL

```
> tree .terraform

.terraform/
├── modules
│   ├── 56e6098163e5b2675d14437a2a723a54
│   │   └── mitchellh-terraform-aws-dynamic-keys-37d94b3
│   │       ├── main.tf
│   │       ├── outputs.tf
│   │       ├── README.md
│   │       └── variables.tf
│   └── modules.json
└── plugins
    └── linux_amd64
        ├── lock.json
        ├── terraform-provider-aws_v1.23.0_x4
        ├── terraform-provider-local_v1.1.0_x4
        ├── terraform-provider-null_v1.0.0_x4
        └── terraform-provider-tls_v1.1.0_x4

4 directories, 10 files
```

TERMINAL

```
> terraform apply
aws_instance.web: Refreshing state... [id=i-03724b60e0ff853bc]

# ...

  # module.keypair.aws_key_pair.generated will be created
  + resource "aws_key_pair" "generated" {
      + fingerprint = (known after apply)
      + id          = (known after apply)
      + key_name    = "robin-training-env-ant-key"
      + public_key  = (known after apply)
    }

# ...

Plan: 5 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

What You Learned

- Import a module from the Terraform Public Module Registry
- Use a module from `main.tf`
- Understand how modules are stored on disk

HashiCorp

EXERCISE

Lab 5: Modules

Duration: 20 minutes

Goal: Refactor your existing code into a separate module and connect a module from the Public Registry.

HashiCorp

Provisioners

HashiCorp

What You'll Learn

- How to create a connection to our server
- Specify the key for connecting to the server
- Provision the server with files and commands

HashiCorp

WARNING

The Server We Created is Useless

We created a server without any running code.

No useful services are running on it.

In this section, we'll see how to provision or instance.

HashiCorp

```
resource "aws_instance" "web" {
  # . . .

  connection = {
    type        = "ssh"
    user        = "ubuntu"
    host        = self.public_ip
    private_key = module.keypair.private_key_pem
  }
}
```

MAIN.TF

```
variable "identity" {}

module "keypair" {
  source = "mitchellh/dynamic-keys/aws"
  path   = "${path.root}/keys"
  name   = "${var.identity}-key"
}
```

```
resource "aws_instance" "web" {
# . . .

  key_name = "${var.identity}-key"
# . . .
}
```

```
resource "aws_instance" "web" {
  # . . .

  provisioner "file" {
    source      = "assets"
    destination = "/tmp/"
  }

}
```

https://www.terraform.io/docs/provisioners/file.html

```
resource "aws_instance" "web" {
  # . . .

  provisioner "remote-exec" {
    inline = [
      "sudo sh /tmp/assets/setup-web.sh"
    ]
  }
}
```

https://www.terraform.io/docs/provisioners/remote-exec.html

TIP

Force Provisioning With the `taint` Command

If we run `apply` after adding provisioners, it won't change the server.

Force it to be destroyed and recreated with the `taint` command:

```
terraform taint aws_instance.web
```

If you're unsure of the correct ID for the resource you want to taint, you can use terraform show:

```
> terraform show
aws_instance.web:
  id = i-07842f0a26236fb91
  ami = ami-0def3275
  . . .
```

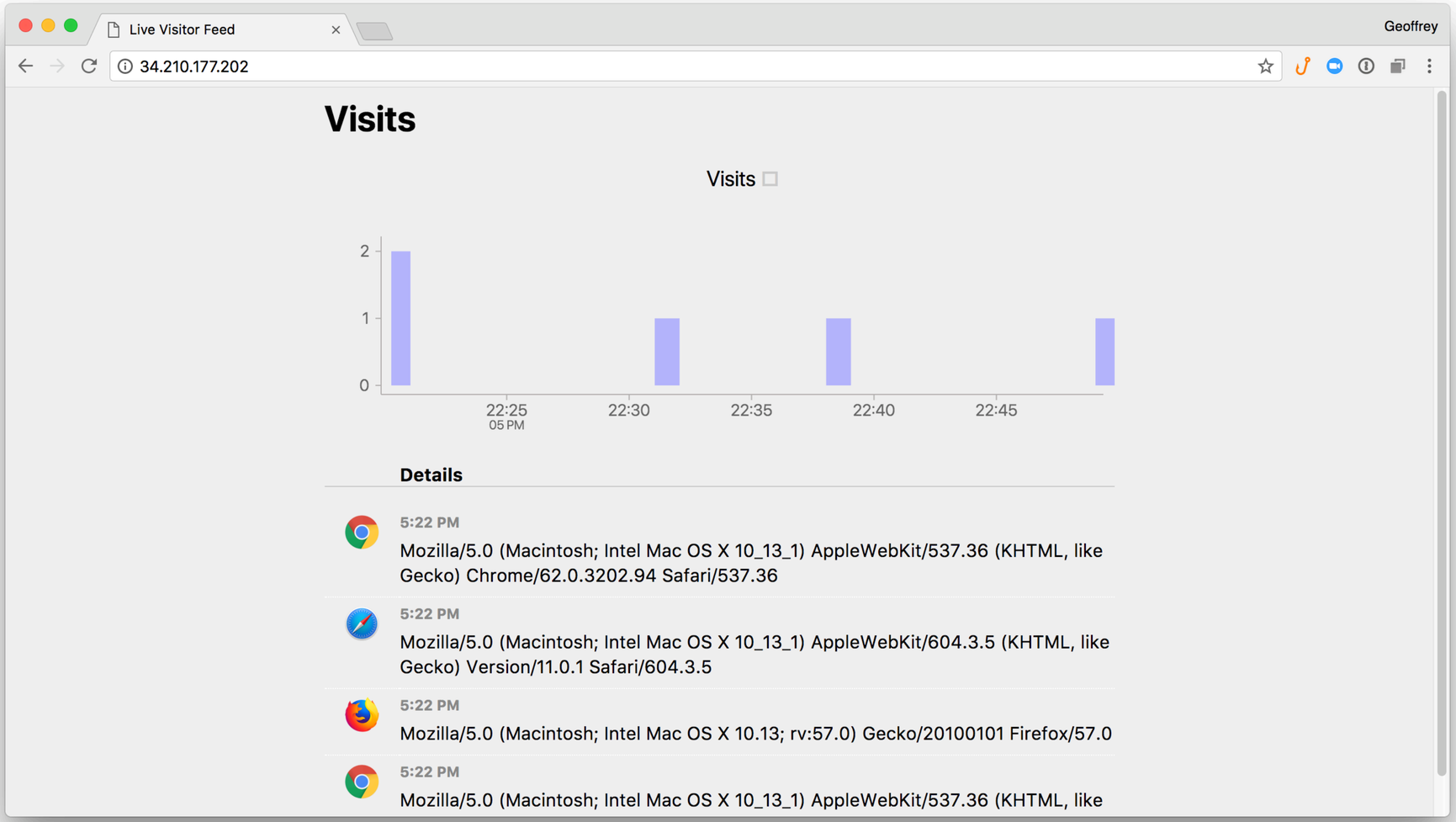

HashiCorp

TERMINAL

```
> terraform taint aws_instance.web
Resource instance aws_instance.web has been marked as
tainted.
```

TERMINAL

```
> terraform apply
# ...
Plan: 1 to add, 0 to change, 1 to destroy.
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

# ...

web: Provisioning with 'remote-exec'...
web (remote-exec): Connecting to remote host via SSH...
web (remote-exec):   Host: 35.162.171.124
web (remote-exec):   User: ubuntu
web (remote-exec):   Password: false
web (remote-exec):   Private key: true
web (remote-exec):   SSH Agent: false
web (remote-exec): Connected!
```


Live Visitor Feed
Geoffrey
34.210.177.202
Visits
Visits
2
1
0
22:25
05 PM
22:30
22:35
22:40
22:45
Details
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/604.3.5 (KHTML, like Gecko) Version/11.0.1 Safari/604.3.5
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:57.0) Gecko/20100101 Firefox/57.0
5:22 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like

Provisioning Best Practices

Good

Automate the creation of infrastructure and initialization of instances with `user_data` or AWS `cloud-init`.

Better

Use `remote-exec` provisioner on a base AMI to run a few commands on instance creation.

Best

Build AMIs with Packer so that minimal configuration is needed.

HashiCorp

What You Learned

- How to create a connection to our server
- Specify the key for connecting to the server
- Provision the server with files and commands

HashiCorp

EXERCISE

Lab 6: Provisioners

Duration: 10 minutes

Goal: Provisioners allow us to create instances that are ready to use at runtime. Use your SSH key from the last lab to connect to and install our web app on a new instance.

Useful Providers

HashiCorp

```
resource "random_id" "server" {
  keepers {
    # Generate a new id each time we switch to a new AMI
    ami_id = var.ami_id
  }
  byte_length = 4
}

output "id" {
  value = random_id.server.hex
}
```

```
variable "animal" {}

data "template_file" "setup" {
  template = file("templates/setup.sh")

  vars {
    animal = var.animal
  }
}

output "template" {
  value = data.template_file.setup.rendered
}
```

```
#!/usr/bin/env sh

echo "Deploying ${animal}"
```

Theory Behind Terraform

HashiCorp

What You'll Learn

- Understand infrastructure as code
- Understand how graph theory makes Terraform more effective
- Generate a graph of your infrastructure

HashiCorp

Infrastructure as Code

Provide a codified workflow to create infrastructure

Expose a workflow for managing updates to existing infrastructure

Integrate with application code workflows (Git, SCM, Code Review)

Provide modular, sharable components for separation of concerns

HashiCorp

Infrastructure as Code (Terraform)

Human-readable configuration (HCL) is designed for human consumption so users can quickly interpret and understand their infrastructure configuration

Terraform can also parse configuration in JSON for machine-generated configurations

Configuration format is very VCS friendly with support for multi-line lists, trailing commas, and auto-formatting

HashiCorp

```
 resource "aws_instance" "web" {
    ami             = "ami-9a562df2"
-   instance_type = "t2.micro"
+   instance_type = "m1.small"
 }
```

Infrastructure as Code (Terraform)

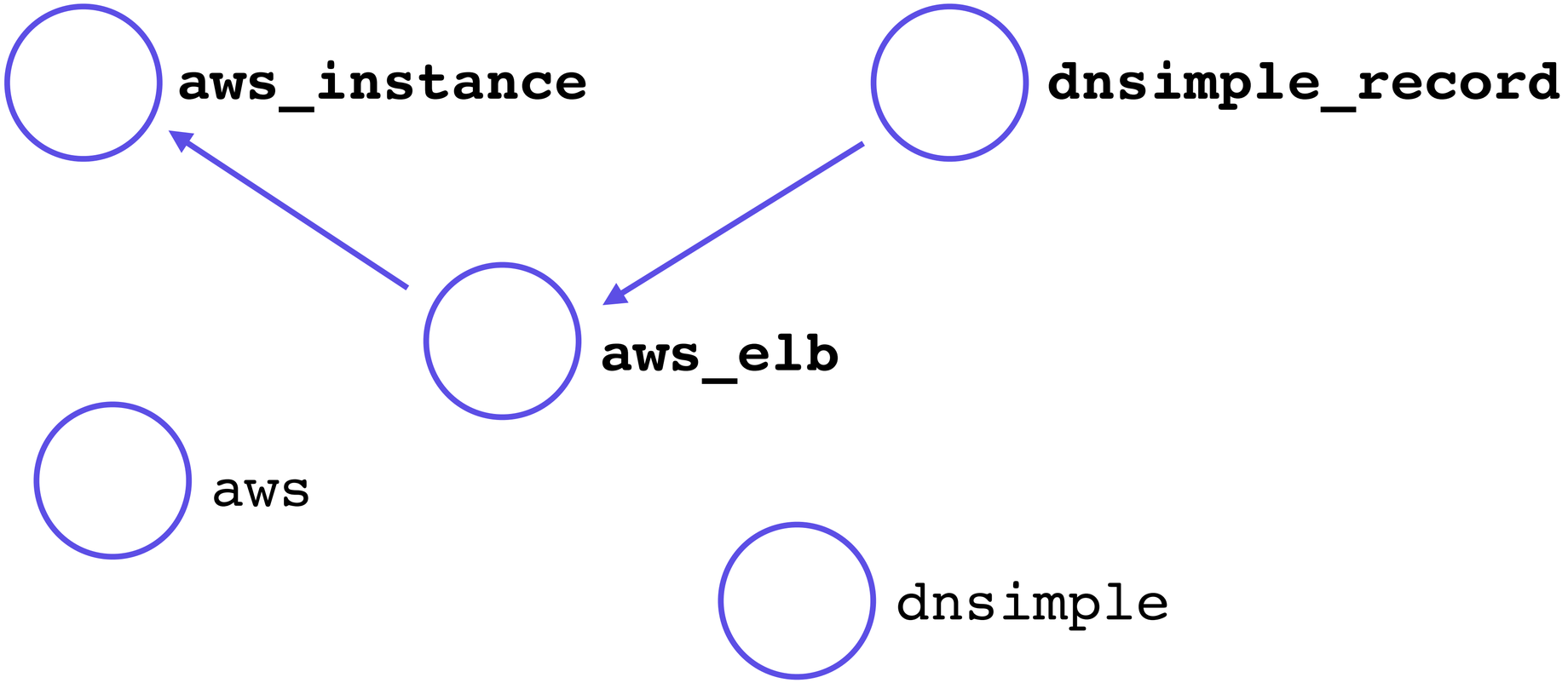
Reference values from other resources, building the implicit dependency graph.

HashiCorp

EXAMPLE.TF

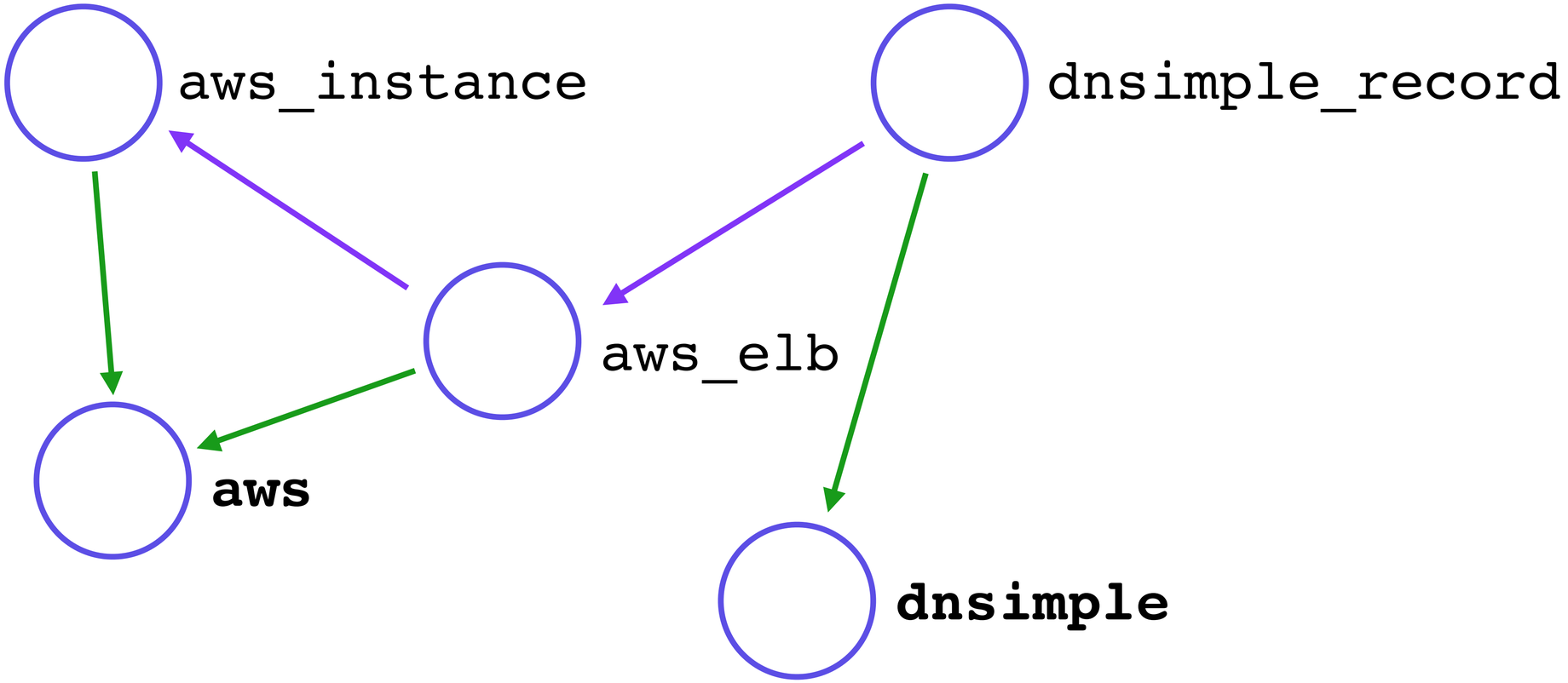
```
resource "aws_instance" "web" {
  ami           = "ami-0375ca3842950ade6"
  instance_type = "t2.micro"
}

resource "dnsimple_record" "web" {
  domain = "hashicorp.com"
  name   = "web"
  ttl    = "3600"
  type   = "A"
  value  = aws_instance.web.public_ip
}
```

Building the Graph



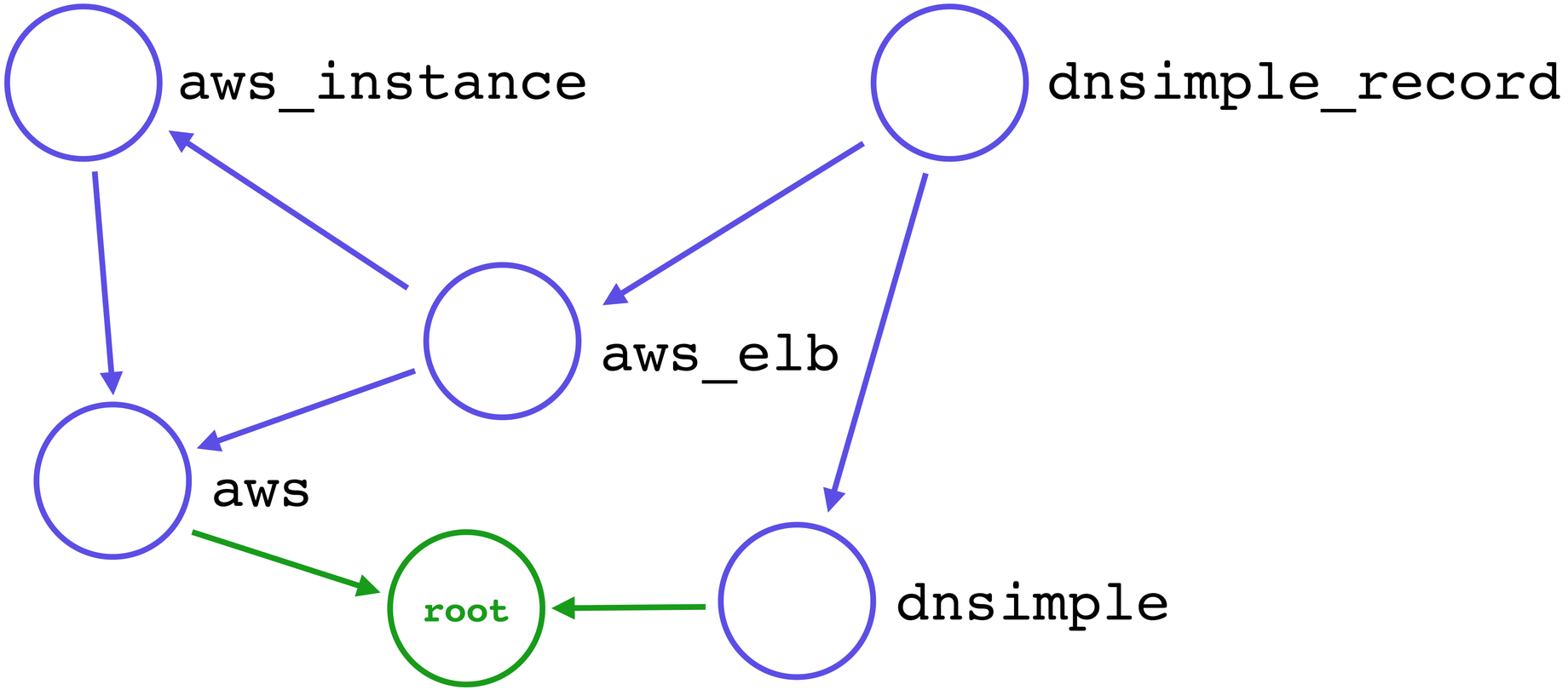
HashiCorp

Building the Graph



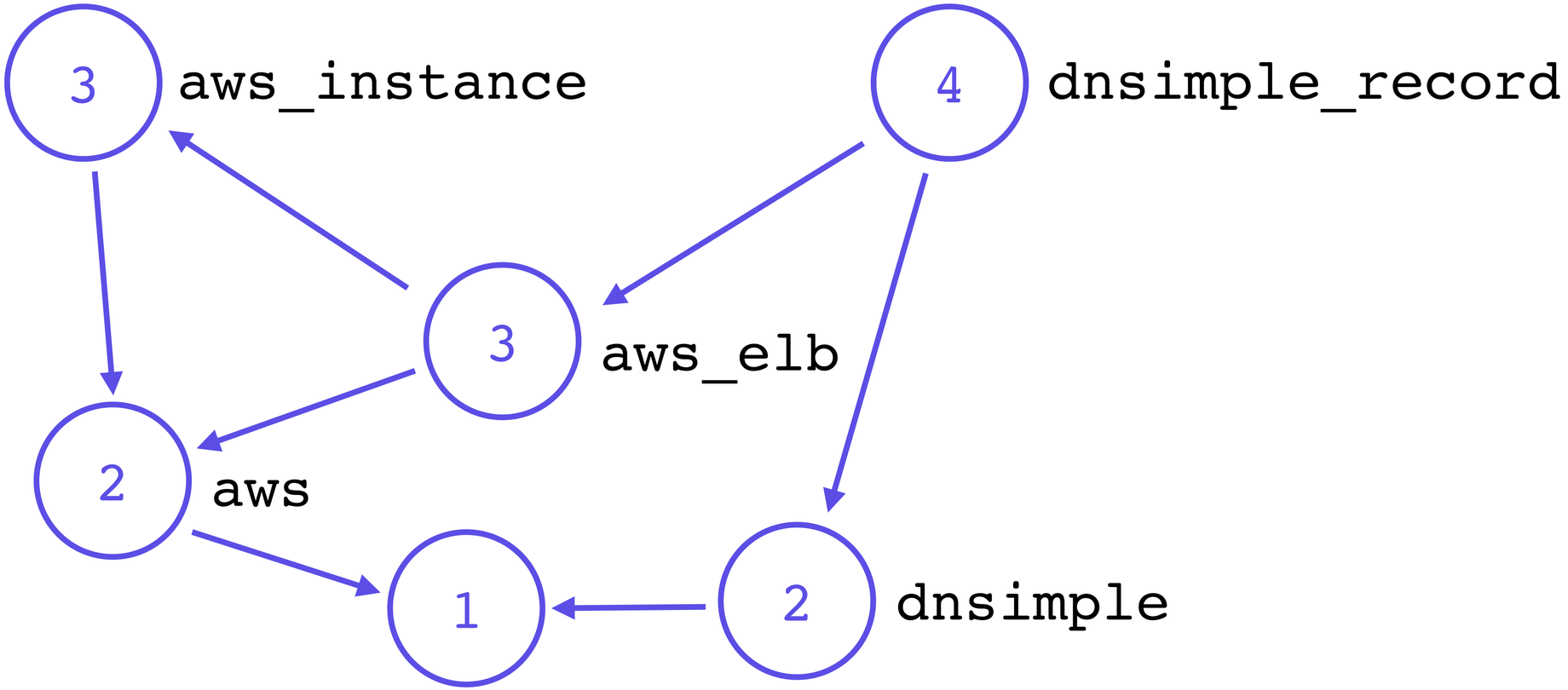
HashiCorp

Building the Graph



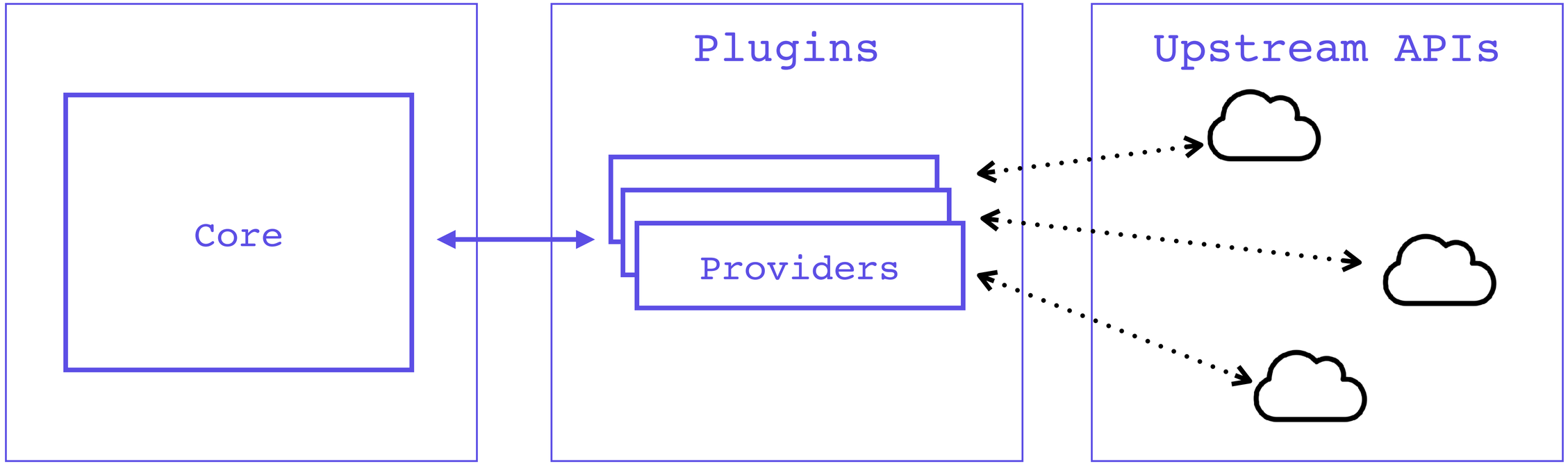
HashiCorp

Walking the Graph



HashiCorp

Terraform's Internals: Structure



HashiCorp

Terraform's Internals: Core Concepts

Config: Target Reality

State: Current Reality

Diff: {Config - State}

Plan: Presents Diff

Apply: Resolves Diff

HashiCorp

Resource Graph

The resource graph is an internal representation of all resources and their dependencies.

A human-readable graph can be generated using the `terraform graph` command.

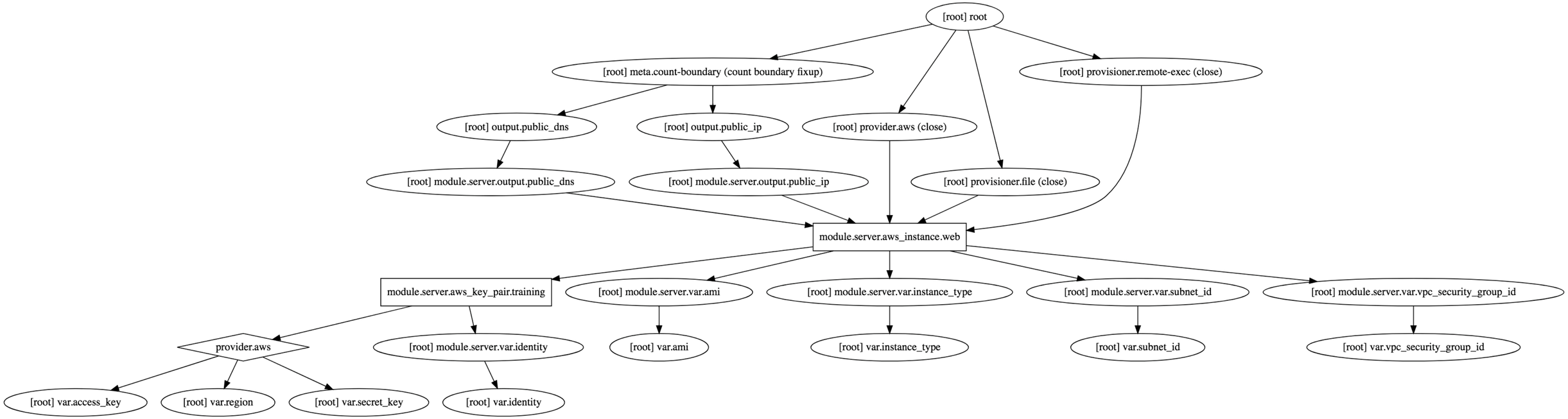
Can optionally draw cycles (advanced).

HashiCorp

TERMINAL

```
> terraform graph
digraph {
    compound = "true"
    newrank = "true"
    subgraph "root" {
        "[root] aws_instance.web" [label = "aws_instance.web", shape = "box"]
        "[root] provider.aws" [label = "provider.aws", shape = "diamond"]
        "[root] aws_instance.web" -> "[root] provider.aws"
        "[root] meta.count-boundary (count boundary fixup)" -> "[root] ..."
        "[root] provider.aws (close)" -> "[root] aws_instance.web"
        "[root] root" -> "[root] meta.count-boundary (count boundary fixup)"
        "[root] root" -> "[root] provider.aws (close)"
    }
}
```

terraform graph



HashiCorp

Terraform Graph

Useful for visualizing infrastructure and dependencies

Builds upon existing visualization technologies and open formats such as DOT

Can optionally draw cycles (resources that depend on each other in a circle)

HashiCorp

EXERCISE

Lab 7: Graphs

Duration: 10 minutes

Goal: Visualize the dependencies in a Terraform configuration with the graph command.

Meta Arguments

HashiCorp

What You'll Learn

- Understand meta arguments
- Use `count` to create multiple instances
- Manage your own metadata

HashiCorp

Meta-Arguments

Meta-arguments allow for higher-level control flow and lifecycle management in Terraform

These don’t map directly to cloud resources or APIs, but help you control Terraform’s actions

HashiCorp

Meta-Arguments

Count

The `count` argument allows for N number of identical resources to be created. This removes the need for iteration with "for" or "while" loops in many cases

HashiCorp

Meta-Arguments

Depends On

The `depends_on` argument allows for declaration of explicit dependencies. This is useful where interpolation is not required, but explicit ordering is desired

HashiCorp

Meta-Arguments

Provider

The `provider` argument allows for a resource to explicitly use a provider by name or alias. This is most useful when there are multiple providers in a single configuration, such as `aws.west` and `aws.east`

HashiCorp

Meta-Arguments

Lifecycle

The `lifecycle` argument allow explicit configuration of resource lifecycle such as preventing destruction or ignoring property changes. This is an advanced option and is not recommended for users who are getting started with Terraform

HashiCorp

```
# ...

resource "aws_instance" "web" {
  count           = 2

  ami             = "ami-07669fc90e6e6cc47"
  instance_type = "t2.micro"

# ...
}
```

```
# ...


output "public_ip" {
  value = aws_instance.web.*.public_ip
}

output "public_dns" {
  value = aws_instance.web.*.public_dns
}
```

EXERCISE

Lab 8: Meta Arguments

Duration: 10 minutes

Goal: Modify the number instances created with count.

HashiCorp

TERMINAL

```
> terraform apply
# module.server.aws_instance.web[1] will be created
+ resource "aws_instance" "web" {
    + ami:                          "ami-0375ca3842950ade6"
    + availability_zone:            "<computed>"
# ...

Plan: 1 to add, 0 to change, 0 to destroy.

# ...

Apply complete! Resources: 1 added, 0 changed, 0
destroyed.
```

```
variable "num_webservers" {
  count   = number
  default = 2
}

# ...

resource "aws_instance" "web" {
  count = var.num_webservers

  ami             = var.ami
  instance_type = "t2.micro"

  subnet_id                = var.subnet_id
  vpc_security_group_ids = [var.vpc_security_group_id]

  tags {
    # ...
    "Name" = "web ${count.index+1}/${var.num_webservers}"
  }
```

State

HashiCorp

What You'll Learn

- Understand how state is managed
- Understand where state can be stored
- Understand Terraform Cloud and Terraform Enterprise

HashiCorp

State

Terraform stores the state of your managed infrastructure from the last time Terraform was run.

Terraform uses this state to create plans and make changes to your infrastructure.

It is critical that this state is maintained appropriately so future runs operate as expected.

HashiCorp

TERMINAL

```
> head terraform.tfstate
{
    "version": 4,
    "terraform_version": "0.12.3",
    "serial": 46,
    "lineage": "b139a29e-9daa-47a9-b00f-33a4462a706b",
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {
                "public_dns": {
                    "sensitive": false,
                    "type": "list",
# ...
```

TERMINAL

```
> terraform state list

module.server.aws_instance.web[0]
module.server.aws_instance.web[1]
module.server.aws_key_pair.training

# ...
```

State Resolutions

Configuration	State	Reality	**Operation**
aws_instance.web			
aws_instance.web	aws_instance.web		
aws_instance.web	aws_instance.web	aws_instance.web	
	aws_instance.web	aws_instance.web	
		aws_instance.web	
aws_instance.web		aws_instance.web	
	aws_instance.web		

HashiCorp

State Resolutions

Configuration	State	Reality	**Operation**
aws_instance.web			**create**
aws_instance.web	aws_instance.web		**create**
aws_instance.web	aws_instance.web	aws_instance.web	**noop**
	aws_instance.web	aws_instance.web	**delete**
		aws_instance.web	**noop**
aws_instance.web		aws_instance.web	**re-create***
	aws_instance.web		**update state**

HashiCorp

State Import

Terraform only manages previous or imported resources.

Terraform can `import` state for existing resources. If you have already spun up an instance with the web console, you may need to import it so Terraform knows about it.

This only writes the state file, not the Terraform configuration.

HashiCorp

State Import

Critical Thinking

After importing an existing resource, what operation(s) would the `terraform plan` show?

HashiCorp

State Import

Critical Thinking

After importing an existing resource, what operation(s) would the `terraform plan` show?

Answer

Destroy (state exists, no configuration)

HashiCorp

State Locking

If supported, the state backend will "lock" to prevent concurrent modifications which could cause corruption.

Not all backends support locking - Terraform's documentation identifies which backends support this functionality (Terraform Enterprise, Terraform Cloud, S3, GCS, Azure Storage do)

HashiCorp

Where is State?

Local State (default)

Stored locally in a JSON format.
(`terraform.tfstate`)

Remote State

Stored on a remote source (Terraform Cloud, Terraform Enterprise, S3, GCS, Azure Storage, etc).

HashiCorp

Local State

State is stored locally on one machine

It is generally acceptable for individuals and small teams

No good way to back up or share state

Tends not to scale for larger teams

Requires a more "mono repo" pattern

HashiCorp

Remote State

State is on a remote source like S3, Terraform Cloud, Terraform Enterprise, or Consul

Remote storage is responsible for handling merging and locking

Can be queried for information in other Terraform configurations

Removes the risk of losing state to a hard drive crash or other data loss

HashiCorp

Sensitive Data in State

State can contain sensitive data depending on the resources used

Sometimes it can contain initial database passwords or other secret data returned by a provider

Some resources support PGP encrypting the values in the state, but this is implemented on a per-resource basis

We recommend never keeping state in source control, for example

HashiCorp

Sensitive Data in State

Local state (JSON) is not encrypted

Remote state encryption is backend-specific

State is only held in memory when remote state is used

Example: S3 bucket can be encrypted + IAM + TLS connection

Example: TFE encrypted in transit and rest + full audit log

HashiCorp

```
resource "aws_iam_access_key" "lb" {
  user    = aws_iam_user.lb.name
  pgp_key = "keybase:sethvargo"
}
```

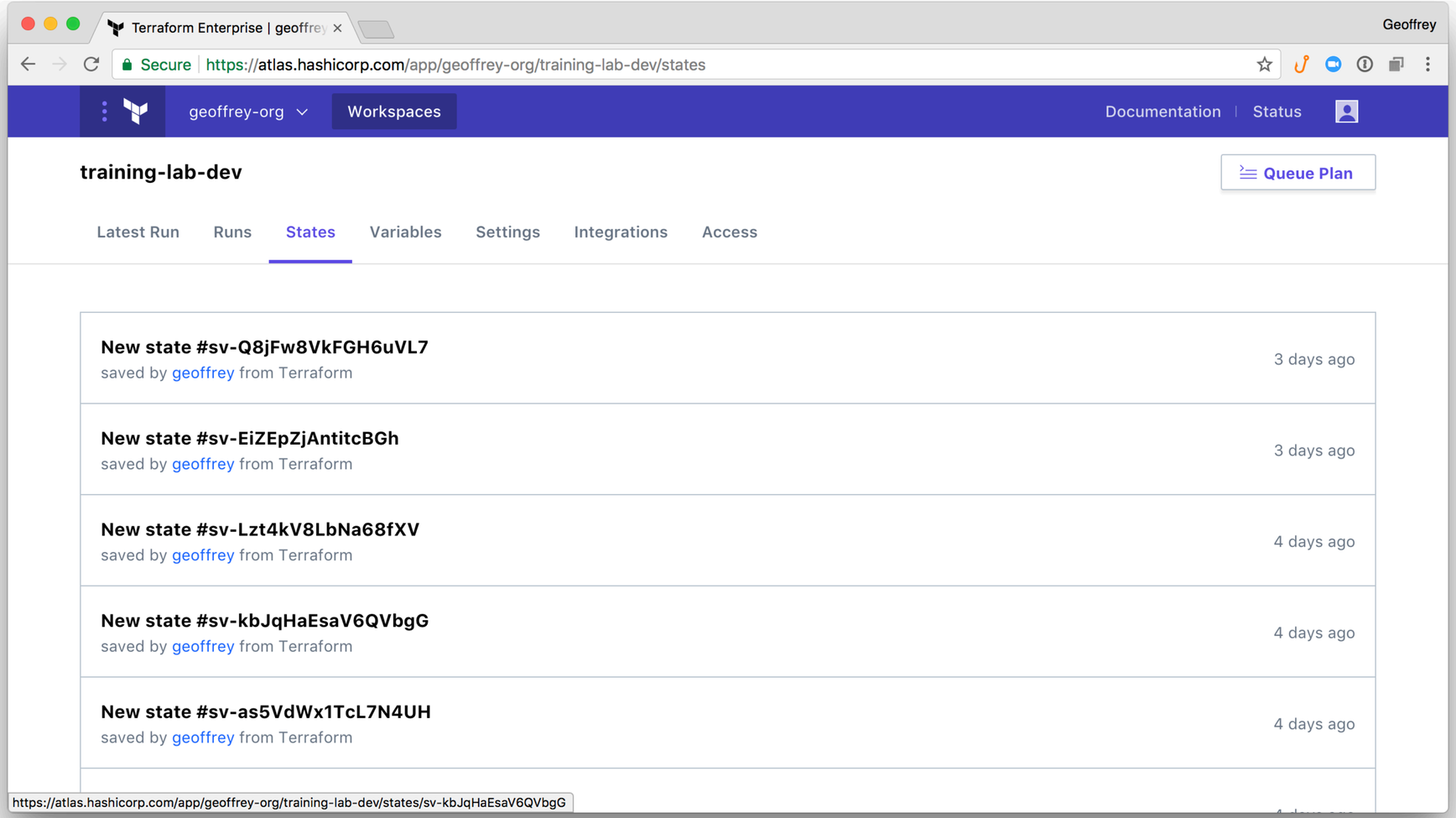
Terraform Cloud

HashiCorp

Terraform Ecosystem

Terraform CLI	Terraform Cloud	Terraform Enterprise
Command line tool	SaaS app	On premise/in your cloud
Infrastructure as Code	Free for individuals and teams	Workspace & team management
Open Source	Remote state	Governance and policy features
150+ Providers	Version Control System Integration	Private module registry
Module Registry		

HashiCorp


Terraform Enterprise | geoffrey
Geoffrey
Secure | https://atlas.hashicorp.com/app/geoffrey-org/training-lab-dev/states
geoffrey-org
Workspaces
Documentation | Status
training-lab-dev
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
New state #sv-Q8jFw8VkFGH6uVL7
saved by geoffrey from Terraform
3 days ago
New state #sv-EiZEpZjAntitcBGh
saved by geoffrey from Terraform
3 days ago
New state #sv-Lzt4kV8LbNa68fXV
saved by geoffrey from Terraform
4 days ago
New state #sv-kbJqHaEsaV6QVbgG
saved by geoffrey from Terraform
4 days ago
New state #sv-as5VdWx1TcL7N4UH
saved by geoffrey from Terraform
4 days ago
https://atlas.hashicorp.com/app/geoffrey-org/training-lab-dev/states/sv-kbJqHaEsaV6QVbgG

HashiCorp

Transitioning from Local to Remote

Transitioning is a one-time operation

After configured, Terraform will no longer store local state

The `terraform` stanza declares the configuration alongside resources

HashiCorp

```
terraform {
  backend "remote" {
    hostname = "app.terraform.io"
    organization = "example-company"

# Single workspace:
    workspaces {
      name = "my-app-prod"
    }

# Or use multiple workspaces:
#    workspaces {
#      prefix = "my-app-"
#    }
  }
}
```

```
data "terraform_remote_state" "vpc" {
  backend = "remote"

  config {
    organization = "example-org"
    hostname = "app.terraform.io"
  }

  workspaces {
    name = "example-workspace"
  }
}

resource “aws_instance" "example" {
  # ...
  subnet_id = data.terraform_remote_state.vpc.subnet_id
}
```

~/.TERRAFORMRC

```
credentials "app.terraform.io" {
  token = "<TOKEN>"
}
```

```
data "terraform_remote_state" "vpc" {
  backend = "remote"

  config {
    organization = "example-org"
    hostname = "app.terraform.io"
    access_token = "xxxxxx"
  }
}

resource "aws_instance" "example" {
  # ...
  subnet_id = data.terraform_remote_state.vpc.subnet_id
}
```

Backend Initialization

Remote State backends must be initialized before use

Initialization prepares the remote backend for storage

Initialization will prompt for any missing values and cache them locally in `.terraform` (which should be ignored from source)

Initialization will also optionally migrate state

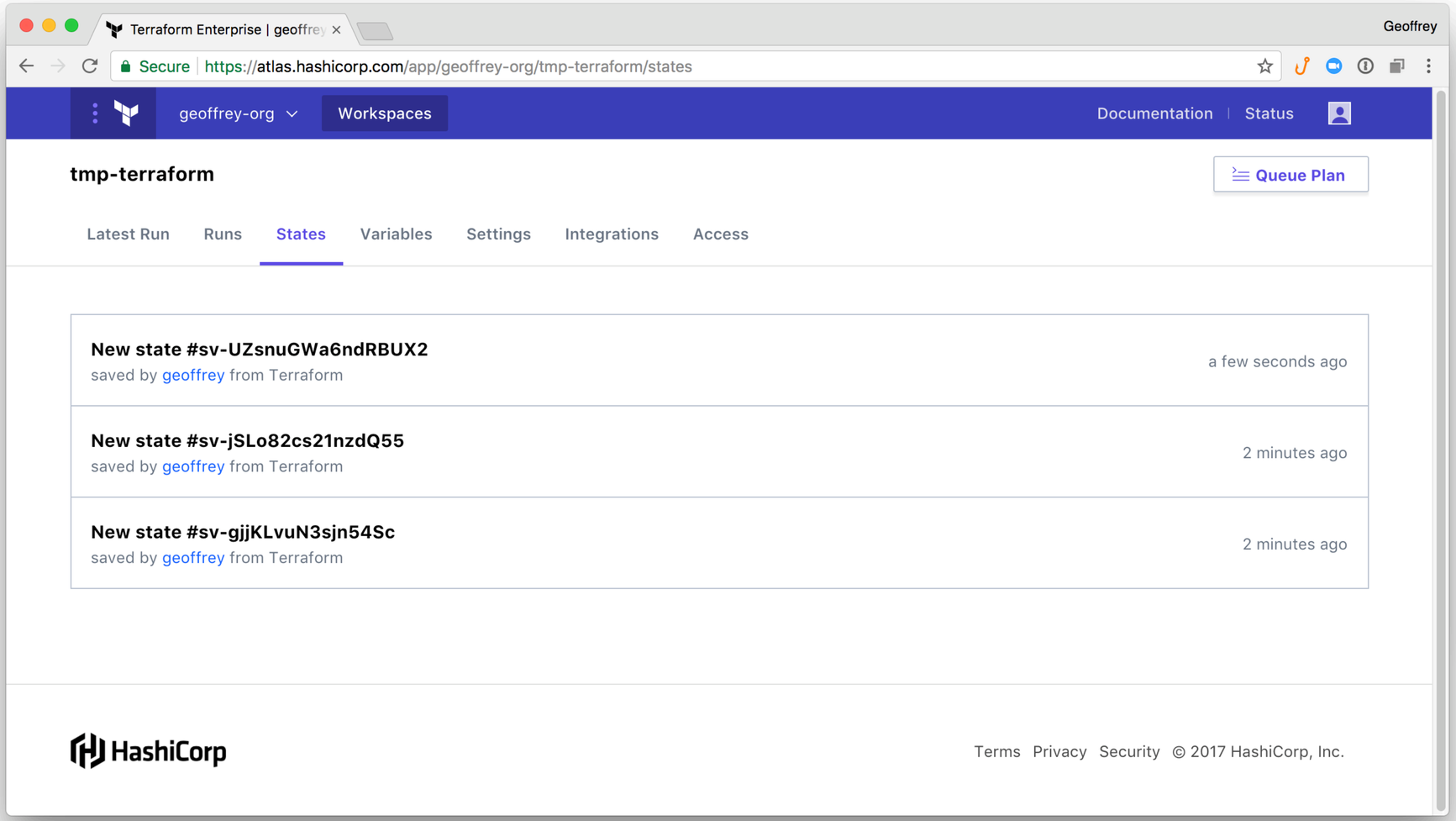
HashiCorp

TERMINAL

```
> terraform init

Initializing the backend...
Acquiring state lock. This may take a few moments...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the
previous "local" backend to the newly configured "remote"
backend. No existing state was found in the newly
configured "remote" backend.
  Do you want to copy this state to the new "remote"
backend? Enter "yes" to copy and "no" to start with an
empty state.

  Enter a value:
```

TERMINAL

```
Successfully configured the backend "remote"! Terraform
will automatically use this backend unless the backend
configuration changes.
```


Terraform Enterprise | geoffrey
Geoffrey
Secure https://atlas.hashicorp.com/app/geoffrey-org/tmp-terraform/states
geoffrey-org
Workspaces
Documentation
Status
tmp-terraform
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
New state #sv-UZsnuGWa6ndRBUX2
saved by geoffrey from Terraform
a few seconds ago
New state #sv-jSLo82cs21nzdQ55
saved by geoffrey from Terraform
2 minutes ago
New state #sv-gjjKLvuN3sjn54Sc
saved by geoffrey from Terraform
2 minutes ago
HashiCorp
Terms Privacy Security © 2017 HashiCorp, Inc.

Remote State

Terraform will no longer write to the `.tfstate` file

Everything else about your local workflow remains the same

HashiCorp

TERMINAL

```
> terraform apply
aws_instance.web.0: Refreshing state... (ID:
i-09ac7e80c7d33b461)

Apply complete! Resources: 0 added, 0 changed, 0
destroyed.

Outputs:

public_dns = [
    ec2-54-174-40-55.compute-1.amazonaws.com,
]
public_ip = [
    54.174.40.55
]
```

EXERCISE

Lab 9: Data Source (Optional)

Duration: 10 minutes

Goal: Use a data source to select the AMI used for your web server

Destroy

HashiCorp

What You'll Learn

- Destroy infrastructure that Terraform has created

HashiCorp

Command: `terraform destroy`

Destroys running infrastructure

Does not touch infrastructure not managed by Terraform

HashiCorp

TERMINAL

```
> terraform destroy
aws_instance.web: Refreshing state...
[id=i-0ec4ffe55abd300a5]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following
symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.web will be destroyed
  - resource "aws_instance" "web" {
# ...
```

EXERCISE

Lab 10: Destroy

Duration: 5 minutes

Goal: Use the destroy command to destroy your infrastructure

What You'll Learn

- Create a repository at GitHub
- Push your code to GitHub
- Create an organization at Terraform Cloud
- Connect to GitHub via OAuth
- Add a workspace that references a GitHub repo
- Commit changes and create infrastructure

HashiCorp

Terraform Cloud

Self-service workflow

Collaboration for teams

Powerful ACLs & auditing

Runs Terraform for you

Control Terraform version

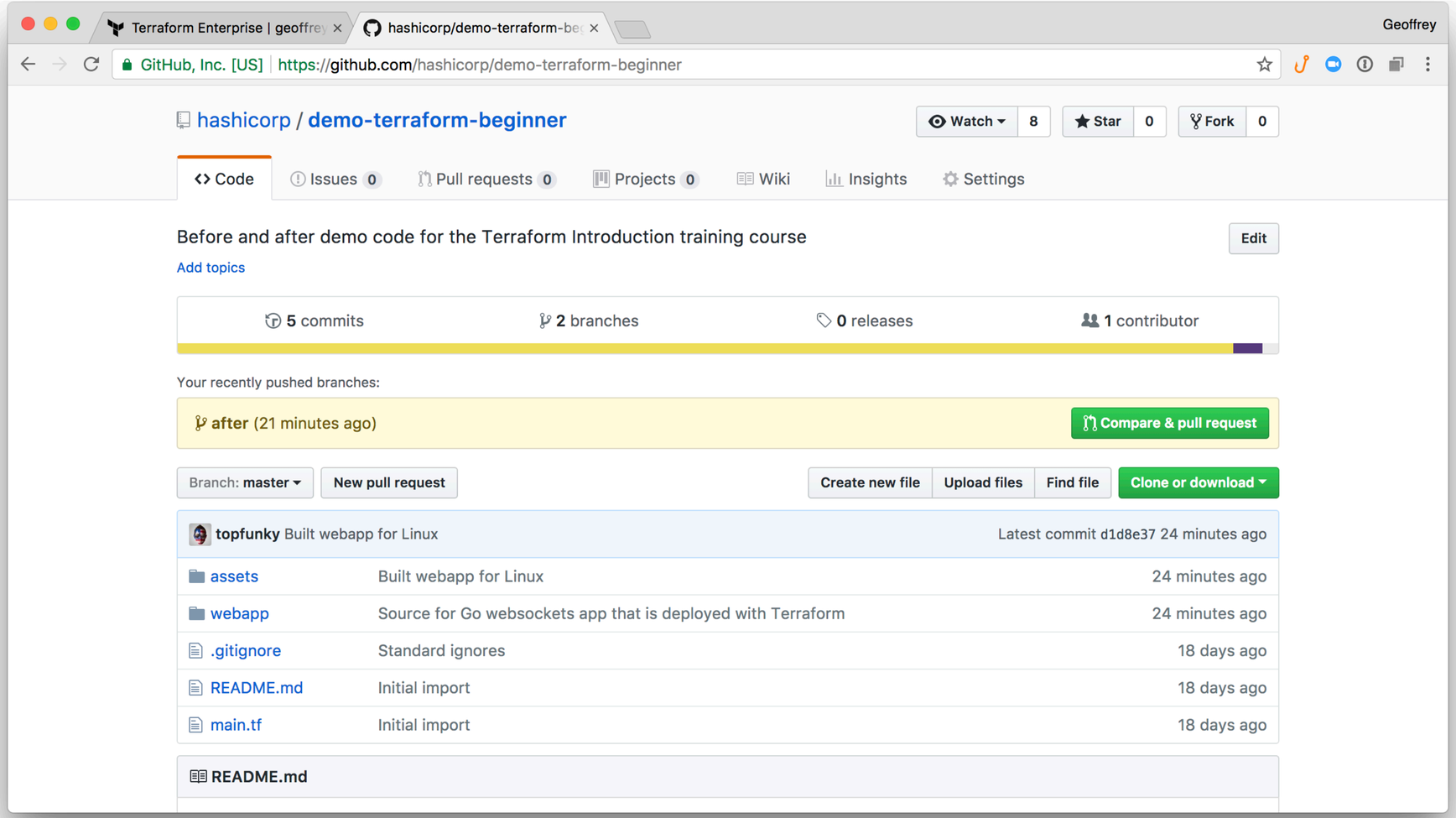
Prevent concurrent changes

Integrate with SCM

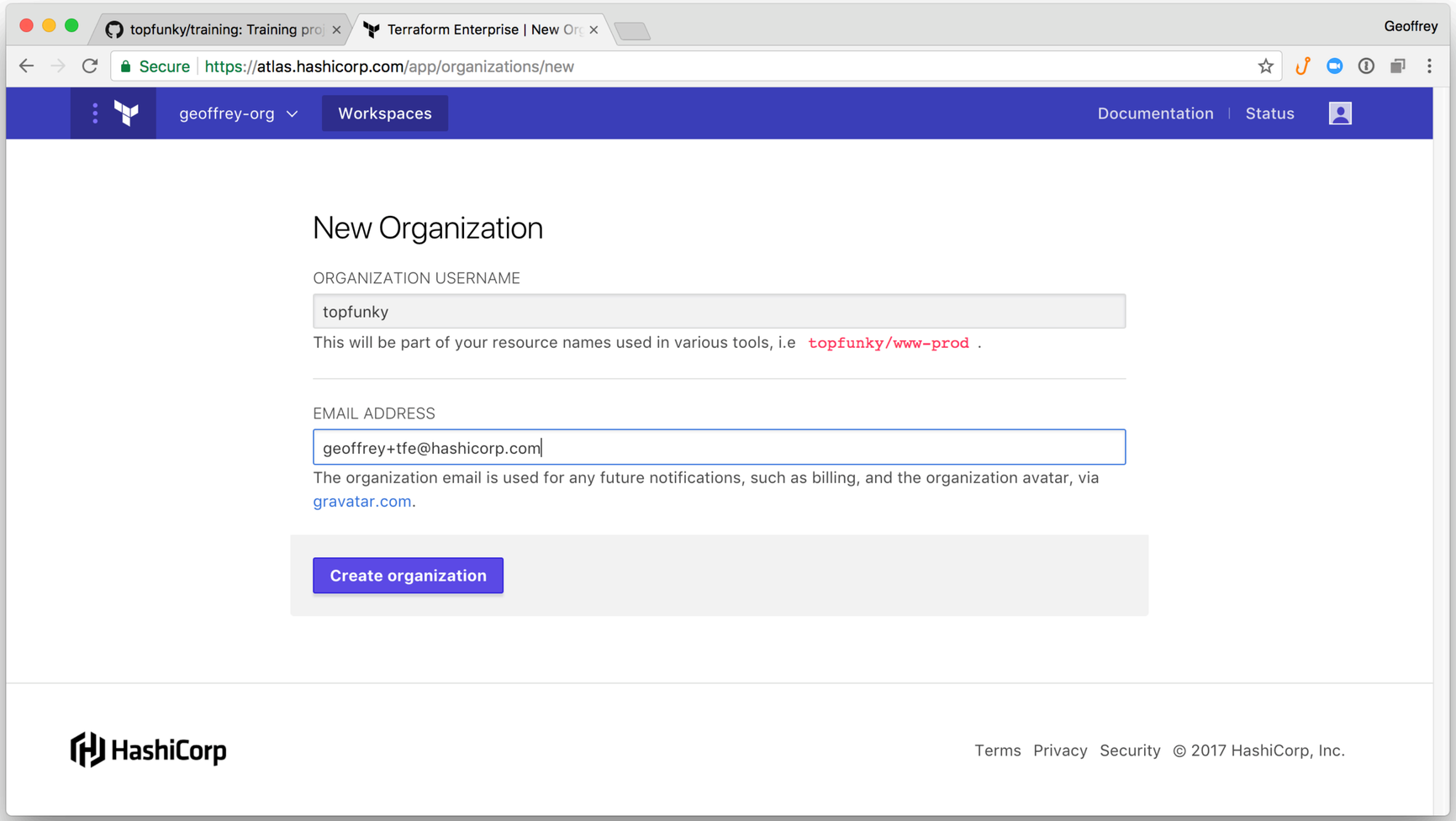
View history of changes

Enforce policies with Sentinel

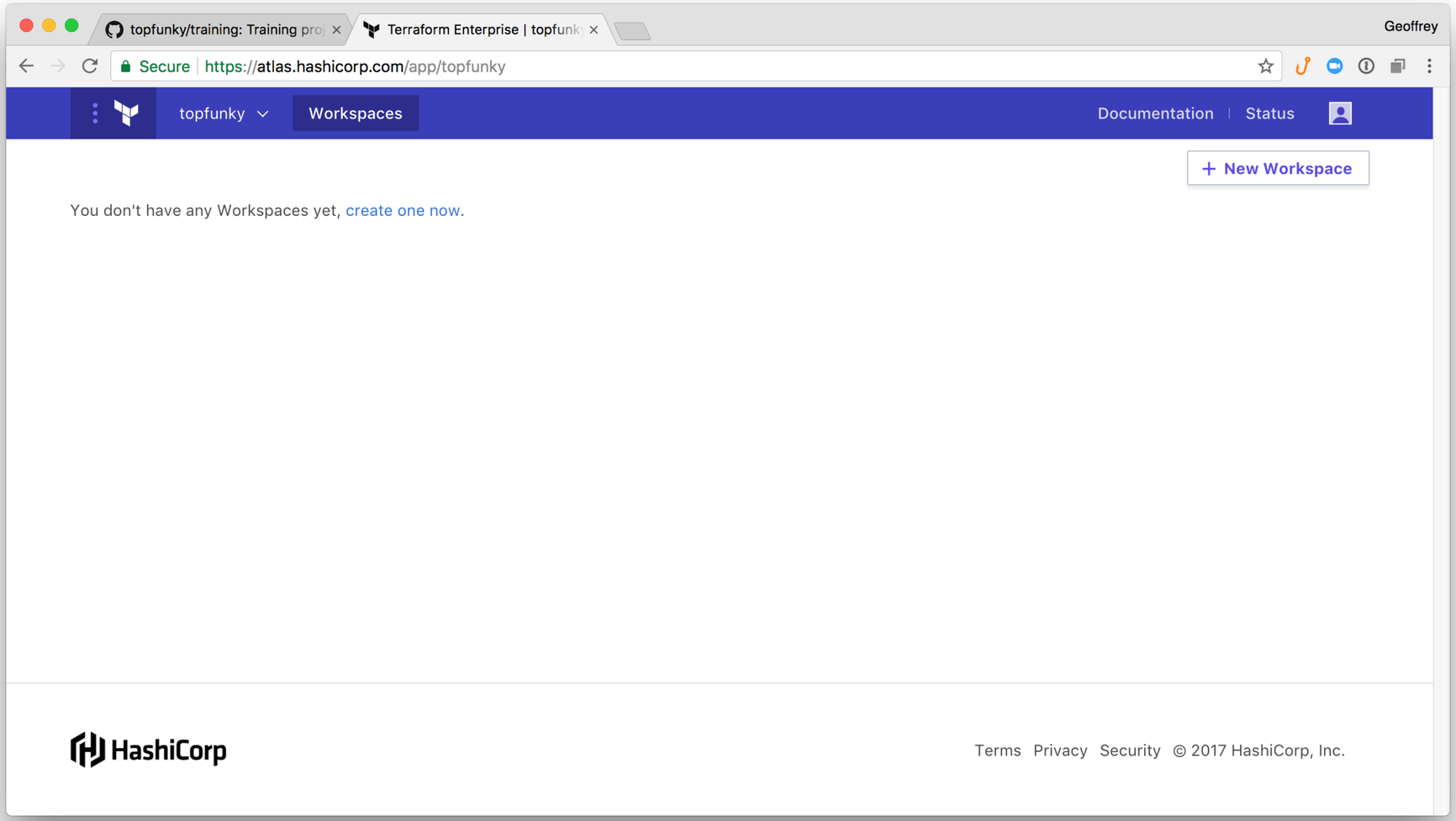
HashiCorp

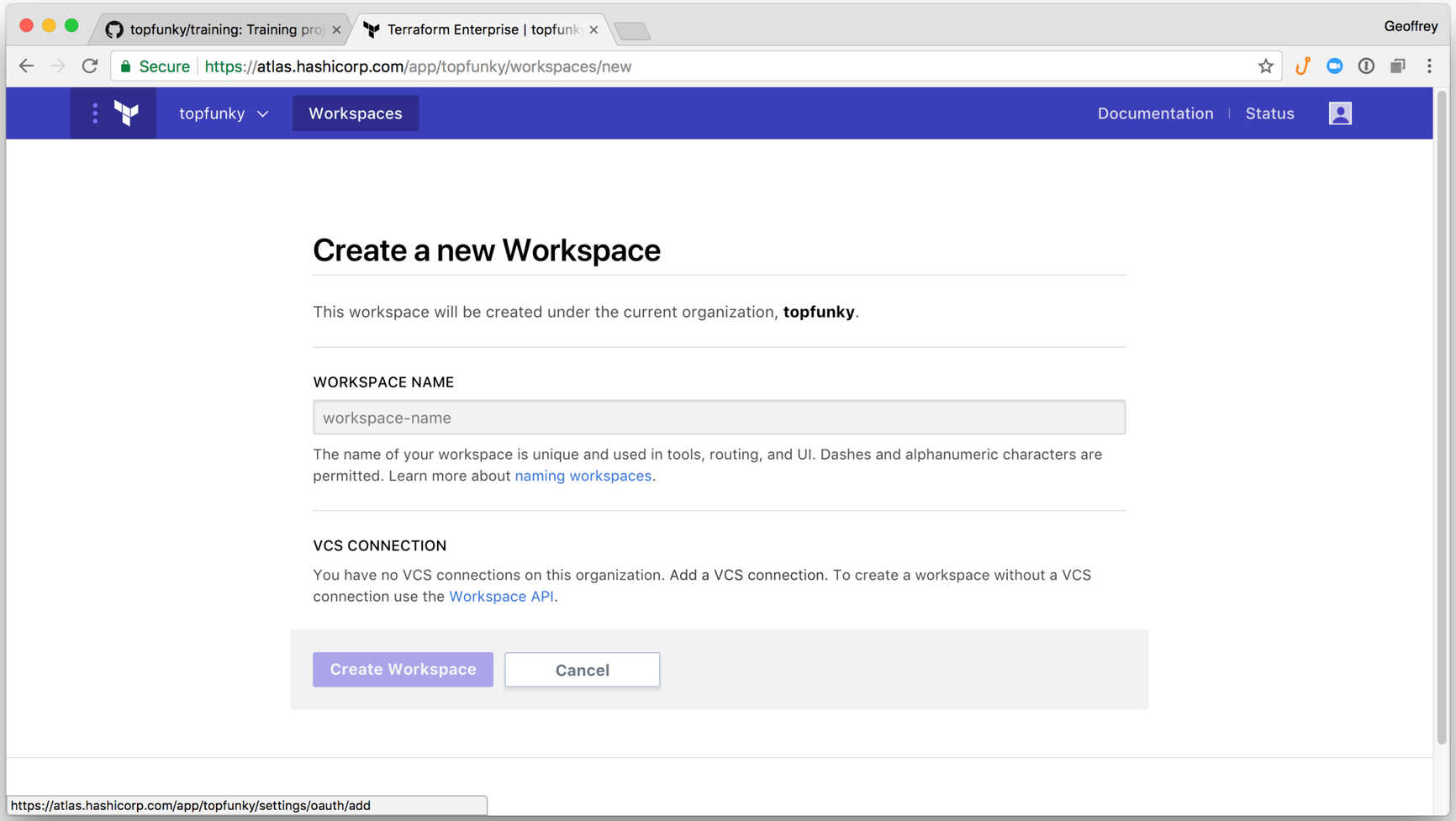

Terraform Enterprise | geoffrey
hashicorp/demo-terraform-beg
Geoffrey
GitHub, Inc. [US] | https://github.com/hashicorp/demo-terraform-beginner
hashicorp / demo-terraform-beginner
Watch 8
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Before and after demo code for the Terraform Introduction training course
Edit
Add topics
5 commits
2 branches
0 releases
1 contributor
Your recently pushed branches:
after (21 minutes ago)
Compare & pull request
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
topfunky Built webapp for Linux
Latest commit d1d8e37 24 minutes ago
assets
Built webapp for Linux
24 minutes ago
webapp
Source for Go websockets app that is deployed with Terraform
24 minutes ago
.gitignore
Standard ignores
18 days ago
README.md
Initial import
18 days ago
main.tf
Initial import
18 days ago
README.md

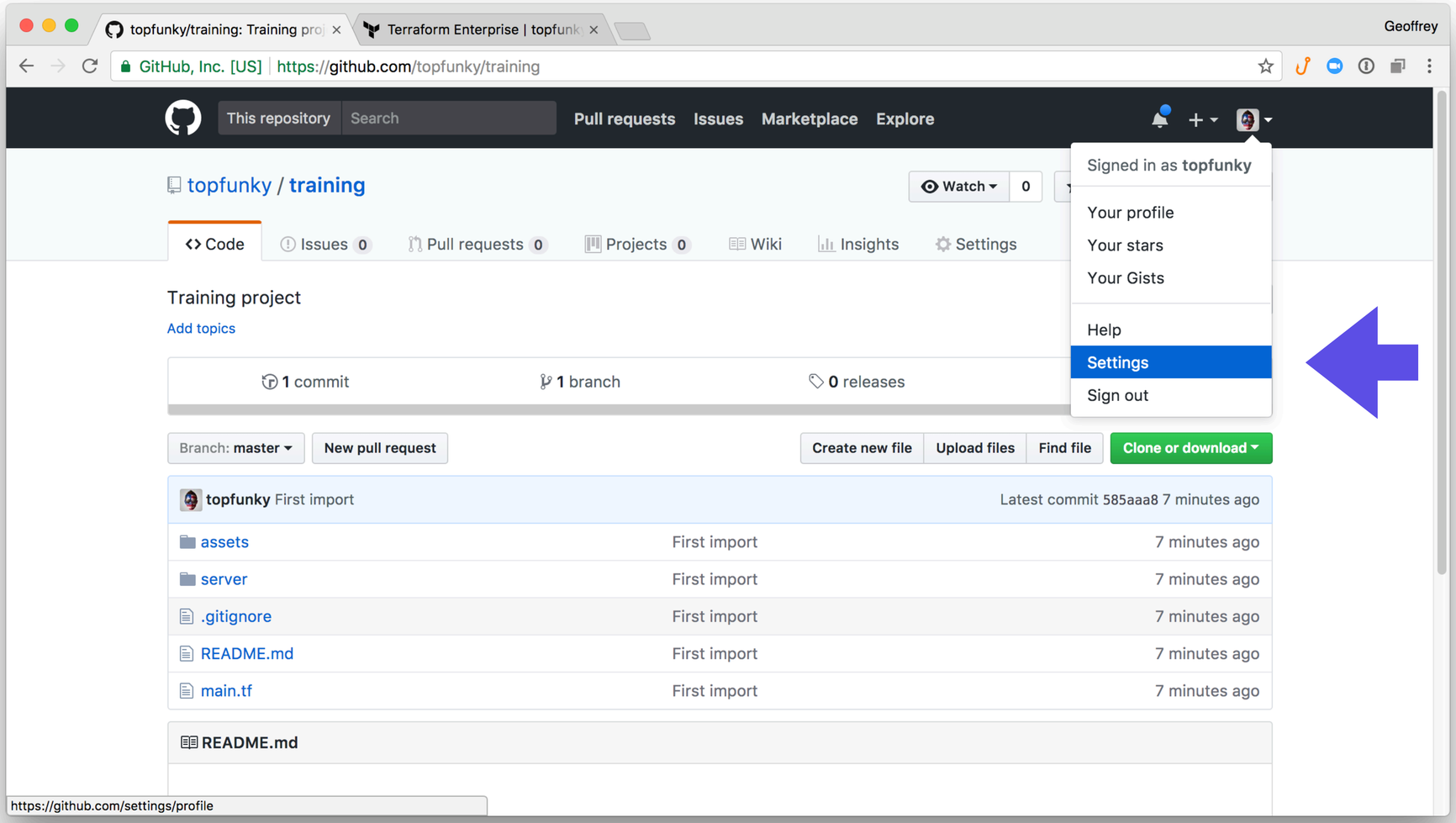
HashiCorp

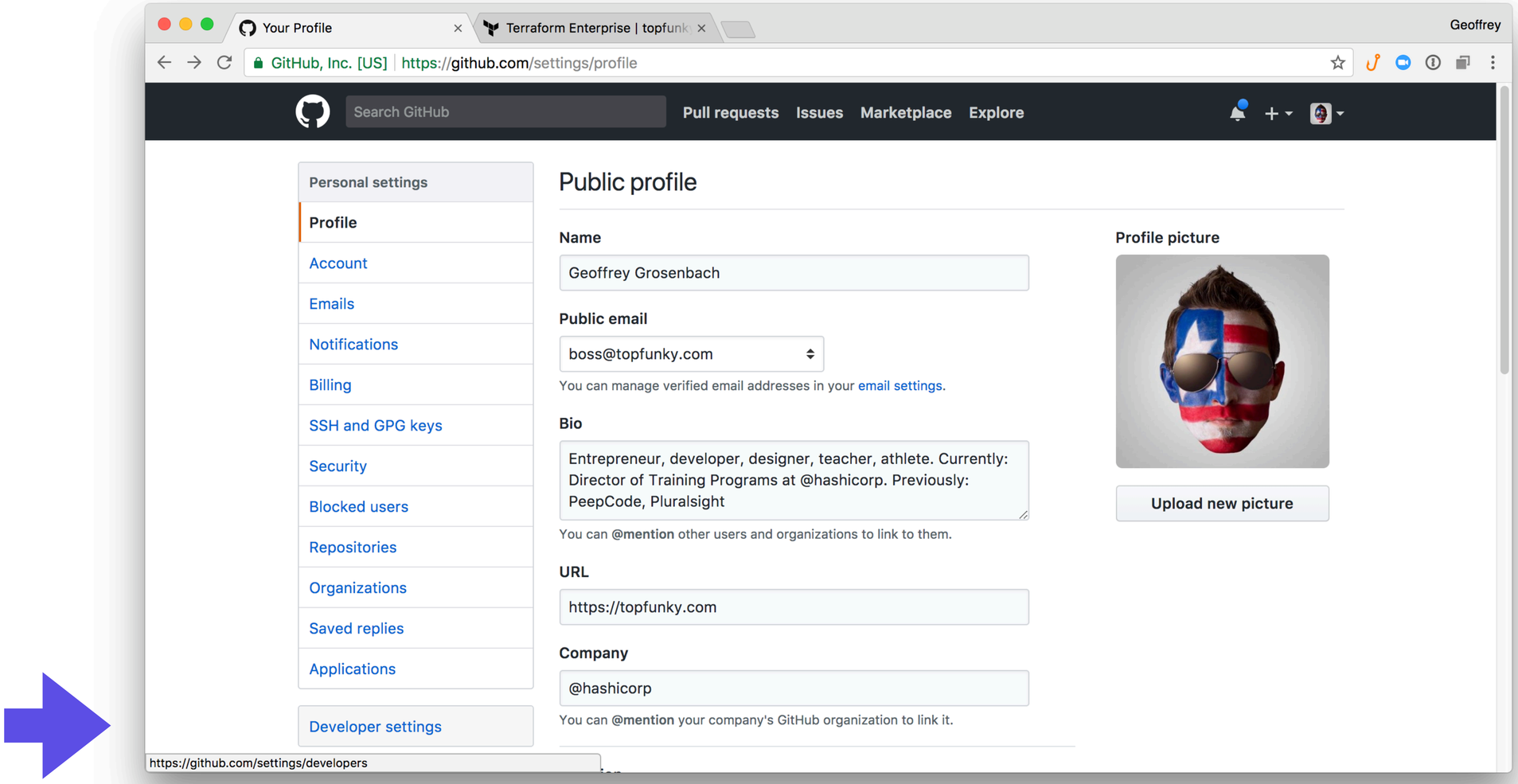


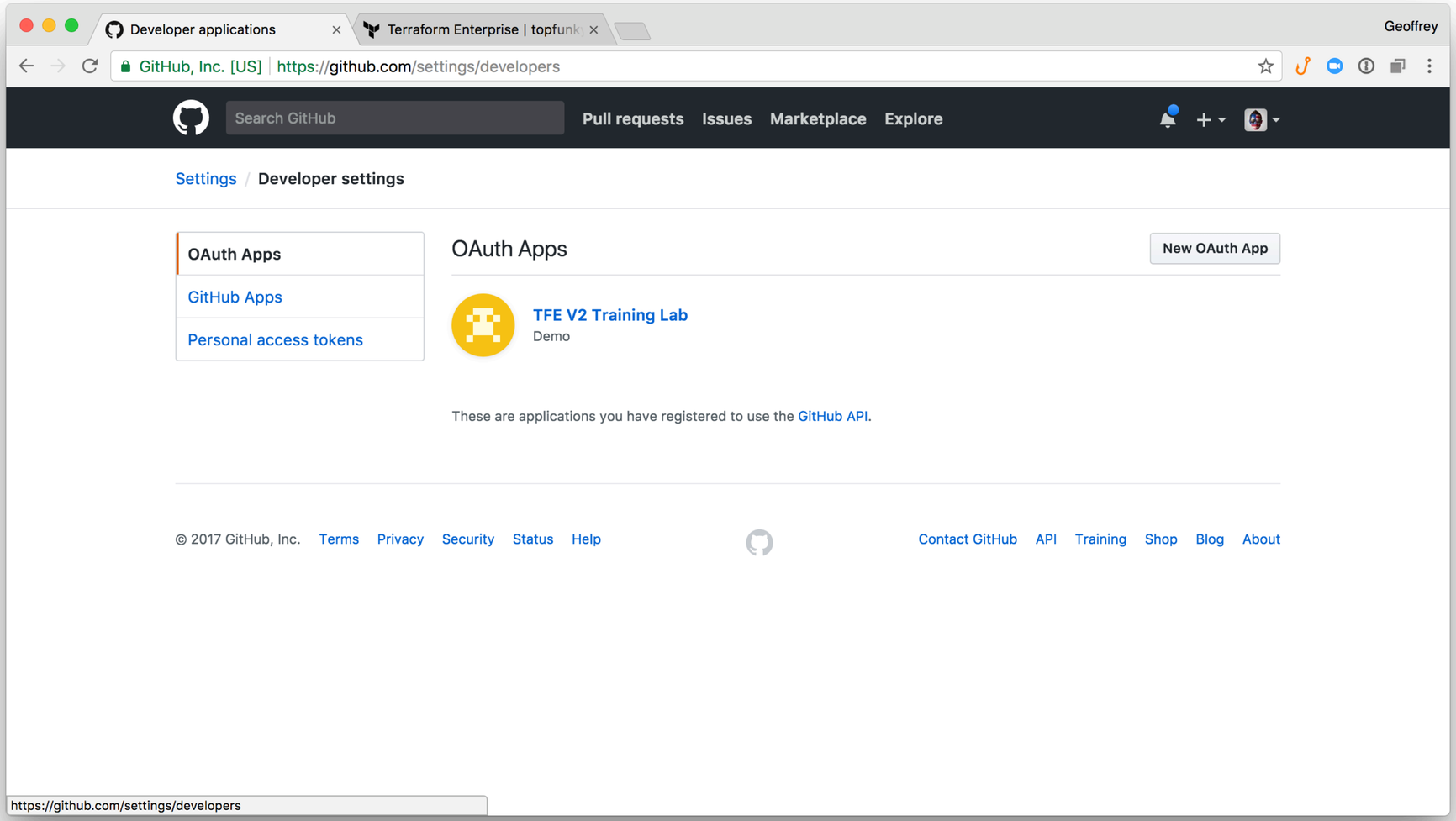
https://www.terraform.io/docs/enterprise/vcs/github.html

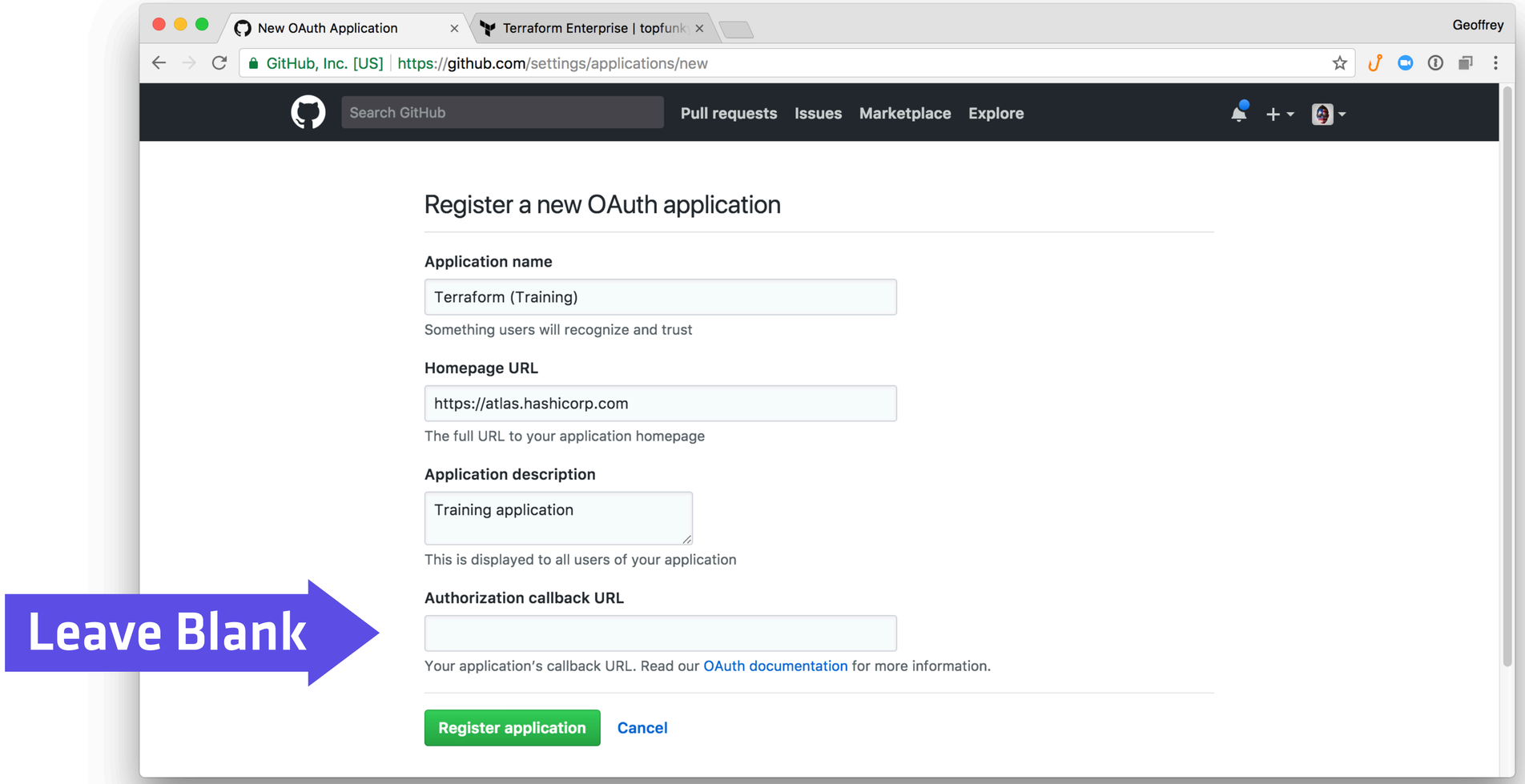

topfunky/training: Training proj
Terraform Enterprise | topfunky
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky
topfunky
Workspaces
Documentation
Status
+ New Workspace
You don't have any Workspaces yet, create one now.
HashiCorp
Terms
Privacy
Security
© 2017 HashiCorp, Inc.

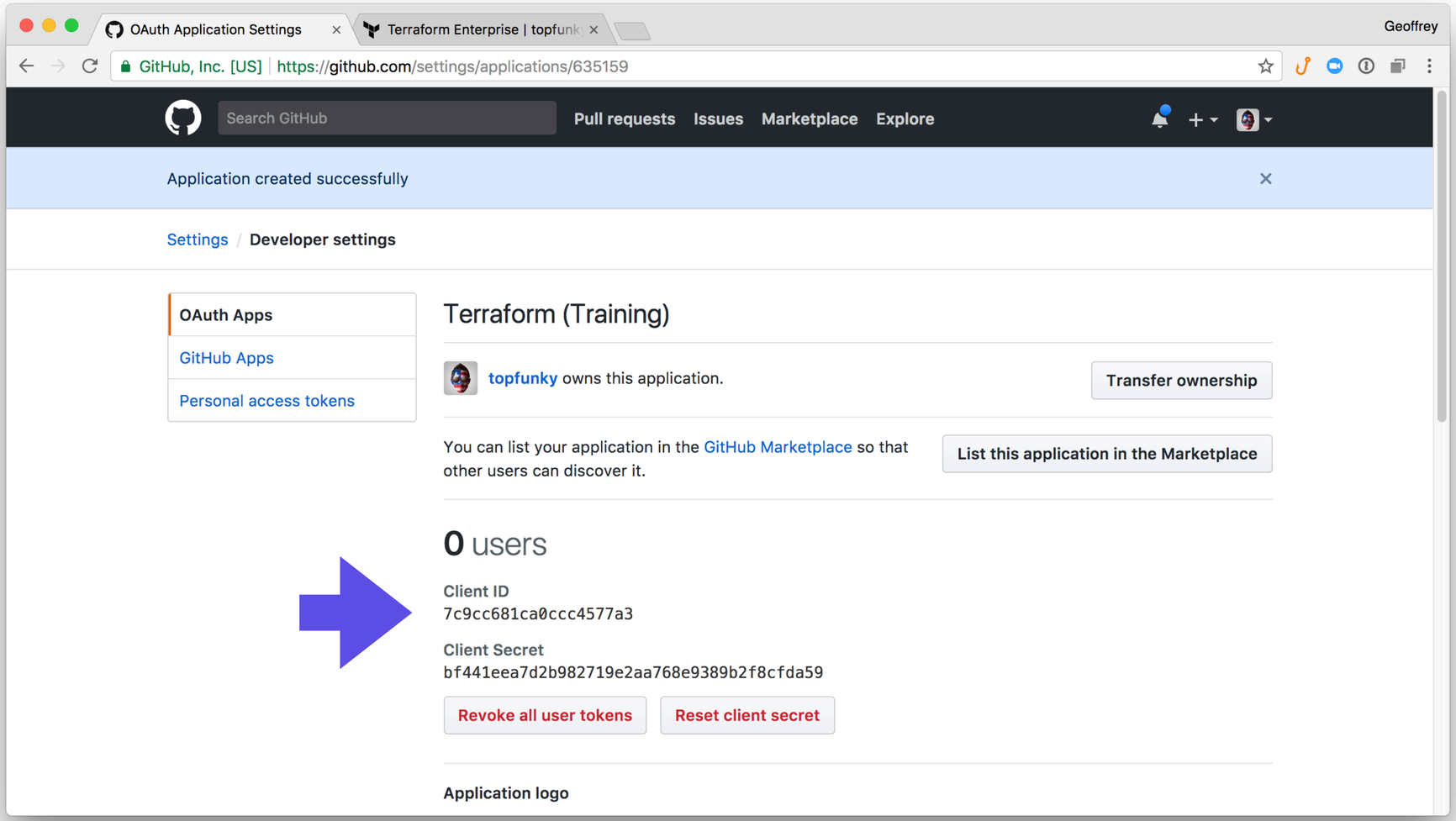

topfunky/training: Training proj
Terraform Enterprise | topfunky
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky/workspaces/new
topfunky
Workspaces
Documentation
Status
Create a new Workspace
This workspace will be created under the current organization, topfunky.
WORKSPACE NAME
workspace-name
The name of your workspace is unique and used in tools, routing, and UI. Dashes and alphanumeric characters are permitted. Learn more about naming workspaces.
VCS CONNECTION
You have no VCS connections on this organization. Add a VCS connection. To create a workspace without a VCS connection use the Workspace API.
Create Workspace
Cancel
https://atlas.hashicorp.com/app/topfunky/settings/oauth/add

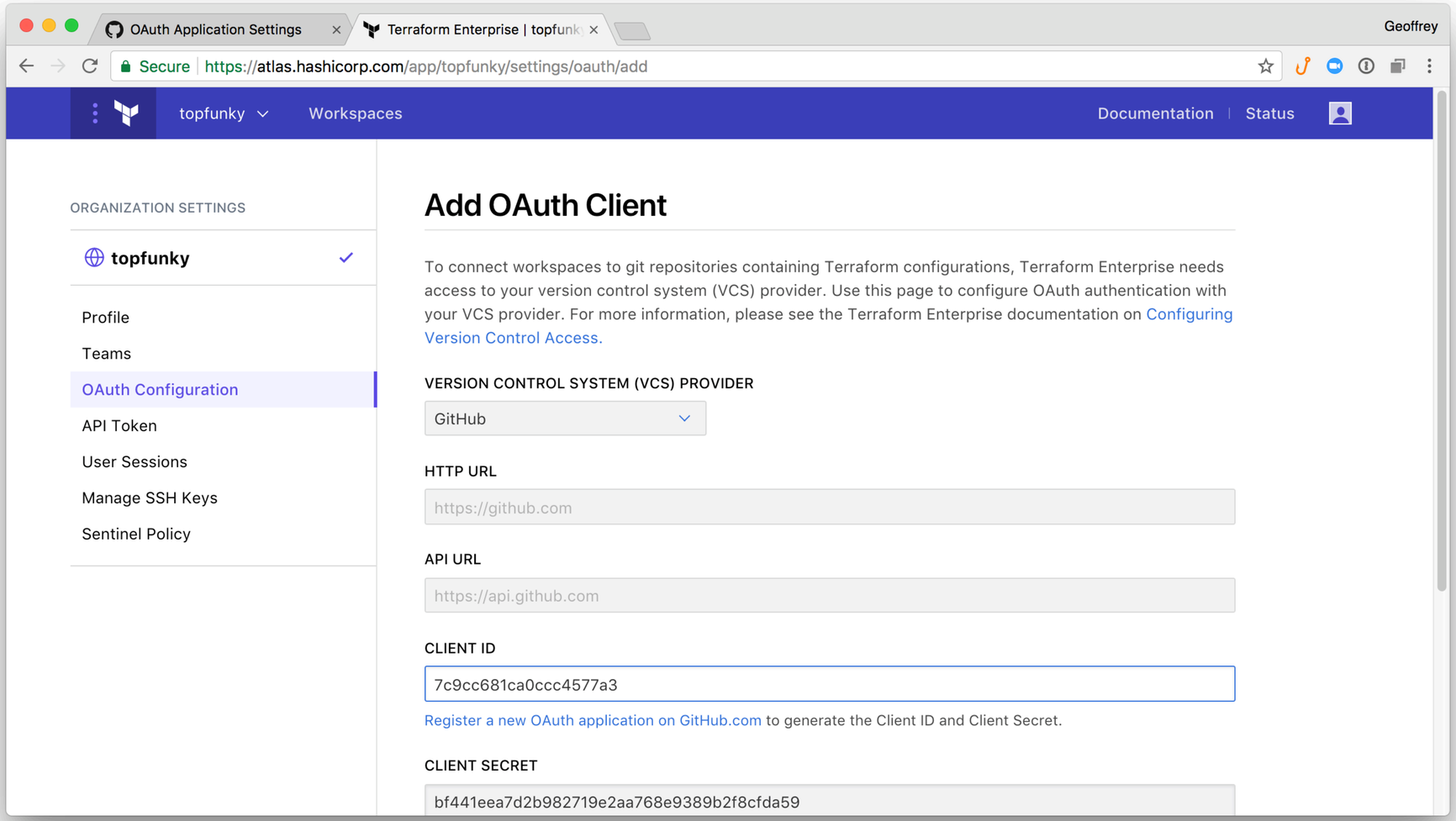

topfunky/training: Training proj
Terraform Enterprise | topfunky
Geoffrey
GitHub, Inc. [US] | https://github.com/topfunky/training
This repository
Search
Pull requests
Issues
Marketplace
Explore
Signed in as topfunky
Your profile
Your stars
Your Gists
Help
Settings
Sign out
topfunky / training
Watch
0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Training project
Add topics
1 commit
1 branch
0 releases
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
topfunky First import
Latest commit 585aaa8 7 minutes ago
assets First import 7 minutes ago
server First import 7 minutes ago
.gitignore First import 7 minutes ago
README.md First import 7 minutes ago
main.tf First import 7 minutes ago
README.md
https://github.com/settings/profile

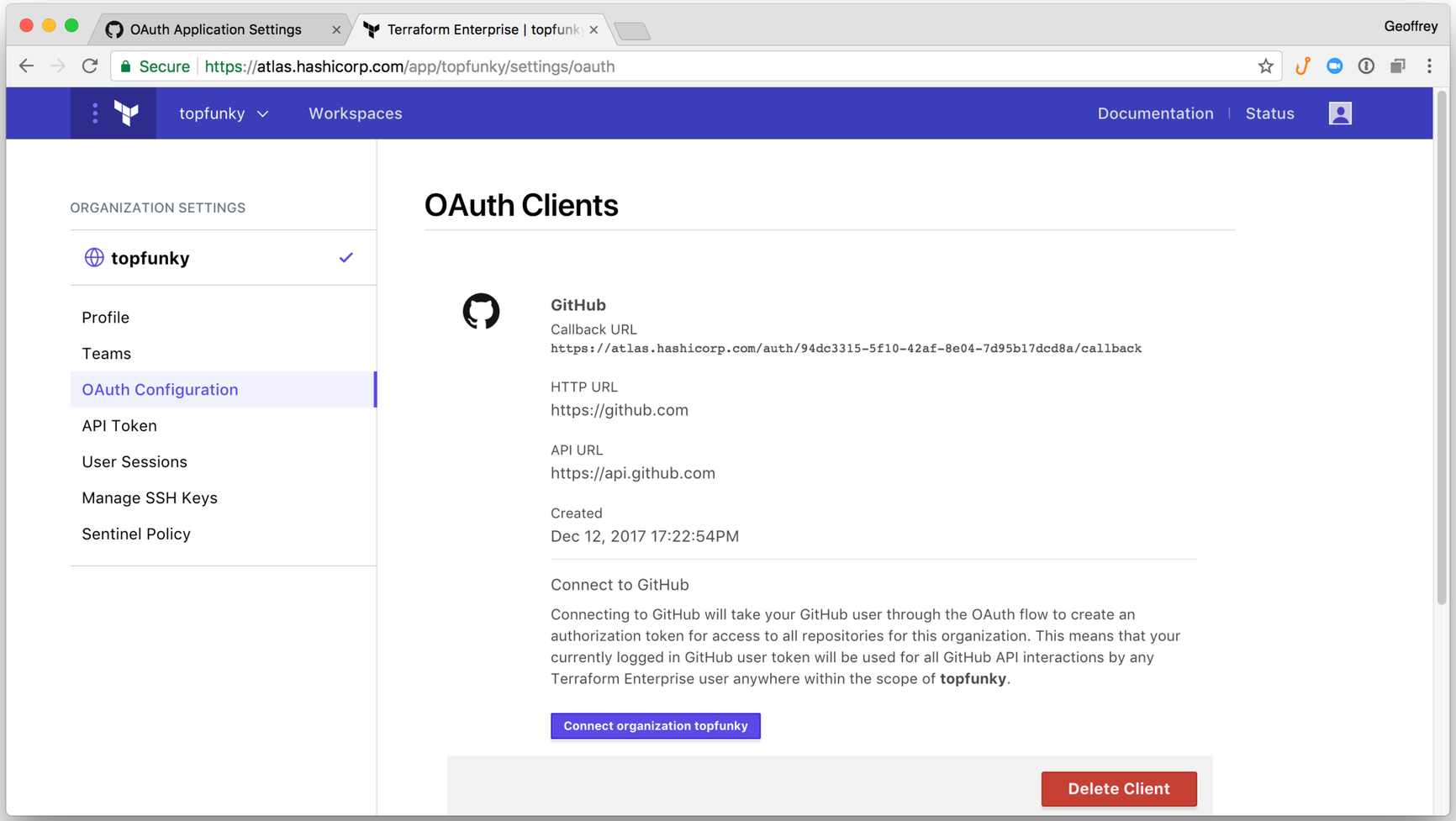

Your Profile
Terraform Enterprise | topfunk
Geoffrey
GitHub, Inc. [US] | https://github.com/settings/profile
Search GitHub
Pull requests
Issues
Marketplace
Explore
Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings
Public profile
Name
Geoffrey Grosenbach
Public email
boss@topfunky.com
You can manage verified email addresses in your email settings.
Bio
Entrepreneur, developer, designer, teacher, athlete. Currently: Director of Training Programs at @hashicorp. Previously: PeepCode, Pluralsight
You can @mention other users and organizations to link to them.
URL
https://topfunky.com
Company
@hashicorp
You can @mention your company's GitHub organization to link it.
Profile picture
Upload new picture
https://github.com/settings/developers

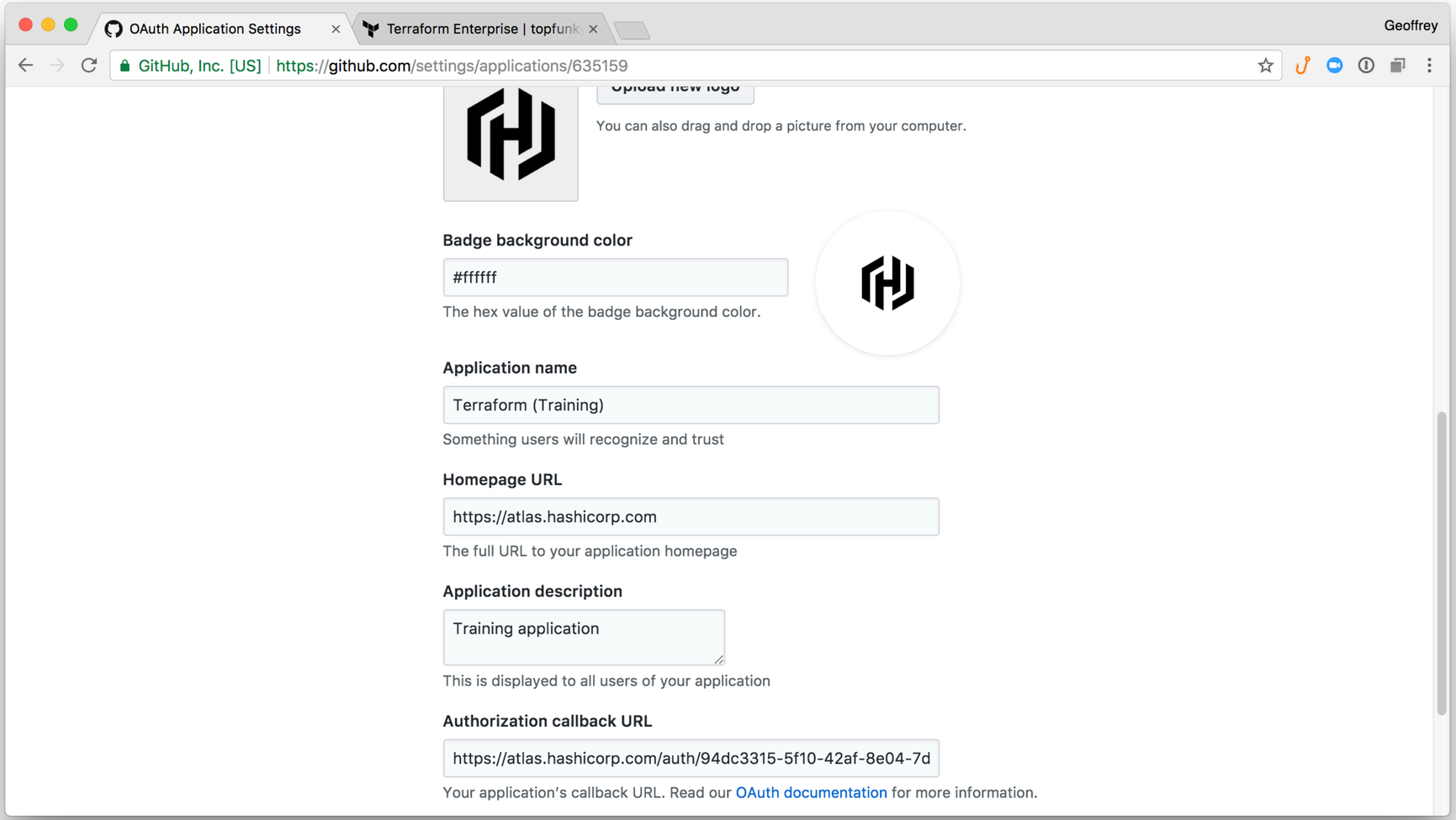

Developer applications
Terraform Enterprise | topfunk
Geoffrey
GitHub, Inc. [US] | https://github.com/settings/developers
Search GitHub
Pull requests
Issues
Marketplace
Explore
Settings / Developer settings
OAuth Apps
GitHub Apps
Personal access tokens
OAuth Apps
New OAuth App
TFE V2 Training Lab
Demo
These are applications you have registered to use the GitHub API.
© 2017 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub API Training Shop Blog About
https://github.com/settings/developers

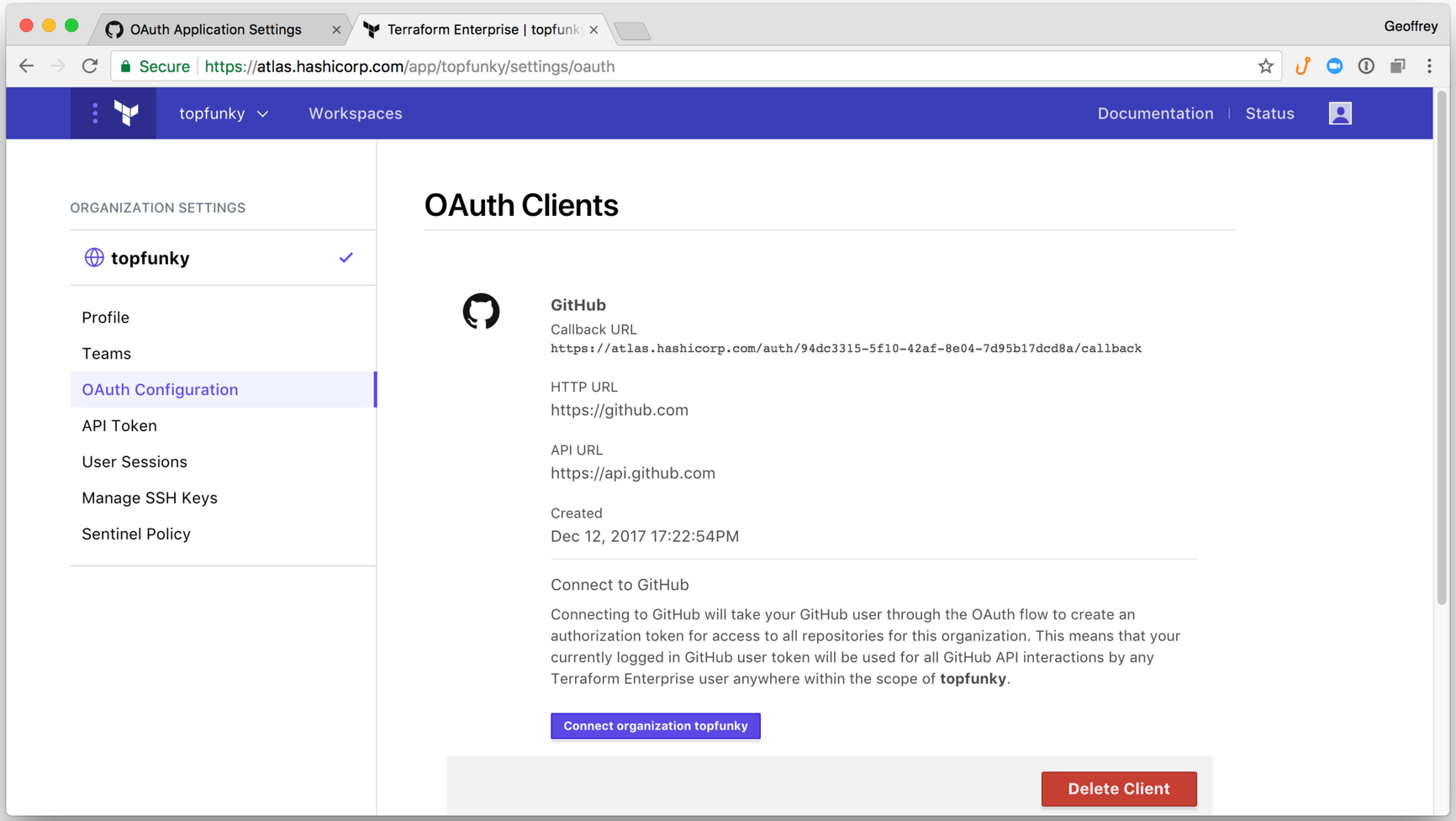

New OAuth Application
Terraform Enterprise | topfunk
Geoffrey
GitHub, Inc. [US] | https://github.com/settings/applications/new
Search GitHub
Pull requests
Issues
Marketplace
Explore
Register a new OAuth application
Application name
Terraform (Training)
Something users will recognize and trust
Homepage URL
https://atlas.hashicorp.com
The full URL to your application homepage
Application description
Training application
This is displayed to all users of your application
Authorization callback URL
Your application's callback URL. Read our OAuth documentation for more information.
Register application
Cancel
Leave Blank

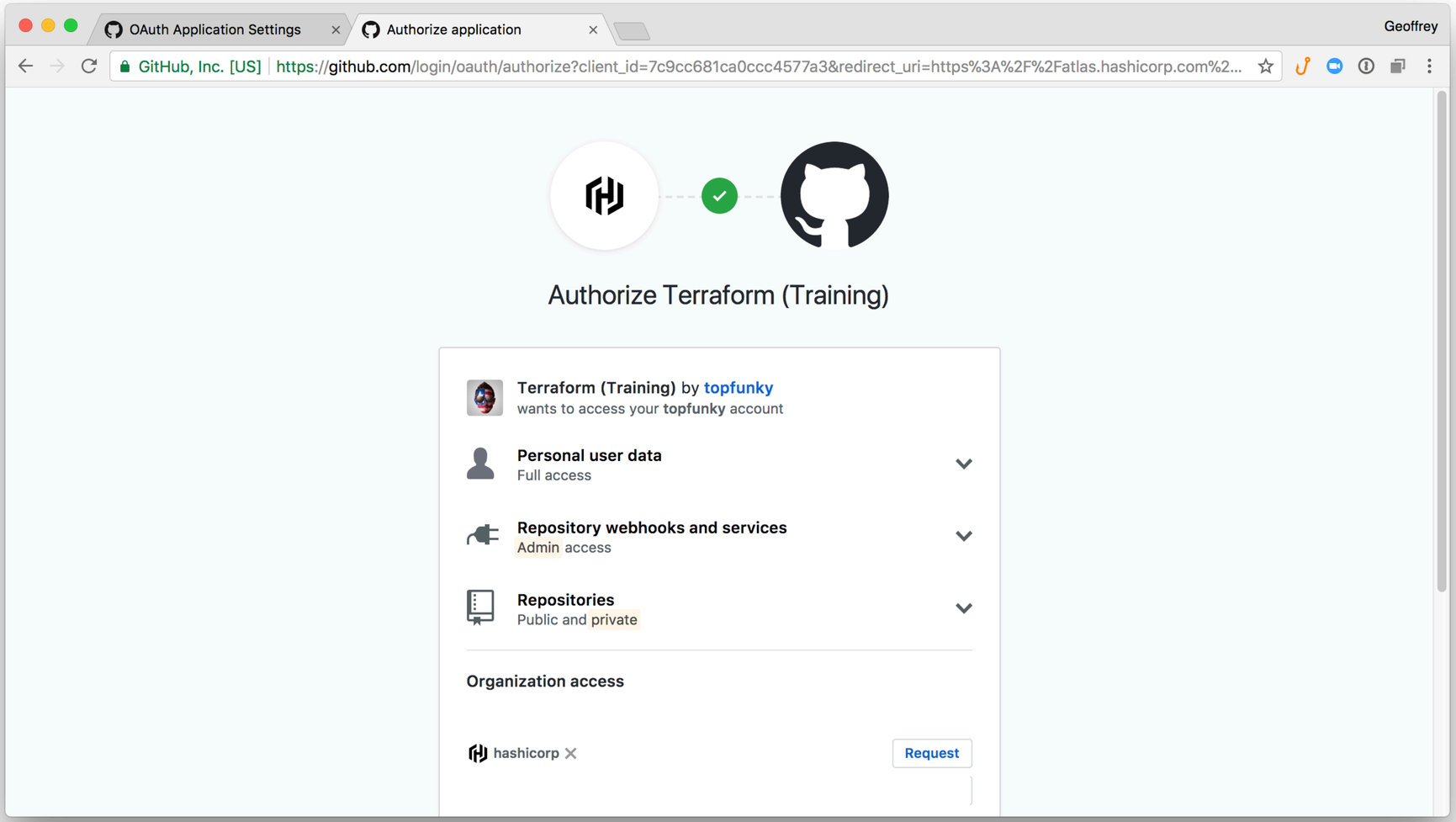

OAuth Application Settings
Terraform Enterprise | topfunk
Geoffrey
GitHub, Inc. [US] | https://github.com/settings/applications/635159
Search GitHub
Pull requests
Issues
Marketplace
Explore
Application created successfully
Settings / Developer settings
OAuth Apps
GitHub Apps
Personal access tokens
Terraform (Training)
topfunky owns this application.
Transfer ownership
You can list your application in the GitHub Marketplace so that other users can discover it.
List this application in the Marketplace
0 users
Client ID
7c9cc681ca0ccc4577a3
Client Secret
bf441eea7d2b982719e2aa768e9389b2f8cfda59
Revoke all user tokens
Reset client secret
Application logo

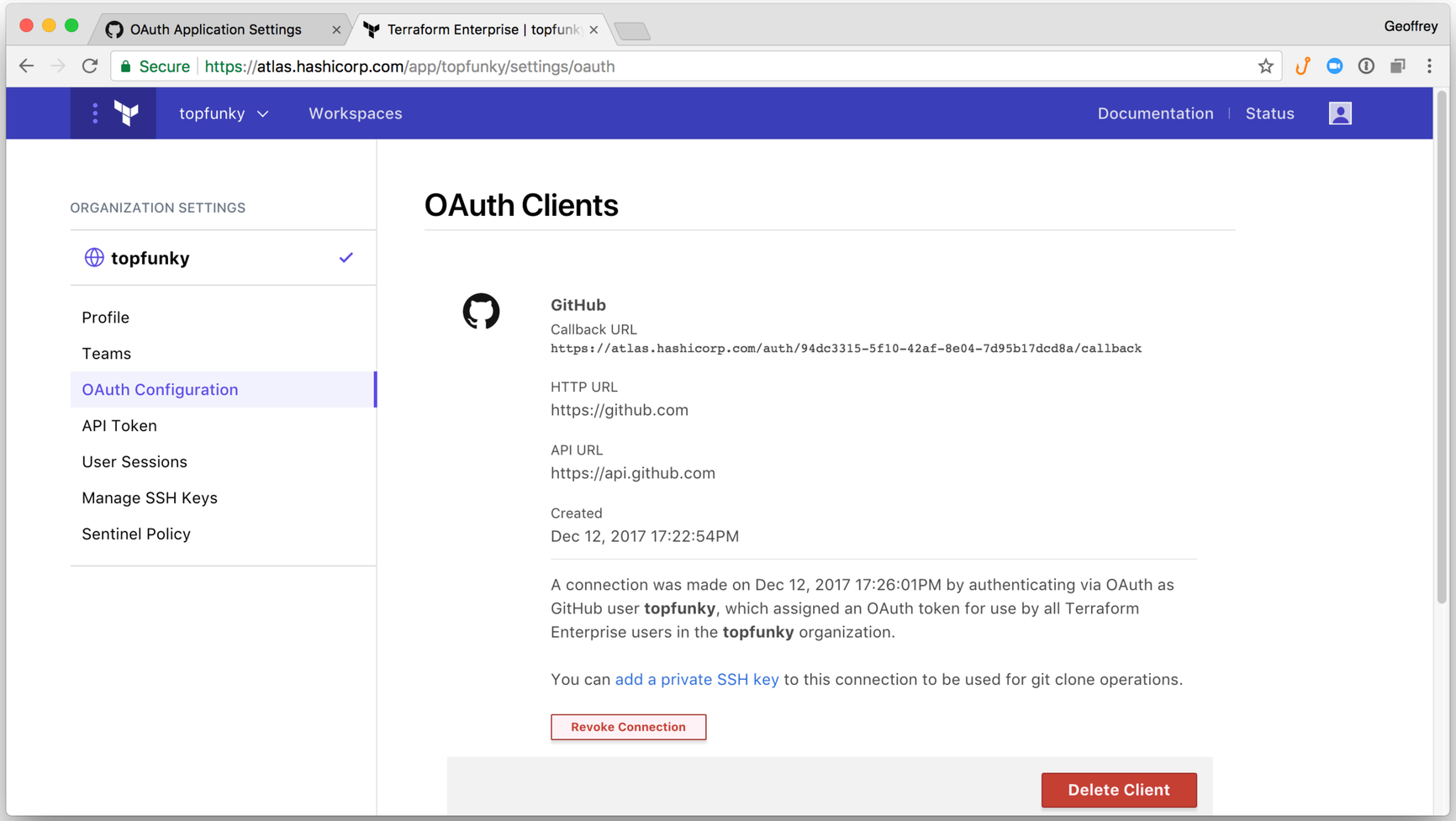

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky/settings/oauth/add
topfunky
Workspaces
Documentation | Status
ORGANIZATION SETTINGS
topfunky
Profile
Teams
OAuth Configuration
API Token
User Sessions
Manage SSH Keys
Sentinel Policy
Add OAuth Client
To connect workspaces to git repositories containing Terraform configurations, Terraform Enterprise needs access to your version control system (VCS) provider. Use this page to configure OAuth authentication with your VCS provider. For more information, please see the Terraform Enterprise documentation on Configuring Version Control Access.
VERSION CONTROL SYSTEM (VCS) PROVIDER
GitHub
HTTP URL
https://github.com
API URL
https://api.github.com
CLIENT ID
7c9cc681ca0ccc4577a3
Register a new OAuth application on GitHub.com to generate the Client ID and Client Secret.
CLIENT SECRET
bf441eea7d2b982719e2aa768e9389b2f8cfda59

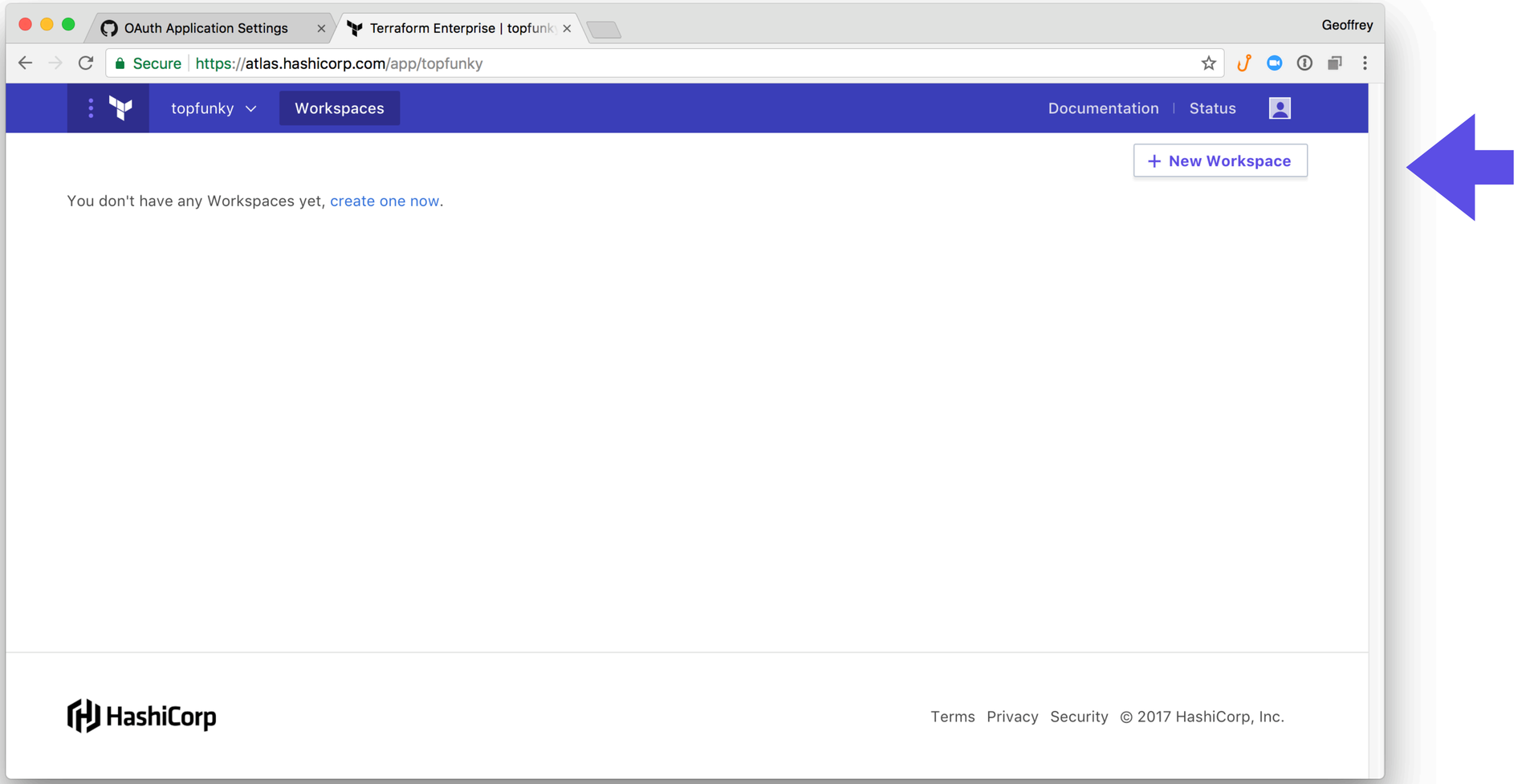

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky/settings/oauth
topfunky
Workspaces
Documentation | Status
ORGANIZATION SETTINGS
topfunky
Profile
Teams
OAuth Configuration
API Token
User Sessions
Manage SSH Keys
Sentinel Policy
OAuth Clients
GitHub
Callback URL
https://atlas.hashicorp.com/auth/94dc3315-5f10-42af-8e04-7d95b17dcd8a/callback
HTTP URL
https://github.com
API URL
https://api.github.com
Created
Dec 12, 2017 17:22:54PM
Connect to GitHub
Connecting to GitHub will take your GitHub user through the OAuth flow to create an authorization token for access to all repositories for this organization. This means that your currently logged in GitHub user token will be used for all GitHub API interactions by any Terraform Enterprise user anywhere within the scope of topfunky.
Connect organization topfunky
Delete Client

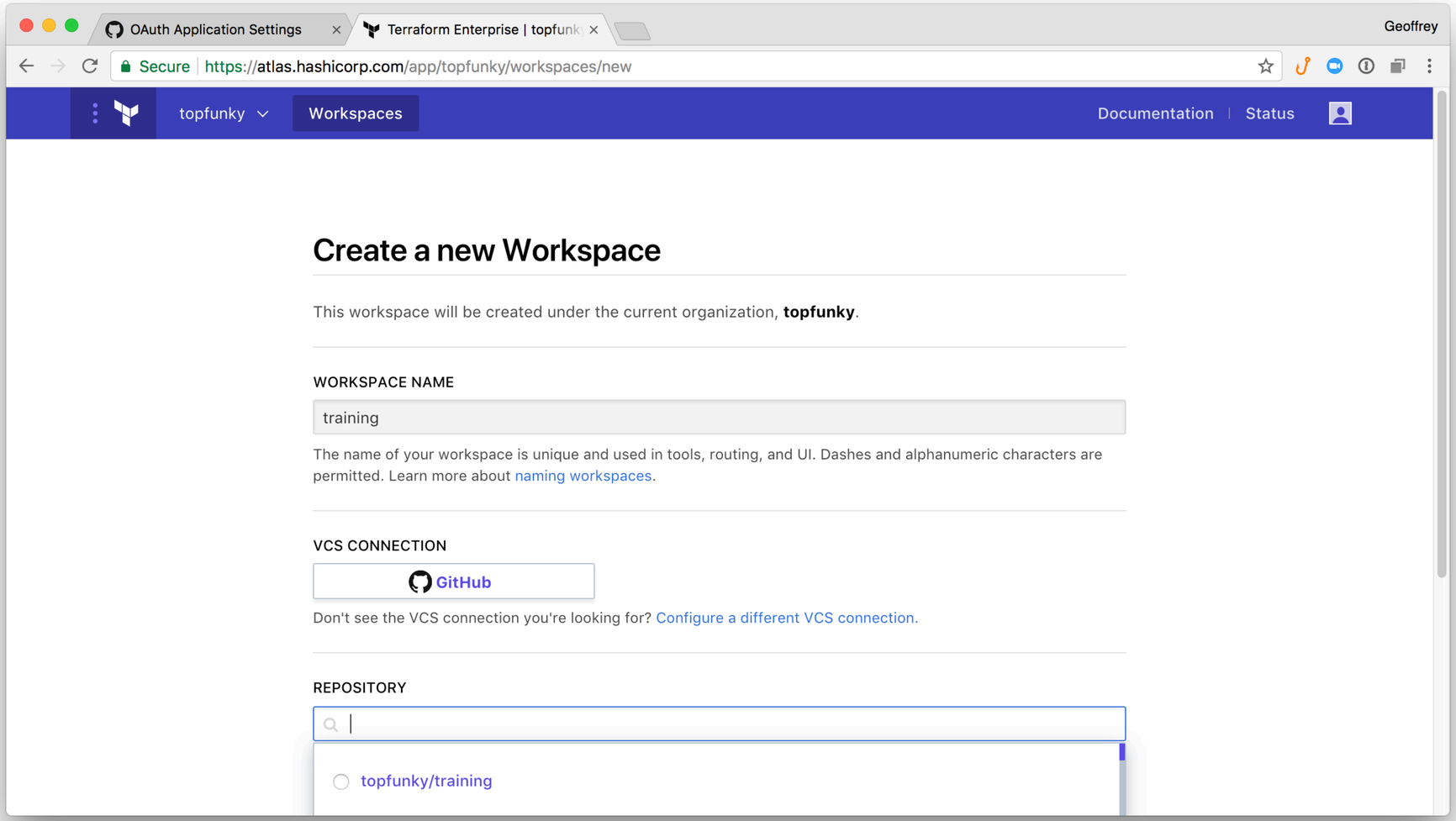

OAuth Application Settings
Terraform Enterprise | topfunk
Geoffrey
GitHub, Inc. [US] | https://github.com/settings/applications/635159
You can also drag and drop a picture from your computer.
Badge background color
#ffffff
The hex value of the badge background color.
Application name
Terraform (Training)
Something users will recognize and trust
Homepage URL
https://atlas.hashicorp.com
The full URL to your application homepage
Application description
Training application
This is displayed to all users of your application
Authorization callback URL
https://atlas.hashicorp.com/auth/94dc3315-5f10-42af-8e04-7d
Your application's callback URL. Read our OAuth documentation for more information.

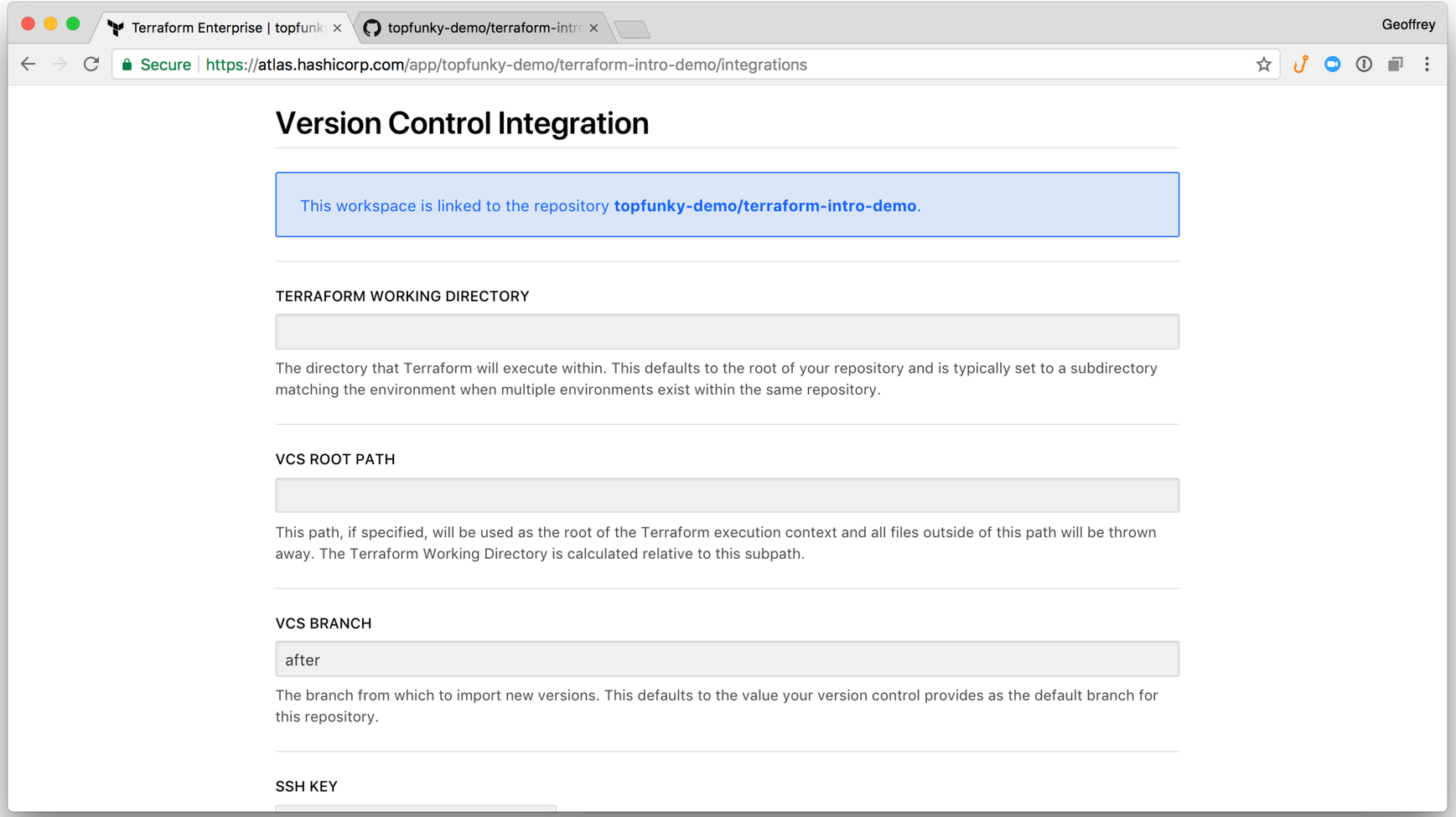

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky/settings/oauth
topfunky
Workspaces
Documentation | Status
ORGANIZATION SETTINGS
topfunky
Profile
Teams
OAuth Configuration
API Token
User Sessions
Manage SSH Keys
Sentinel Policy
OAuth Clients
GitHub
Callback URL
https://atlas.hashicorp.com/auth/94dc3315-5f10-42af-8e04-7d95b17dcd8a/callback
HTTP URL
https://github.com
API URL
https://api.github.com
Created
Dec 12, 2017 17:22:54PM
Connect to GitHub
Connecting to GitHub will take your GitHub user through the OAuth flow to create an authorization token for access to all repositories for this organization. This means that your currently logged in GitHub user token will be used for all GitHub API interactions by any Terraform Enterprise user anywhere within the scope of topfunky.
Connect organization topfunky
Delete Client


OAuth Application Settings
Authorize application
Geoffrey
GitHub, Inc. [US] | https://github.com/login/oauth/authorize?client_id=7c9cc681ca0ccc4577a3&redirect_uri=https%3A%2F%2Fatlas.hashicorp.com%2...
Authorize Terraform (Training)
Terraform (Training) by topfunky
wants to access your topfunky account
Personal user data
Full access
Repository webhooks and services
Admin access
Repositories
Public and private
Organization access
hashicorp
Request

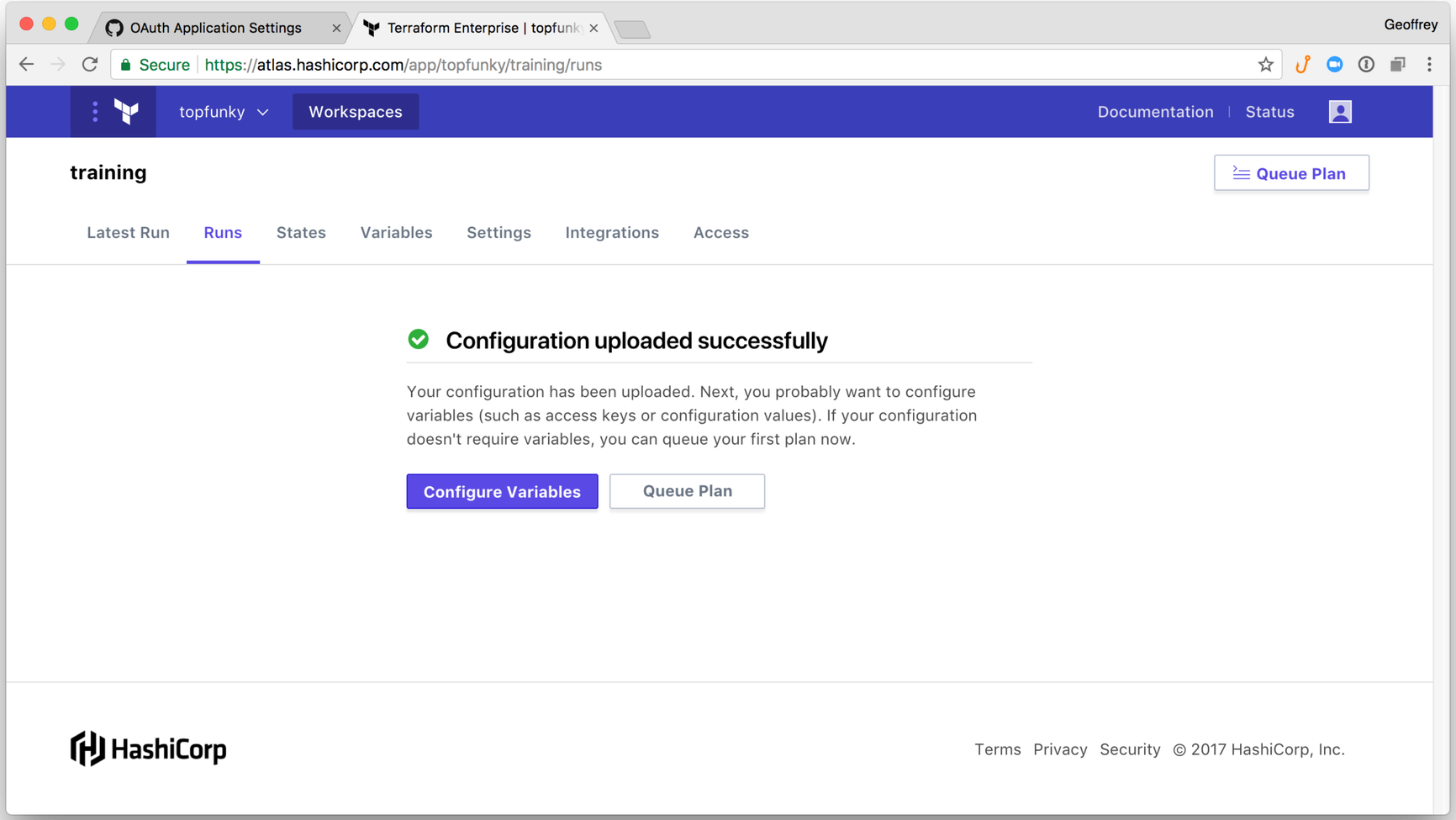

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky/settings/oauth
topfunky
Workspaces
Documentation | Status
ORGANIZATION SETTINGS
topfunky
Profile
Teams
OAuth Configuration
API Token
User Sessions
Manage SSH Keys
Sentinel Policy
OAuth Clients
GitHub
Callback URL
https://atlas.hashicorp.com/auth/94dc3315-5f10-42af-8e04-7d95b17dcd8a/callback
HTTP URL
https://github.com
API URL
https://api.github.com
Created
Dec 12, 2017 17:22:54PM
A connection was made on Dec 12, 2017 17:26:01PM by authenticating via OAuth as GitHub user topfunky, which assigned an OAuth token for use by all Terraform Enterprise users in the topfunky organization.
You can add a private SSH key to this connection to be used for git clone operations.
Revoke Connection
Delete Client

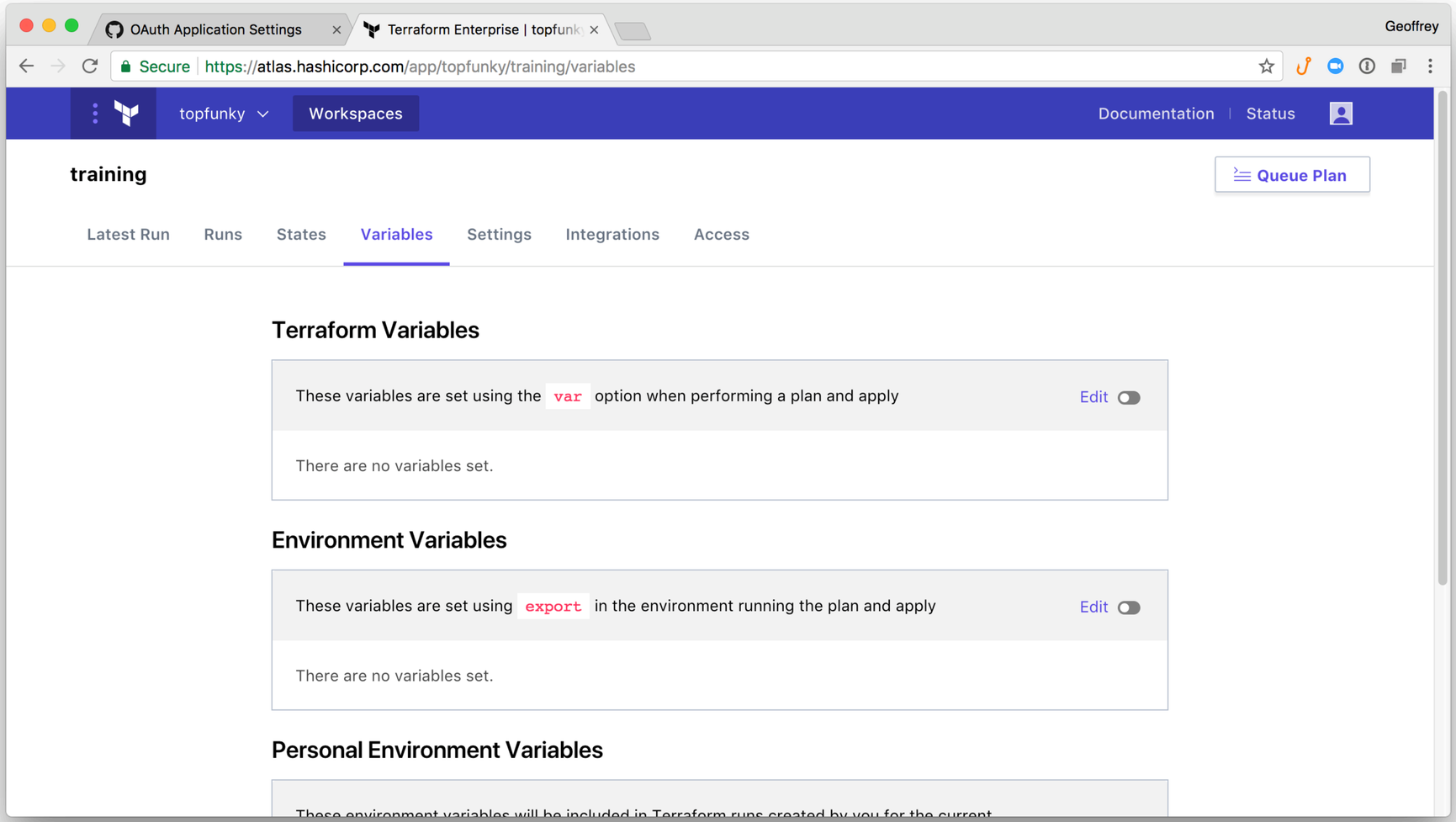

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky
topfunky
Workspaces
Documentation | Status
+ New Workspace
You don't have any Workspaces yet, create one now.
HashiCorp
Terms Privacy Security © 2017 HashiCorp, Inc.

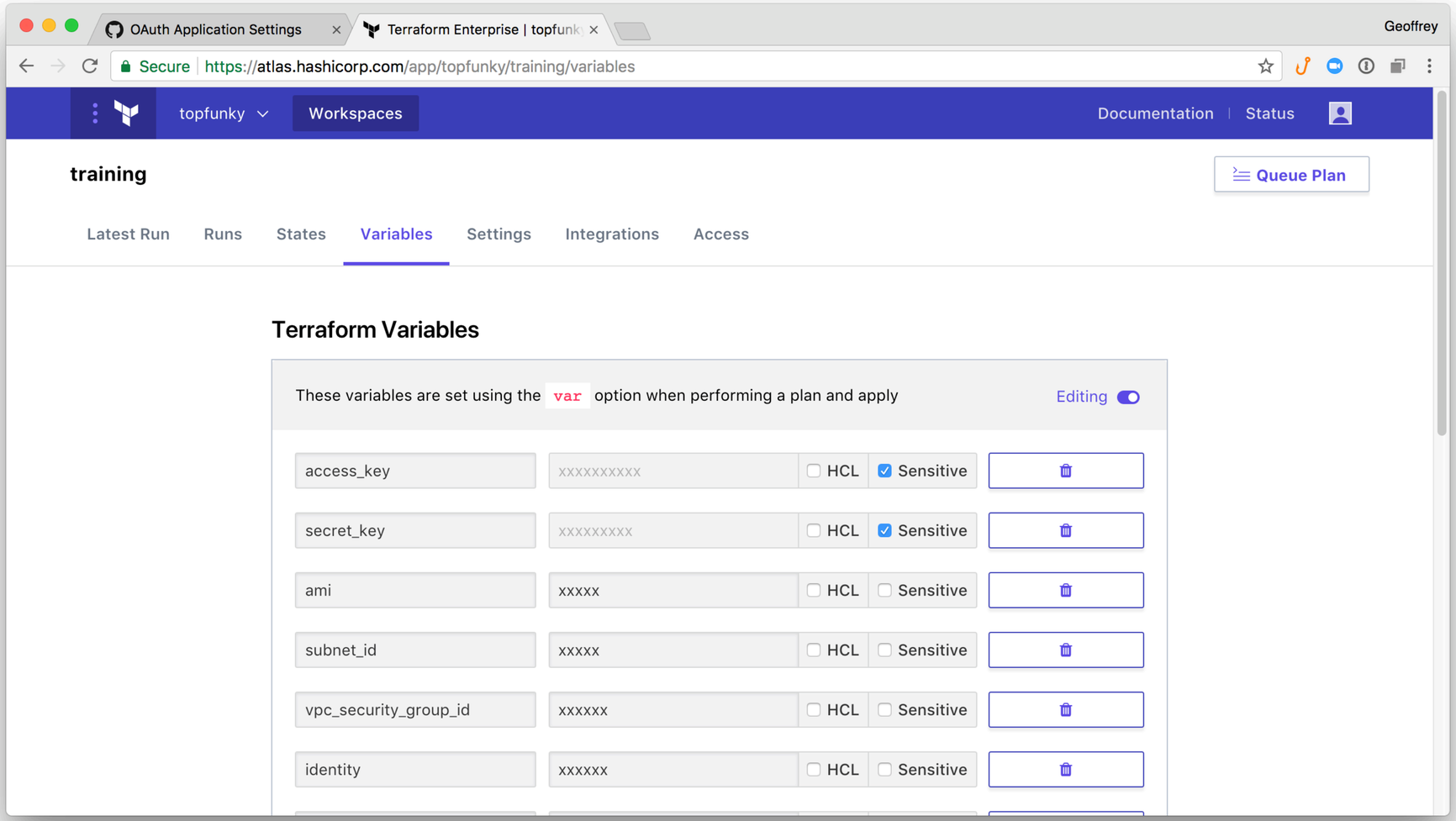

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky/workspaces/new
topfunky
Workspaces
Documentation
Status
Create a new Workspace
This workspace will be created under the current organization, topfunky.
WORKSPACE NAME
training
The name of your workspace is unique and used in tools, routing, and UI. Dashes and alphanumeric characters are permitted. Learn more about naming workspaces.
VCS CONNECTION
GitHub
Don't see the VCS connection you're looking for? Configure a different VCS connection.
REPOSITORY
topfunky/training

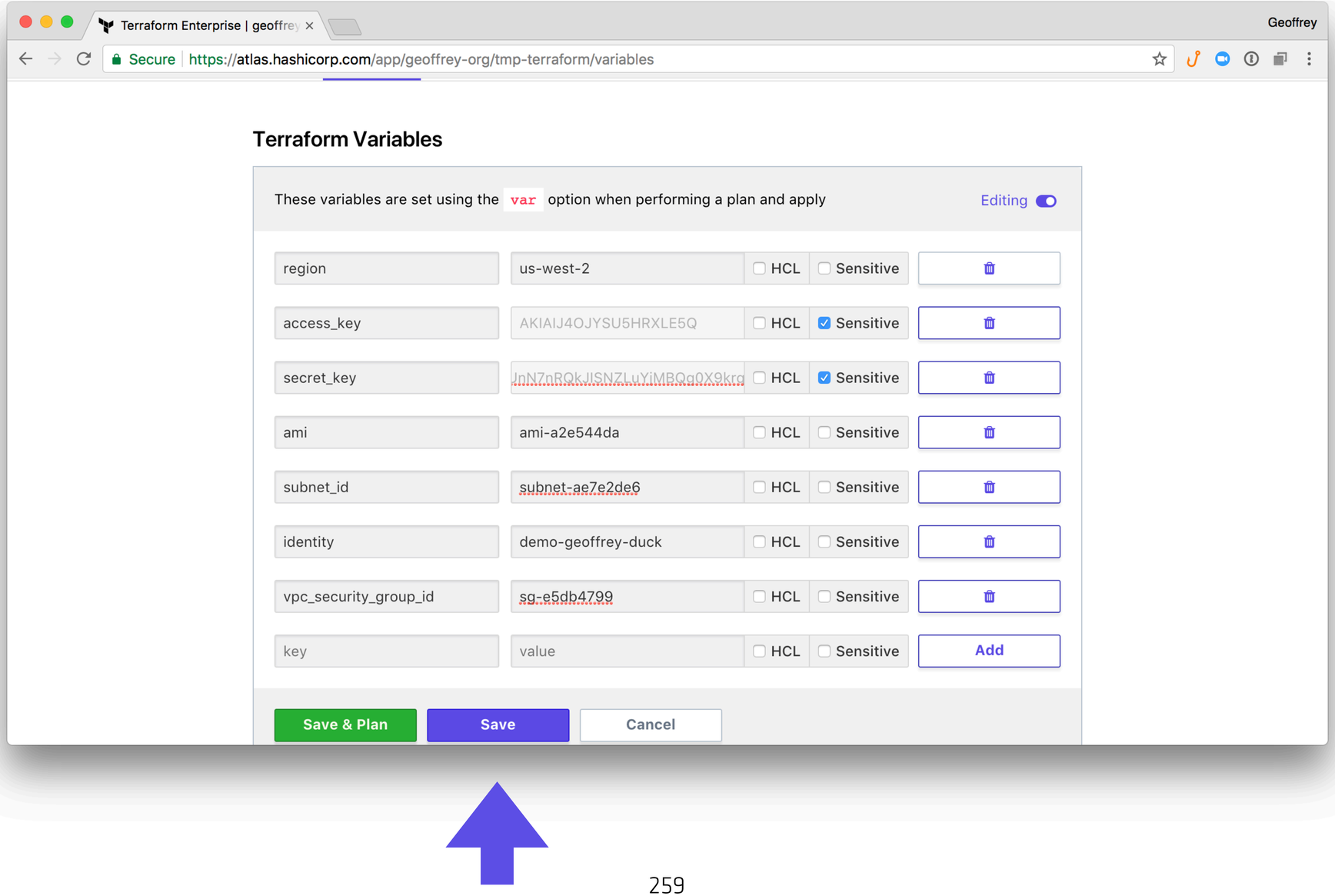

Terraform Enterprise | topfunk
topfunky-demo/terraform-intro
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/integrations
Version Control Integration
This workspace is linked to the repository topfunky-demo/terraform-intro-demo.
TERRAFORM WORKING DIRECTORY
The directory that Terraform will execute within. This defaults to the root of your repository and is typically set to a subdirectory matching the environment when multiple environments exist within the same repository.
VCS ROOT PATH
This path, if specified, will be used as the root of the Terraform execution context and all files outside of this path will be thrown away. The Terraform Working Directory is calculated relative to this subpath.
VCS BRANCH
after
The branch from which to import new versions. This defaults to the value your version control provides as the default branch for this repository.
SSH KEY

HashiCorp

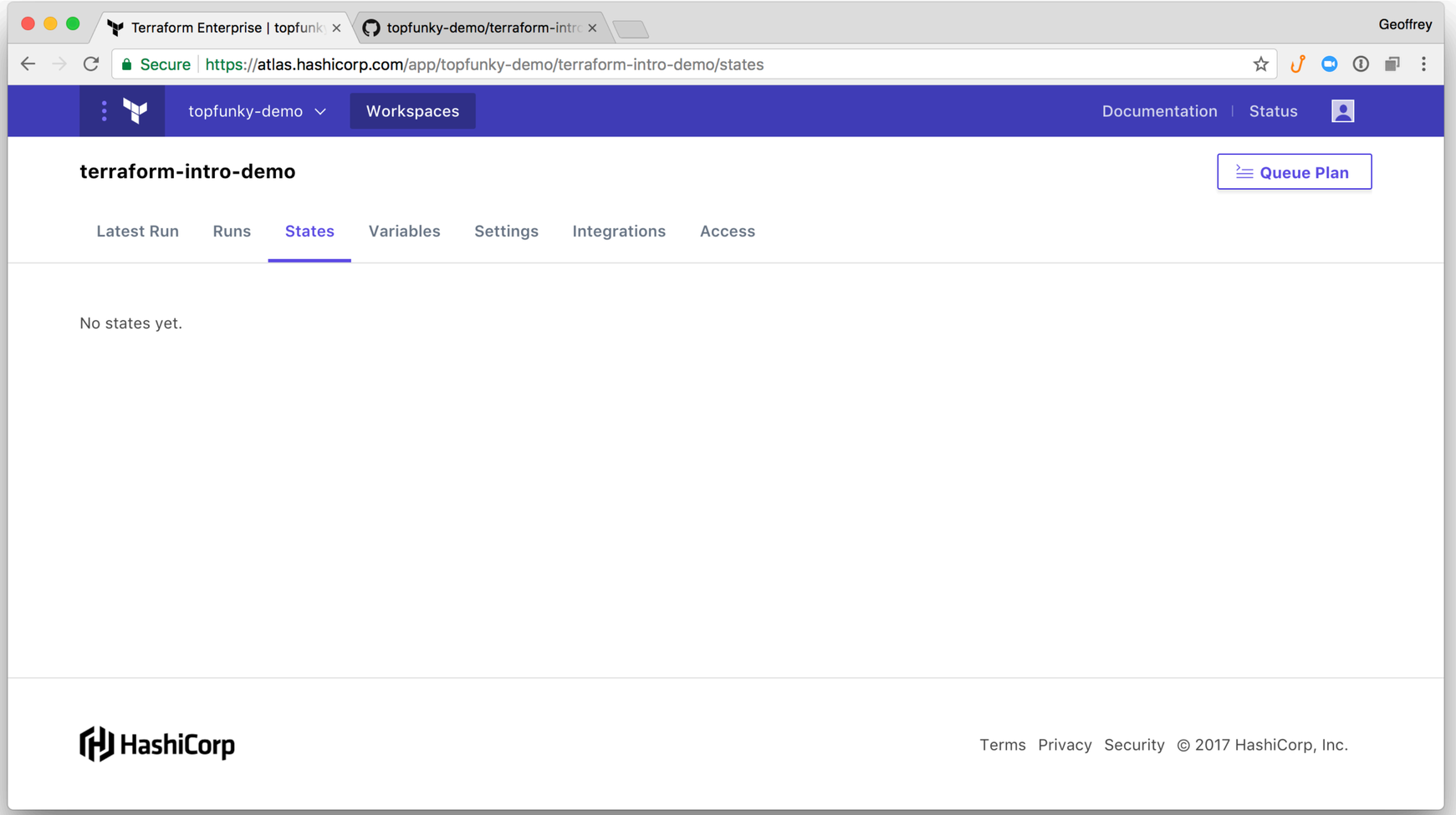

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure https://atlas.hashicorp.com/app/topfunky/training/runs
topfunky
Workspaces
Documentation
Status
training
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
Configuration uploaded successfully
Your configuration has been uploaded. Next, you probably want to configure variables (such as access keys or configuration values). If your configuration doesn't require variables, you can queue your first plan now.
Configure Variables
Queue Plan
HashiCorp
Terms Privacy Security © 2017 HashiCorp, Inc.


OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky/training/variables
topfunky
Workspaces
Documentation
Status
training
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
Terraform Variables
These variables are set using the var option when performing a plan and apply
Edit
There are no variables set.
Environment Variables
These variables are set using export in the environment running the plan and apply
Edit
There are no variables set.
Personal Environment Variables

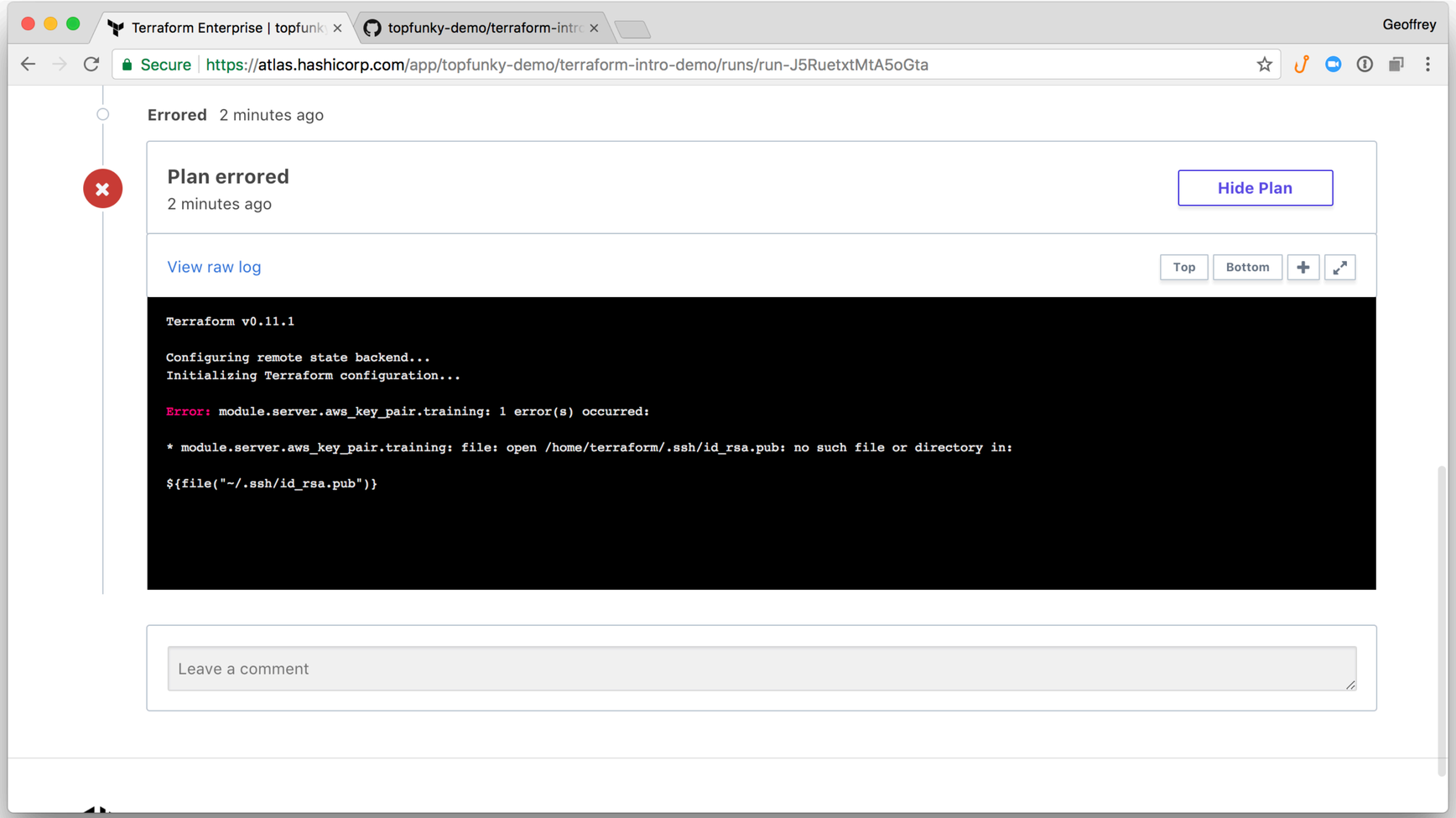

OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky/training/variables
topfunky
Workspaces
Documentation | Status
training
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
Terraform Variables
These variables are set using the var option when performing a plan and apply
Editing
access_key xxxxxxxxxx HCL Sensitive
secret_key xxxxxxxxx HCL Sensitive
ami xxxxx HCL Sensitive
subnet_id xxxxx HCL Sensitive
vpc_security_group_id xxxxxx HCL Sensitive
identity xxxxxx HCL Sensitive


Terraform Enterprise | geoffrey
Geoffrey
Secure
https://atlas.hashicorp.com/app/geoffrey-org/tmp-terraform/variables
Terraform Variables
These variables are set using the var option when performing a plan and apply
Editing
region
us-west-2
HCL
Sensitive
access_key
AKIAIJ4OJYSU5HRXLE5Q
HCL
Sensitive
secret_key
JnN7nRQkJISNZLuYiMBQq0X9krq
HCL
Sensitive
ami
ami-a2e544da
HCL
Sensitive
subnet_id
subnet-ae7e2de6
HCL
Sensitive
identity
demo-geoffrey-duck
HCL
Sensitive
vpc_security_group_id
sg-e5db4799
HCL
Sensitive
key
value
HCL
Sensitive
Add
Save & Plan
Save
Cancel

HashiCorp


Terraform Enterprise | topfunk
topfunky-demo/terraform-intro
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/states
topfunky-demo
Workspaces
Documentation
Status
terraform-intro-demo
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
No states yet.
HashiCorp
Terms
Privacy
Security
© 2017 HashiCorp, Inc.

HashiCorp


Terraform Enterprise | topfunk
topfunky-demo/terraform-intro
Geoffrey
Secure https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/runs/run-J5RuetxtMtA5oGta
Errored 2 minutes ago
Plan errored
2 minutes ago
Hide Plan
View raw log
Top
Bottom
Terraform v0.11.1
Configuring remote state backend...
Initializing Terraform configuration...
Error: module.server.aws_key_pair.training: 1 error(s) occurred:
* module.server.aws_key_pair.training: file: open /home/terraform/.ssh/id_rsa.pub: no such file or directory in:
${file("~/.ssh/id_rsa.pub")}
Leave a comment

HashiCorp

TIP

Remove/Repair References to the Local Filesystem

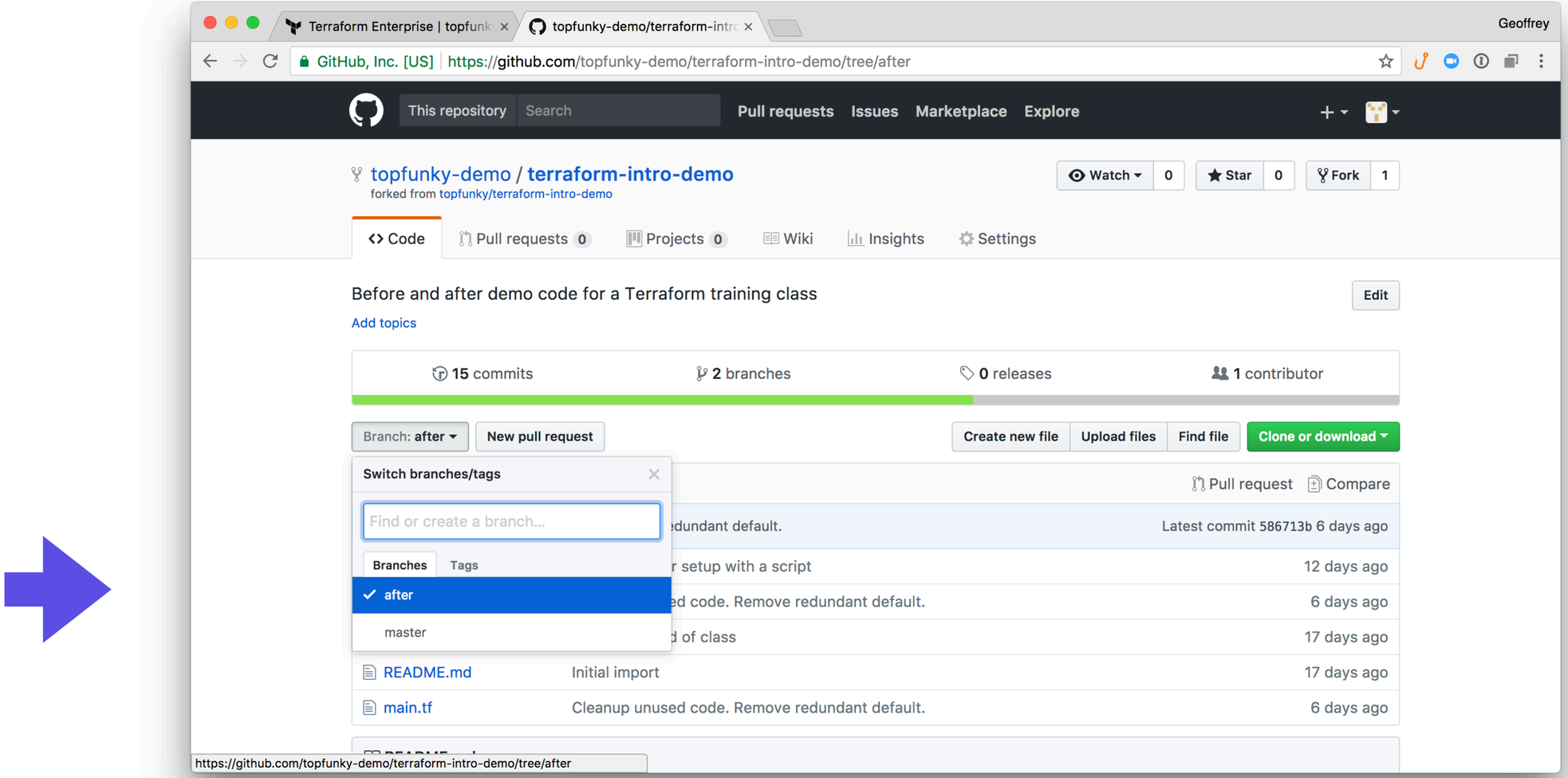
Unlike a CI platform, Terraform Cloud doesn't allow you to execute shell scripts before or after a run.

You're limited to the exact contents of your Git repository plus the variables you've set up.

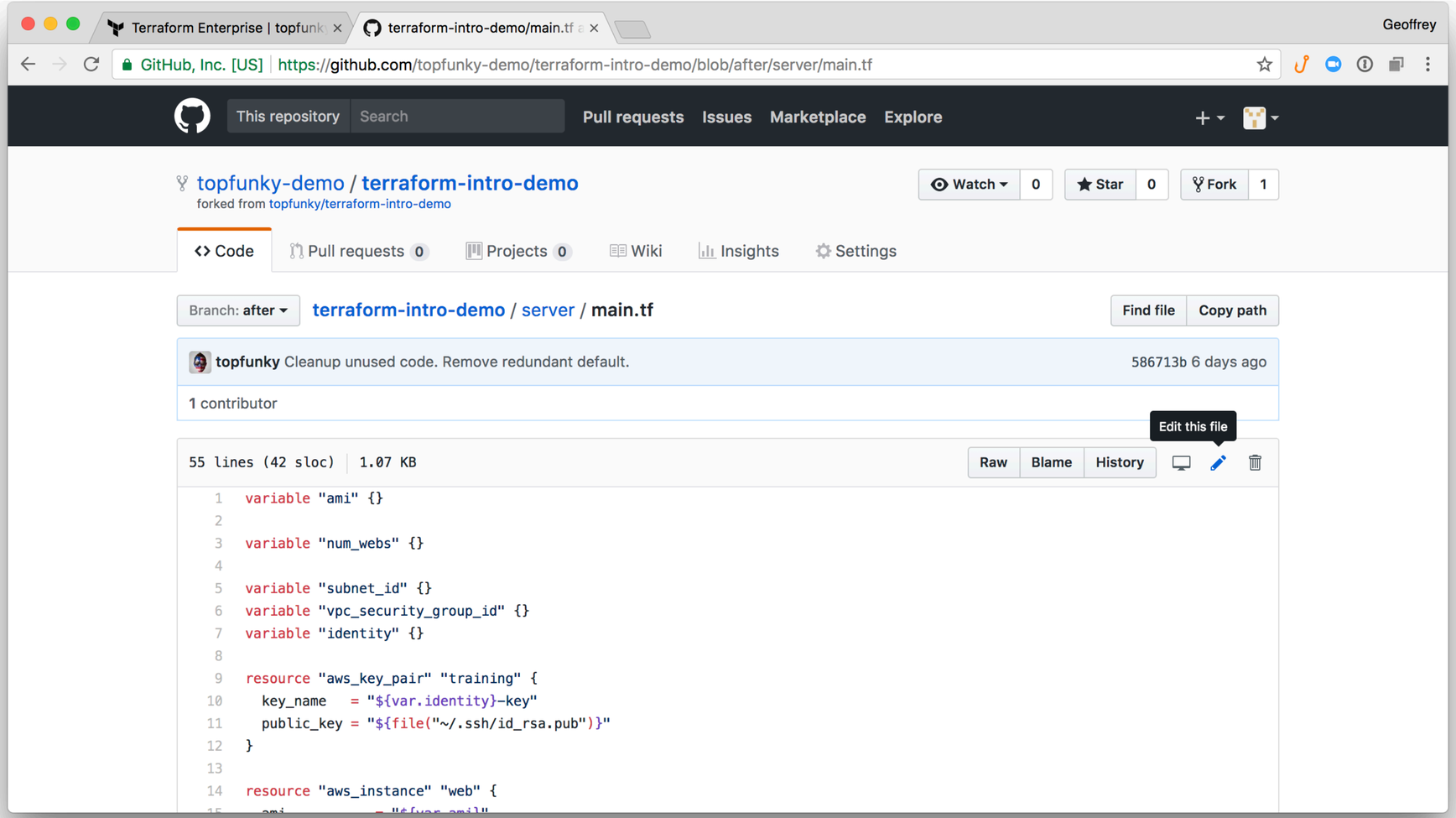
The SSH keys we used were part of our instance but aren't part of the Terraform environment.

Instead of reading files, use content from variables.

HashiCorp

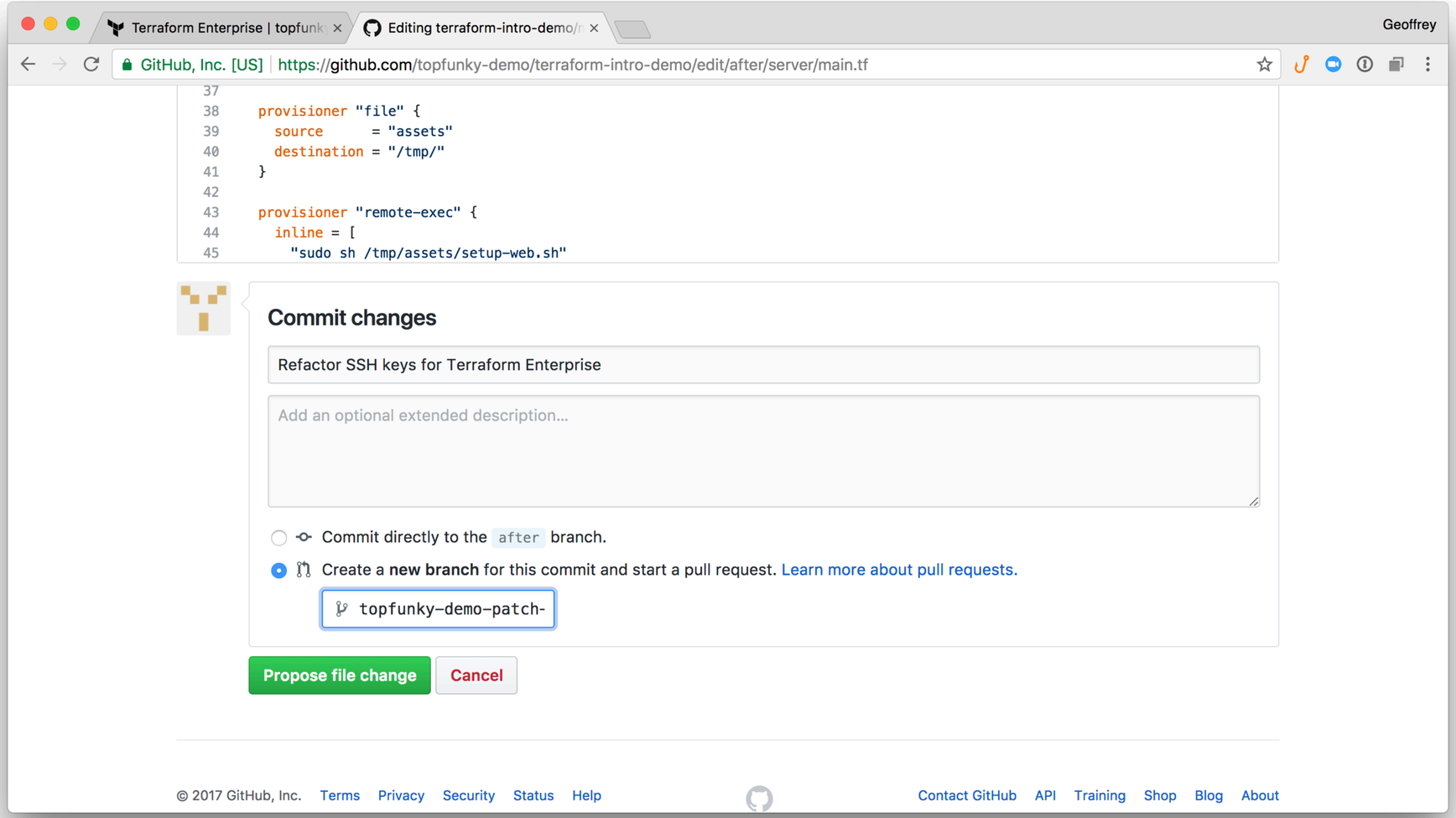

Terraform Enterprise | topfunk
topfunky-demo/terraform-intro
Geoffrey
GitHub, Inc. [US] https://github.com/topfunky-demo/terraform-intro-demo/tree/after
This repository
Search
Pull requests
Issues
Marketplace
Explore
topfunky-demo / terraform-intro-demo
forked from topfunky/terraform-intro-demo
Watch 0
Star 0
Fork 1
Code
Pull requests 0
Projects 0
Wiki
Insights
Settings
Before and after demo code for a Terraform training class
Edit
Add topics
15 commits
2 branches
0 releases
1 contributor
Branch: after
New pull request
Create new file
Upload files
Find file
Clone or download
Switch branches/tags
Find or create a branch...
Branches
Tags
after
master
Pull request
Compare
Latest commit 586713b 6 days ago
12 days ago
6 days ago
17 days ago
README.md
Initial import
17 days ago
main.tf
Cleanup unused code. Remove redundant default.
6 days ago
https://github.com/topfunky-demo/terraform-intro-demo/tree/after

HashiCorp

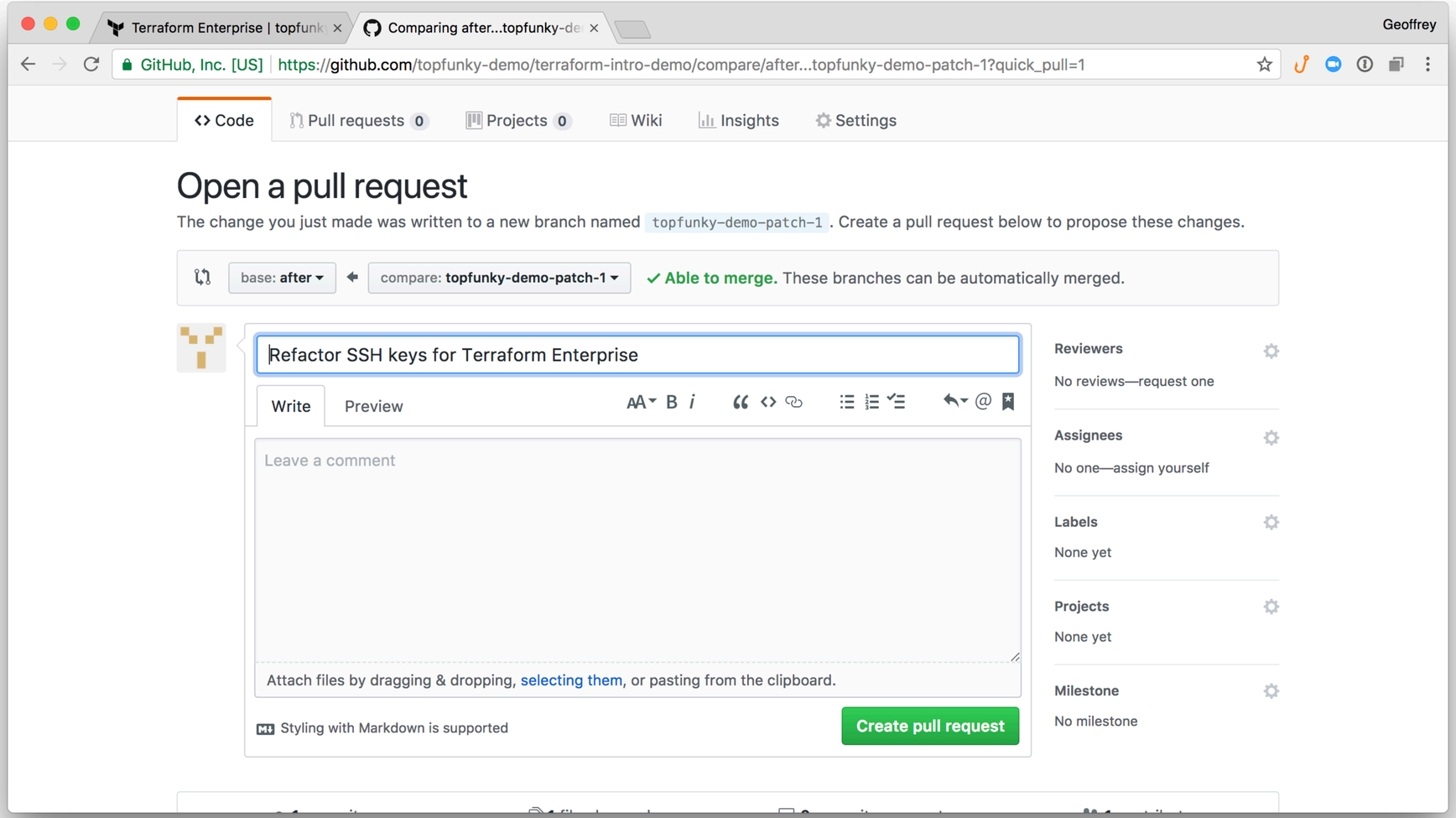

Terraform Enterprise | topfunk
terraform-intro-demo/main.tf
Geoffrey
GitHub, Inc. [US] https://github.com/topfunky-demo/terraform-intro-demo/blob/after/server/main.tf
This repository
Search
Pull requests
Issues
Marketplace
Explore
topfunky-demo / terraform-intro-demo
forked from topfunky/terraform-intro-demo
Watch 0
Star 0
Fork 1
Code
Pull requests 0
Projects 0
Wiki
Insights
Settings
Branch: after
terraform-intro-demo / server / main.tf
Find file
Copy path
topfunky Cleanup unused code. Remove redundant default.
586713b 6 days ago
1 contributor
Edit this file
55 lines (42 sloc) 1.07 KB
Raw
Blame
History
1 variable "ami" {}
2
3 variable "num_webs" {}
4
5 variable "subnet_id" {}
6 variable "vpc_security_group_id" {}
7 variable "identity" {}
8
9 resource "aws_key_pair" "training" {
10 key_name = "${var.identity}-key"
11 public_key = "${file("~/.ssh/id_rsa.pub")}"
12 }
13
14 resource "aws_instance" "web" {

HashiCorp

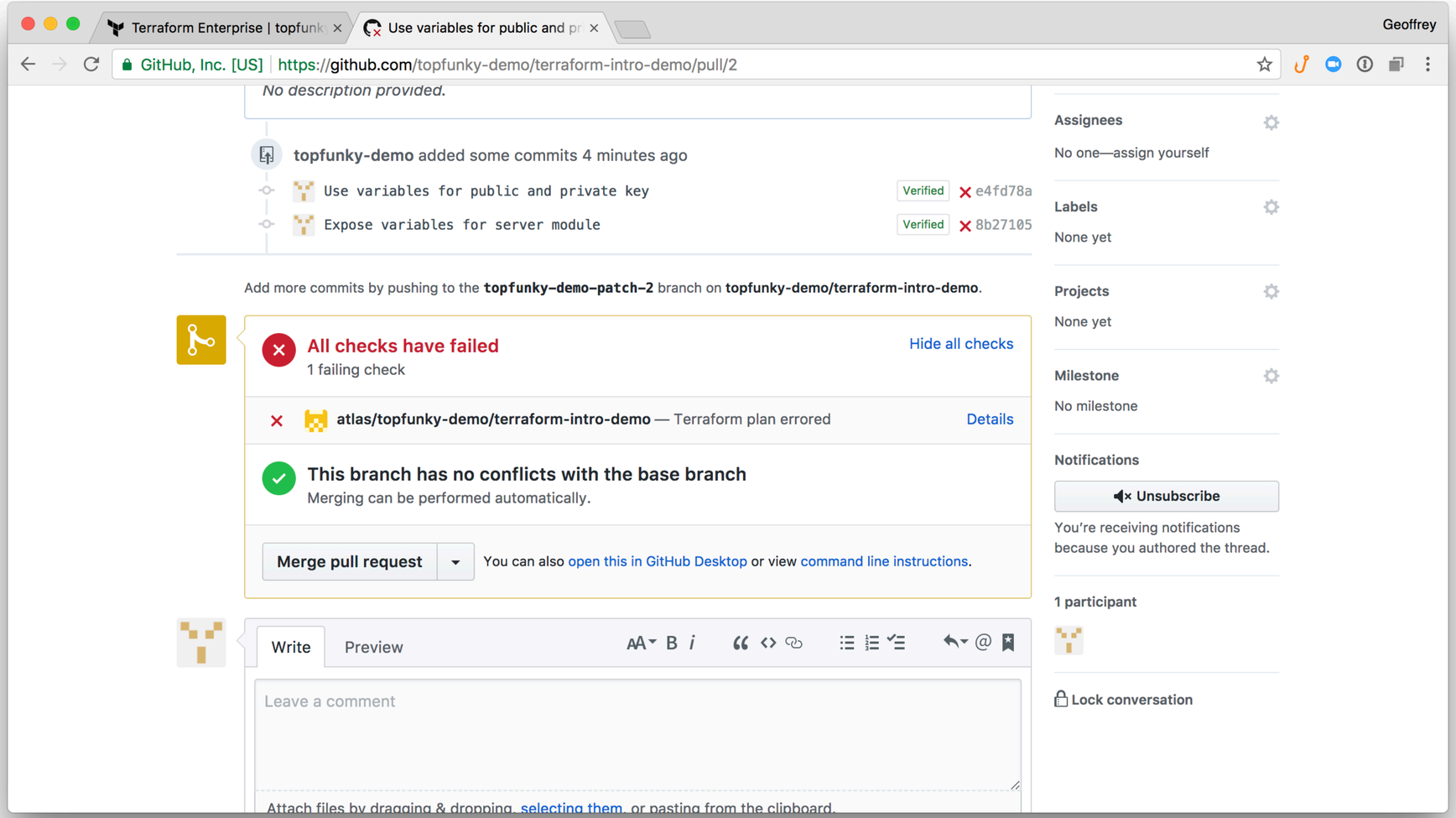
```
variable "public_key" {}
variable "private_key" {}

resource "aws_key_pair" "training" {
  key_name   = "${var.identity}-key"
  public_key = var.public_key
}

resource "aws_instance" "web" {
  ami           = var.ami
  # . . .

  connection {
    user        = "ubuntu"
    private_key = var.private_key
  }
}
```


Terraform Enterprise | topfunk
Editing terraform-intro-demo/r
Geoffrey
GitHub, Inc. [US] https://github.com/topfunky-demo/terraform-intro-demo/edit/after/server/main.tf
37
38 provisioner "file" {
39 source = "assets"
40 destination = "/tmp/"
41 }
42
43 provisioner "remote-exec" {
44 inline = [
45 "sudo sh /tmp/assets/setup-web.sh"
Commit changes
Refactor SSH keys for Terraform Enterprise
Add an optional extended description...
Commit directly to the after branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
topfunky-demo-patch-
Propose file change
Cancel
© 2017 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub API Training Shop Blog About

HashiCorp


Terraform Enterprise | topfunk
Comparing after...topfunky-de
Geoffrey
GitHub, Inc. [US] https://github.com/topfunky-demo/terraform-intro-demo/compare/after...topfunky-demo-patch-1?quick_pull=1
Code
Pull requests 0
Projects 0
Wiki
Insights
Settings
Open a pull request
The change you just made was written to a new branch named topfunky-demo-patch-1. Create a pull request below to propose these changes.
base: after
compare: topfunky-demo-patch-1
Able to merge. These branches can be automatically merged.
Refactor SSH keys for Terraform Enterprise
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Create pull request
Reviewers
No reviews—request one
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone

HashiCorp


Terraform Enterprise | topfunk
Use variables for public and pr
Geoffrey
GitHub, Inc. [US] | https://github.com/topfunky-demo/terraform-intro-demo/pull/2
No description provided.
topfunky-demo added some commits 4 minutes ago
Use variables for public and private key
Verified
e4fd78a
Expose variables for server module
Verified
8b27105
Add more commits by pushing to the topfunky-demo-patch-2 branch on topfunky-demo/terraform-intro-demo.
All checks have failed
1 failing check
Hide all checks
atlas/topfunky-demo/terraform-intro-demo — Terraform plan errored
Details
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Notifications
Unsubscribe
You're receiving notifications because you authored the thread.
1 participant
Lock conversation

HashiCorp

WARNING

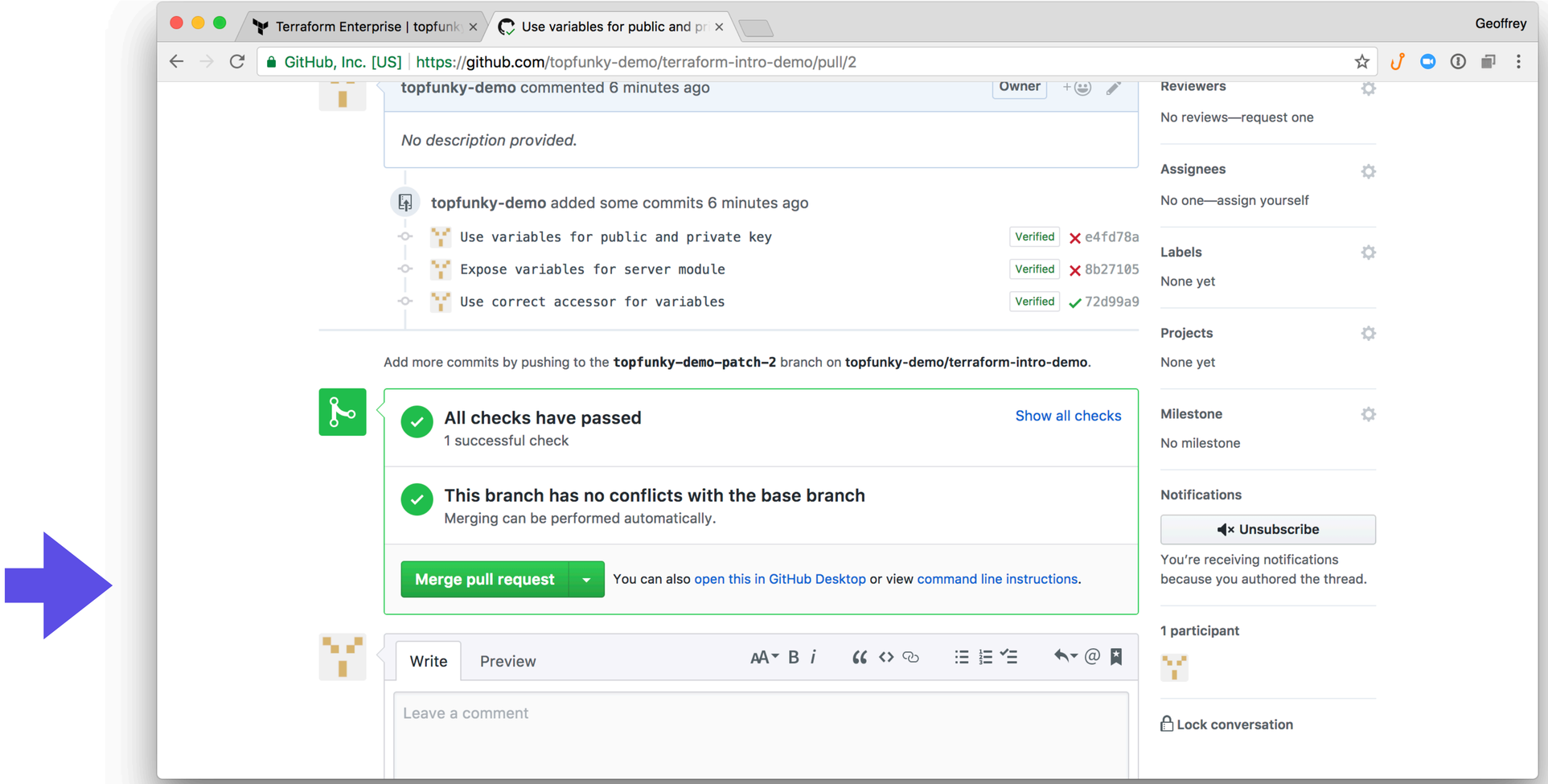
Pull Request Plan Runs Will Not Appear in TFC

When Terraform Cloud runs a plan against a pull request, the only way to see the result is through the link in the pull request at GitHub.

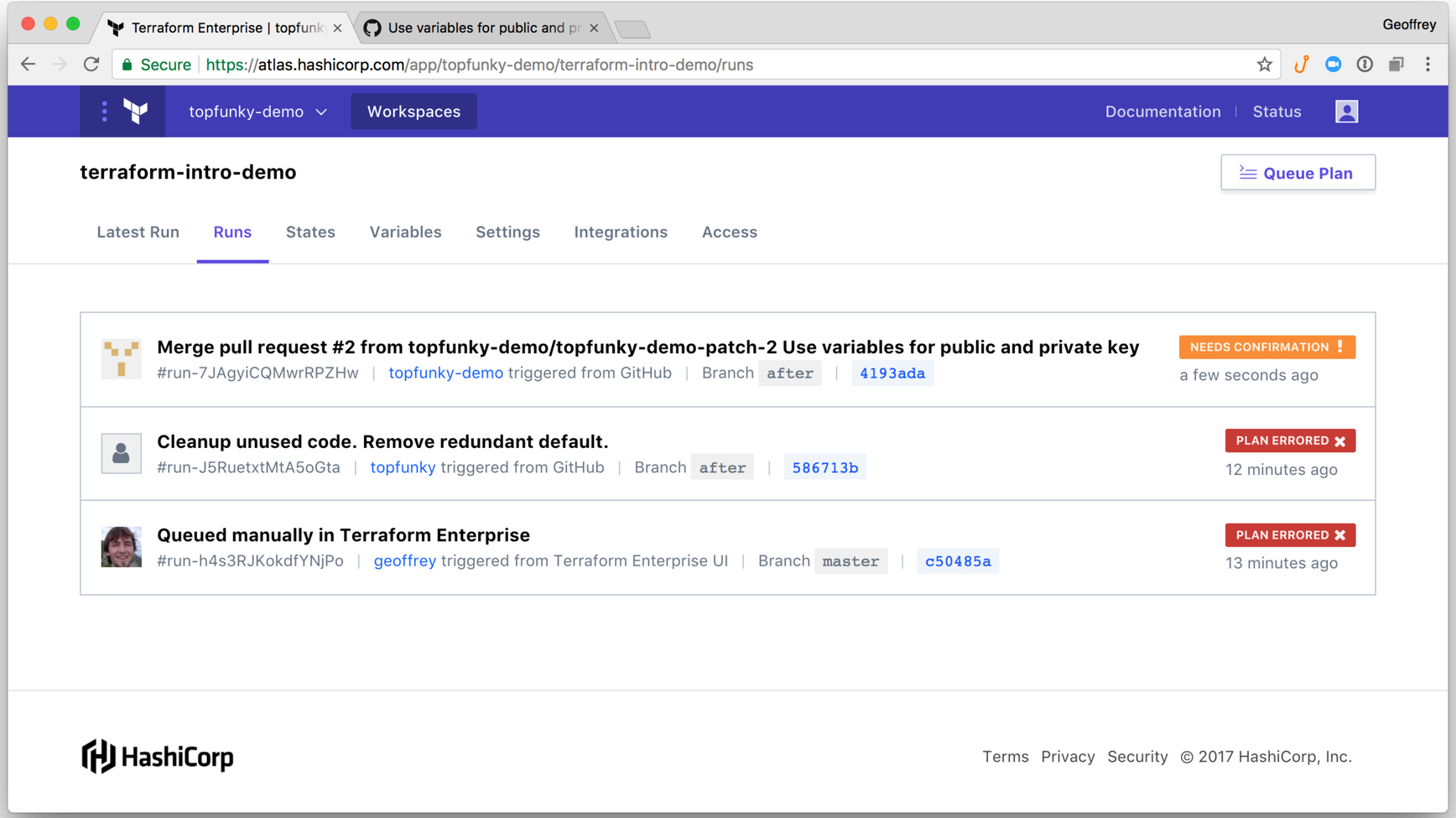
The plan is ephemeral and can not be navigated to in the Terraform Cloud UI.

HashiCorp

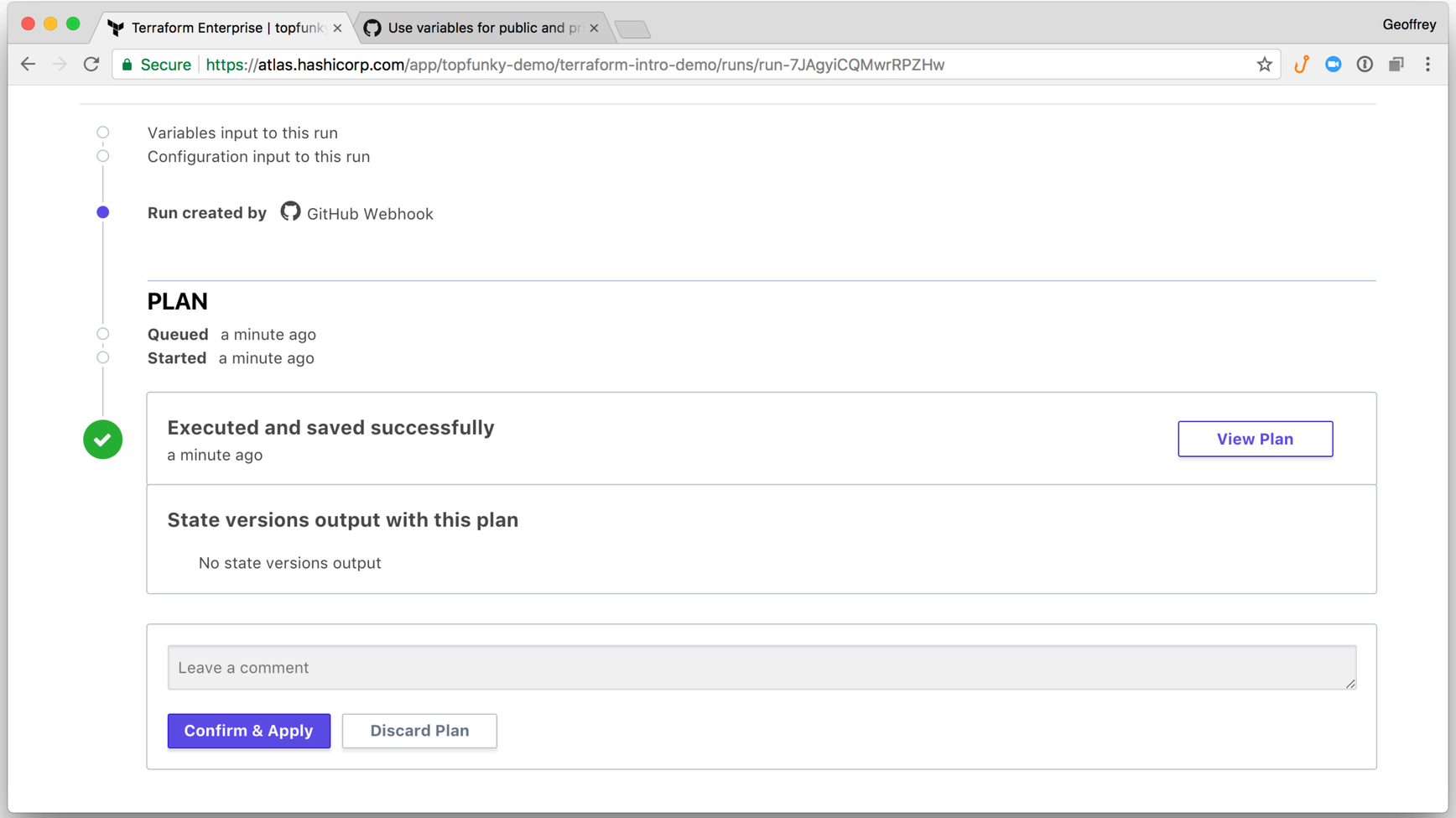
```
variable "public_key" {}
variable "private_key" {}

module "server" {
  source = "./server"

  num_webs                = var.num_webs
  ami                     = var.ami
  subnet_id               = var.subnet_id
  vpc_security_group_id = var.vpc_security_group_id
  identity                = var.identity
  public_key              = var.public_key
  private_key             = var.private_key
}
```


Terraform Enterprise | topfunk
Use variables for public and pr
Geoffrey
GitHub, Inc. [US] | https://github.com/topfunky-demo/terraform-intro-demo/pull/2
topfunky-demo commented 6 minutes ago
Owner
No description provided.
topfunky-demo added some commits 6 minutes ago
Use variables for public and private key
Verified
e4fd78a
Expose variables for server module
Verified
8b27105
Use correct accessor for variables
Verified
72d99a9
Add more commits by pushing to the topfunky-demo-patch-2 branch on topfunky-demo/terraform-intro-demo.
All checks have passed
1 successful check
Show all checks
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
Write
Preview
Leave a comment
Reviewers
No reviews—request one
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Notifications
Unsubscribe
You're receiving notifications because you authored the thread.
1 participant
Lock conversation

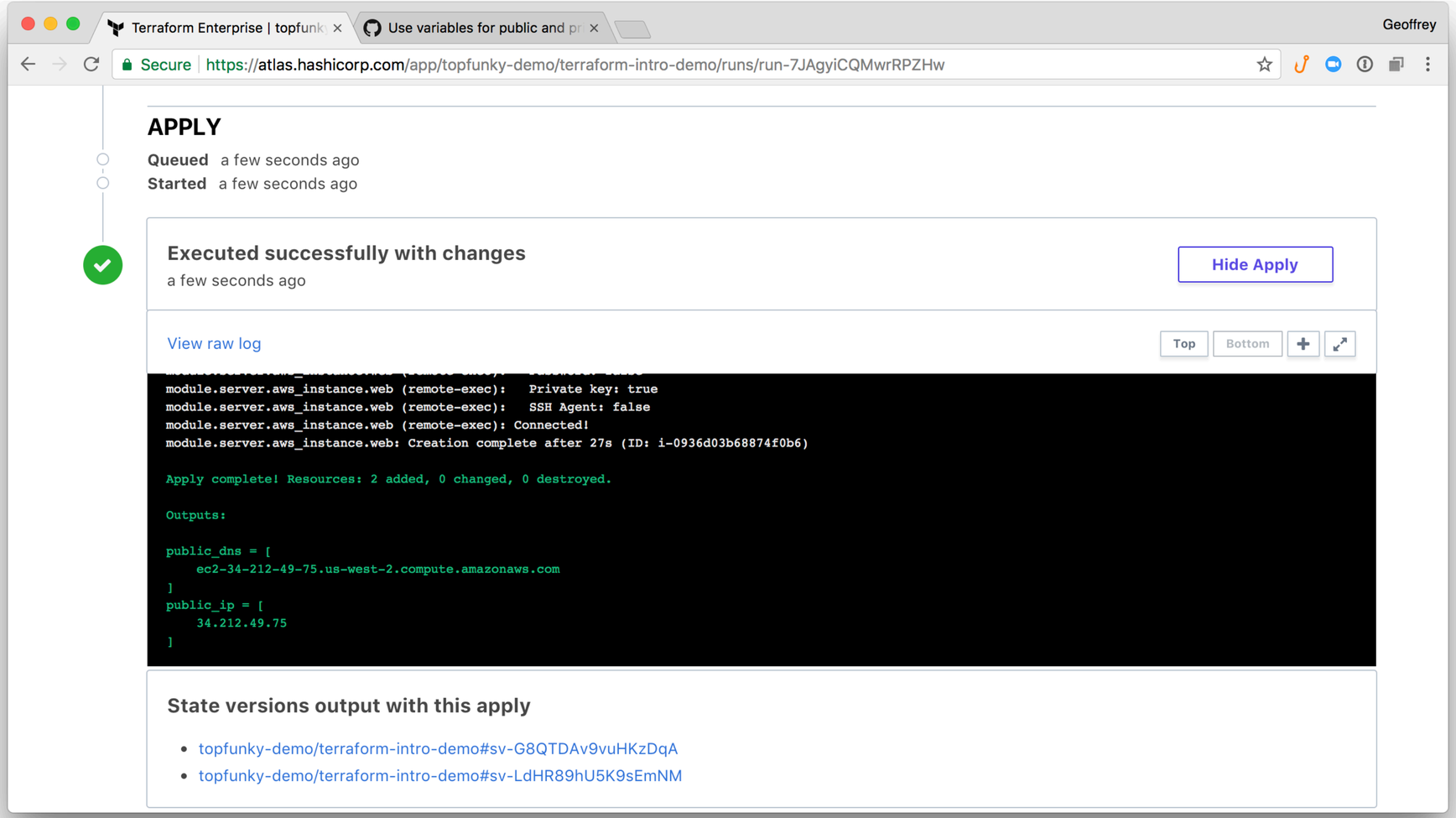
HashiCorp


Terraform Enterprise | topfunk
Use variables for public and pr
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/runs
topfunky-demo
Workspaces
Documentation | Status
terraform-intro-demo
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
Merge pull request #2 from topfunky-demo/topfunky-demo-patch-2 Use variables for public and private key
#run-7JAgyiCQMwrRPZHw | topfunky-demo triggered from GitHub | Branch after | 4193ada
NEEDS CONFIRMATION
a few seconds ago
Cleanup unused code. Remove redundant default.
#run-J5RuetxtMtA5oGta | topfunky triggered from GitHub | Branch after | 586713b
PLAN ERRORED
12 minutes ago
Queued manually in Terraform Enterprise
#run-h4s3RJKokdfYNjPo | geoffrey triggered from Terraform Enterprise UI | Branch master | c50485a
PLAN ERRORED
13 minutes ago
HashiCorp
Terms Privacy Security © 2017 HashiCorp, Inc.

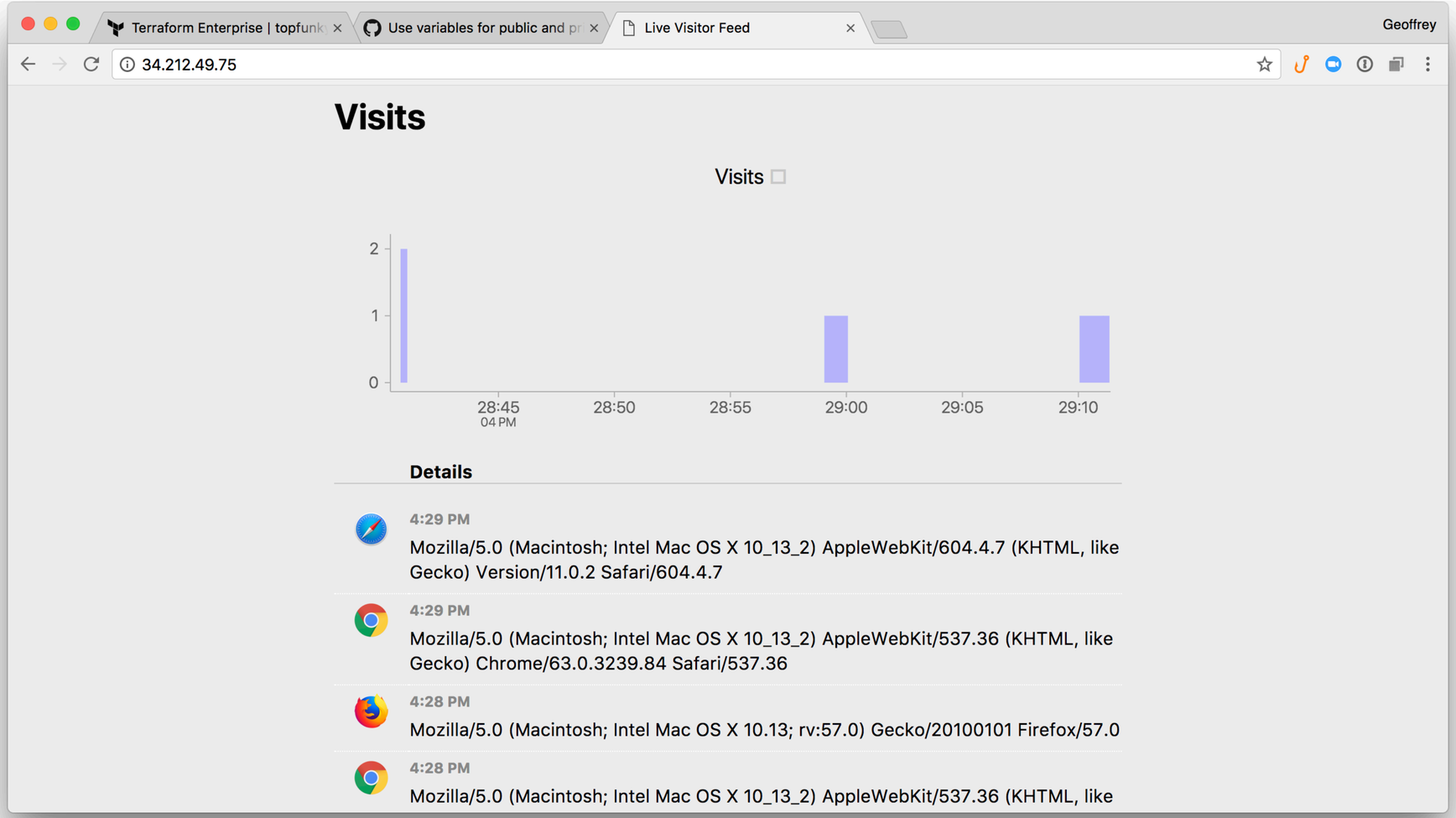
HashiCorp


Terraform Enterprise | topfunk
Use variables for public and pr
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/runs/run-7JAgyiCQMwrRPZHw
Variables input to this run
Configuration input to this run
Run created by
GitHub Webhook
PLAN
Queued a minute ago
Started a minute ago
Executed and saved successfully
a minute ago
View Plan
State versions output with this plan
No state versions output
Leave a comment
Confirm & Apply
Discard Plan

HashiCorp


Terraform Enterprise | topfunk
Use variables for public and pr
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/runs/run-7JAgyiCQMwrRPZHw
APPLY
Queued a few seconds ago
Started a few seconds ago
Executed successfully with changes
a few seconds ago
Hide Apply
View raw log
Top
Bottom
module.server.aws_instance.web (remote-exec): Private key: true
module.server.aws_instance.web (remote-exec): SSH Agent: false
module.server.aws_instance.web (remote-exec): Connected!
module.server.aws_instance.web: Creation complete after 27s (ID: i-0936d03b68874f0b6)
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
Outputs:
public_dns = [
ec2-34-212-49-75.us-west-2.compute.amazonaws.com
]
public_ip = [
34.212.49.75
]
State versions output with this apply
topfunky-demo/terraform-intro-demo#sv-G8QTDAv9vuHKzDqA
topfunky-demo/terraform-intro-demo#sv-LdHR89hU5K9sEmNM

HashiCorp


Terraform Enterprise | topfunk
Use variables for public and pr
Live Visitor Feed
Geoffrey
34.212.49.75
Visits
Visits
2
1
0
28:45
04 PM
28:50
28:55
29:00
29:05
29:10
Details
4:29 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/604.4.7 (KHTML, like Gecko) Version/11.0.2 Safari/604.4.7
4:29 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36
4:28 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:57.0) Gecko/20100101 Firefox/57.0
4:28 PM
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/537.36 (KHTML, like

HashiCorp

Instructor Discretion Exercises

If time permits, the instructor may suggest extra exercises to complete using Terraform Cloud.

HashiCorp

Terraform Cloud: Review

Integrates with SCM and application workflows

Provides a dry-run experience for infrastructure changes

Continuous Integration, but for Infrastructure as Code

HashiCorp

What You Learned

- Create a repository at GitHub
- Push your code to GitHub
- Create an organization at Terraform Cloud
- Connect to GitHub via OAuth
- Add a workspace that references a GitHub repo
- Commit changes and create infrastructure

HashiCorp

What You'll Learn

- Destroy infrastructure that Terraform has created

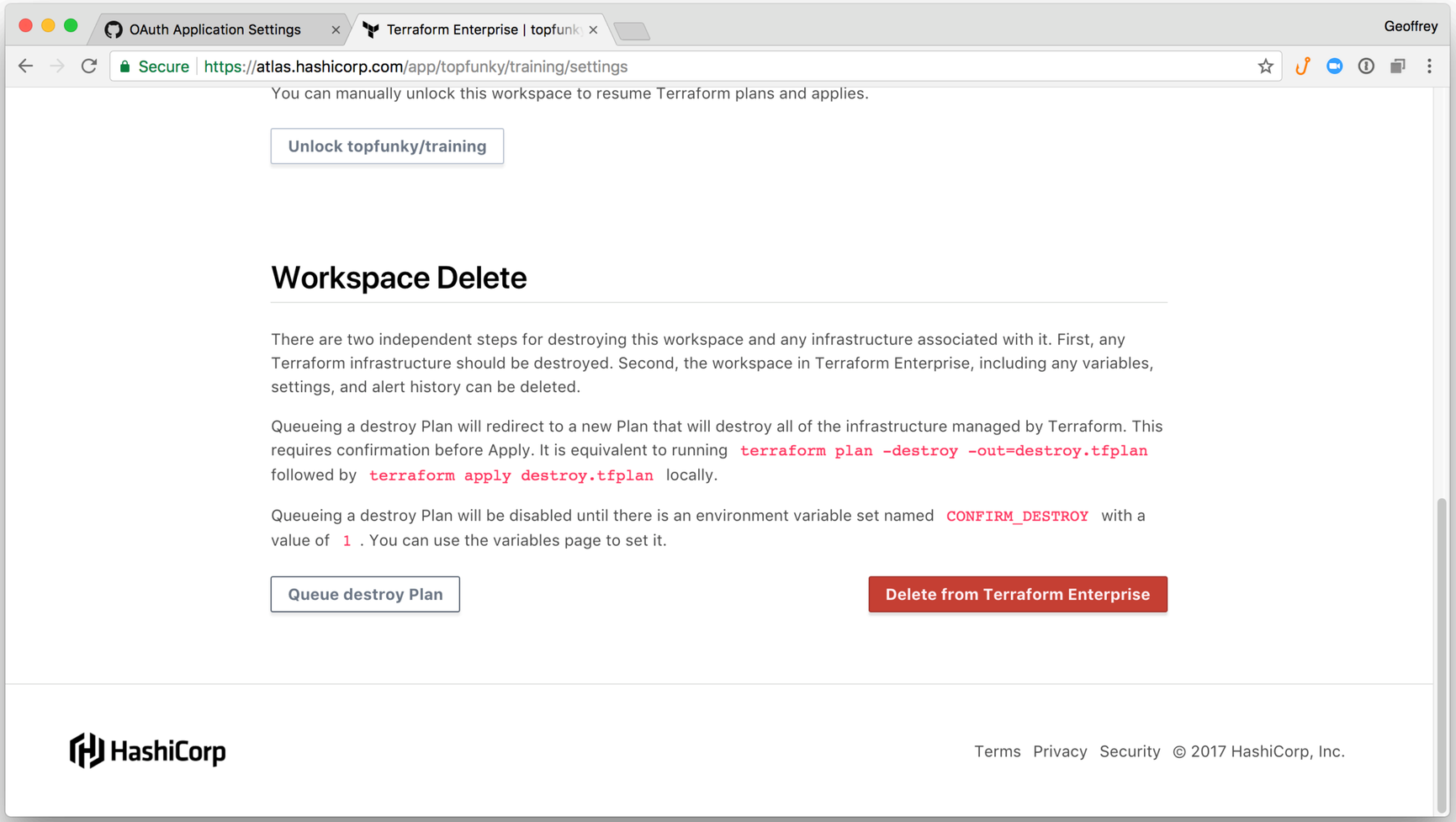
HashiCorp

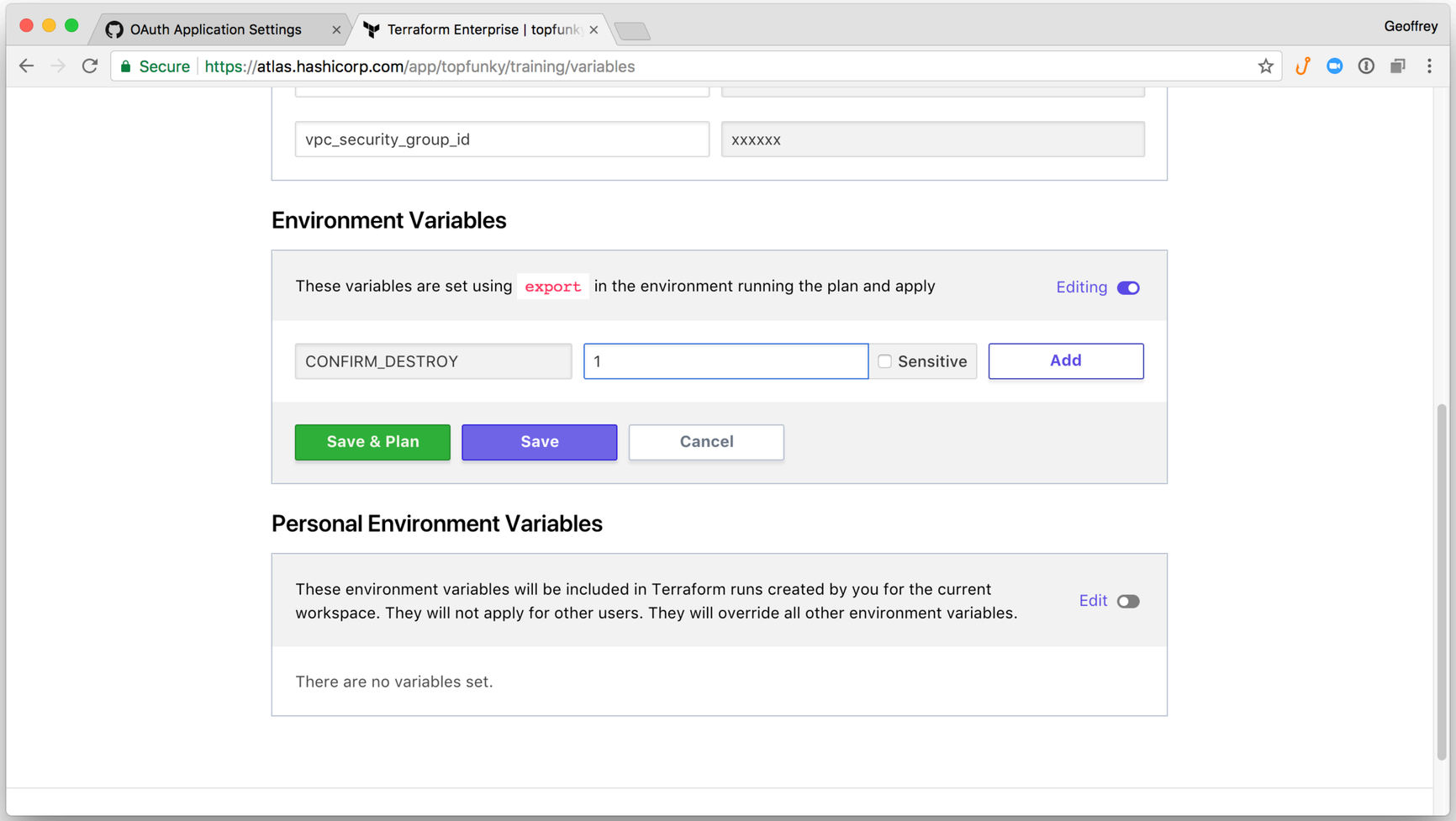
Command: `terraform destroy`

Destroys running infrastructure

Does not touch infrastructure not managed by Terraform

HashiCorp


OAuth Application Settings
Terraform Enterprise | topfunk
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky/training/settings
You can manually unlock this workspace to resume Terraform plans and applies.
Unlock topfunky/training
Workspace Delete
There are two independent steps for destroying this workspace and any infrastructure associated with it. First, any Terraform infrastructure should be destroyed. Second, the workspace in Terraform Enterprise, including any variables, settings, and alert history can be deleted.
Queueing a destroy Plan will redirect to a new Plan that will destroy all of the infrastructure managed by Terraform. This requires confirmation before Apply. It is equivalent to running terraform plan -destroy -out=destroy.tfplan followed by terraform apply destroy.tfplan locally.
Queueing a destroy Plan will be disabled until there is an environment variable set named CONFIRM_DESTROY with a value of 1 . You can use the variables page to set it.
Queue destroy Plan
Delete from Terraform Enterprise
HashiCorp
Terms Privacy Security © 2017 HashiCorp, Inc.


OAuth Application Settings
Terraform Enterprise | topfunk
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky/training/variables
vpc_security_group_id
xxxxxx
Environment Variables
These variables are set using
export
in the environment running the plan and apply
Editing
CONFIRM_DESTROY
1
Sensitive
Add
Save & Plan
Save
Cancel
Personal Environment Variables
These environment variables will be included in Terraform runs created by you for the current workspace. They will not apply for other users. They will override all other environment variables.
Edit
There are no variables set.

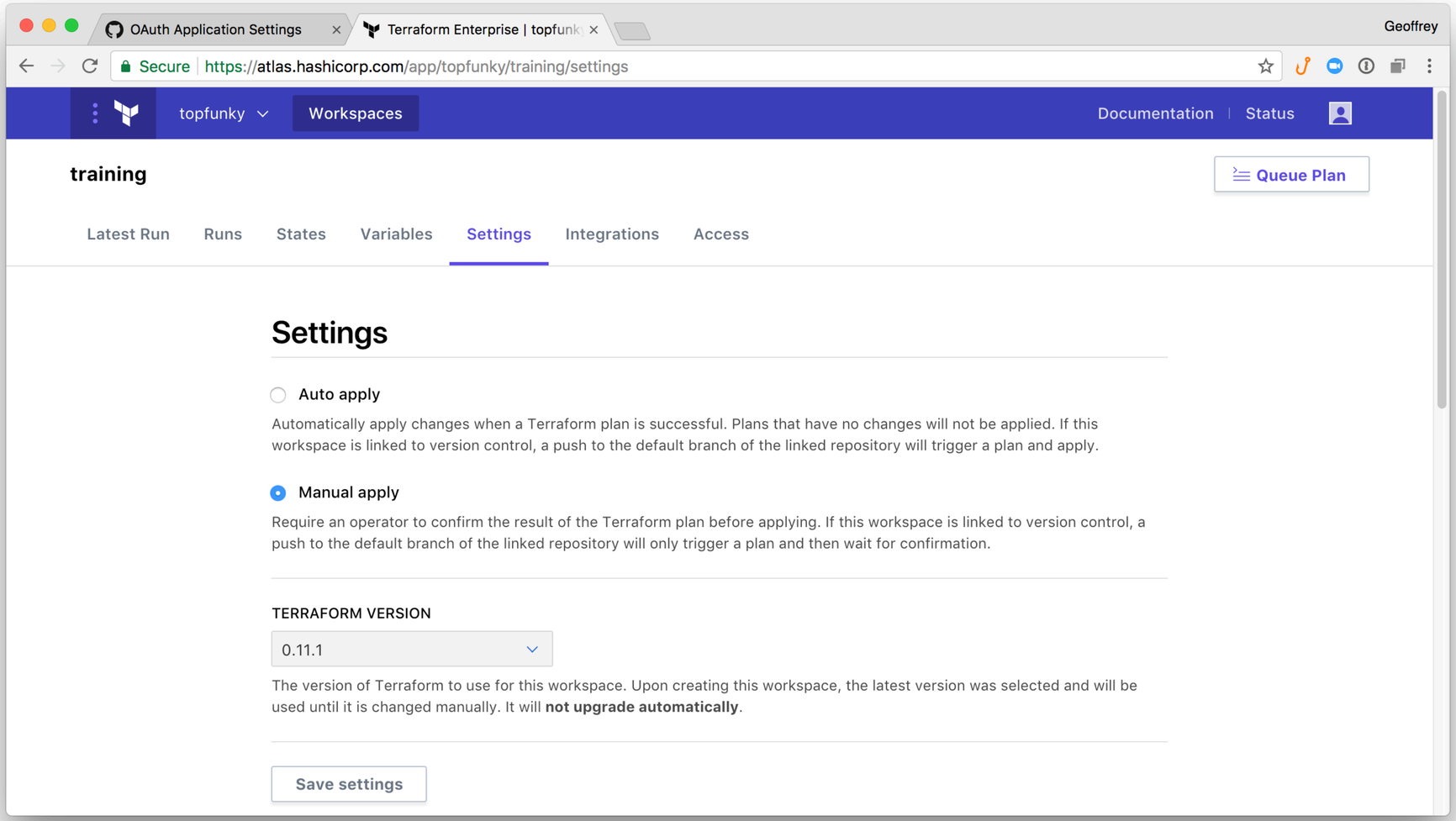
TIP

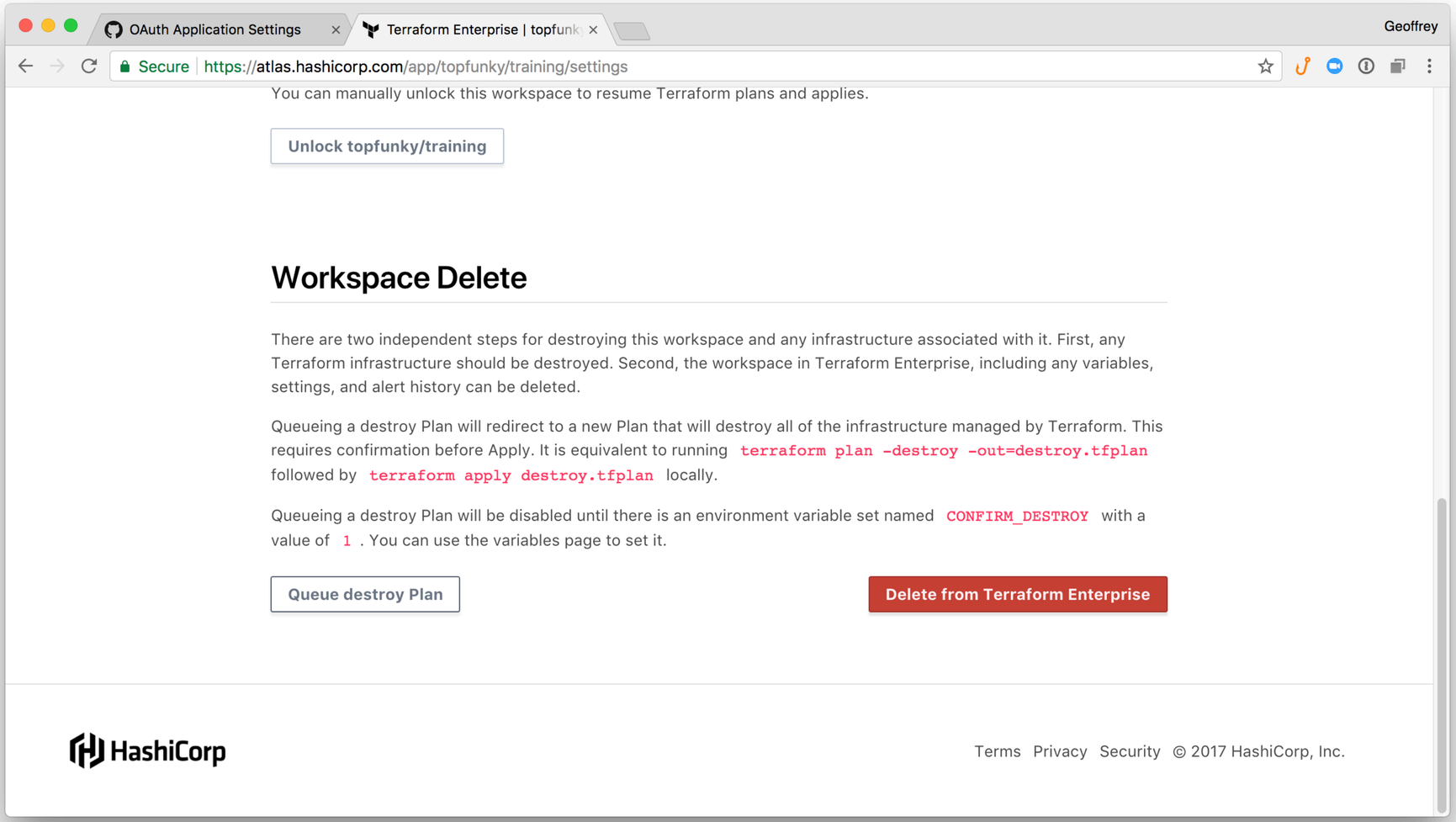
Destroying in Terraform Cloud

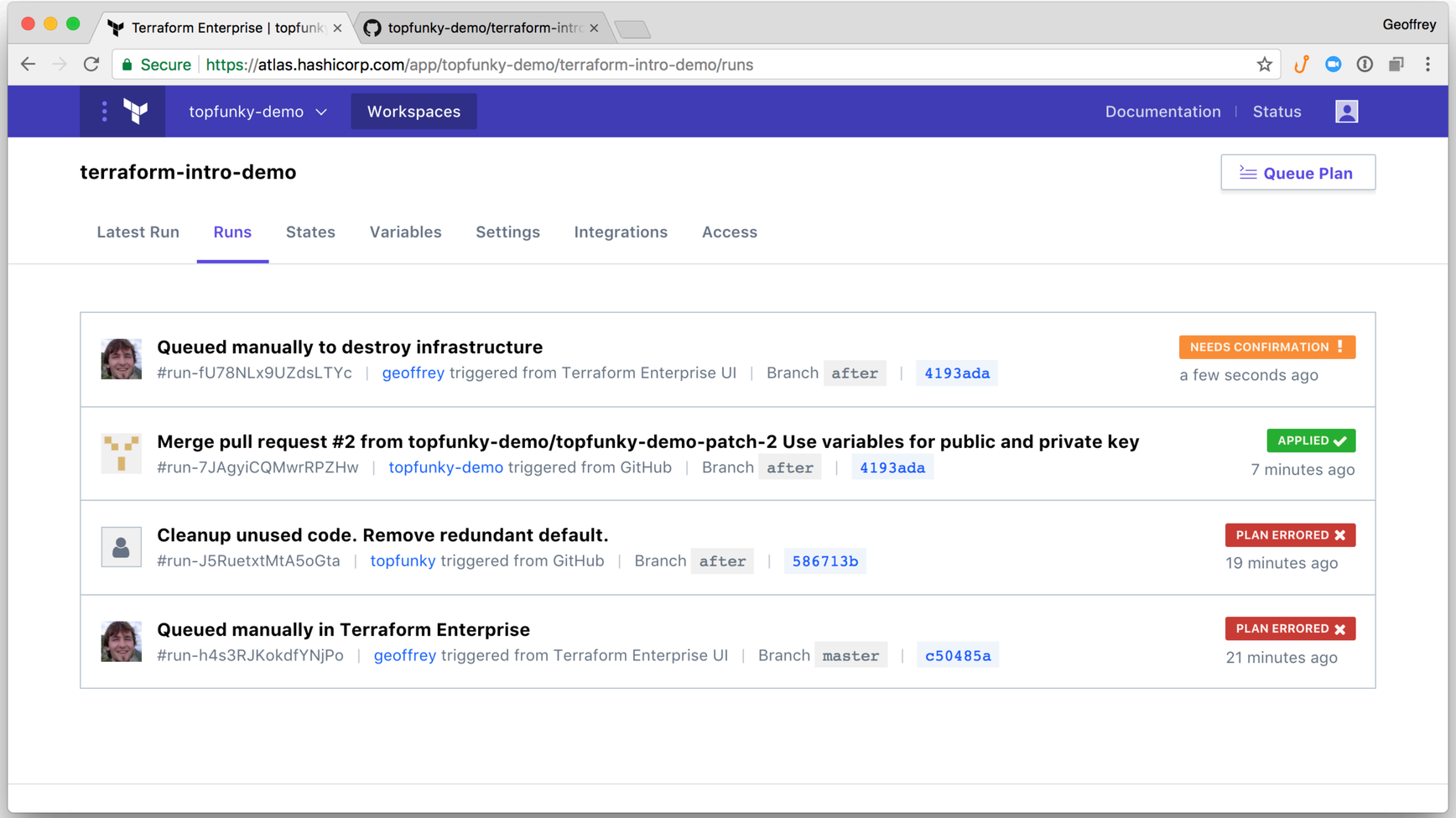
It must be an environment variable, not any other kind of variable

Do not click “Save and Plan.” This will not queue a `destroy` plan (it will queue a normal plan)

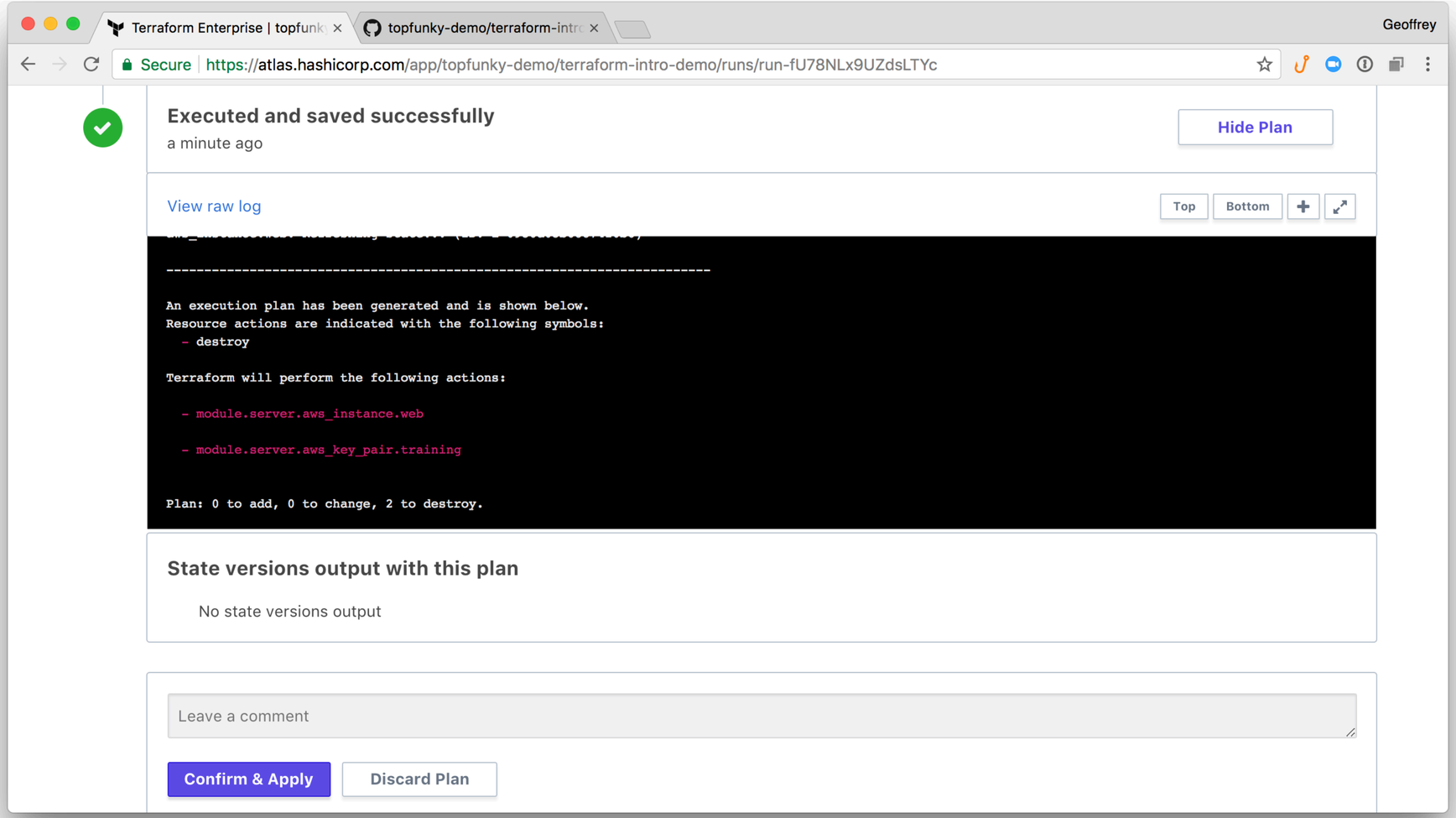
HashiCorp


OAuth Application Settings
Terraform Enterprise | topfunky
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky/training/settings
topfunky
Workspaces
Documentation
Status
training
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
Settings
Auto apply
Automatically apply changes when a Terraform plan is successful. Plans that have no changes will not be applied. If this workspace is linked to version control, a push to the default branch of the linked repository will trigger a plan and apply.
Manual apply
Require an operator to confirm the result of the Terraform plan before applying. If this workspace is linked to version control, a push to the default branch of the linked repository will only trigger a plan and then wait for confirmation.
TERRAFORM VERSION
0.11.1
The version of Terraform to use for this workspace. Upon creating this workspace, the latest version was selected and will be used until it is changed manually. It will not upgrade automatically.
Save settings

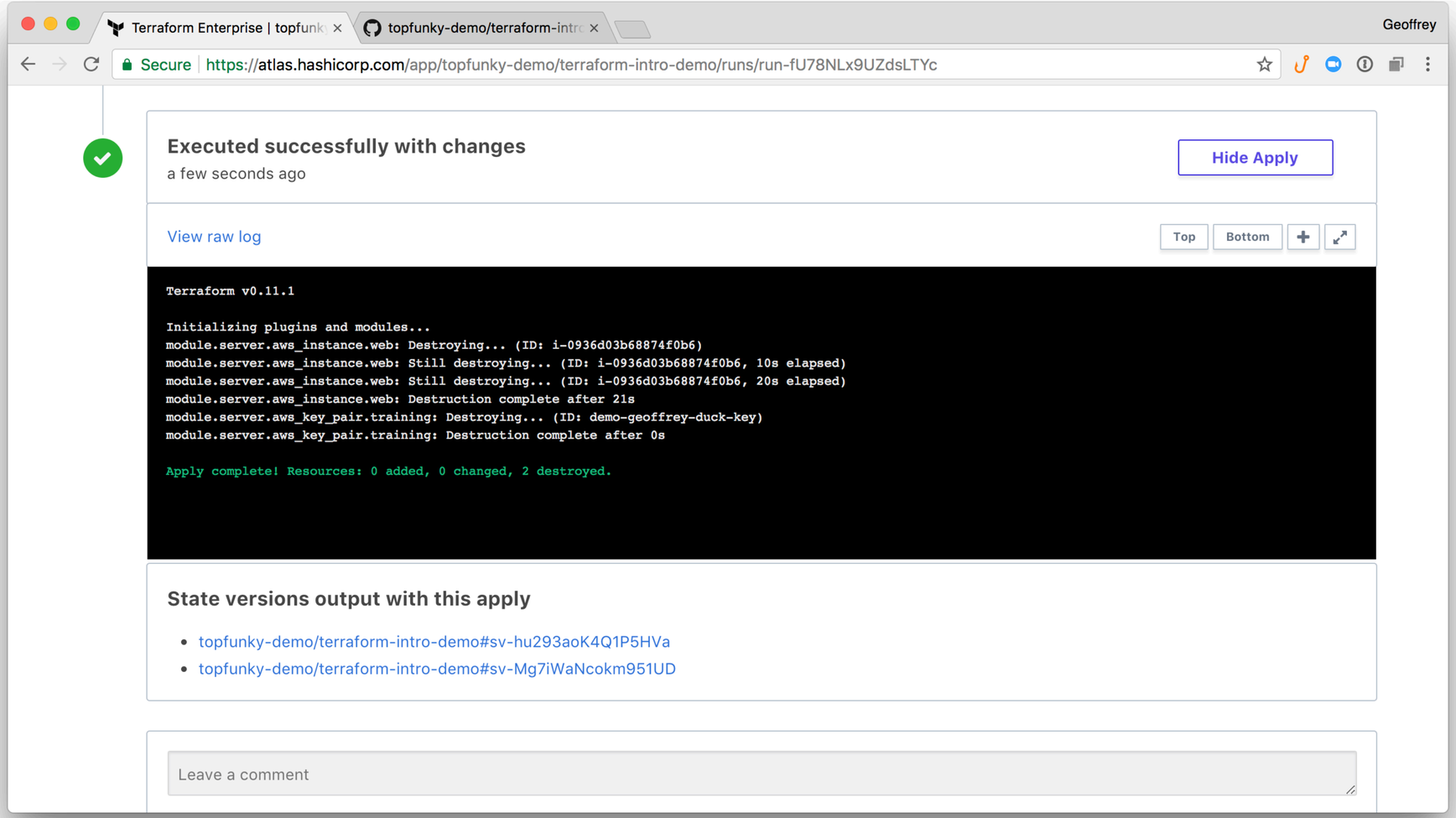

OAuth Application Settings
Terraform Enterprise | topfunk
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky/training/settings
You can manually unlock this workspace to resume Terraform plans and applies.
Unlock topfunky/training
Workspace Delete
There are two independent steps for destroying this workspace and any infrastructure associated with it. First, any Terraform infrastructure should be destroyed. Second, the workspace in Terraform Enterprise, including any variables, settings, and alert history can be deleted.
Queueing a destroy Plan will redirect to a new Plan that will destroy all of the infrastructure managed by Terraform. This requires confirmation before Apply. It is equivalent to running terraform plan -destroy -out=destroy.tfplan followed by terraform apply destroy.tfplan locally.
Queueing a destroy Plan will be disabled until there is an environment variable set named CONFIRM_DESTROY with a value of 1 . You can use the variables page to set it.
Queue destroy Plan
Delete from Terraform Enterprise
HashiCorp
Terms Privacy Security © 2017 HashiCorp, Inc.


Terraform Enterprise | topfunk
topfunky-demo/terraform-intro
Geoffrey
Secure | https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/runs
topfunky-demo
Workspaces
Documentation | Status
terraform-intro-demo
Queue Plan
Latest Run
Runs
States
Variables
Settings
Integrations
Access
Queued manually to destroy infrastructure
#run-fU78NLx9UZdsLTYc | geoffrey triggered from Terraform Enterprise UI | Branch after | 4193ada
NEEDS CONFIRMATION
a few seconds ago
Merge pull request #2 from topfunky-demo/topfunky-demo-patch-2 Use variables for public and private key
#run-7JAgyiCQMwrRPZHw | topfunky-demo triggered from GitHub | Branch after | 4193ada
APPLIED
7 minutes ago
Cleanup unused code. Remove redundant default.
#run-J5RuetxtMtA5oGta | topfunky triggered from GitHub | Branch after | 586713b
PLAN ERRORED
19 minutes ago
Queued manually in Terraform Enterprise
#run-h4s3RJKokdfYNjPo | geoffrey triggered from Terraform Enterprise UI | Branch master | c50485a
PLAN ERRORED
21 minutes ago

HashiCorp


Terraform Enterprise | topfunk
topfunky-demo/terraform-intro
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/runs/run-fU78NLx9UZdsLTYc
Executed and saved successfully
a minute ago
Hide Plan
View raw log
Top
Bottom
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
- destroy
Terraform will perform the following actions:
- module.server.aws_instance.web
- module.server.aws_key_pair.training
Plan: 0 to add, 0 to change, 2 to destroy.
State versions output with this plan
No state versions output
Leave a comment
Confirm & Apply
Discard Plan

HashiCorp


Terraform Enterprise | topfunk
topfunky-demo/terraform-intro
Geoffrey
Secure
https://atlas.hashicorp.com/app/topfunky-demo/terraform-intro-demo/runs/run-fU78NLx9UZdsLTYc
Executed successfully with changes
a few seconds ago
Hide Apply
View raw log
Top
Bottom
Terraform v0.11.1
Initializing plugins and modules...
module.server.aws_instance.web: Destroying... (ID: i-0936d03b68874f0b6)
module.server.aws_instance.web: Still destroying... (ID: i-0936d03b68874f0b6, 10s elapsed)
module.server.aws_instance.web: Still destroying... (ID: i-0936d03b68874f0b6, 20s elapsed)
module.server.aws_instance.web: Destruction complete after 21s
module.server.aws_key_pair.training: Destroying... (ID: demo-geoffrey-duck-key)
module.server.aws_key_pair.training: Destruction complete after 0s
Apply complete! Resources: 0 added, 0 changed, 2 destroyed.
State versions output with this apply
topfunky-demo/terraform-intro-demo#sv-hu293aoK4Q1P5HVa
topfunky-demo/terraform-intro-demo#sv-Mg7iWaNcokm951UD
Leave a comment

HashiCorp

Topics for Self-Exploration

State Environments

terraform.io/docs/state/environments.html

Custom Providers

terraform.io/docs/plugins/index.html

Destroy Provisioners

hashicorp.com/blog/terraform-0-9/#destroy-provisioners

HashiCorp

Q & A

HashiCorp