TERRAFORM

INFRASTRUCTURE AS CODE

TERRAFORM

INFRASTRUCTURE AS CODE

Slides and example code @

github.com/jdforsythe/presentations/terraform-iac-basic

http://bit.ly/hackysu-terraform

Jeremy Forsythe

Director of Software Engineering

SENSOURCE

https://github.com/jdforsythe

https://linkedin.com/in/jdforsythe

jdforsythe@gmail.com


WHAT'S
YOUR
PROBLEM?


JASSY BE LIKE...
aws
IS EASY!

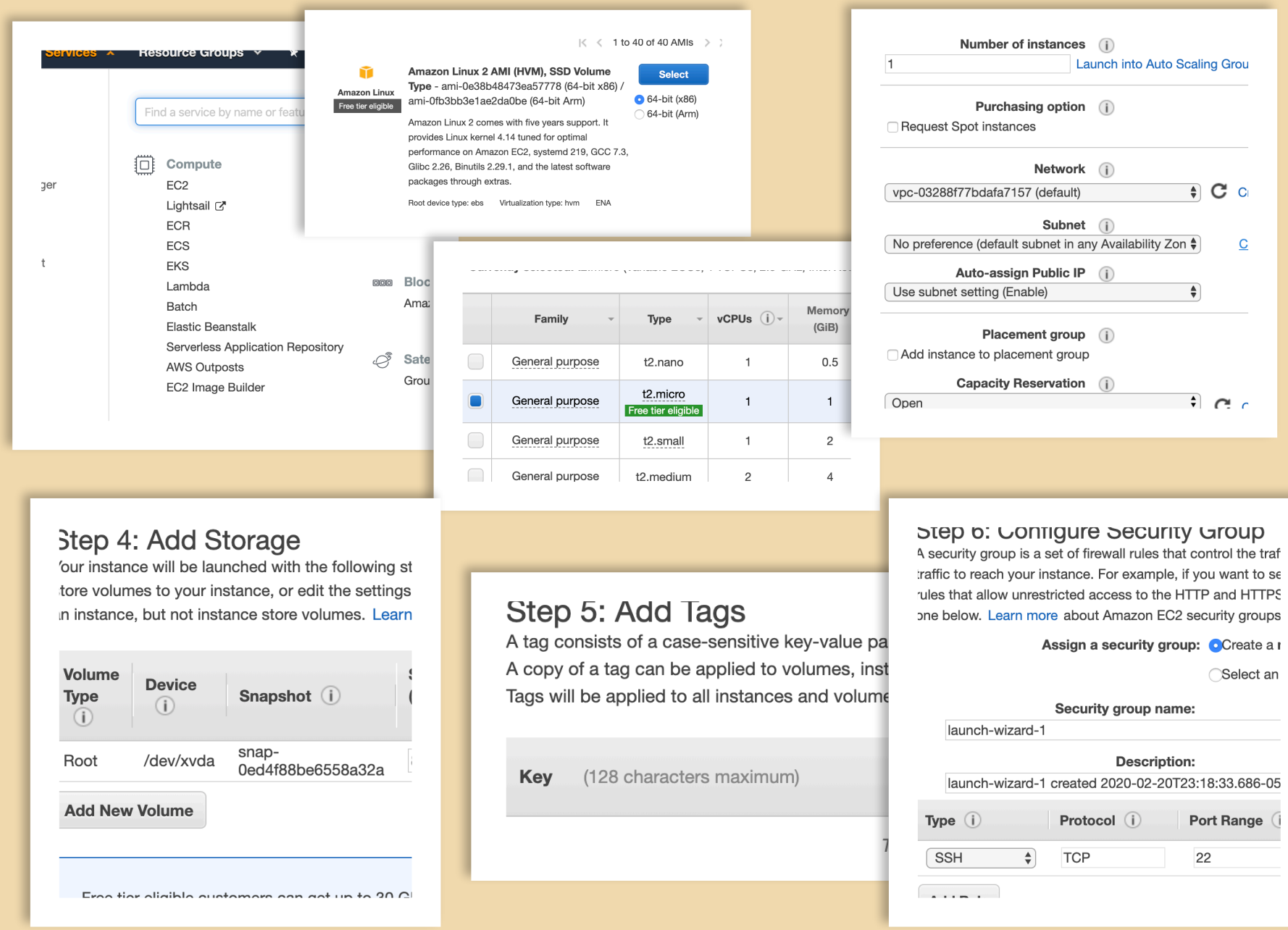
How to draw an owl

1.

1. Draw some circles

2.

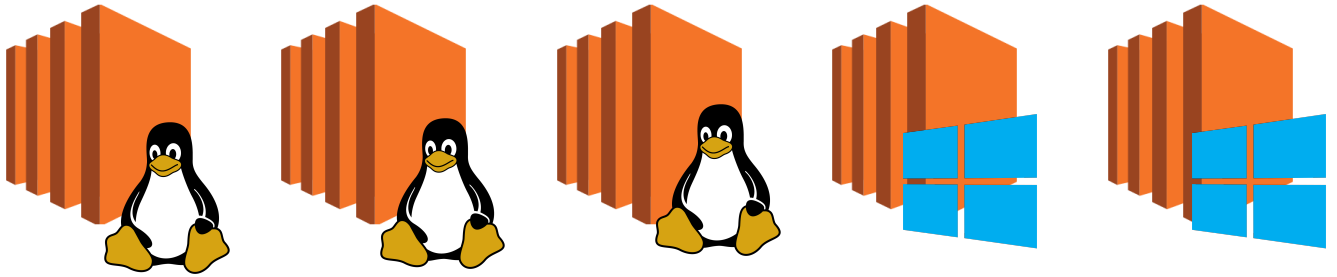
2. Draw the rest of the owl



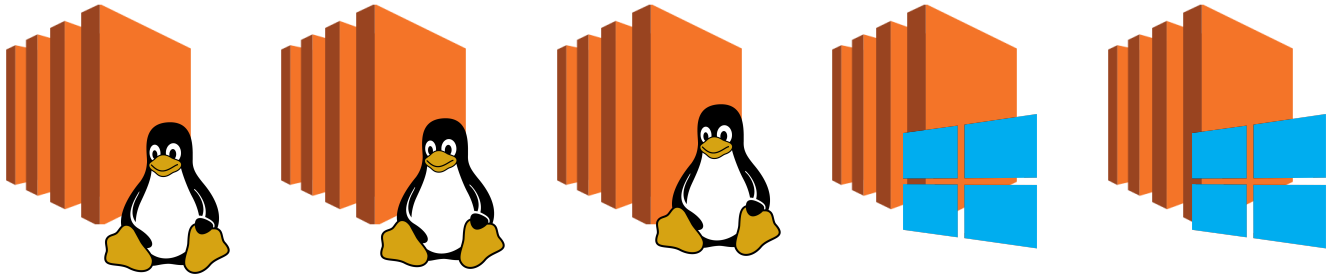
Creating a Single VM:

7 steps

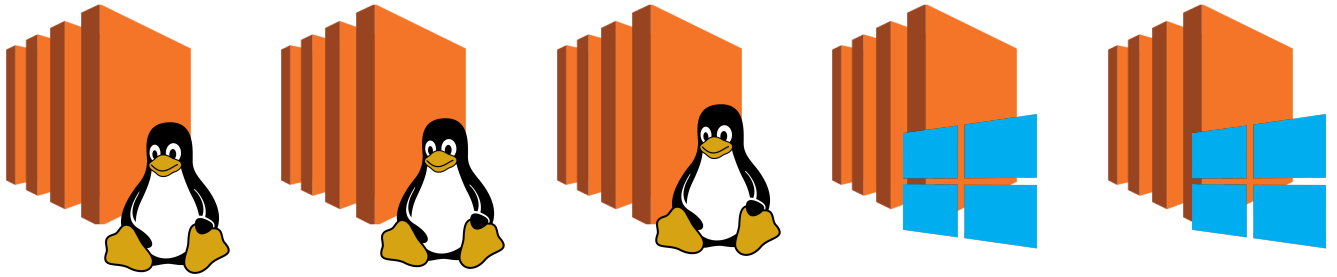
Dozens of options



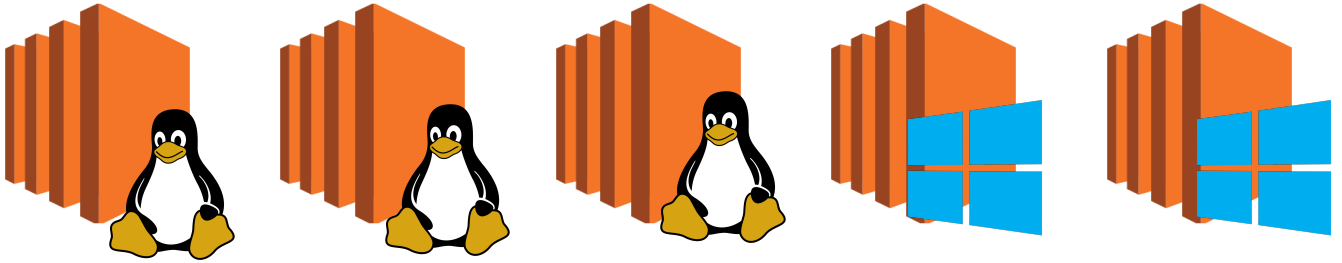
Staging



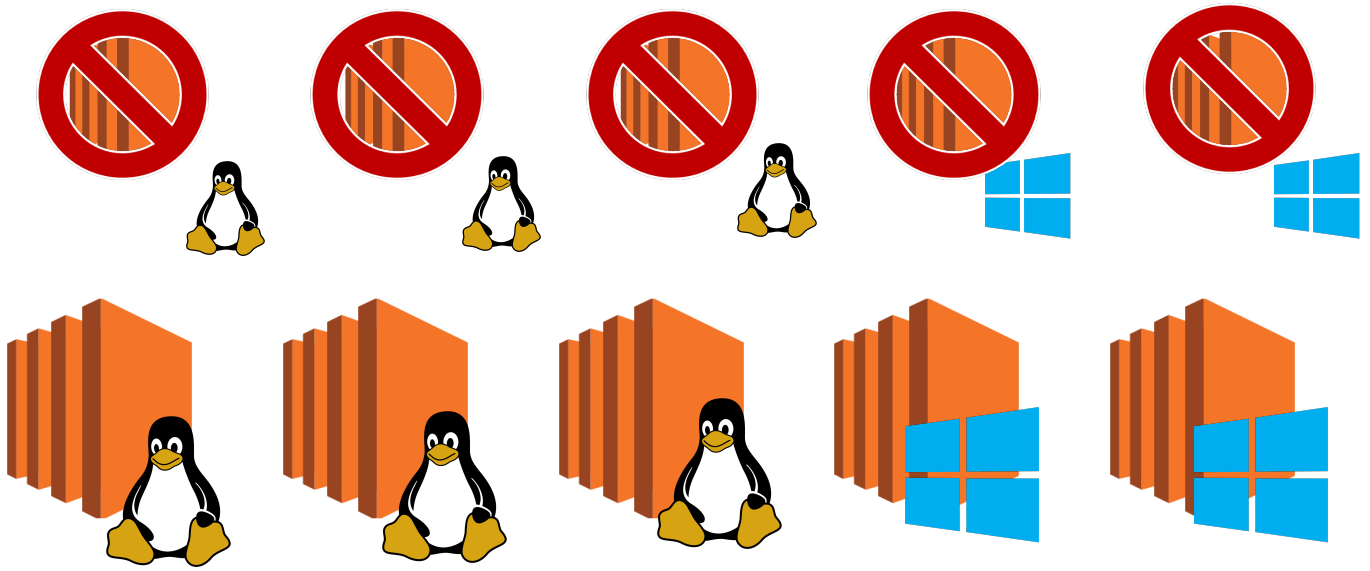
Staging



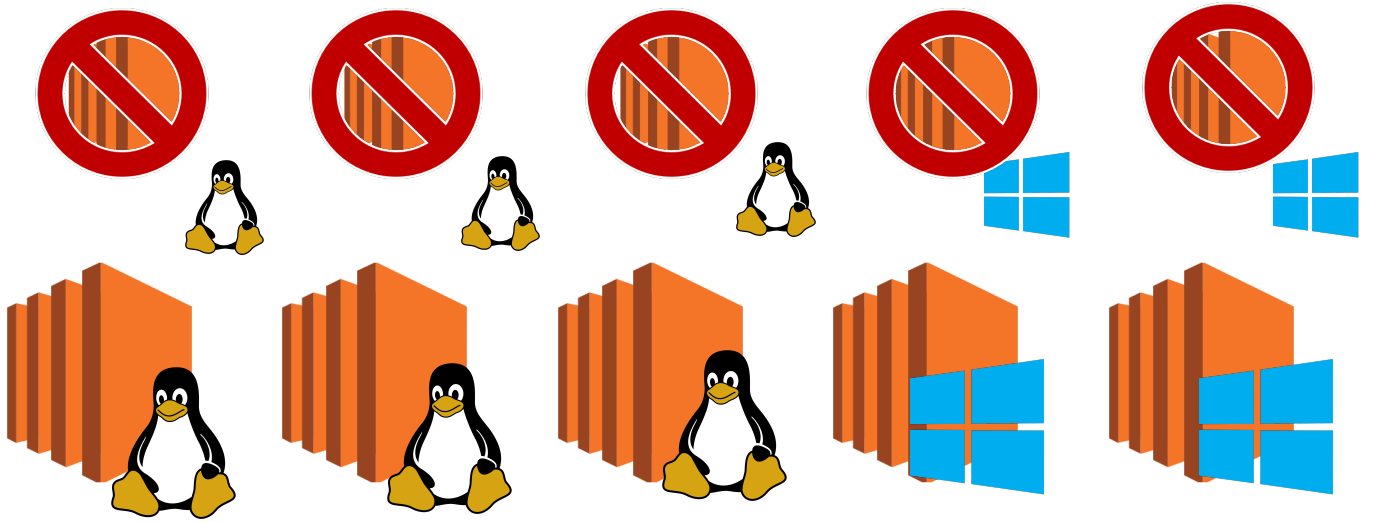
Production

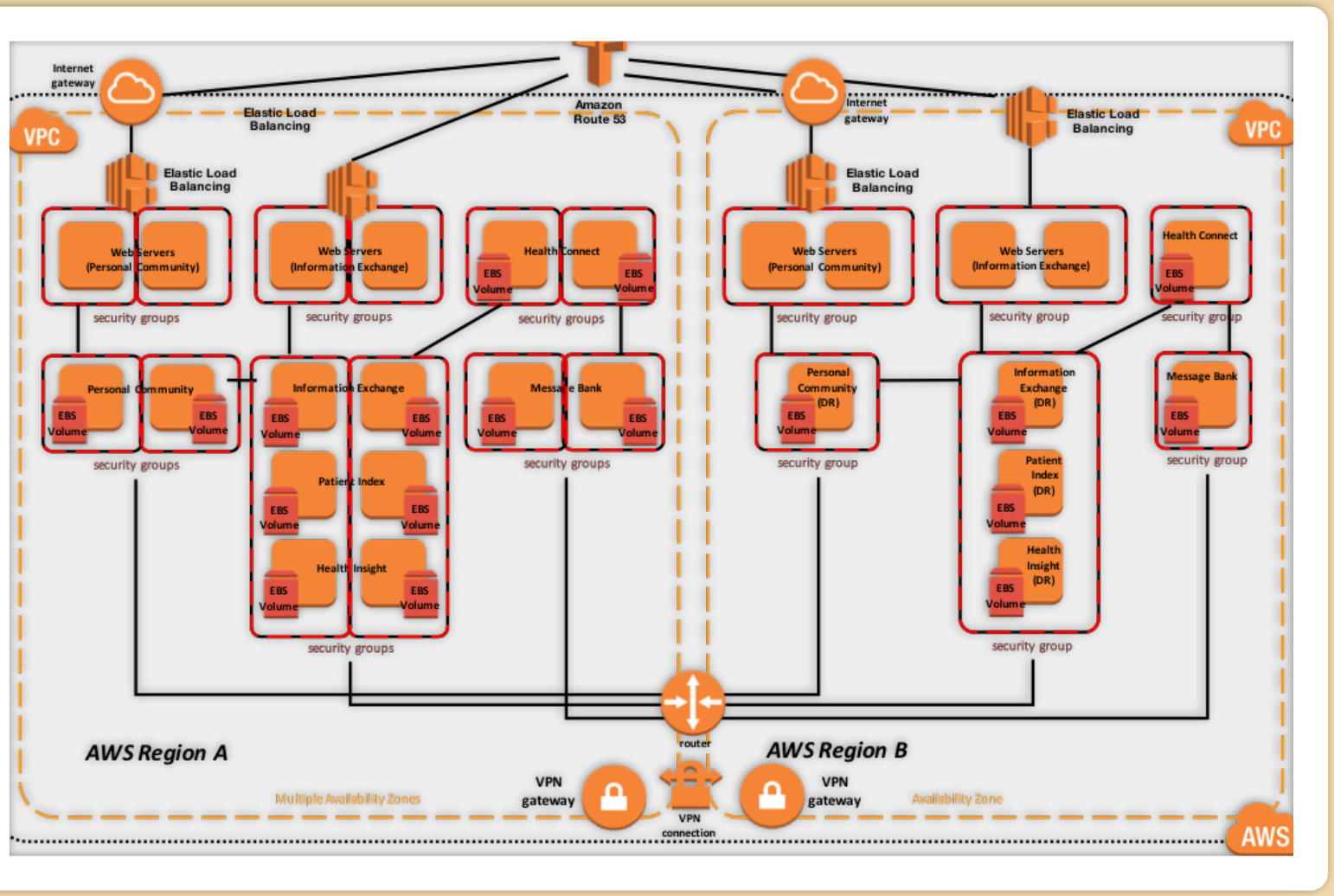


Staging



Production





Real infrastructure is much more complicated and redundant

High availability * High reliability * High resiliency * High scalability

Infrastructure Changes Over Time

Day 1	***Provision original infrastructure***
Day 2-n	***Add, upgrade, destroy infrastructure***
Day X	***Shut down - destroy***

PROVISIONING INFRASTRUCTURE

- Tedious
- Error-prone
- Difficult to track
- Difficult to change
- Not easily reproducible
- Multi-cloud? Multi-problems

INFRASTRUCTURE AS CODE

Imperative

- Step-by-step
- Shell scripts
- Idempotent or very complicated
- Error-prone
- Better than nothing?
- Custom code

Declarative

- Describe desired state
- Configuration files
- Tool makes necessary changes
- Easy to track state
- Easy to collaborate on changes
- Custom configuration

Provider-Native Infrastructure Automation

How many DSLs do you want to learn?

How many APIs do you want to learn?

Terraform

What is it? What does it do?

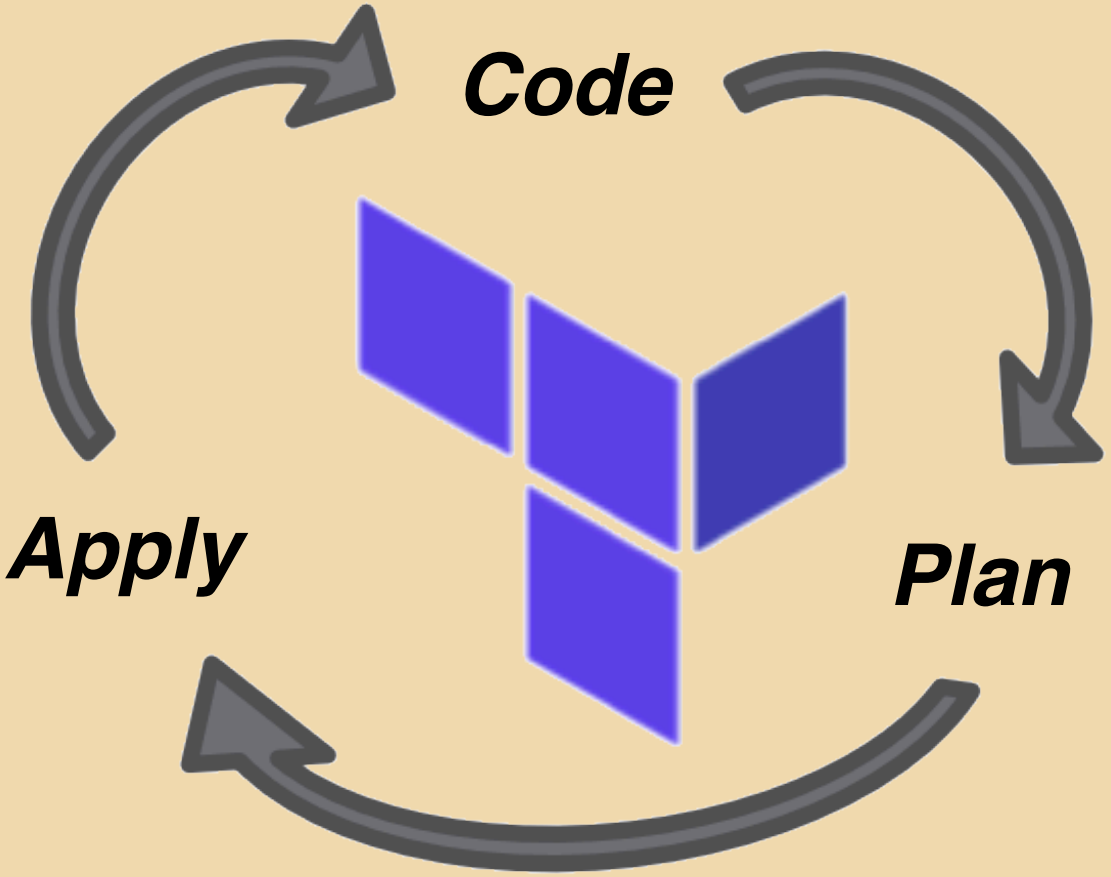
- Tool for ***building***, ***changing***, and ***versioning*** infrastructure
- Uses declarative configuration files that can be checked into VCS
- Keeps track of state – makes only necessary changes
- Allows reproducible provisioning even across multiple providers
- Allows using reusable ***modules*** for consistency

What does it not do?

- Does not teach you cloud architecture
- Does not make you a better cloud architect
- Does not **enforce** best practices
- Does not stop you from ***destroying your entire infrastructure***

Terraform

Basic Workflow



Installing

```
~$ brew install terraform
```

Create repository

```
~$ mkdir ~/dev/terraform-demo && cd ~/dev/terraform-demo

~/dev/terraform-demo$ git init
Initialized empty Git repository in /Users/username/dev/terraform-demo/.git/

~/dev/terraform-demo$ code .
```

Simple example *

```
provider "aws" {
  version = "~> 2.0"
  region = "us-east-2"

  # use AWS credentials from ~/.aws/credentials
  profile = "default"

  # NEVER put your credentials in the repository!
}

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

** Do not use this example for production – **never** deploy to the default VPC*

There is no substitute for knowing AWS best practices

Initialize providers

```
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 2.50.0...

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.
```

Terraform View vs Desired Config vs Real World

Refresh	***TF View***	⟷	***Real World***
Plan	***Real World***	⟷	***Desired Config***
Apply	***Plan***	⟷	***Real World***
Destroy	***Plan***	⟷	***Real World***

Plan

```
$ terraform plan
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.

------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.example will be created
  + resource "aws_instance" "example" {
      + ami                          = "ami-2757f631"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)

  ...


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

Apply

```
$ terraform apply
data.aws_ami.ubuntu: Refreshing state...

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.example will be created
  + resource "aws_instance" "example" {
      + ami                          = "ami-2757f631"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)

  ...


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

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

Name	Instance ID	Instance Type	Availability Zone	Instance State	Stat
	i-05b682b45d2cd9e8b	t2.micro	us-east-2c	pending	

Destroy

```
$ terraform destroy
data.aws_ami.ubuntu: Refreshing state...

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.example will be destroyed
  - resource "aws_instance" "example" {
      - ami                          = "ami-2757f631"
      - arn                          = (known after apply)
      - associate_public_ip_address  = (known after apply)

  ...


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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to approve.

  Enter a value: yes
```

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Che
app	i-03184767b42de1c14	t2.micro	us-east-2c	terminated	

Real World Workflow

Branch → ***Code*** → ***Plan*** → ***Apply*** → ***Apply***

from infrastructure repo

Commit changes

Export to file
Commit
Make PR

To staging env

Via automated process on merge

CI or GH Actions

Run tests

To production

Automated process on release tag

CI or GH Actions

120+ Official Providers

- Major and minor cloud providers
  - AWS, Azure, GCP, Alibaba, Digital Ocean
- On-Prem
  - VMWare, OpenStack
- PaaS
  - Heroku, K8S, Lambda
- SaaS
  - Datadog, Fastly, GitHub
- Resources / Misc
  - PostgreSQL, InfluxDB, Netlify, TLS

Awesome Ecosystem

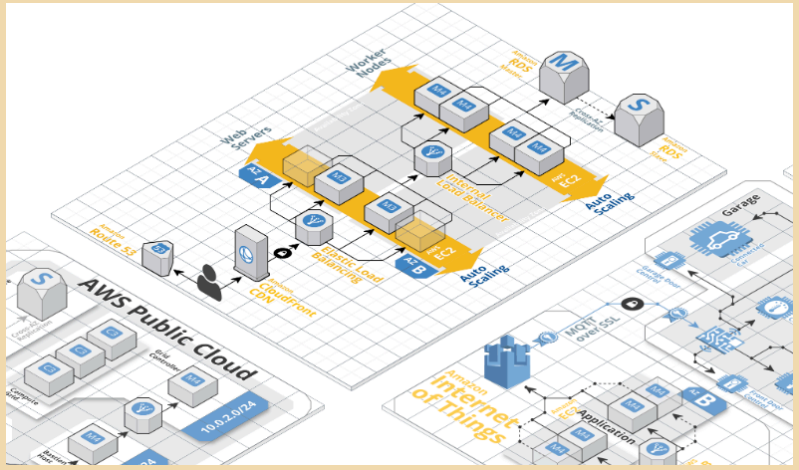
Hundreds of easy-to-use modules ready to use

Awesome Ecosystem

Many additional tools for complicated use cases

Awesome Ecosystem

+



Cloudcraft: Draw architecture diagrams

modules.tf: Generate Terraform config from diagrams

LIVE DEMO

We are hiring Junior Developers

www.sensourceinc.com

Jeremy Forsythe

Director of Software Engineering

SENSOURCE

https://github.com/jdforsythe

https://linkedin.com/in/jdforsythe

jdforsythe@gmail.com

TERRAFORM

INFRASTRUCTURE AS CODE