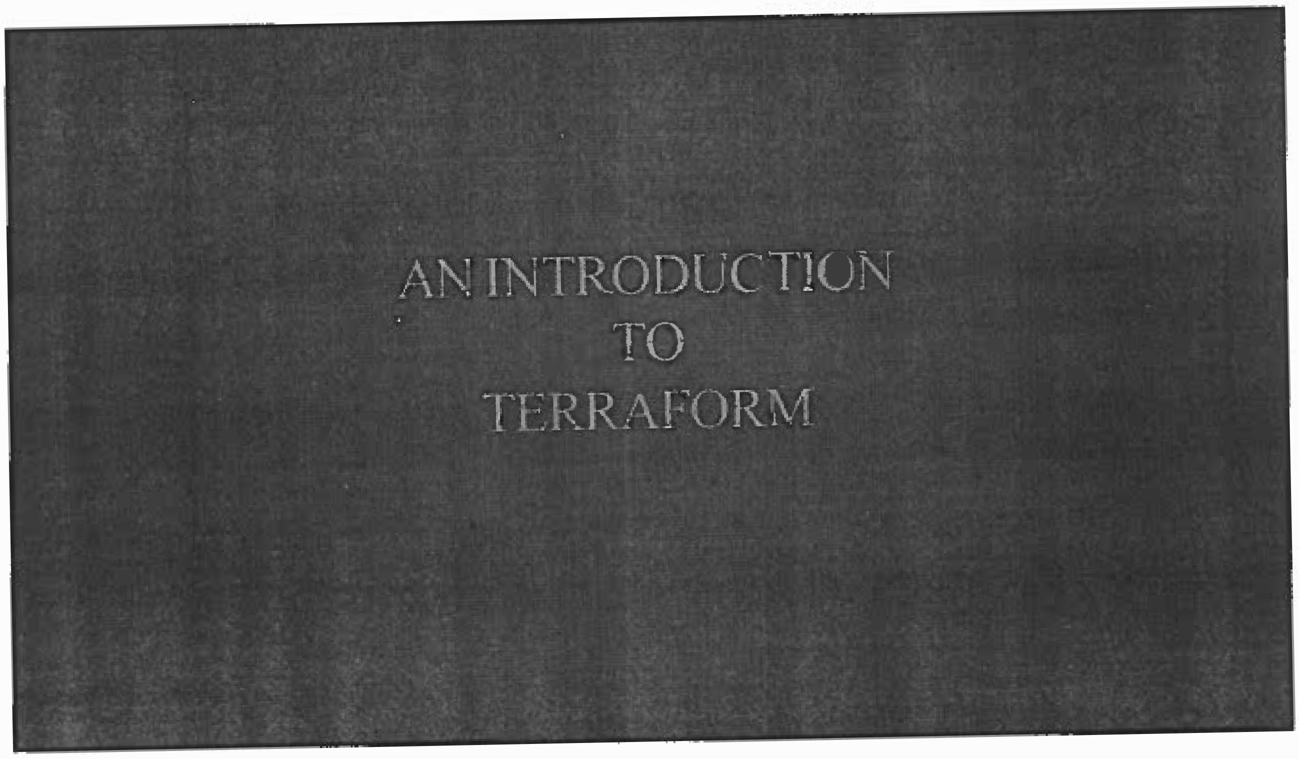
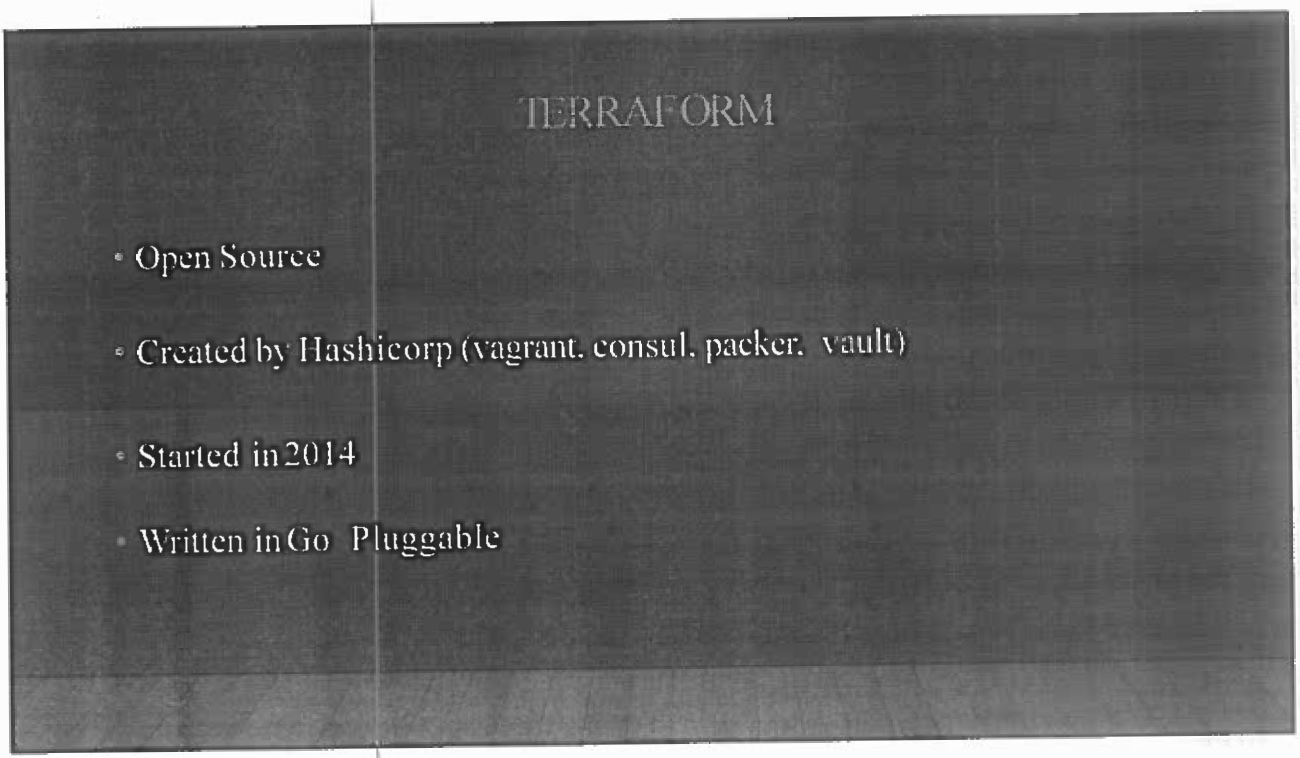
AWS-7



AN INTRODUCTION TO TERRAFORM

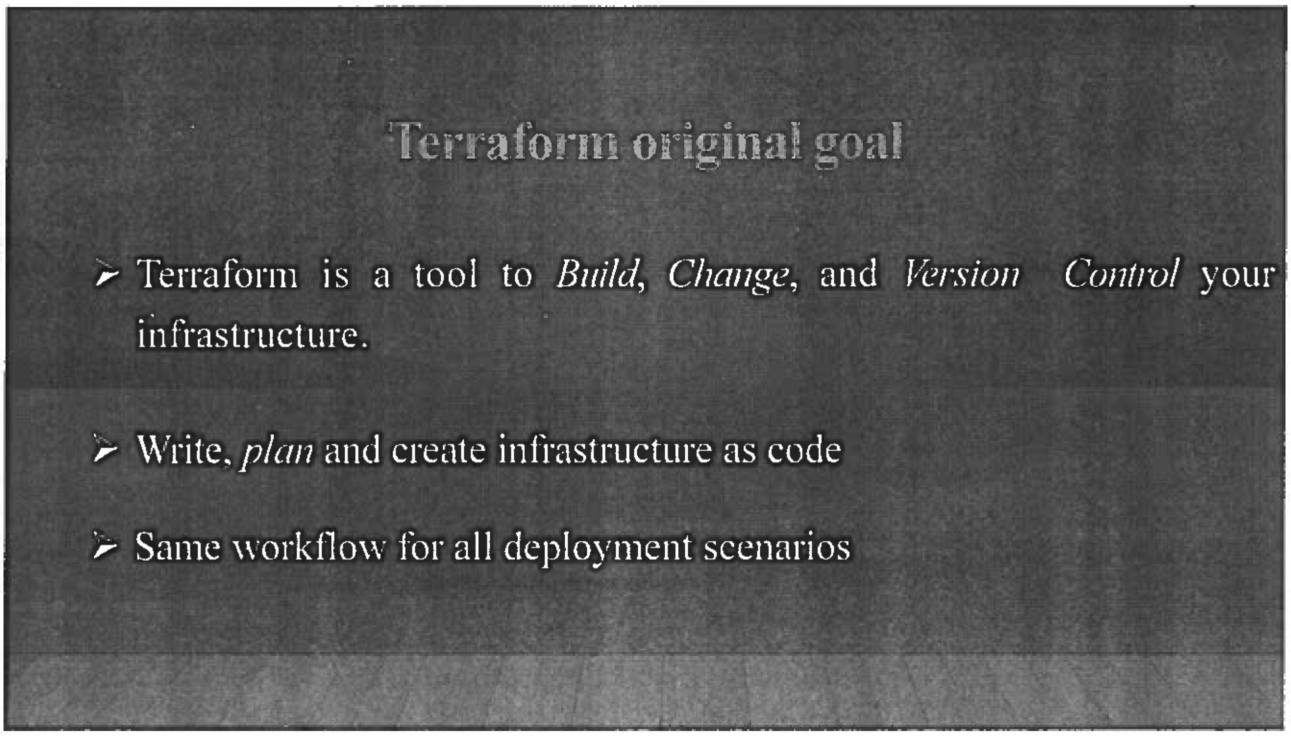
1



TERRAFORM

- Open Source
- Created by Hashicorp (vagrant. consul. packer. vault)
- Started in 2014
- Written in Go Pluggable

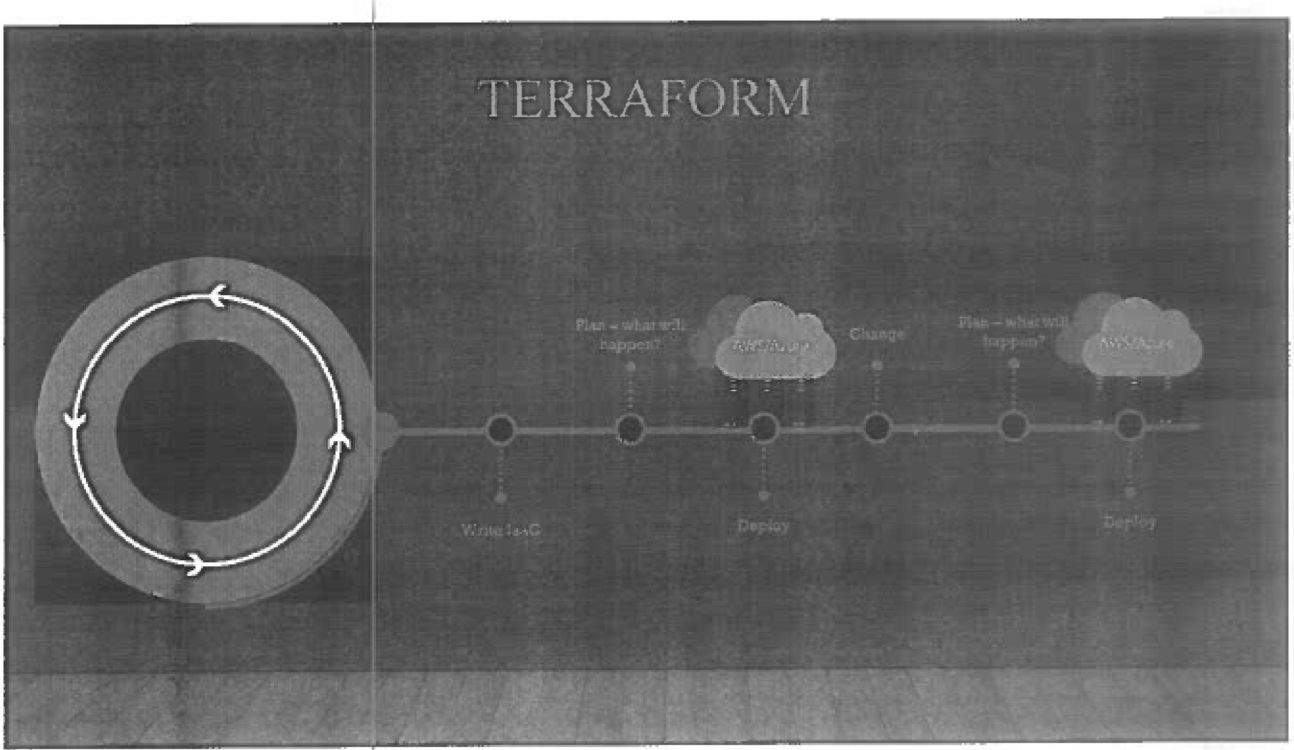
2



Terraform original goal

- ➢ Terraform is a tool to *Build*, *Change*, and *Version Control* your infrastructure.
- ➢ Write, *plan* and create infrastructure as code
- ➢ Same workflow for all deployment scenarios

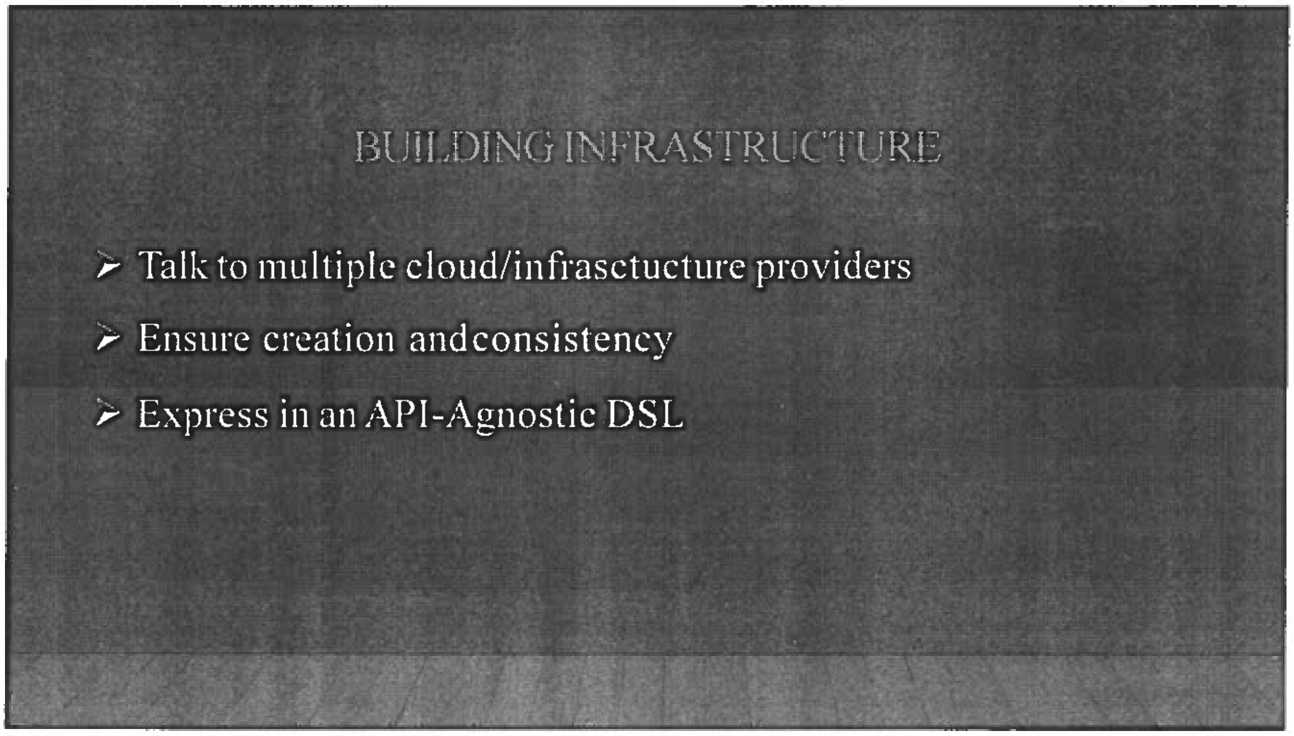
3



TERRAFORM

Write IaaC
Plan – what will happen?
AWS/Azure
Deploy
Change
Plan – what will happen?
AWS/Azure
Deploy

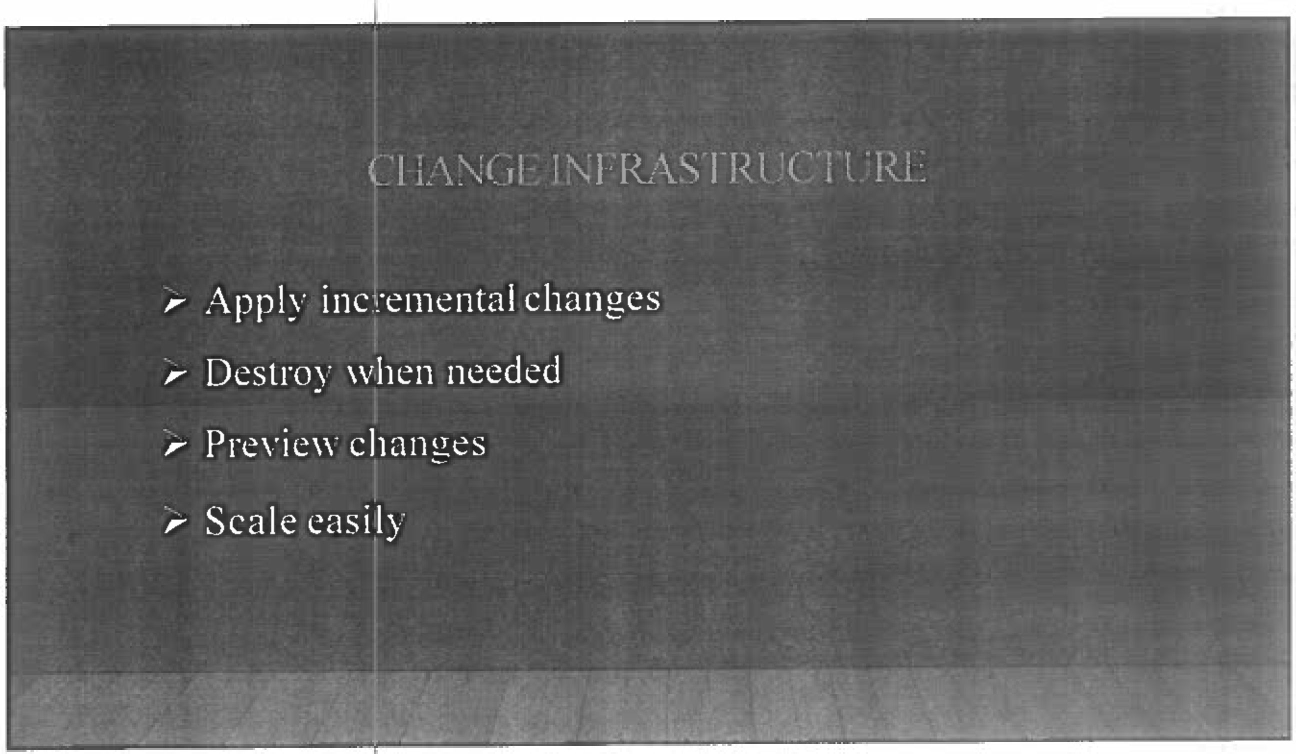
4



BUILDING INFRASTRUCTURE

- ➢ Talk to multiple cloud/infrasctucture providers
- ➢ Ensure creation andconsistency
- ➢ Express in an API-Agnostic DSL

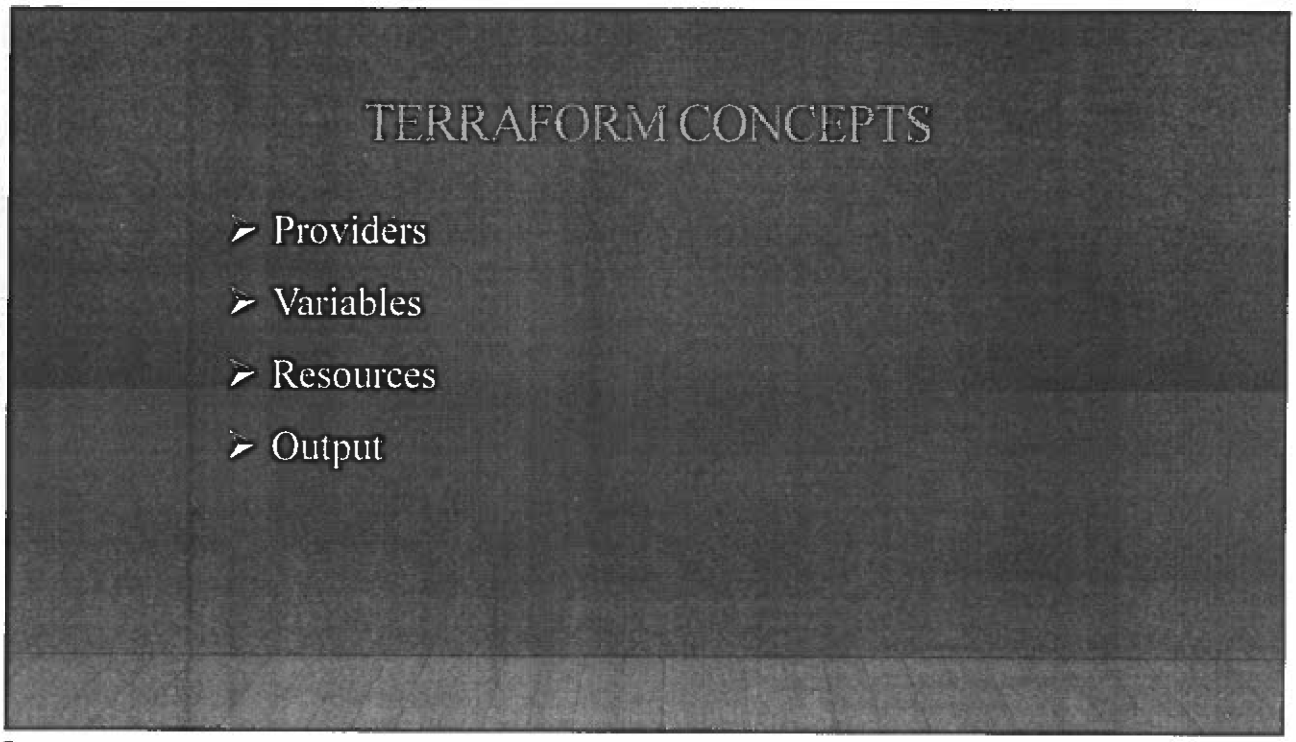
5



CHANGE INFRASTRUCTURE

- Apply incremental changes
- Destroy when needed
- Preview changes
- Scale easily

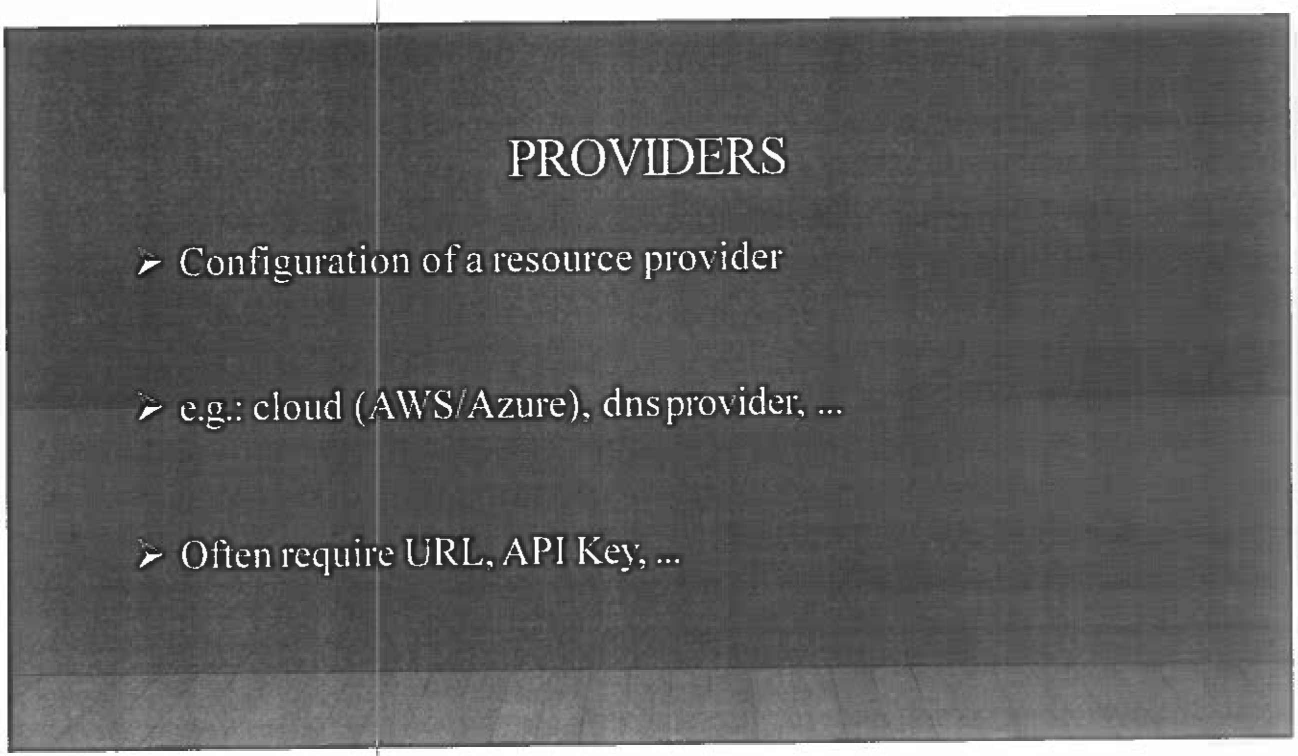
6



TERRAFORM CONCEPTS

- Providers
- Variables
- Resources
- Output

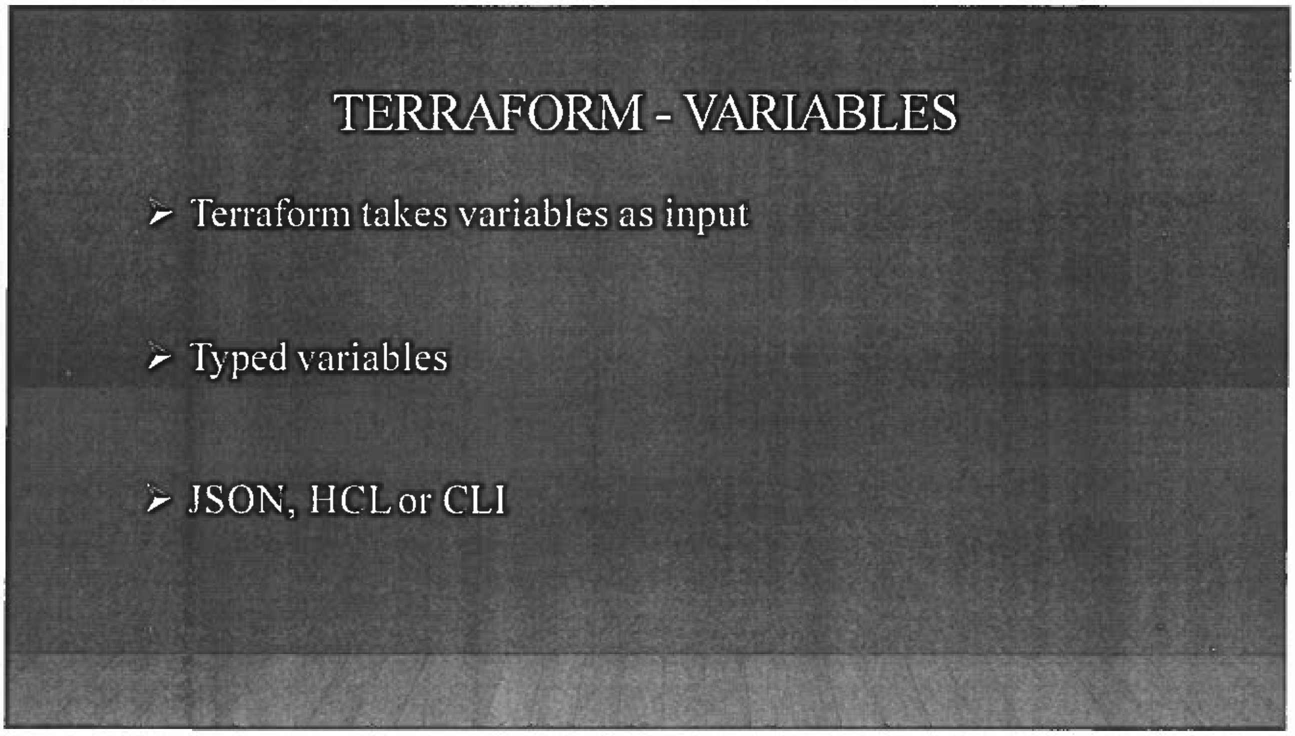
7



PROVIDERS

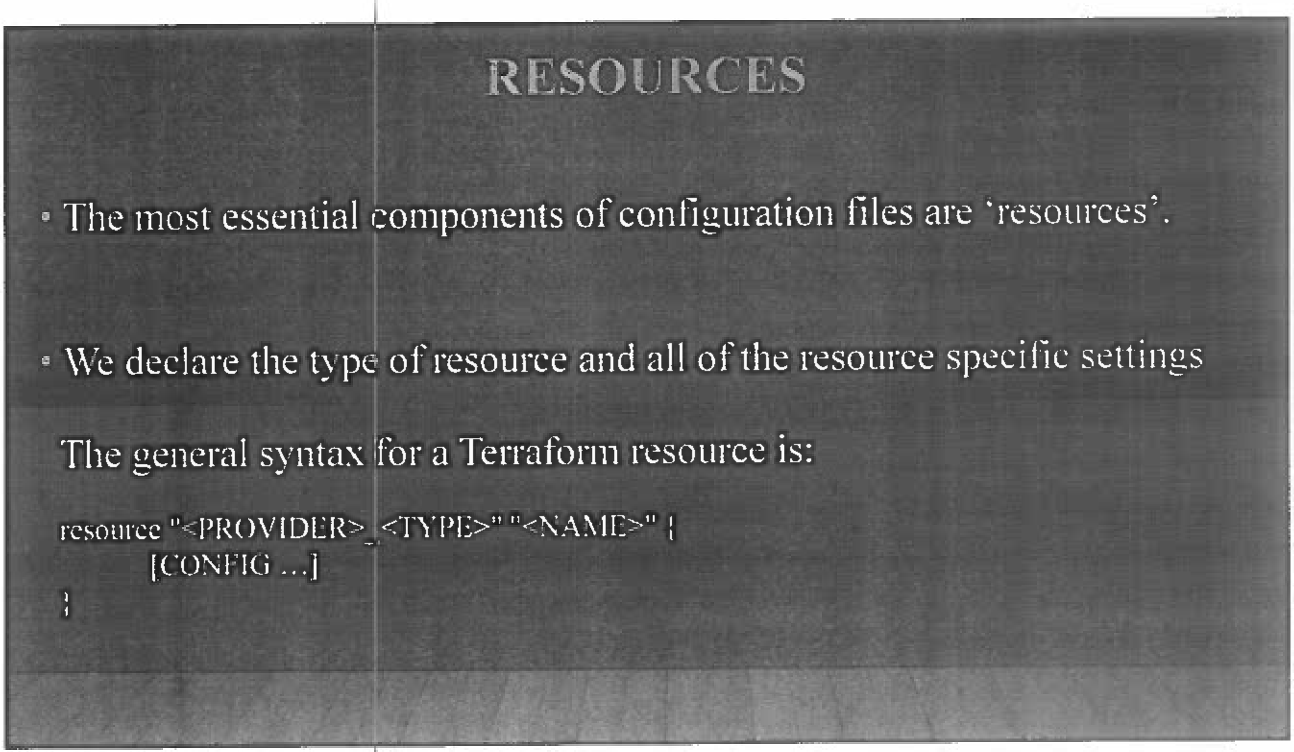
- ➢ Configuration of a resource provider
- ➢ e.g.: cloud (AWS/Azure), dns provider, ...
- ➢ Often require URL, API Key, ...

8



TERRAFORM - VARIABLES

- ➢ Terraform takes variables as input
- ➢ Typed variables
- ➢ JSON, HCL or CLI



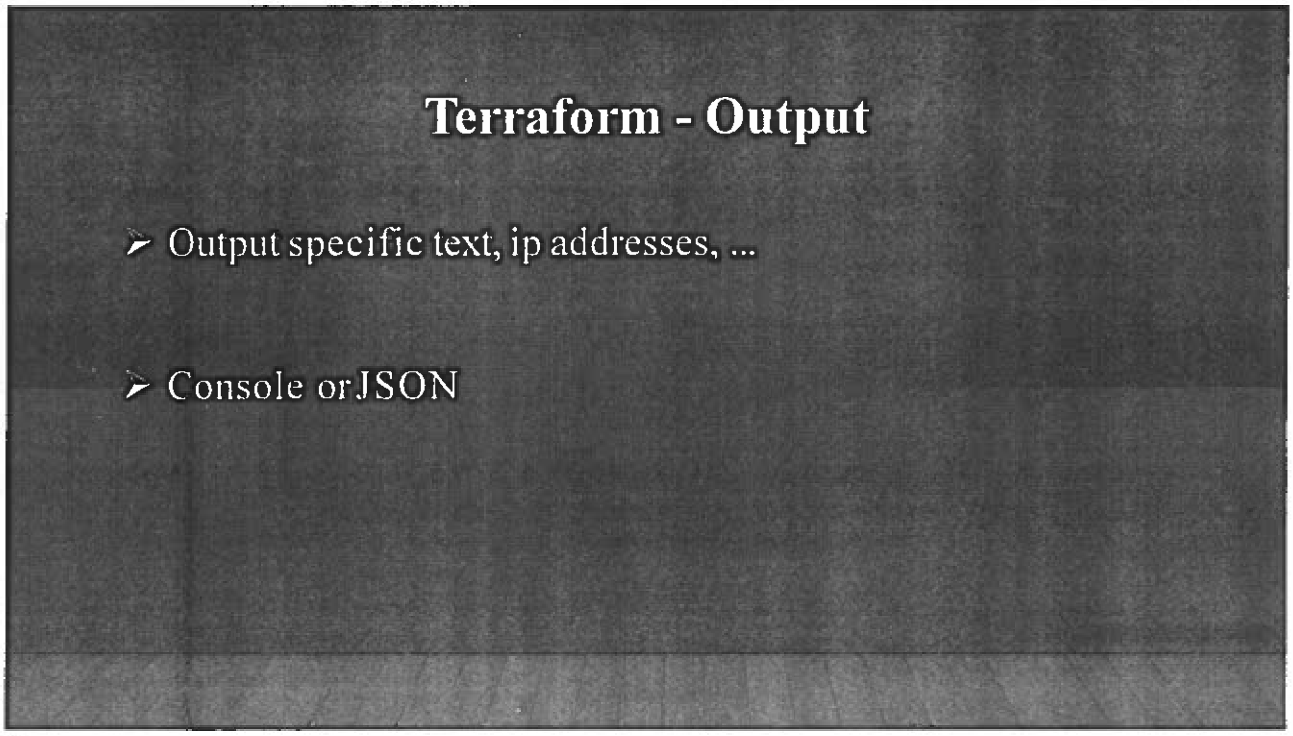
RESOURCES

- The most essential components of configuration files are 'resources'.
- We declare the type of resource and all of the resource specific settings

The general syntax for a Terraform resource is:

```
resource "<PROVIDER>_<TYPE>" "<NAME>" {
        [CONFIG ...]
}
```

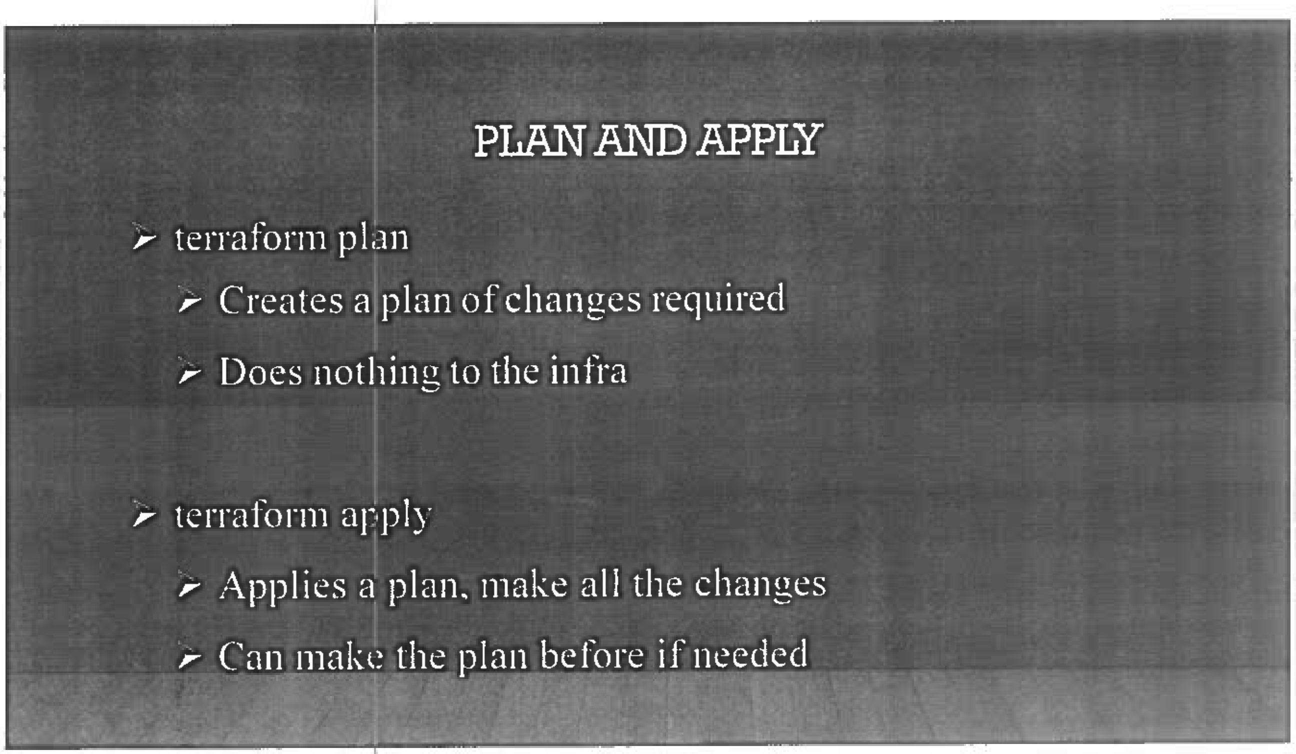
10



Terraform - Output

- Output specific text, ip addresses, ...
- Console or JSON

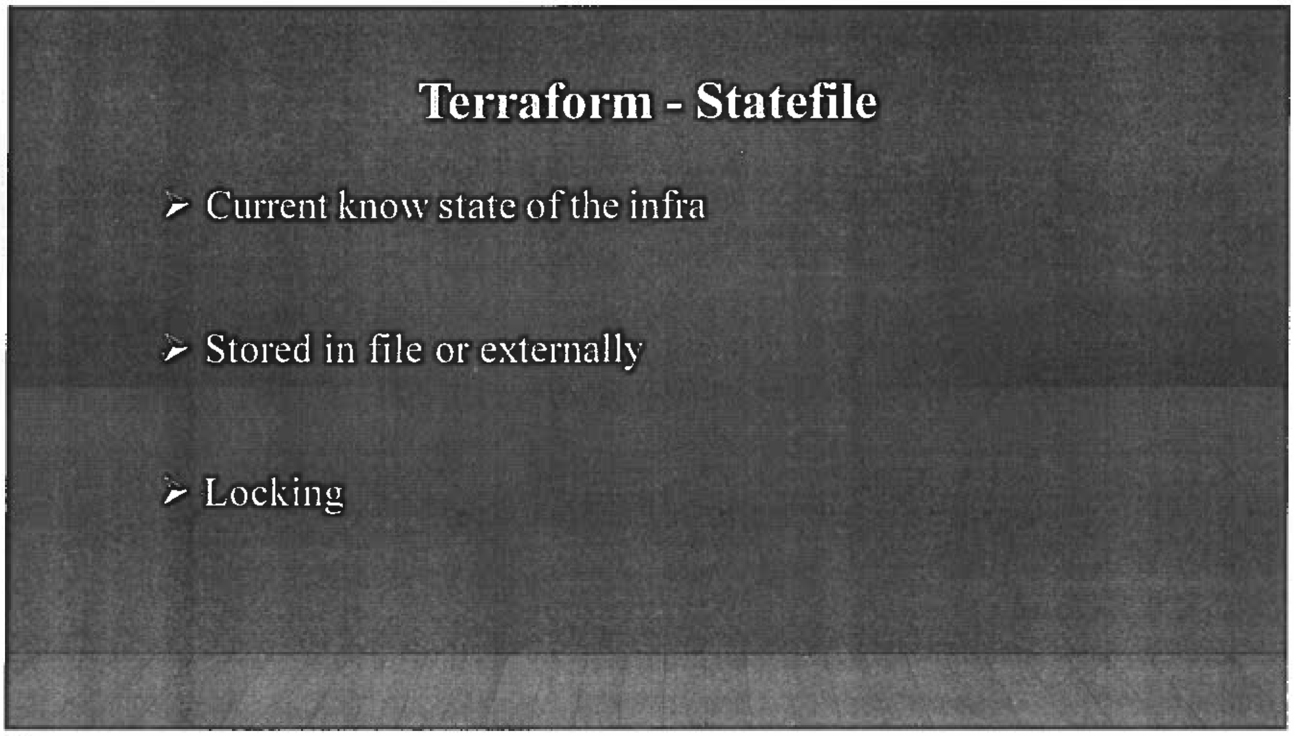
11



PLAN AND APPLY

- terraform plan
  - Creates a plan of changes required
  - Does nothing to the infra
- terraform apply
  - Applies a plan, make all the changes
  - Can make the plan before if needed

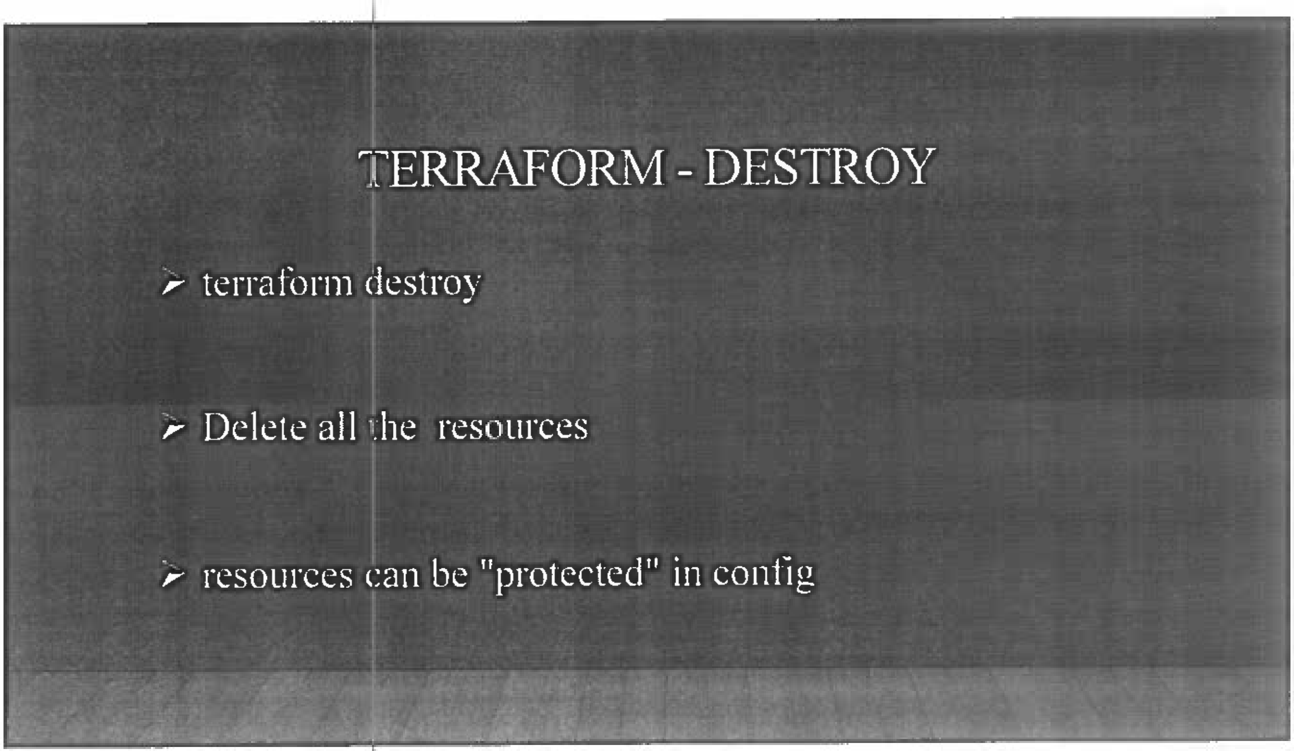
12



Terraform - Statefile

- Current know state of the infra
- Stored in file or externally
- Locking

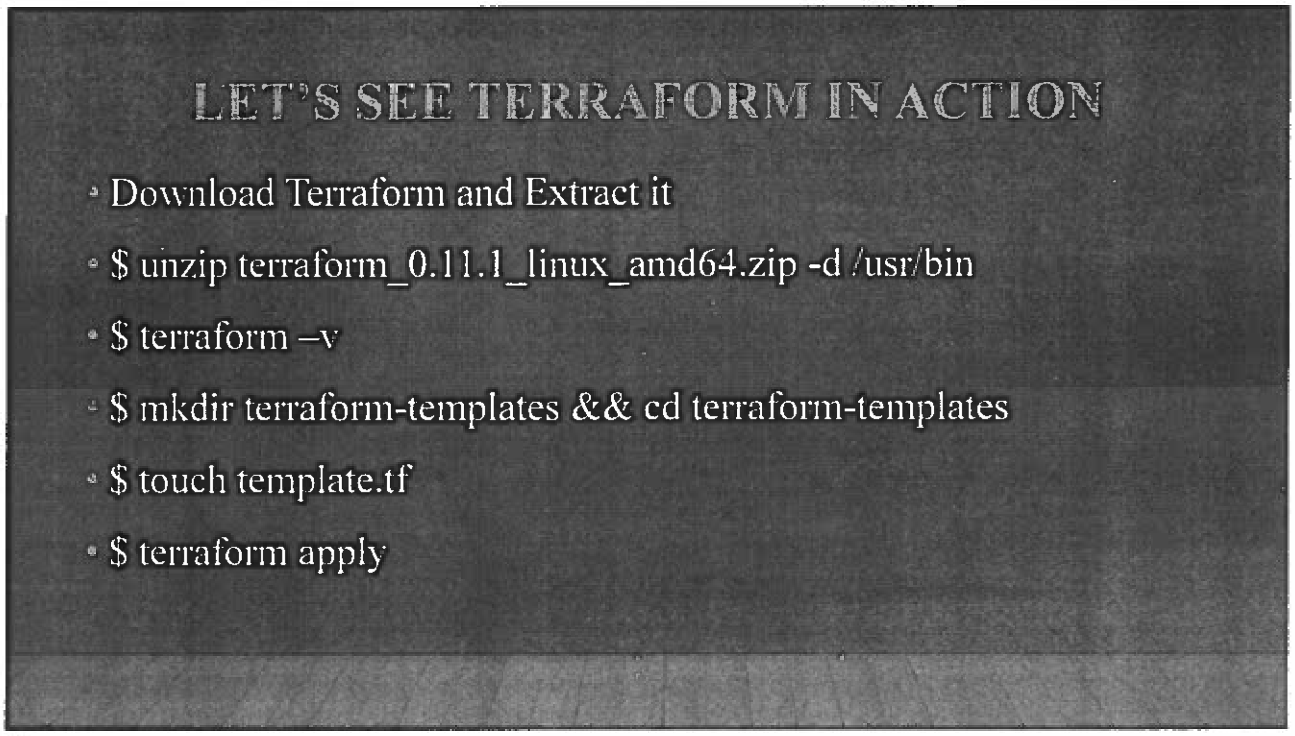
13



TERRAFORM - DESTROY

- terraform destroy
- Delete all the resources
- resources can be "protected" in config

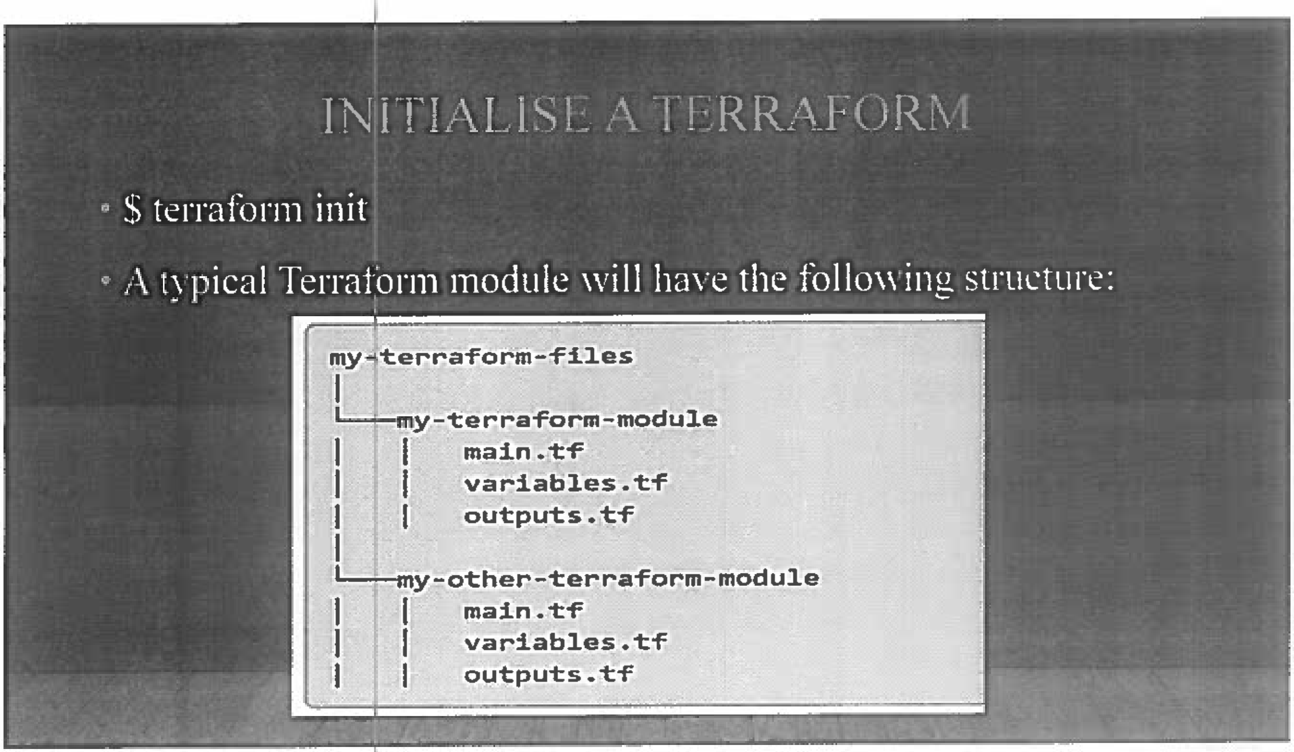
14



LET'S SEE TERRAFORM IN ACTION

- Download Terraform and Extract it
- $ unzip terraform_0.11.1_linux_amd64.zip -d /usr/bin
- $ terraform –v
- $ mkdir terraform-templates && cd terraform-templates
- $ touch template.tf
- $ terraform apply

15

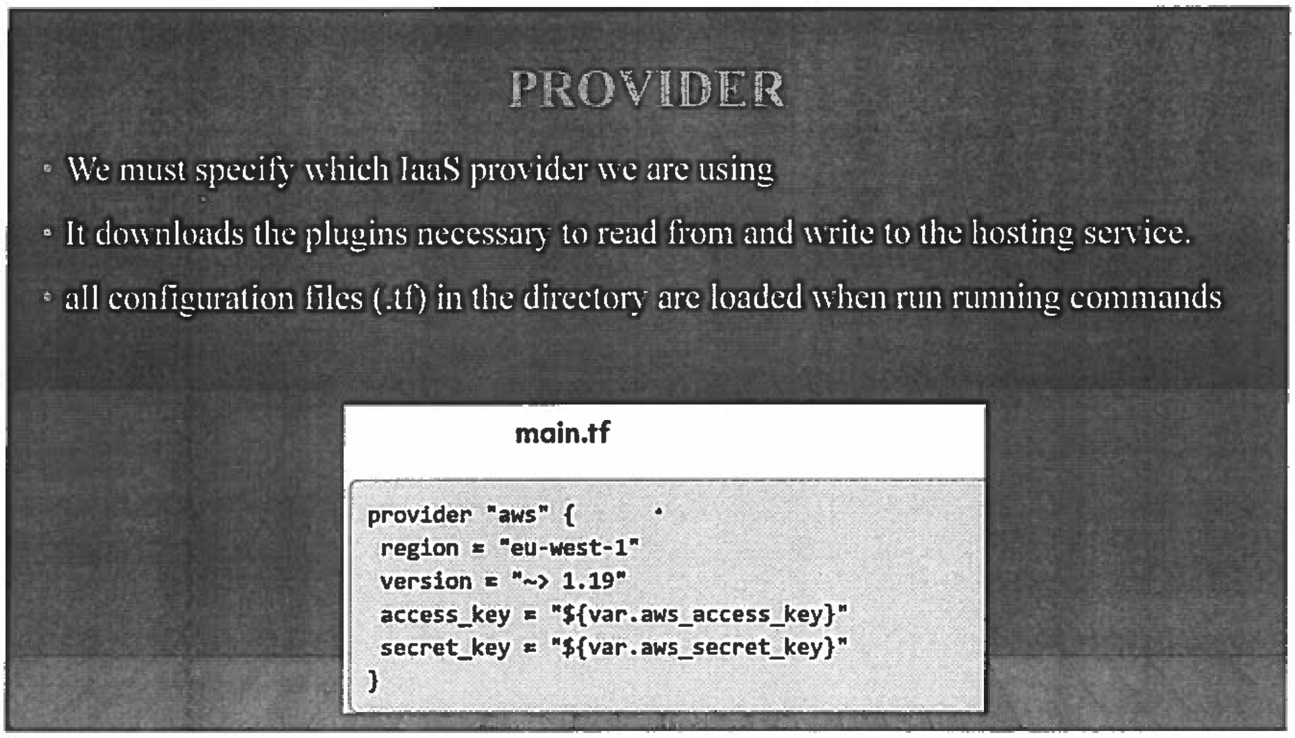


INITIALISE A TERRAFORM

- $ terraform init
- A typical Terraform module will have the following structure:

```
my-terraform-files
|
└───my-terraform-module
|   |    main.tf
|   |    variables.tf
|   |    outputs.tf
|
└───my-other-terraform-module
|   |    main.tf
|   |    variables.tf
|   |    outputs.tf
```

16



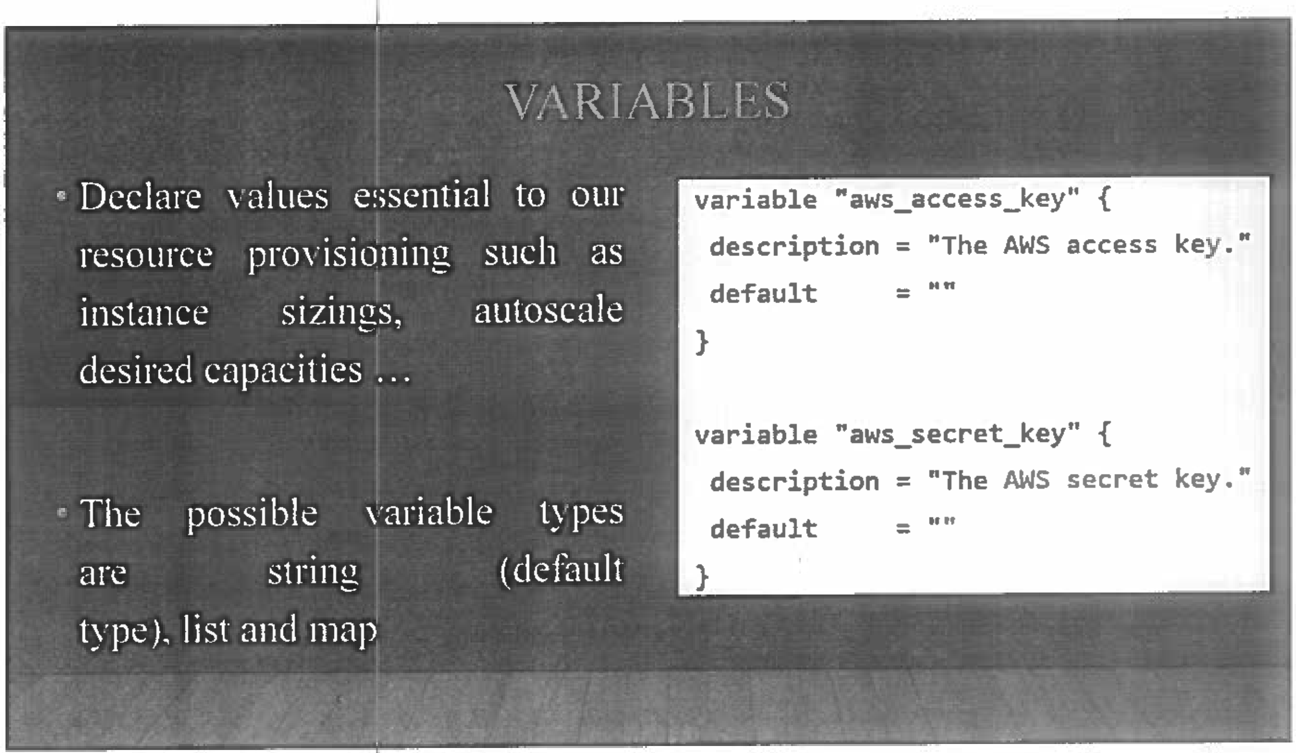
PROVIDER

- We must specify which IaaS provider we are using
- It downloads the plugins necessary to read from and write to the hosting service.
- all configuration files (.tf) in the directory are loaded when run running commands

main.tf

```
provider "aws" {
 region = "eu-west-1"
 version = "~> 1.19"
 access_key = "${var.aws_access_key}"
 secret_key = "${var.aws_secret_key}"
}
```

17



VARIABLES

- Declare values essential to our resource provisioning such as instance sizings, autoscale desired capacities …

- The possible variable types are string (default type), list and map

```
variable "aws_access_key" {
 description = "The AWS access key."
 default     = ""
}

variable "aws_secret_key" {
 description = "The AWS secret key."
 default     = ""
}
```

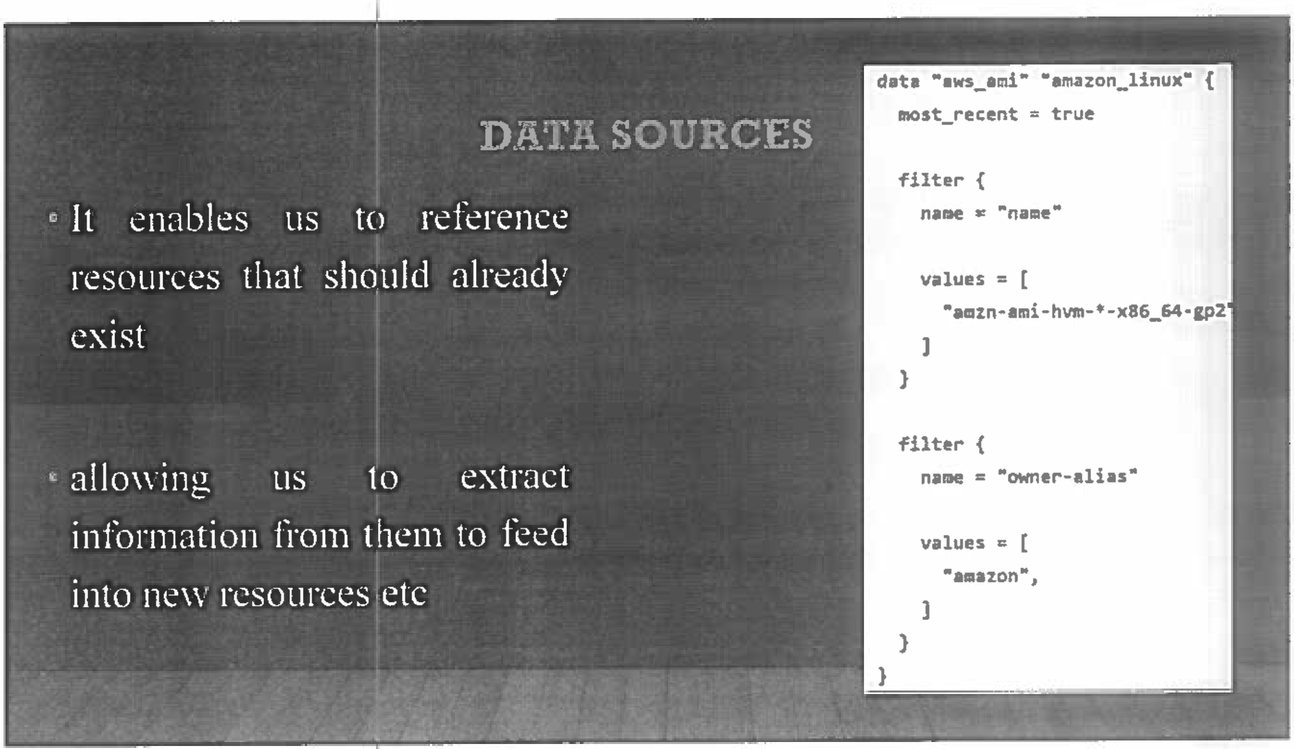
18

RESOURCES

- The most essential components of configuration files are 'resources'.
- We declare the type of resource and all of the resource specific settings

```
resource "aws_volume_attachment" "this_ec2" {
  device_name = "/dev/sdh"
  volume_id   = "${aws_ebs_volume.this.id}"
  instance_id = "${module.ec2.id[0]}"
}

resource "aws_ebs_volume" "this" {
  availability_zone = "${module.ec2.availability_zone[0]}"
  size              = 1
}
```

19

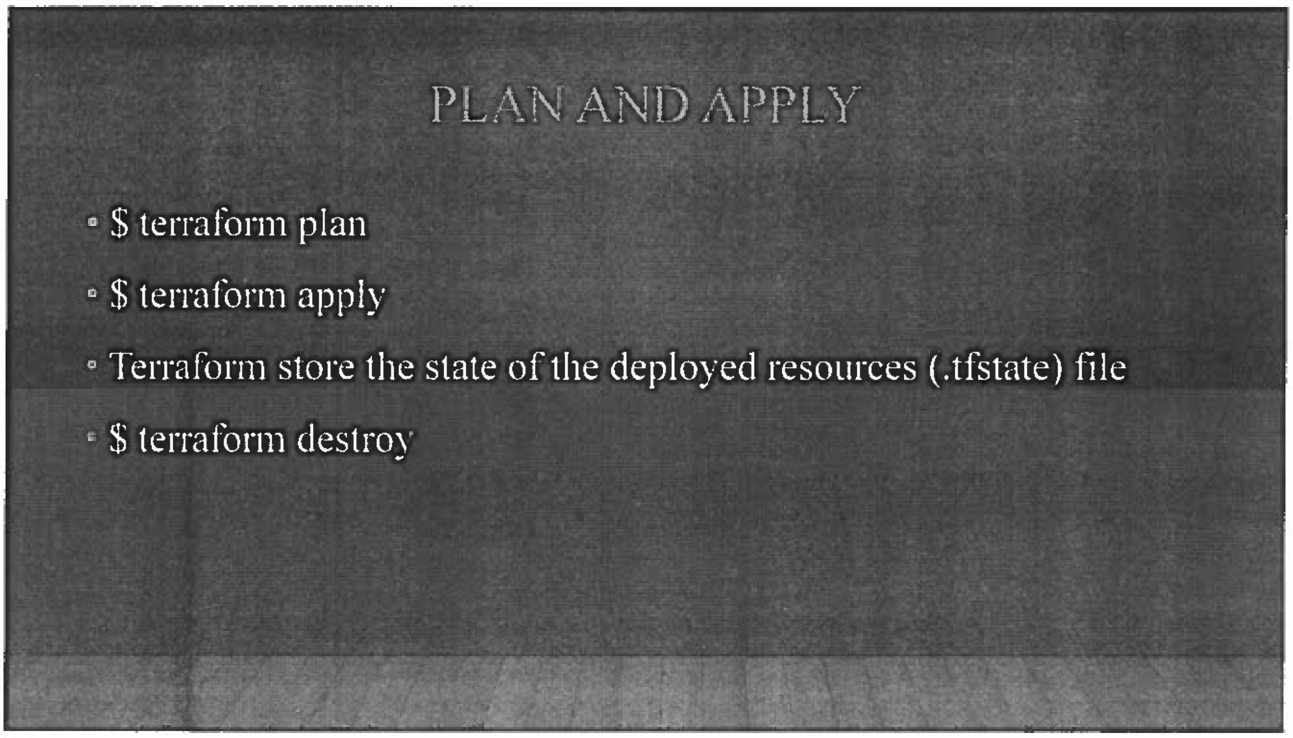


DATA SOURCES

- It enables us to reference resources that should already exist
- allowing us to extract information from them to feed into new resources etc

```
data "aws_ami" "amazon_linux" {
  most_recent = true

  filter {
    name = "name"

    values = [
      "amzn-ami-hvm-*-x86_64-gp2"
    ]
  }

  filter {
    name = "owner-alias"

    values = [
      "amazon",
    ]
  }
}
```

20



PLAN AND APPLY

- $ terraform plan
- $ terraform apply
- Terraform store the state of the deployed resources (.tfstate) file
- $ terraform destroy

21

MORE!

- https://terraform.io

22