Pivotal

Shishir Amin
Solutions Architect
xetamus
samin@pivotal.io

What is Terraform

- ❖ IaaS deployment tool
- ❖ IaaS as code
- ❖ Similar to CloudFormation
- ❖ Version control and automate your infrastructure

HashiCorp Terraform

Why Terraform?

- Easily source control and version your infrastructure
- Create consistent repeatable and re-usable automation
- No more snowflakes!
- Not limited to a single IaaS provider or service
- Its Open-Source!

Terraform

Concepts

Terraform Concepts

Providers

https://www.terraform.io/docs/providers/index.html

- Generally an IaaS provider or some other service you want to provision an object for
  - AWS/GCP/Azure
  - Openstack/vSphere
  - MySQL, DNS, consul, Docker, Datadog, Terraform, etc...
- Responsible for understanding API interactions and exposing resources
- We'll focus on the AWS provider

Terraform Concepts

Example

https://www.terraform.io/docs/providers/aws/index.html

```
provider "aws" {
  access_key = "${var.access_key}"
  secret_key = "${var.secret_key}"
  region     = "us-east-2"
}
```

Terraform Concepts

Resources

- Individual objects that can be created by a provider
    - VPCs
    - Security Groups
    - VMs
    - Load Balancers
    - Etc...
- Consist of arguments (inputs) and attributes (outputs)
- Resource attributes can be referenced by other resources
- Arguments/attributes that are not passed in will be generated
- https://www.terraform.io/docs/providers/aws/d/vpc.html

Terraform Concepts

Example

```
resource "aws_vpc" "test" {
  cidr_block = "10.0.0.0/22"
  tags {
    Name = "myawsvpc"
  }
}
resource "aws_subnet" "test" {
  vpc_id = "${aws_vpc.test.id}"
  cidr_block = "10.0.0.0/24"
  tags {
    Name = "myawssubnet"
  }
}
resource "aws_internet_gateway" "test" {
  vpc_id = "${aws_vpc.test.id}"
}
```

Terraform Concepts

Variables

https://www.terraform.io/docs/configuration/variables.html

- Allows for inputs to be passed in
- Must be declared in a `*.tf` file to be passed in
- Allows for multiple data types
  - Strings
  - Lists
  - Maps
- Can be passed as environment variables, on the command line or through files
- `TF_VAR_<variable_name>`

Terraform Concepts

Variables

```
variable "key" {
  type = "string"
}
variable "images" {
  type = "map"

  default = {
    us-east-1 = "image-1234"
    us-west-2 = "image-4567"
  }
}
variable "zones" {
  default = ["us-east-1a", "us-east-1b"]
}
```

Providers

Resources

➔ Variables

Provisioners

Outputs

State

Pivotal

Terraform Concepts

Variables

```
$ terraform apply -var-file=foo.tfvars -var ‘foo=bar’
```

tfvars:

```
foo = "bar"
xyz = "abc"
somelist = [
  "one",
  "two",
]
somemap = {
  foo = "bar"
  bax = "qux"
}
```

Terraform Concepts

Provisioners

https://www.terraform.io/docs/provisioners/index.html

- Generally used to run some command or interact with instances
- Some examples include
  - Config management tools like chef or salt
    - chef, salt-masterless
  - Transferring files or running a command locally or remotely
    - file, local-exec, remote-exec
  - Setting up connections to remote
    - connection

Terraform Concepts

Provisioners

```
resource "aws_instance" "web" {
  # ...

  provisioner "local-exec" {
    command = "echo ${self.private_ip} > file.txt"
  }
}
```

Terraform Concepts

Outputs

https://www.terraform.io/docs/configuration/outputs.html

- Strings that can be generated and displayed, usually used to display resource attributes
- Secrets can be redacted using the `sensitive` argument
- Will be shown after a `terraform apply`
- Can be shown at any time with `terraform output`

```
output "address" {
  value = "${aws_instance.db.public_dns}"
}
output "sensitive" {
  sensitive = true
  value     = VALUE
}
```

Terraform Concepts

State

- A file containing a snapshot of the currently deployed infrastructure
- Source of truth
- Created when you run a `terraform apply` and can be created with a `terraform plan`
- Updated when there are changes performed
    - `terraform apply`
    - `terraform delete`

Terraform Files

`*.tf`

- Files containing your provider/resource/provisioner objects

`terraform.tfstate`

- A current snapshot of deployed resources
- Can be stored remotely with backends
- A backup of the previous statefile is saved to `terraform.tfstate.backup`

`terraform.tfvars`

- Default file for loading variables
- Can add other var files using the `-var-file` flag

Pivotal

Syntax

HCL

- JSON-like and interoperable with JSON
- https://www.terraform.io/docs/configuration/syntax.html
- https://github.com/hashicorp/hcl

Interpolation

- Allows referencing the attributes of other resources
- Many functions built-in for use
  - Data manipulation, math functions etc...
- https://www.terraform.io/docs/configuration/interpolation.html

Terraform

CLI

Creating Infrastructure

```
terraform init
```

- Terraform switched over to being more pluggable recently
- Downloads all of the plugins necessary
- Plugins are your providers and provisioners generally

Creating Infrastructure

`terraform plan`

- Performs a dry-run
- Gives an overview of proposed changes

`terraform apply`

- Creates the infrastructure laid out in your `*.tf` files
- Creates the `terraform.tfstate`file with the objects that were created
- Subsequent runs will show the any changes to the currently deployed infrastructure

`terraform show`

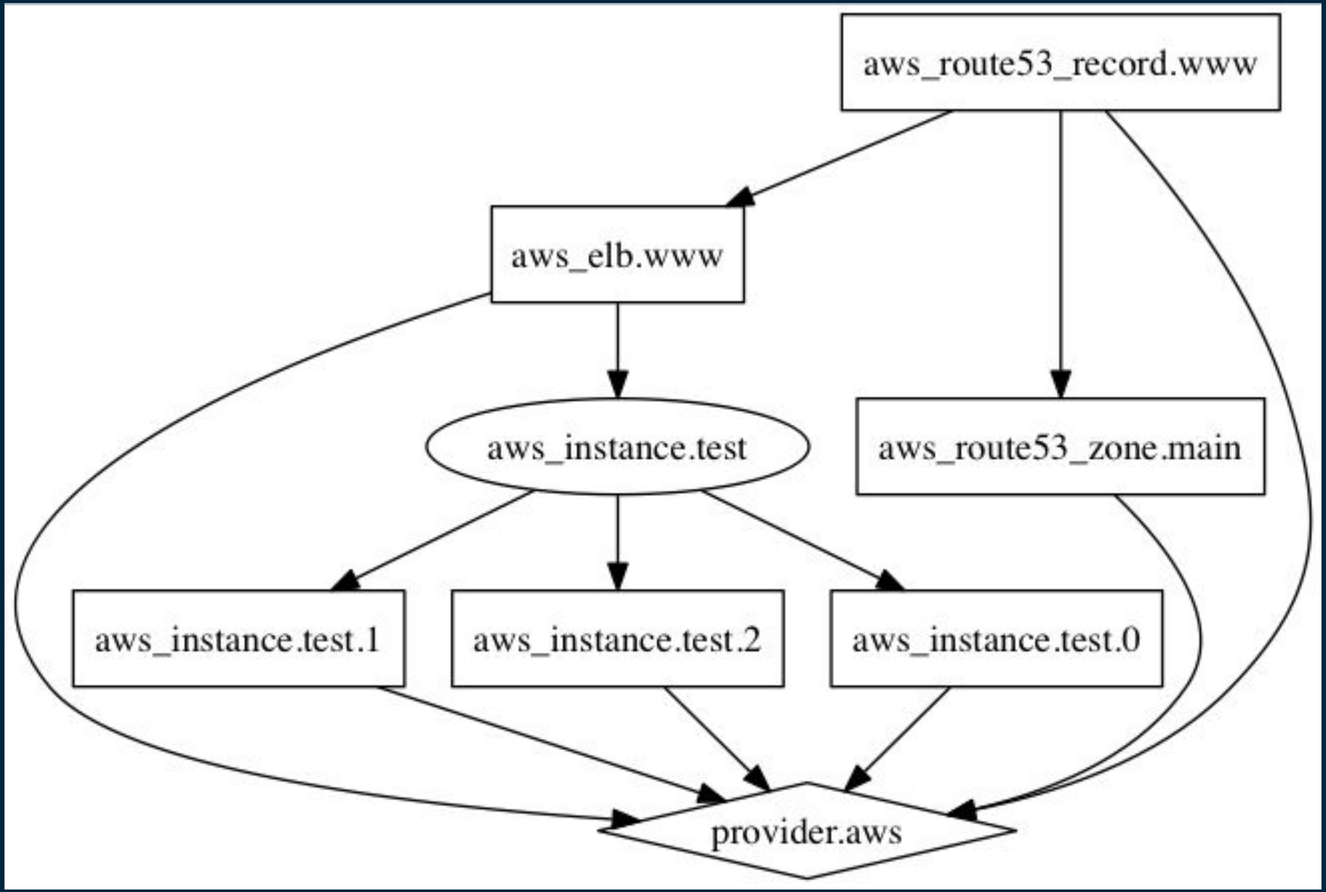
- Will output the attributes of your resources from the plan

Dependency Graphing

`terraform graph`

- Creates a dependency mapping of your providers/resources/provisioners
- Can you help with debugging
- Allows you to visualize how your components work together
- https://www.terraform.io/docs/commands/graph.html
- `terraform graph | dot -Tsvg > graph.svg`

Dependency Graphing



Tearing Down

```
terraform destroy
```

- Goes through in reverse order and tries to delete all objects
- Deletes based on what is in the statefile and the tf files
- Statefile will still reflect the current state of infra on failure

Terraform

Demo

https://github.com/xetamus/terraform-example

Pivotal

Questions?

Pivotal

Further Topics

- Destroy provisioners
- Tainting resources
  - https://www.terraform.io/docs/commands/taint.html
- Null resource
  - https://www.terraform.io/docs/provisioners/null_resource.html
- Modules
  - https://www.terraform.io/docs/configuration/modules.html
- Backends
  - https://www.terraform.io/docs/state/remote.html
- Explicit Dependencies
  - https://www.terraform.io/docs/configuration/resources.html#depends_on
- Data sources
  - https://www.terraform.io/docs/configuration/data-sources.html
- Overrides
  - https://www.terraform.io/docs/configuration/override.html

Pivotal

Terraform

THE DOCS ARE YOUR FRIENDS!

https://www.terraform.io/docs/index.html

Pivotal