Modern Provisioning and CI/CD with Terraform, Terratest & Jenkins

Duncan Hutty

Overview

1. Introduction: Context, Philosophy
2. Provisioning Exercises
    1. MVP
    2. Testing
    3. CI/CD
    4. Refactoring
3. Coping with complexity & scale

http://gitlab.com/dhutty/modern-provisioning_code

Introduction

- Context
- Philosophy

Context

- Infrastructure
- Configuration [Management]
- Orchestration
- Provisioning
- CI/CD
- Pipelines

Philosophy

"Everything As Code"

- Use Code for Everything
- Test
- Review
- Lifecycle
- Engineer All The Code

Infrastructure As Code

Treating the tooling that provisions and manages your infrastructure with the same respect as other code.

"Engineering Services"

Engineering Engineering

Benefits of "As Code"

Easier to:

- Consistently regenerate
- Test
- Review
- Grok
- Audit
- Iteratively improve
- Reuse, compose and hide complexity

Tech

- VCS: git+ github
- Scheduler: Jenkins
- PaaS, IaaS: AWS
- Provisioning, making somewhere to deploy *to*: Terraform
- Testing: Jenkinsfiles, simple scripts, Terratest

And with all that said, let’s get on to making things happen, showing some code.

Provisioning Exercises

1. MVP
2. Terratest
3. Containerization
4. CI/CD
5. Refactoring

Setup

Clone the Repository

```
$ git clone https://gitlab.com/dhutty/modern-provisioning_code
```

Terraform installation

Hashicorp provides installation instructions for Terraform.

- install the binary as `terraform-<version>`

```
ln -s ~/bin/terraform-<version> ~/bin/terraform
alias tf=terraform
```

- Install jq

Terratest

- Install terratest

```
for M in $(cat ../terratest_modules.txt); do go get github.com/gruntwork-io/terratest/modules/${M}; done
for P in $(cat ../go_packages.txt); do go get ${P}; done
```

AWS

Install CLI tooling

```
$ virtualenv -p python3 --no-site-packages venv
$ source venv/bin/activate
$ pip install awscli
```

AWS provider for Terraform

Terraform ships a provider for AWS, all you need is to configure:

```
provider "aws" {
  region = "us-east-1"  # the only required argument
  version = "~> 1.36"
}
```

terraform init

```
$ cd tf
$ terraform init
Initializing provider plugins...

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

AWS account

- Create a API key to interact with AWS itself

Environment variables

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export AWS_DEFAULT_REGION=us-east-1
```

Configuration files

- `~/.aws/config`
- `~/.aws/credentials`

Proof

```
aws --output table  ec2 describe-regions
-------------------------------------------------------------
|                     DescribeRegions                       |
+-----------------------------------------------------------+
||                         Regions                         ||
|+-----------------------------------+-------------------+|
||             Endpoint              |   RegionName      ||
|+-----------------------------------+-------------------+|
||  ec2.ap-south-1.amazonaws.com     |  ap-south-1       ||
||  ec2.eu-west-3.amazonaws.com      |  eu-west-3        ||
||  ec2.eu-west-2.amazonaws.com      |  eu-west-2        ||
||  ec2.eu-west-1.amazonaws.com      |  eu-west-1        ||
||  ec2.ap-northeast-2.amazonaws.com |  ap-northeast-2   ||
||  ec2.ap-northeast-1.amazonaws.com |  ap-northeast-1   ||
||  ec2.sa-east-1.amazonaws.com      |  sa-east-1        ||
||  ec2.ca-central-1.amazonaws.com   |  ca-central-1     ||
```

MVP

- an instance
- a security group
- a key_pair so it can be reached
- lots of networking-fu that can be ignored if you have a Default VPC

Provider Configuration

Variables

Outputs

Proof

```
source local.sh.example
tf plan
ssh ec2-user@${PUBLIC_IP} 'echo $(hostname --fqdn)'
```

Provisioning

Configure with `user_data`

- Pass a shell script and it will be run upon launch
- Proof: `curl -v http://<public_ip_of_new_instance>:8080`

EC2 Docs on `user_data`

Configuring with Ansible

```
resource "null_resource" "install-python" {
  provisioner "remote-exec" {
    inline = [
      "sudo yum install -y python-virtualenv",
    ]
  }

  connection {
    type        = "ssh"
    private_key = "${file(var.ssh_private_key_path)}"
    user        = "${var.ssh_user}"
    host        = "${aws_instance.mvp.public_ip}"
  }
}
```

```
provisioner "local-exec" {
  command = "ansible-playbook -vD -i ansible/hosts playbook.yml"
}
```

Templated output for Ansible Inventory

Proof

- `curl -v http://$(awk -F'=' '/ansible_host/ {print $2}' ansible/hosts):8080`
- `curl -v http://$(awk -F'=' '/ansible_host/ {print $2}' ansible/hosts):8081`

```
ansible -i ansible/hosts -m shell -a 'hostname --fqdn && uptime' all
ansible-playbook -vD -i ansible/hosts playbook.yml
```

Terratest

- `Go` testing: write files `*_test.go`, run `go test`

Containers

Containerize pythonhttp

Containerize Jenkins

CI/CD

Refactoring

Terraform Modules

Modules are encapsulated Terraform configuration that are used to:

- better organize TF code
- make TF code more easily resuable
- https://www.terraform.io/docs/modules/create.html#standard-module-structure

Terraform State

The End

Repositories

- http://gitlab.com/dhutty/modern-provisioning_code
- http://gitlab.com/dhutty/pythonhttp

Extras

- https://registry.terraform.io/
- https://github.com/segmentio/terraform-docs

Further Work

- Port to OCI, Azure, GCP
- Add support for other infrastructure resource types, including non-IaaS

Colophon

This repository contains both the class (presentation) and the demonstration code.

The Presentation

- written in asciidoc
- uses asciidoctor and revealjs for slideshow functionality
- needs nodejs/npm to run a local server for speaker notes.

Software Setup

Install asciidoctor-revealjs. I used `rvm` and:

```
$ rm -f Gemfile.lock
  bundle config --local github.https true
  bundle --path=.bundle/gems --binstubs=.bundle/.bin
```

Clone this repository. From the root of this repo, clone revealjs repository with:

```
git clone https://github.com/hakimel/reveal.js.git
```

Point revealjs app at this presentation with:

```
ln -sf ../index.html index.html
```

Install all the node-fu and start the webserver with:

```
npm install && npm start -- --port=5000
```

Usage

From the top repo, generate the presentation with:

```
$ bundle exec asciidoctor-revealjs -a revealjsdir=. presentation/index.adoc
```

Visit http://localhost:5000

- Exporting Slides to PDF

```
$ docker run --rm -v `pwd`:/home/user astefanutti/decktape /home/user/index.html /home/user/slides.pdf
```