Build Infrastructure

We'll build infrastructure on AWS for the getting started guide since it is popular and generally understood, but Terraform can manage many providers. If you don't have an AWS account, create one now.

» Configuration

The set of files used to describe infrastructure in Terraform is simply known as a Terraform *configuration*. We're going to write our first configuration now to launch a single AWS EC2 instance. Save the contents to a file named example.tf. Verify that there are no other *.tf files in your directory, since Terraform loads all of them.

```
provider "aws" {
  access_key = "ACCESS_KEY_HERE"
  secret_key = "SECRET_KEY_HERE"
  region     = "us-east-1"
}

resource "aws_instance" "example" {
  ami           = "ami-2757f631"
  instance_type = "t2.micro"
}
```

Note: The above configuration is designed to work on most EC2 accounts, with access to a default VPC. For EC2 Classic users, please use t1.micro for instance_type, and ami-408c7f28 for the ami. If you use a region other than us-east-1 then you will need to choose an AMI in that region as AMI IDs are region specific.

Replace the ACCESS_KEY_HERE and SECRET_KEY_HERE with your AWS access key and secret key,

The provider block is used to configure the named provider, in our case "aws". A provider is responsible for creating and managing resources.

» Initialization

Save the file where terraform binary exists. The first command to run for a new configuration -- or after checking out an existing configuration from version control -- is **terraform init**, which initializes various local settings and data that will be used by subsequent commands.

Terraform uses a plugin based architecture to support the numerous infrastructure and service providers available. As of Terraform version 0.10.0, each "Provider" is its own encapsulated binary distributed separately from Terraform itself. The terraform init command will automatically download and install any Provider binary for the providers in use within the configuration, which in this case is just the aws provider:

>> terraform init

» Apply Changes

Note: The commands shown in this guide apply to Terraform 0.11 and above. Earlier versions require using the terraform plan command to see the execution plan before applying it. Use terraform version to confirm your running version.

In the same directory as the example.tf file you created, run **terraform apply**.

>> terraform apply

This output shows the *execution plan*, describing which actions Terraform will take in order to change real infrastructure to match the configuration. When the value displayed is (known after apply), it means that the value won't be known until the resource is created.

If terraform apply failed with an error, read the error message and fix the error that occurred. At this stage, it is likely to be a syntax error in the configuration.

If the plan was created successfully, Terraform will now pause and wait for approval before proceeding. If anything in the plan seems incorrect or dangerous, it is safe to abort here with no changes made to your infrastructure. In this case the plan looks acceptable, so type yes at the confirmation prompt to proceed.

Executing the plan will take a few minutes since Terraform waits for the EC2 instance to become available:

After this, Terraform is all done! You can go to the EC2 console to see the created EC2 instance. (Make sure you're looking at the same region that was configured in the provider configuration!)

Terraform also wrote some data into the terraform.tfstate file. This state file is extremely important; it keeps track of the IDs of created resources so that Terraform knows what it is managing.

Note: You can inspect the current state using terraform show: