# Ansible & Terraform for django projects

Move faster, break fewer things, by Ezequiel Golub

# About me

- Originally Python backend dev
- (but) Exclusively devopsing for 2 years.
- Working High traffic revenue generating sites
- Python <3 for 5 years

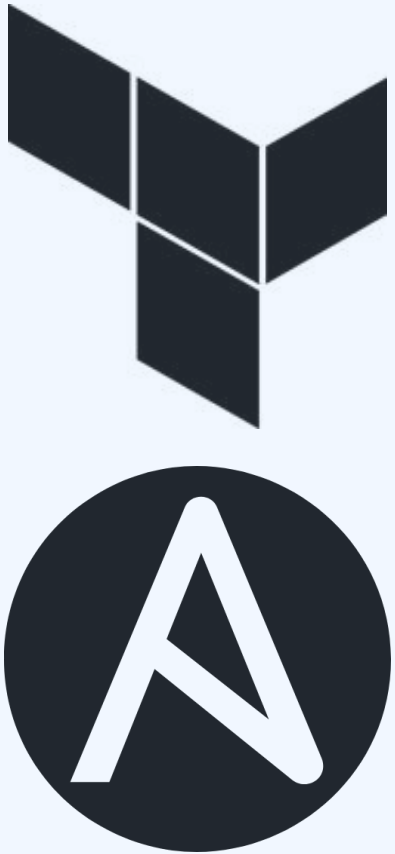
# Why complicate yourself?

My current release process works fine, why invest time on this?

- Break down silos between dev team and ops team
- New releases every day (or every hour or whatever)
- Enable better telemetry of your app
- Infrastructure as code
- Servers as cattle, not pets

# Introducing...

- Terraform: Provisioning cloud resources
- Ansible: For everything software related



# Terraform facts

- V0.10.3 -- 88 releases in ~ 3 years
- Open source (https://github.com/hashicorp/terraform)
- Backed by Hashicorp
- Written in Golang
- Sometimes might be unstable for new features

# Terraform key features

- Infrastructure as (readable) Code
- Execution Plans
- Resource Graph
- Works with most cloud providers
- Supports saving state in a remote location, with locking

```
# Create a new instance of the latest
Ubuntu 16.10 on an
# t2.micro node with an AWS Tag naming
it "HelloWorld"

resource "aws_instance" "web" {
  ami           = "ami-1d4e7a66"
  instance_type = "t2.micro"

  tags {
    Name = "HelloWorld"
  }
}
```

# Terraform workflow

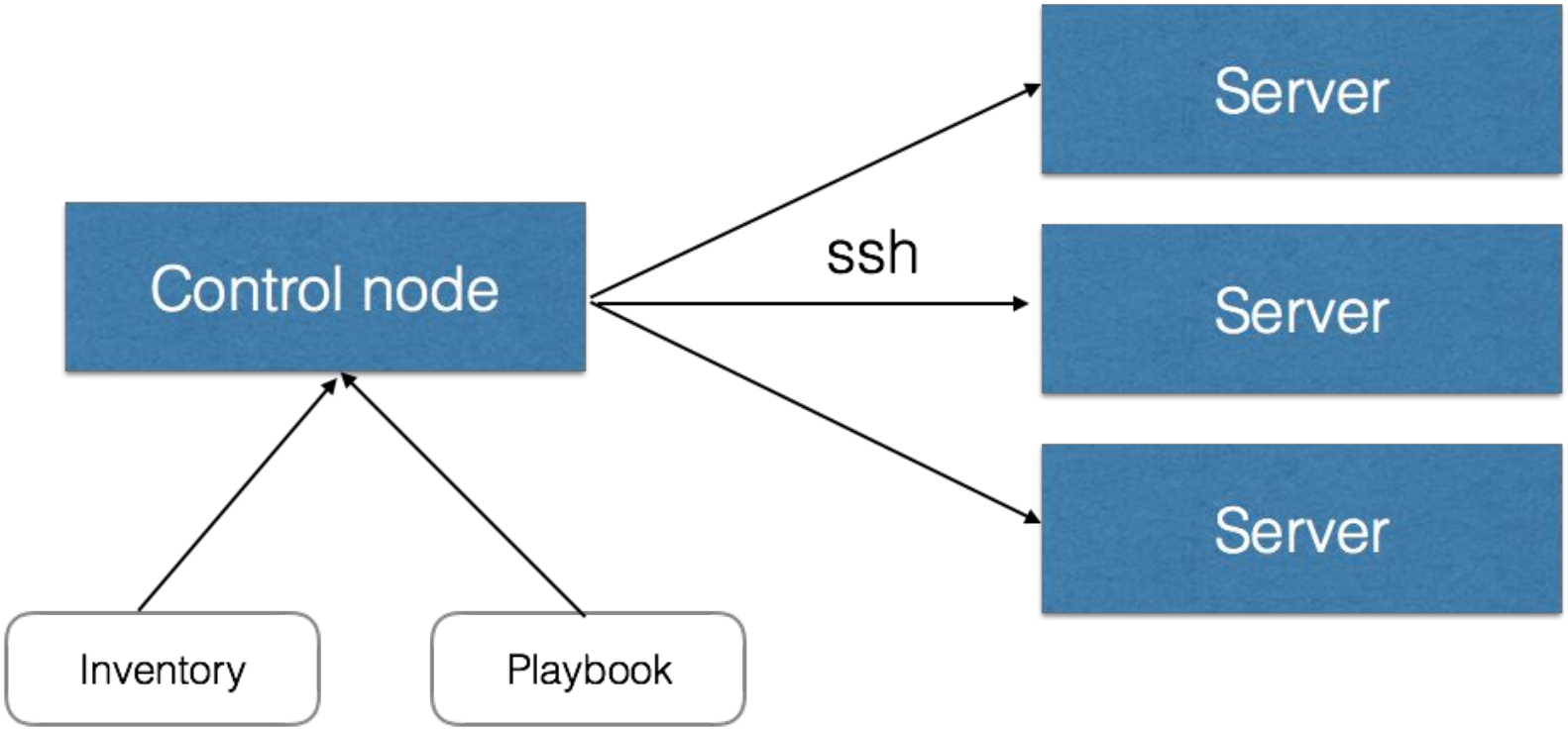
1. `$ terraform plan`
2. `$ terraform apply`

```
$ terraform plan
aws_elb.web: Refreshing state... (ID:
terraform-example-elb)

...  # disclaimers and more info

+ aws_instance.web
    ami: "ami-1d4e7a66"
    associate_public_ip...: "<computed>"

...  # more info abt the resource

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

# Ansible facts

- Mature tool
- Open source
- Backed by Red Hat
- Written in Python
- Agentless
- Lots and lots of modules

# Ansible: How does it work?



# Ansible: Playbook

- Written in pure YAML
- Describes a desired state
- Supports conditionals and loops
- Human readable
- Supports error handling

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest
version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2
dest=/etc/httpd.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
```

# Ansible: Inventory

- Static: A file with a list of servers
- Dynamic: Pulls server list from cloud provider
- AWS and other clouds are supported as first class citizens
- Customizable

```
[atlanta]
host1
host2

[raleigh]
host2
host3

[southeast:children]
atlanta
raleigh

[southeast:vars]
some_server=foo.southeast.example.com
escape_pods=2

[usa:children]
southeast
northeast
southwest
northwest
```


LET'S DO
A DEMO
memegenerator.net

# Ansible: how it's used in the real world

```
group_vars/
   group1                 # group specific vars
host_vars/
   hostname1              # host specific vars
site.yml                  # master playbook - initial setup
deploy.yml                # executed just for deploy
roles/
    web/                  # role name
        tasks/            #
            main.yml      #  <-- main.yml can include other files
        handlers/         #
            main.yml      #  <-- handlers go here
        templates/        #  <-- templates for files
        files/            #  <-- files that we just copy
        vars/             #
            main.yml      #  <-- role level variables
        defaults/         #
            main.yml      #  <-- default values for role vars
        meta/             #
            main.yml      #  <-- defines dependencies between roles
```

# Terraform: how it's used in the real world

```
modules/
   web/                        # this web module can be configured and reused
      main.tf
      variables.tf
      outputs.tf
      policy_template.json
environments/
   alpha/                      # environment specific settings, less code replication
     main.tf
     webservers.tf
     databases.tf
   production/
```

# Making servers smarter

## Make servers apply the ansible playbooks and download current code on boot

```
# apt get basic libraries
apt-get update -qq
apt-get install -yqq git-core python-dev python-virtualenv python-pip awscli
build-essential libssl-dev libffi-dev
# get latest release
aws --region us-east-1 s3 cp s3://pycon-alpha-releases/devops/$(aws --region us-east-1 s3
ls s3://pycon-alpha-releases/devops/ | awk '{print $4}' | tail -n1) /tmp/devops.tar.gz
cd /tmp; tar -xzf devops.tar.gz
# install devops ansible folder reqs
pip install -qr /tmp/devops/ansible/requirements.txt
# execute site and deploy playbooks
cd /tmp/devops/ansible; EC2_INI_PATH="/tmp/devops/ansible/ec2.local.ini" ansible-playbook
-i ec2.py site.yml --connection=local --limit `ec2metadata
--local-ipv4`,localhost,127.0.0.1 -e user_data=yes
cd /tmp/devops/ansible; EC2_INI_PATH="/tmp/devops/ansible/ec2.local.ini" ansible-playbook
-i ec2.py deploy.yml --connection=local --limit `ec2metadata
--local-ipv4`,localhost,127.0.0.1 -e user_data=yes
```

# Thanks a lot!
# Questions?


**Slides and code samples are available at:**
**https://github.com/ezegolub/terraform_and_ansible_for_django**