AUTOMATING AWS INFRASTRUCTURE CREATION WITH TERRAFORM

AGENDA

- Why Infrastructure as Code?
- Terraform introduction
- Provisioning AWS with Terraform

WHY DO WE NEED INFRASTRUCTURE AS CODE?

WHAT IS REQUIRED TO DELIVER YOUR CODE TO THE CUSTOMER?

SO WHAT'S THE PROBLEM?

SO WHAT'S THE PROBLEM?

- Time consuming

SO WHAT'S THE PROBLEM?

- Time consuming
- Error prone

SO WHAT'S THE PROBLEM?

- Time consuming
- Error prone
- Configuration
  Drift

WHAT IS INFRASTRUCTURE AS CODE?

INFRASTRUCTURE AUTOMATION

… AS CODE

"When we compared high performers to low performers, we found that high performers are doing significantly less manual work" - State of DevOps

"By performing operations as code, you limit human error and enable consistent responses to events." - AWS

INFRASTRUCTURE AS CODE IS A FUNDAMENTAL PART OF DEVOPS

INFRASTRUCTURE AS CODE IS A FUNDAMENTAL PART OF DEVOPS

- Culture
- **AUTOMATIO N**
- Lean
- Measurement
- Sharing

WHEN IS SOFTWARE "DONE"?

YOU AREN'T DONE UNTIL YOU DELIVER IAC!

HashiCorp

Terraform

EXAMPLE APPLICATION STACK

EXAMPLE APPLICATION STACK

- app.war

EXAMPLE APPLICATION STACK

- app.war
- Tomcat

EXAMPLE APPLICATION STACK

- app.war
- Tomcat
- Ubuntu

EXAMPLE APPLICATION STACK

- app.war
- Tomcat
- Ubuntu
- Virtual machine

EXAMPLE APPLICATION STACK

- app.war
- Tomcat
- Ubuntu
- Virtual machine
- Infrastructure: network, load balancer etc

TERRAFORM IS "CLOUD AGNOSTIC"

HASHICORP CONFIGURATION LANGUAGE

TERRAFORM-PROVIDERS-AWS

```
provider "aws" {
  region = "eu-central-1"
}
```

Resource

```
resource "aws_ecr_repository" "ecr" {
    name = "acme-business-portal"
}
```

Resource

```
resource "aws_ecr_repository" "ecr" {
    name = "acme-business-portal"
}
```

Resource

```
resource "aws_ecr_repository" "ecr" {
    name = "acme-business-portal"
}
```

AWS Provider

The Amazon Web Services (AWS) provider is used to interact with the many resources supported by AWS provider needs to be configured with the proper credentials before it can be used.

Use the navigation to the left to read about the available resources.

Example Usage

```
# Configure the AWS Provider
provider "aws" {
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
  region     = "us-east-1"
}

# Create a web server
resource "aws_instance" "web" {
  # ...
}
```

Complete configuration

```
provider "aws" {
  region = "eu-central-1"
}

resource "aws_ecr_repository" "ecr" {
    name = "acme-business-portal"
}
```

DEMO?!

Interpolation syntax

```
"${}"
```

Variables

```
variable "region" {
  type    = "string"
  default = "eu-central-1"
}
```

Variables

```
variable "region" {
  type    = "string"
  default = "eu-central-1"
}
```

Variables

```
variable "region" {
  type    = "string"
  default = "eu-central-1"
}
```

Setting variable

```
$ TF_VAR_region=eu-west-1 terraform apply
```

Using a variable

```
provider "aws" {
  region = "${var.region}"
}
```

IMPLICIT DEPENDENCY

```
resource "aws_vpc" "foo" {
  cidr_block = "198.18.0.0/16"
}
```

IMPLICIT DEPENDENCY

```
resource "aws_vpc" "foo" {
  cidr_block = "198.18.0.0/16"
}
```

```
resource "aws_subnet" "bar" {
  vpc_id     = "${aws_vpc.foo.id}"
  cidr_block = ...
}
```

EXPLICIT DEPENDENCY

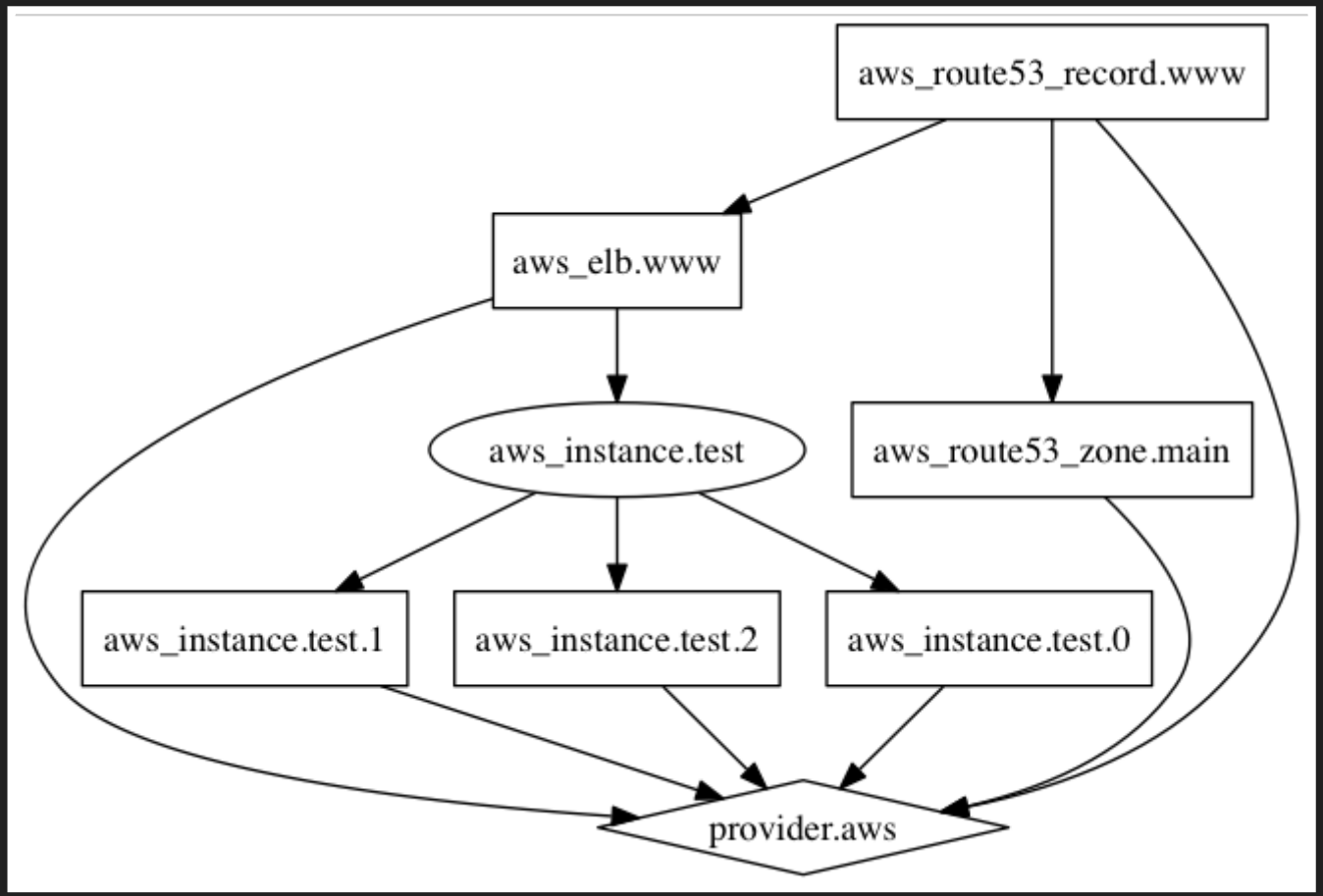
EXPLICIT DEPENDENCY

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

  depends_on = ["aws_s3_bucket.example"]
}
```

RESOURCE GRAPH

RESOURCE GRAPH



RUNNING TERRAFORM IN AUTOMATION

https://github.com/oscr/circleci-terraform-aws

TERRAFORM AND STATE

terraform.tfstate

```
terraform {
  backend "s3" {
```

```
terraform {
  backend "s3" {
```

```
bucket = "circle-terraform-state"
```

```
terraform {
  backend "s3" {
```

```
bucket = "circle-terraform-state"
```

```
key    = "terraform.tfstate"
```

```
terraform {
  backend "s3" {
```

```
bucket = "circle-terraform-state"
```

```
key    = "terraform.tfstate"
```

```
    region = "eu-west-1"
  }
}
```

CIRCLECI CONFIGURATION

```
docker:
  - image: hashicorp/terraform:light
```

CIRCLECI CONFIGURATION

```
docker:
  - image: hashicorp/terraform:light
```

```
steps:
  - checkout
  - run:
    name: INIT
    command: >
      terraform init
      -input=false
      -backend-config='key='${CIRCLE_BRANCH}
```

CIRCLECI CONFIGURATION

```
- run:
    name: APPLY
    command: >
      terraform apply
      -input=false
```

AUTOMATING AWS INFRASTRUCTURE CREATION WITH TERRAFORM

Terraform workflow

Terraform workflow

```
$ terraform init

Initializing provider plugins...
- Checking for available provider plugins ...
- Downloading plugin for provider "aws" (1.18.0)...

(...)

* provider.aws: version = "~> 1.18"
```

Terraform workflow

```
$ 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.

(...)

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

Terraform workflow

```
$ terraform apply

Terraform will perform the following actions:

(...)


Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```