Introduction to OpenTofu/Terraform

Craig Reeder

Who am I?

Craig Reeder

Senior Cloud Engineer @ Uturn Data Solutions

I write a lot of OpenTofu & Terraform

It changed the way I think about infrastructure and configuration.

What is Terraform?

It is an infrastructure as code provisioning tool.

Originally written in 2014, it quickly became the most popular way to provision infrastructure.

What does that mean?

It means that your infrastructure is represented as code.

Why?

Allows provisioning to be automated, repeatable, and less error prone.

What is OpenTofu?

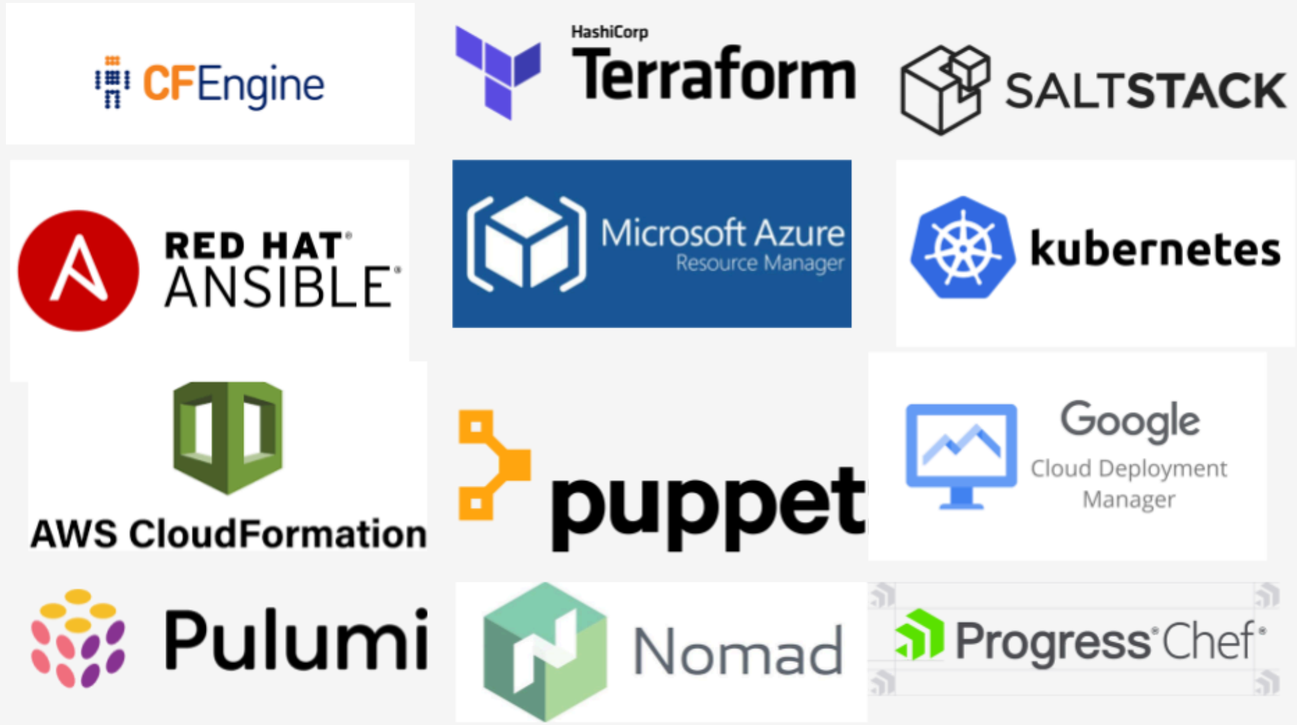
OpenTofu is an open source fork of Terraform

OpenTofu forked in August, 2023 after Terraform announced it would move from an open-source license (MPL) to the Business Source License.

Why do we prefer OpenTofu?

- Is Open-Source and Community Governed
- Member project of the CNCF
- Implemented long-awaited wish list items missing from Terraform

There's a lot of tools that do IaC



Why not those?

- Cross-platform
- Cloud-agnostic
- Popular

So lets learn some TF!

First, some important concepts

What's a Provider?

Providers connect OpenTofu to the underlying infrastructure

OpenTofu is just a tool to execute those providers

The provider performs the "magic" of converting configuration to the result

Common Providers

We're going to be talking about the AWS provider.

Other common ones are PostgreSQL, Cloudflare, and Kubernetes.

Dominos Provider?

You can order pizza with OpenTofu

Configuring a Provider

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 6.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"
}
```

What is a Resource?

A resource is an infrastructure object that is managed by the provider

Resources are the bread and butter of OpenTofu

```
resource "aws_instance" "example" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "ExampleServer"
  }
}
```

Resources have parameters

Parameters are for configuring a resource

`ami`, `instance_type`, and `tags` are parameters here

```
resource "aws_instance" "example" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "ExampleServer"
  }
}
```

Resources have attributes

Attributes get information out of a resource

`private_ip` is the attribute here

```
resource "aws_instance" "example" {
  ...
}

# You can later use this value
aws_instance.example.private_ip
```

Resource Dependencies

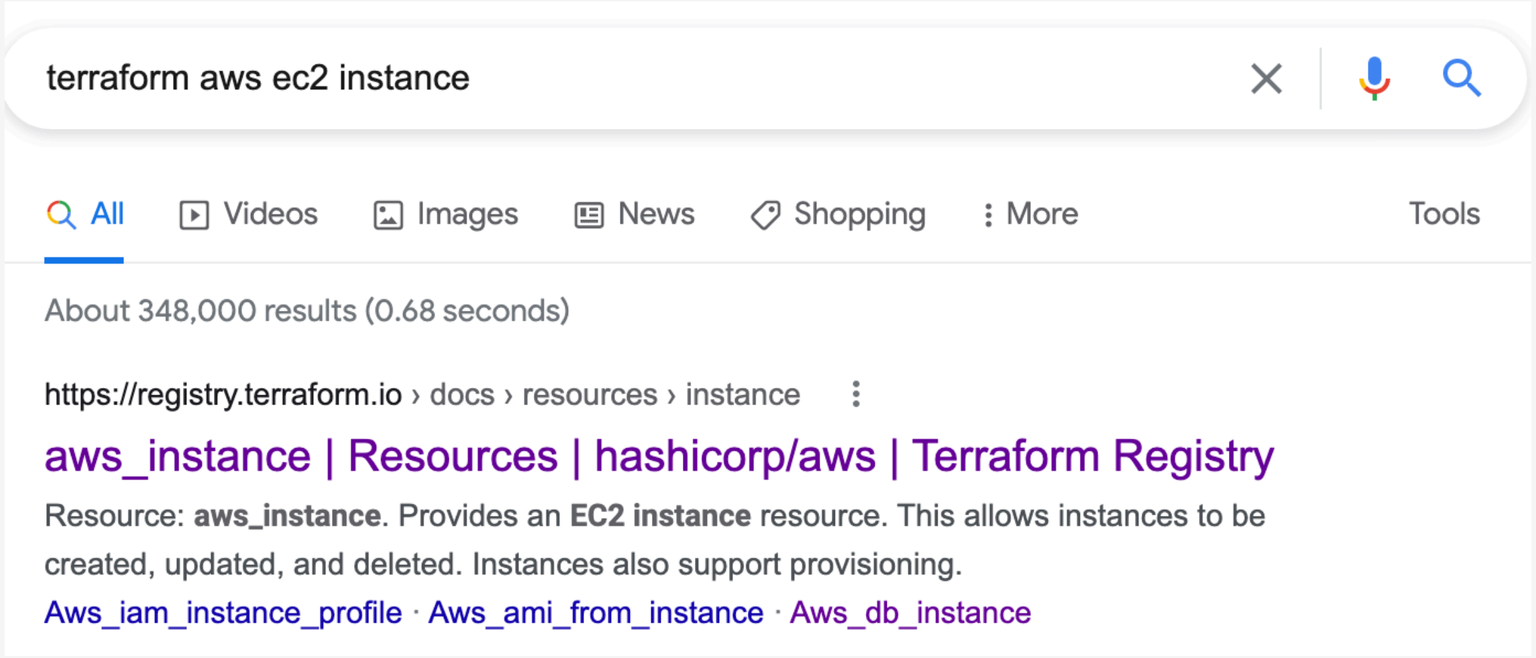
OpenTofu automatically determines the order to create resources

When one resource references another, OpenTofu knows to create them in the correct order

How do I know which parameters and attributes exist?

The Terraform & OpenTofu Registries have provider documentation

Google search tends to work better for the registry


terraform aws ec2 instance
All
Videos
Images
News
Shopping
More
Tools
About 348,000 results (0.68 seconds)
https://registry.terraform.io › docs › resources › instance
aws_instance | Resources | hashicorp/aws | Terraform Registry
Resource: aws_instance. Provides an EC2 instance resource. This allows instances to be created, updated, and deleted. Instances also support provisioning.
Aws_iam_instance_profile · Aws_ami_from_instance · Aws_db_instance

Input Variables

Input variables allow for changing configuration values

```
variable "username" {
  type = string
  description = "Username to apply the changes to"
  default = "Steve"
}

# You can later use this value
var.username
```

Variables can be set in a number of ways

- environment variables
- tfvars files
- passed during apply

Unset variables

If a variable doesn't have a default and isn't set it will prompt you on apply

Variables cannot change during runtime

Local Values

Local values allow for computed or specified values to be defined once and reused

```
locals {
  department = "Operations"
  # ...
  # you can also define more here
}

# You can later use this value
local.department
```

Output Values

Output values do exactly what they sound like

```
output "public_ip" {
  value = aws_instance.example.public_ip
}
```

Data Source

Data sources allow you to get information from a remote location into your code

Data sources are implemented by providers, much like resources

Example

A common data source is to get the most recent version of an AMI

```
data "aws_ami" "windows_server_2019" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["Windows_Server-2019-English-Full-Base-*"]
  }
}

# You can later use this value
# Note the "image_id" attribute here
data.aws_ami.windows_server_2019.image_id
```

State Files

OpenTofu keeps track of what it creates.

That information is stored in a state file.

Never modify the statefile directly

Corrupting your state means that OpenTofu doesn't know what exists

That's bad

It may create new copies of infrastructure, or destroy the existing ones

Local state storage

Without configuration, statefiles are stored locally

(In a file called "terraform.tfstate")

Remote state storage

In AWS, we use S3 buckets (Object storage)

This enables collaboration and safety

Configuring Remote State

```
terraform {
  backend "s3" {
    bucket         = "my-terraform-state"
    key            = "project/terraform.tfstate"
    region         = "us-east-1"
    use_lockfile   = true
  }
}
```

Modules

Modules allow you to reuse OpenTofu code multiple times, in multiple places

Kind of like a software library

Controlling modules

Modules use input variables as their parameters and output values as attributes

Modules can do a lot

Modules can define multiple resources, or use multiple data sources

Modules are written in the same OpenTofu code that projects are written in

Modules can be sourced remotely or locally

```
# This is just an identifier for local access,
# not the name of the module
module "servers" {
  # This is a local "relative" source reference
  source = "../modules/example-cluster"

  servers = 5 # Module's input variables are parameters
}

# Modules's outputs become attributes
# (you can use this elsewhere)
module.servers.ip_list
```

How do you use a module?

To know a modules parameters and attributes, you need to refer to its documentation (or code)

Quick Note: tenv

tenv is an OpenTofu / Terraform version manager

This will allow different versions and tools where appropriate

tenv `tf` alias

`tf` is an alias in tenv for either Terraform or OpenTofu depending on the folder's contents

Specifically, an `.opentofu-version` or `.terraform-version` file

Now what?

We've written some OpenTofu code now

How do we actually apply it?

First, you need to initialize

```
$ tf init
```

This sets up the neccessary providers and downloads modules

You don't need to init everytime

OpenTofu will generally tell you if it's needed again

To see the changes OpenTofu will make:

```
$ tf plan
```

Planning will show you what OpenTofu will change on apply, but will not change anything

Ready to make your infrastructure changes?

```
$ tf apply
```

This will prompt you for confirmation

Idempotency

Running `tf apply` multiple times with the same configuration will not recreate resources

OpenTofu only makes changes when the desired state differs from the actual state

Tear it all down!

```
$ tf destroy
```

This is dangerous, as it will destroy everything if you confirm.

Do not destroy

Outside of sandbox/testing environments, it is not recommended to run destroy. Instead, remove the resources that are no longer needed from the code and let an apply destroy them.

Some Final Notes

File Names

- Filenames don't matter in OpenTofu (only extensions)
- All files with .tf or .tofu will be combined for the apply

Remote State Locking

Two people can't apply the same project at the same time

Official Language

- In OpenTofu, every folder is a "module"
- A "root module" is a module that is applied directly
- A "child module" is one that is referenced
- U:Turn typically uses slightly different language

We Say → Docs

Projects → Root Modules

Stacks → Child Modules

Modules → Child Modules

Questions?