SERVER PROVISIONING WITH TERRAFORM, ANSIBLE AND PACKER

MARCUS AHNVE

TABLE OF CONTENTS

- About This Talk
- About Terraform
- Terraform Demo
- About Provisioning Tools
- Ansible
- Ansible Demo
- About Packer
- Packer demo
- Architecture
- Finally

ABOUT THIS TALK

INTRODUCTORY TALK ON

- Terraform
- Ansible
- Packer

WE WILL DEMO ON DIGITALOCEAN

- Simple to set up a single server
- Cheap

OR LOCAL QEMU

- Backup

WHAT WE WILL DO

- Launch a server (Terraform)
- Provision it (Ansible)
- Build an image for it (Packer)

CODE ONLINE

- https://github.com/mahnve/server-provisioning

ABOUT TERRAFORM

- Launches Infrastructure
- Handles Many Providers
- Files written in HCL
- Handles state

PROVIDERS

- AWS
- Azure
- DigitalOcean
- Heroku
- GitHub
- Mailgun
- ...

TERRAFORM DEMO

CODE REPOSITORY

CREATE DIRECTORY

```
$ cd <favorite_src_dir>
$ git init server_provisioning
$ cd server_provisioning
$ mkdir terraform
$ cd terraform
```

CHECKING THAT TERRAFORM IS INSTALLED

```
$ terraform
usage: terraform [--version] [--help] <command> [args]
[...]
```

THE TERRAFORM.TFVARS FILE

- If not set, Terraform will ask for them
- Do not check into version control

```
do_token = "very_long_token"
public_key = "~/.ssh/key.pub"
```

BASIC TERRAFORM FILE

- All .tf files are included by default by the terraform command

```
variable "do_token" {}
variable "public_key" {}

provider "digitalocean" {
  token="${var.do_token}"
}
resource "digitalocean_ssh_key" "default" {
  name = "tf-key"
  public_key = "${file("${var.public_key}")}"
}

resource "digitalocean_droplet" "test-server" {
  name="test-server"
  region="ams2"
  size="512mb"
  image="debian-8-x64"
  ssh_keys= ["${digitalocean_ssh_key.default.fingerprint}"]
}
```

TERRAFORM PLAN TO CHECK THAT ALL IS OK

$ terraform plan

TERRAFORM APPLY TO ACTUALLY BUILD STUFF

$ terraform apply

FOR EXTRA CREDIT, CREATE A MAKEFILE

- Because Makefiles

```
.PHONY: tf-apply tf-plan

tf-apply:
  terraform apply

tf-plan:
  terraform plan
```

ABOUT PROVISIONING TOOLS

WHAT'S WRONG WITH BASH?

- Procedural vs Declarative
- Reinventing wheel

WHAT'S WRONG WITH CHEF/PUPPET?

- 1st Generation
- Really assumes server based

WHAT'S WRONG WITH ANSIBLE?

- YAML
- YAML
- YAML
- Not well known for accepting outside contributions
- Occasional versioning problems

WHAT'S GOOD ABOUT ANSIBLE?

- Serverless (SSH/WinRM)
  - Easy to setup
  - Scales well enough
- Easy to understand
- Works on Windows (?)

ABOUT ANSIBLE

- Open Source
- Acquired by RedHat 2015
- Tower is the commercial offering

ANSIBLE

PARTS

PLAYBOOKS

```
- hosts: webservers
  vars:
    http_port: 80
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    apt: name=apache2 state=latest
  - name: write the apache config file
    template: src=site.conf.j2 dest=/etc/sites-available/site.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

INVENTORY

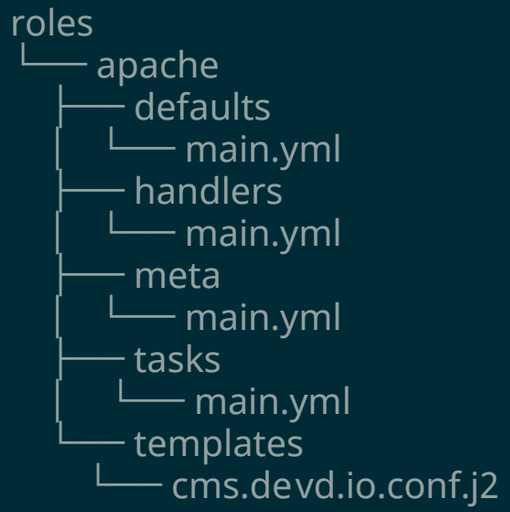
```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

COMMANDS

```
$ ansible-playbook playbook.yml -i hosts
```

ROLES

- Group related tasks together



ROLES IN PLAYBOOKS

```
- hosts: all
  roles:
    - sshd
- hosts: db
  roles:
    - postgresql
- hosts: webservers
  roles:
    - apache
```

ANSIBLE GALAXY

- Reuse other peoples stuff, yay!
- Reuse other peoples stuff, yuk!

MODULES

Where things happen

CLOUD

- Overlap with Terraform
- AWS
- Azure
- Digital Ocean
- ...

CLUSTERING

- Consul
- Kubernetes
- ZooKeeper

COMMANDS

Run stuff directly

DATABASE

- PostgreSQL
- MySQL
- MSSQL
- Mongo
- Redis
- ...

FILES

- Upload
- Modify
- Templating

MESSAGING

RabbitMQ

MONITORING

- Airbrake
- Pingdom
- UptimeRobot
- ...

NETWORK

- Download files
- Manage DNS
- ...

NOTIFICATION

- Slack
- IRC
- ...

PACKAGING

- apt
- yum
- pip
- maven
- bower
- npm
- ...

SOURCE CONTROL

- git
- github
- ...

SYSTEM

- service
- systemd
- mount
- cron
- ...

WEB INFRA

- apache2_module
- letsencrypt

WINDOWS

Lots of stuff

ANSIBLE DEMO

CREATE ROLE DIRECTORY

```
$ mkdir -p ansible/roles

$ cd ansible/roles

$ mkdir -p firewall/{defaults, tasks, handlers}
$ mkdir -p base/tasks
$ mkdir -p app/{defaults,tasks,templates,handlers}
```

BASE TASKS

```
---
- name: Install base packages
  apt: name={{ item }}
  with_items:
    - supervisor
```

FIREWALL TASKS

```yaml
---
- name: install ufw
  apt: name=ufw

- name: open our web port
  ufw:
    rule: allow
    port: 9090
  notify: reload ufw

- name: open SSH port
  ufw:
    rule: allow
    port: 22
  notify: reload ufw

- name: enable ufw
  ufw:
    state: enabled
    policy: deny
```

FIREWALL HANDLERS

```
---
- name: reload ufw
  ufw: state=reloaded
```

APP

```
---
- name: Create app directory
  file: path=/opt/app-0.1/ state=directory

- name: Download app
  get_url:
  url: https://github.com/mahnve/go-web/releases/download/0.1/run
   dest: /opt/app-0.1/app
   mode: 0755
  notify: restart app

- name: symlink
  file: src=/opt/app-0.1 dest=/opt/app state=link

- name: upload supervisord template
  template: src=supervisord.template dest=/etc/supervisor/conf.d/app.conf
  notify: restart app

- name: load app
  supervisorctl: name=app state=present
```

APP HANDLERS

```
---
- name: restart app
  supervisorctl: name=app state=restarted
```

APP SUPERVISOR TEMPLATE

```
[program:app]
command=/opt/app/app
```

THE HOSTS FILE

```
[digital_ocean]
<ip>  ansible_ssh_user=root
```

THE PLAYBOOK FILE

- digitalocean.yaml

```
---
- hosts: digital_ocean
  roles:
  - role: firewall
```

RUN ANSIBLE TO PROVISION

```
$ ansible-playbook -i hosts digitalocean.yaml
```

REALLY, CREATE A MAKEFILE

```
digitalocean:
  ansible-playbook -i hosts digitalocean.yaml
```

ABOUT PACKER

WHAT IS IT

- Built by Hashicorp
  - They're everywhere!
- JSON-files

WHAT IT DOES

- Boots an image
- Runs provisioning
- Takes snapshot
- Shuts down

WHAT CAN WE USE IT FOR

- Immutable infrastructure
  - Slow
- Bake-off images, Immutable with constructor params
  - Load changes on boot
  - Cloud-Config

PACKER DEMO

CREATE DIRECTORY

```
$ mkdir -p packer
$ cd packer
```

THE DO.JSON FILE

```
{
"builders": [
    {
        "type": "digitalocean",
        "api_token": "token",
        "image": "debian-8-7-x64",
        "region": "FRA1",
        "size": "512mb",
        "ssh_username": "root",
        "user_data_file": "user_data.yaml"
    }
 ],

 "provisioners": [
    {
        "type": "ansible",
        "playbook_file": "../image.yaml"
    }
 ]
 }
```

THE DO-IMAGE.YAML FILE

```
---
- hosts: default
  roles:
    - role: firewall
    - role: base
    - role: app
```

CLOUD-CONFIG

- Runs code on boot
- If you're running the right Linux
- On the right cloud
- Maybe

ARCHITECTURE

DIFFERENT LEVELS

SERVERS

- Individual machines
- Real hardware
- Virtual Servers
  - OpenStack
  - VMWare
  - Proxmox`
- Cloud
  - AWS
  - Azure
  - Google Cloud
  - DigitalOcean

CONTAINER SERVICES

- Kubernetes
  - Deis
- Apache Mesos
  - Mesosphere
- Docker Cloud
- Dokku

PAAS

- Heroku
- AWS Beanstalk
- Azure App Service

STANDARD SERVERS

- Two ways to run standard servers

THE STANDARD WAY

- Install OS
  - Start Server
  - Terraform
- Provision Server
  - Make sure it has the right packages
  - Ansible
  - Chef
  - Puppet
  - Salt
- Deploy
  - Ansible?
  - Fabric
  - Capistrano

USE SERVER IMAGES

- Build Image
  - Packer
  - Set up deploy through Cloud Init
- Launch server from image

WHY RUN YOUR OWN?

OWN YOUR DATA

- Monitoring
- Logging

FLEXIBILITY

- Change Platform
  - HAProxy over cloud provided load balancer?
  - Cloud provider
  - Database
  - Container Orchestration
- Run local development

FINALLY

- Tooling is not devops
- Know your architecture
- Thanks