

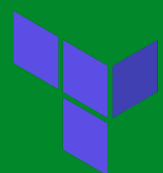
Introduction to Terraform

Presented by: Farley
farley@olindata.com

What is Terraform?

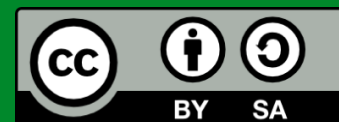
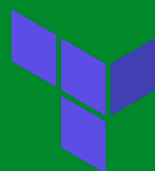


- **Multi-Service Provider Automation Tool**
- **Define infrastructure-as-code**
- **Complete Infrastructure Lifecycle Management**
 - **Environment Creation**
 - **Updating and Versioning Environment**
 - **Re-aligning Environment (from external modifications)**
 - **Destroying Environment**



What is Terraform (more details)

- **A tool to create complex environments across numerous cloud server providers (including AWS, DigitalOcean, Azure, 1&1, etc)**
- **It can also manage your physical/virtual infrastructure (VMWare vSphere, Kubernetes, OpenStack)**
- **Open Source Software, written in Go**
- **Everything in terraform is a plugin, you can actively develop/add/update features to Terraform if desired/needed in a fairly easy fashion**
- **Extremely active development and community**

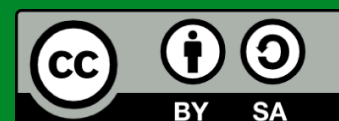
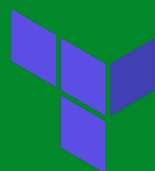


Ok... so what is Terraform?

- It's just a file (or files) that create/modify/destroy your infrastructure on one or many providers.
- The files are in HCL, a format created by its authors at HashiCorp (looks like a hybrid of YAML + JSON)

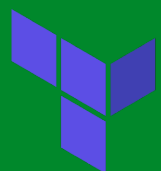
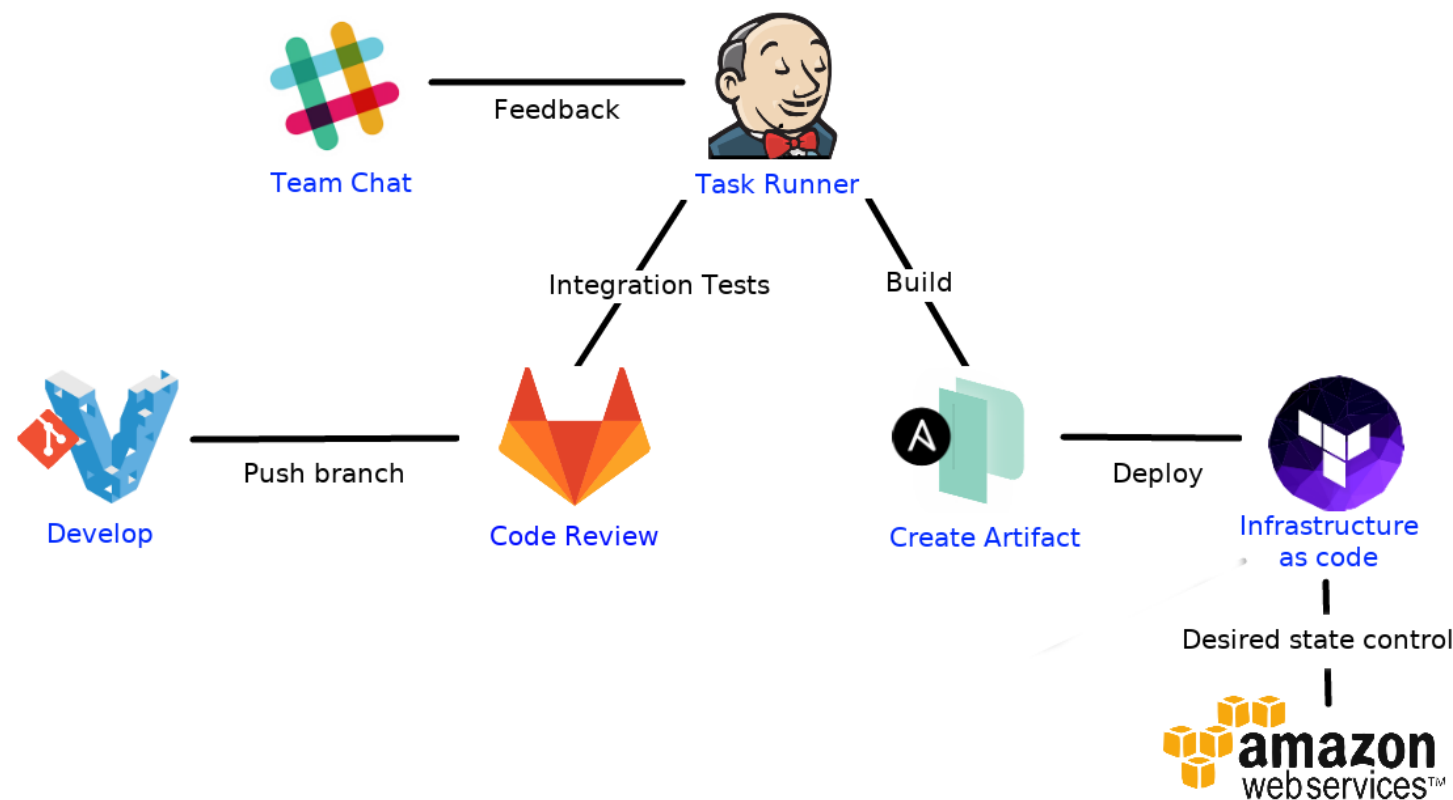
```
# This is the provider we want to use / configure
provider "aws" {
  access_key = "123123123123" # Optional, can use AWS CLI Profiles
  secret_key = "abcabcabcab" # Same as above
  region = "us-east-1"
}

# This is the resource we want to create
resource "aws_instance" "my-dev-server" {
  ami = "ami-12312312"
  instance_type = "t2.micro"
}
```

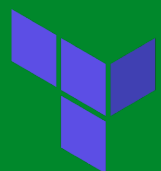
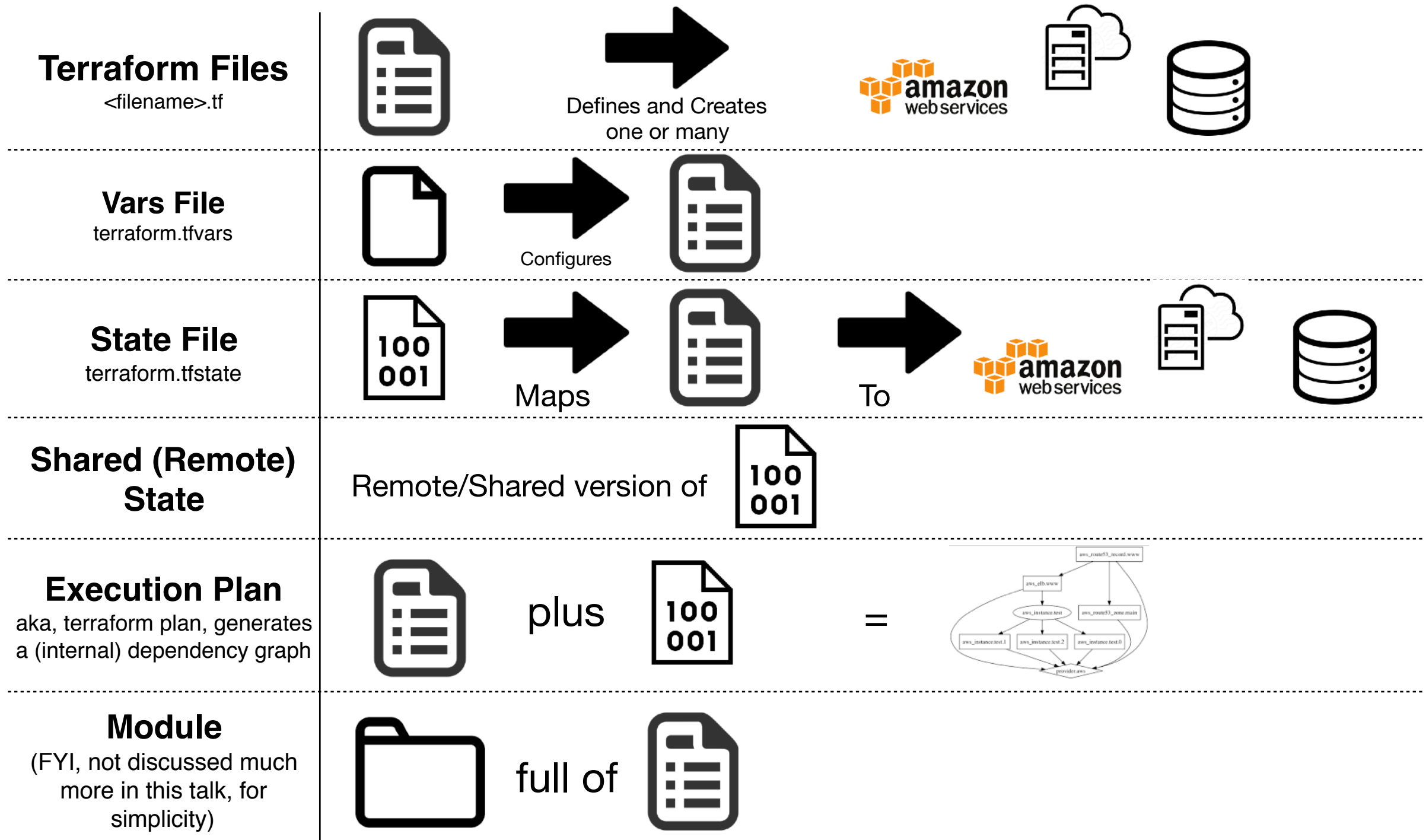


Versioned Infrastructure

- Terraform empowers you to version your infrastructure.
- This allows for Continuous Integration of your infrastructure! Aka, Jenkins builds, for your infra!



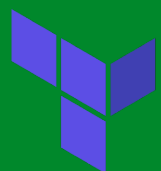
Core Terraform Concepts



Terraform File

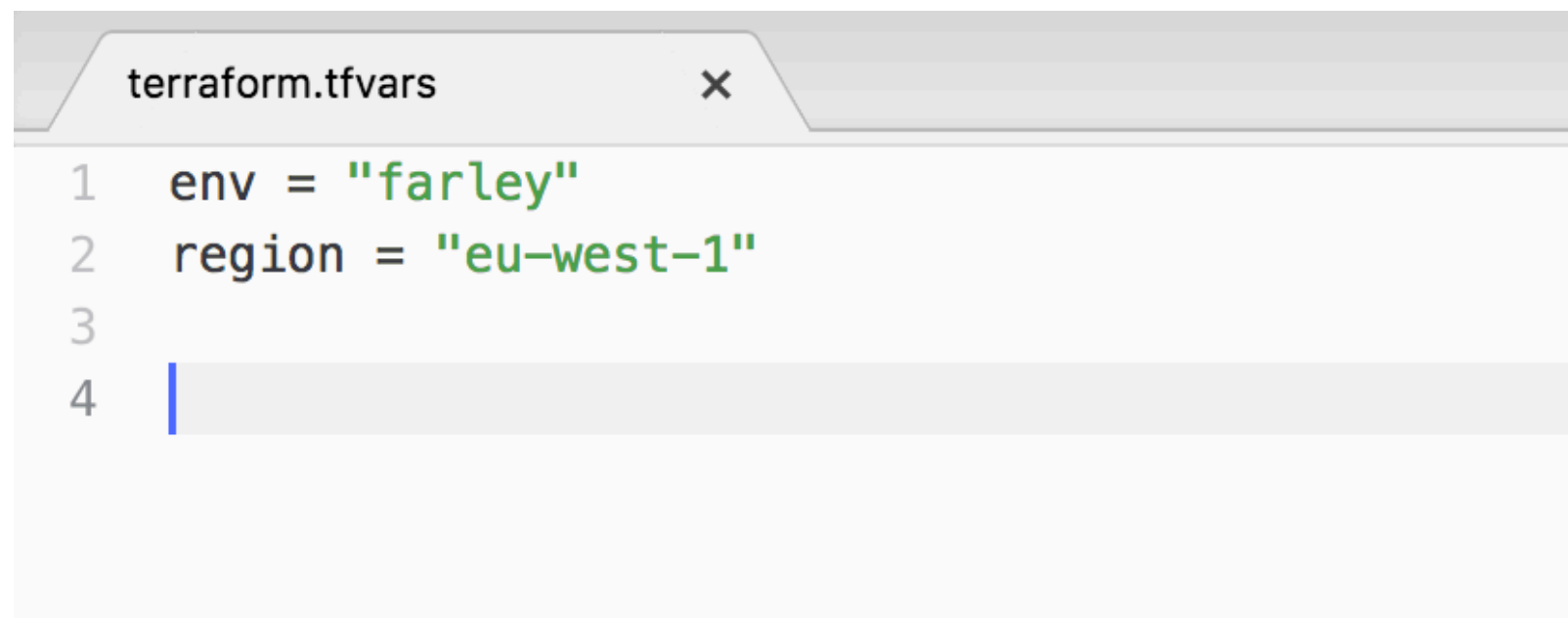
“This is the magic, people!”

```
main.tf x
1
2 # This is the provider we want to use / configure
3 provider "aws" {
4     // access_key = "123123123123" # Intentional, use IAM Instance Role or CLI AWS Profile instead
5     // secret_key = "abcabcabcabc" # Intentional, use IAM Instance Role or CLI AWS Profile instead
6     region = "${var.region}"
7 }
8
9 # This is a data provider, allowing us to use data from AWS within' our terraform below
10 data "aws_caller_identity" "current" {}
11
12 # These are inputs we need to define, these two are fairly common (on basically every stack ever)
13 variable "region" { } # The aws region we want to be working with
14 variable "env" { }    # This is a "prefix" which we will add to the name of everything tag to everything
15 variable "project" { # The name of this project, often used in naming of resources created also
16     default = "terraform-intro"
17 }
18
19 # This is the resource we want to create (or manage/modify) in this case, a S3 bucket
20 resource "aws_s3_bucket" "sample-bucket" {
21     bucket = "${var.env}-sample-bucket-${var.region}-${data.aws_caller_identity.current.account_id}"
22     acl    = "private"
23
24     tags {
25         Environment = "${var.env}"
26         Project      = "${var.project}"
27     }
28 }
29
30
```

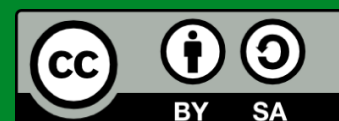
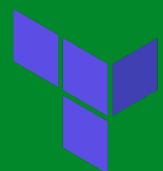


Terraform Config (vars)

Optional, user / environment specific settings or overrides. If not specified in here, terraform will prompt you. Generally this file is not committed to a repository, as it can contain private user information such as passwords, keys, etc. It's best practice if you can avoid this practice. This file could/should point to pointers to sensitive information which you store elsewhere (such as Vault).



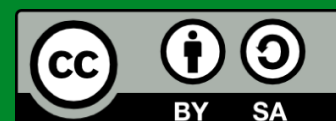
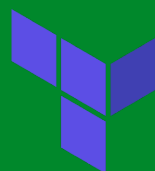
```
1  env = "farley"
2  region = "eu-west-1"
3
4
```



Terraform State

Reminder: This is a GENERATED file, you don't create it (ew!)

```
1 {
2   "version": 3,
3   "terraform_version": "0.9.11",
4   "serial": 0,
5   "lineage": "8642d5fe-7071-4185-8bcb-9bedff3146da",
6   "modules": [
7     {
8       "path": [
9         "root"
10      ],
11      "outputs": {},
12      "resources": {
13        "aws_s3_bucket.sample-bucket": {
14          "type": "aws_s3_bucket",
15          "depends_on": [
16            "data.aws_caller_identity.current"
17          ],
18          "primary": {
19            "id": "farley-sample-bucket-eu-west-1-179381606272",
20            "attributes": {
21              "acceleration_status": "",
22              "acl": "private",
23              "arn": "arn:aws:s3:::farley-sample-bucket-eu-west-1-179381606272",
24              "bucket": "farley-sample-bucket-eu-west-1-179381606272",
25              "bucket_domain_name": "farley-sample-bucket-eu-west-1-179381606272.s3.amazonaws.com",
26              "force_destroy": "false",
27              "hosted_zone_id": "Z1BKCTXD74EZPE",
28              "id": "farley-sample-bucket-eu-west-1-179381606272",
29              "logging.#": "0",
30              "region": "eu-west-1",
31              "request_payer": "BucketOwner",
32              "tags.%": "1",
```



Terraform Plan

This is the “dry-run”, the preview of what Terraform will do

```
[afarley@Farleys-Macbook-Pro:~/OlinData/Grab/terraform-intro-presentation$ terraform plan
var.env
  Enter a value: farley
```

```
var.region
  Enter a value: eu-west-1
```

Refreshing Terraform state in-memory prior to plan...

The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

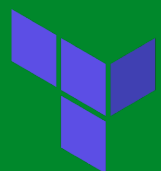
data.aws_caller_identity.current: Refreshing state...

The Terraform execution plan has been generated and is shown below. Resources are shown in alphabetical order for quick scanning. Green resources will be created (or destroyed and then created if an existing resource exists), yellow resources are being changed in-place, and red resources will be destroyed. Cyan entries are data sources to be read.

Note: You didn't specify an "-out" parameter to save this plan, so when "apply" is called, Terraform can't guarantee this is what will execute.

```
+ aws_s3_bucket.sample-bucket
  acceleration_status: "<computed>"
  acl:                  "private"
  arn:                  "<computed>"
  bucket:               "farley-sample-bucket-eu-west-1-179381606272"
  bucket_domain_name:  "<computed>"
  force_destroy:       "false"
  hosted_zone_id:      "<computed>"
  region:              "<computed>"
  request_payer:       "<computed>"
  tags.%:              "1"
  tags.Environment:    "farley"
  versioning.#:        "<computed>"
  website_domain:      "<computed>"
  website_endpoint:    "<computed>"
```

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



Terraform Apply

This is us modifying our infrastructure

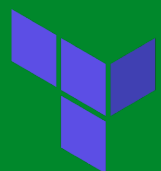
```
afarley@Farleys-Macbook-Pro:~/OlinData/Grab/terraform-intro-presentation$ terraform apply
var.env
  Enter a value: farley

var.region
  Enter a value: eu-west-1

data.aws_caller_identity.current: Refreshing state...
aws_s3_bucket.sample-bucket: Creating...
  acceleration_status: "" => "<computed>"
  acl:                  "" => "private"
  arn:                  "" => "<computed>"
  bucket:               "" => "farley-sample-bucket-eu-west-1-179381606272"
  bucket_domain_name:   "" => "<computed>"
  force_destroy:        "" => "false"
  hosted_zone_id:       "" => "<computed>"
  region:               "" => "<computed>"
  request_payer:         "" => "<computed>"
  tags.%:                "" => "1"
  tags.Environment:     "" => "farley"
  versioning.#:          "" => "<computed>"
  website_domain:        "" => "<computed>"
  website_endpoint:      "" => "<computed>"
aws_s3_bucket.sample-bucket: Creation complete (ID: farley-sample-bucket-eu-west-1-179381606272)
```

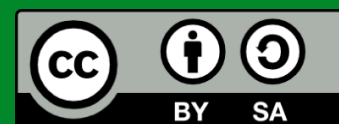
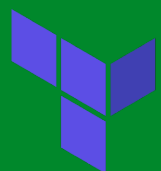
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

The state of your infrastructure has been saved to the path below. This state is required to modify and destroy your infrastructure, so keep it safe. To inspect the complete state use the `terraform show` command.



Popular Use-Case

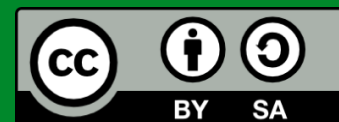
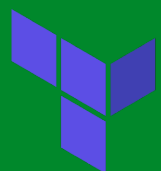
- **The most popular use-case for Terraform is to encapsulate and fully automate the deployment of one entire service and all its dependencies. When fully achieved, this “stack” of infrastructure automation + software + configuration can be a fully working deliverable.**
- **This deliverable can be handed off to QA who can perform their own complete deployment (either updating an existing deployment or creating a new one) and perform their rounds of testing. Who can then hand off the code to a ops engineer to deploy/update to customers, or even handing off your code directly to a customer (when desired, eg: using a shared responsibility model).**



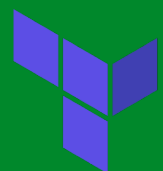
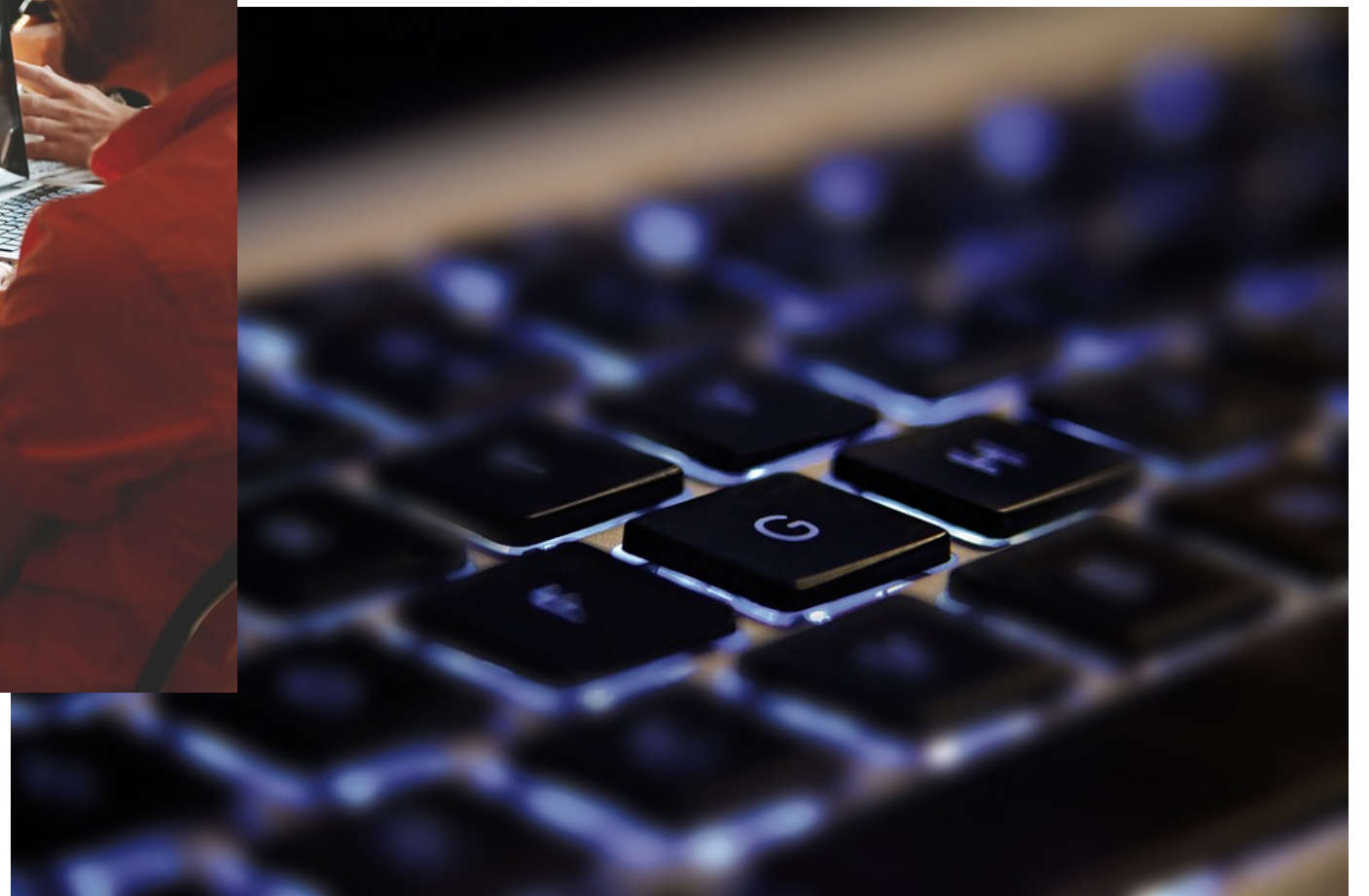
Advanced Topics

For next time!

- Terraform best-practices
- Terraform Security
- Using Terraform modules, and creating reusable terraform modules
- Using existing infrastructure with terraform (aka, terraform import)
- Shared State / Shared Variables
- Continuous Integration / Pipelining Terraform
- user-data, before/after creation scripts, inline scripts
- Integration with other languages and tools
- Using an existing stack's state as an input for another stack



(simple) Demo



Your first steps...

First, download Terraform for your OS
<https://www.terraform.io/downloads.html>

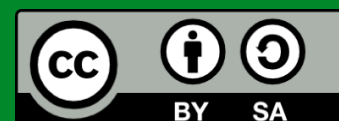
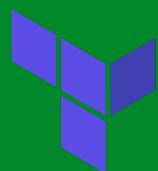
THEN

First start with the example from this presentation
<https://github.com/AndrewFarley/introduction-to-terraform>

Then google around for examples / sample code to do what you want!

Don't forget to use the terraform docs to figure out all the options, features, variables, and outputs that all the terraform objects have. You'll never remember them all even as a seasoned veteran because they are constantly added to!

Terraform Docs: <https://www.terraform.io/docs/index.html>



Feature-set vs Alternatives



Terraform



CloudFormation



Ansible / Shell

	Terraform	CloudFormation	Ansible / Shell
Providers	<u>Many</u>	AWS	Many
Manage Existing Resources	Yes (but time consuming)	No	No (hard)
State Locking	Yes	Yes	(no state support)
Modification Detection and Realignment	Yes	No	No (self-implemented / hard)
Shared State	Yes	Yes	(no state support)
Editable State	Yes	No	(no state support)
Syntax	HCL (JSON also)	JSON & YAML	YAML
Versioned Infra	Yes	Yes	No (since no state support)
“Magic” (things happen that you didn't define and can't control)	No	Yes (which can be bad, but can save effort)	No
Previewing Changes	Yes	Yes	Yes (—diff, but without state will be comparably inaccurate)
Multi-Region	Yes	No	Yes
Modules / Roles (re-use code parts of your stack)	Yes	Yes (Cross-stack references)	Yes

Thanks!

Questions?

Ask them now, or...

farley@olindata.com

