Module 2

Terraform Foundations

HashiCorp
Terraform

Agenda

- Installing Terraform
- Authentication
- Terraform files with introduction to Modules
- Terraform Init, Plan and Apply
- Parallelism
- Implicit vs explicit dependencies
- Lab 2

Installing Terraform

Operating Systems:

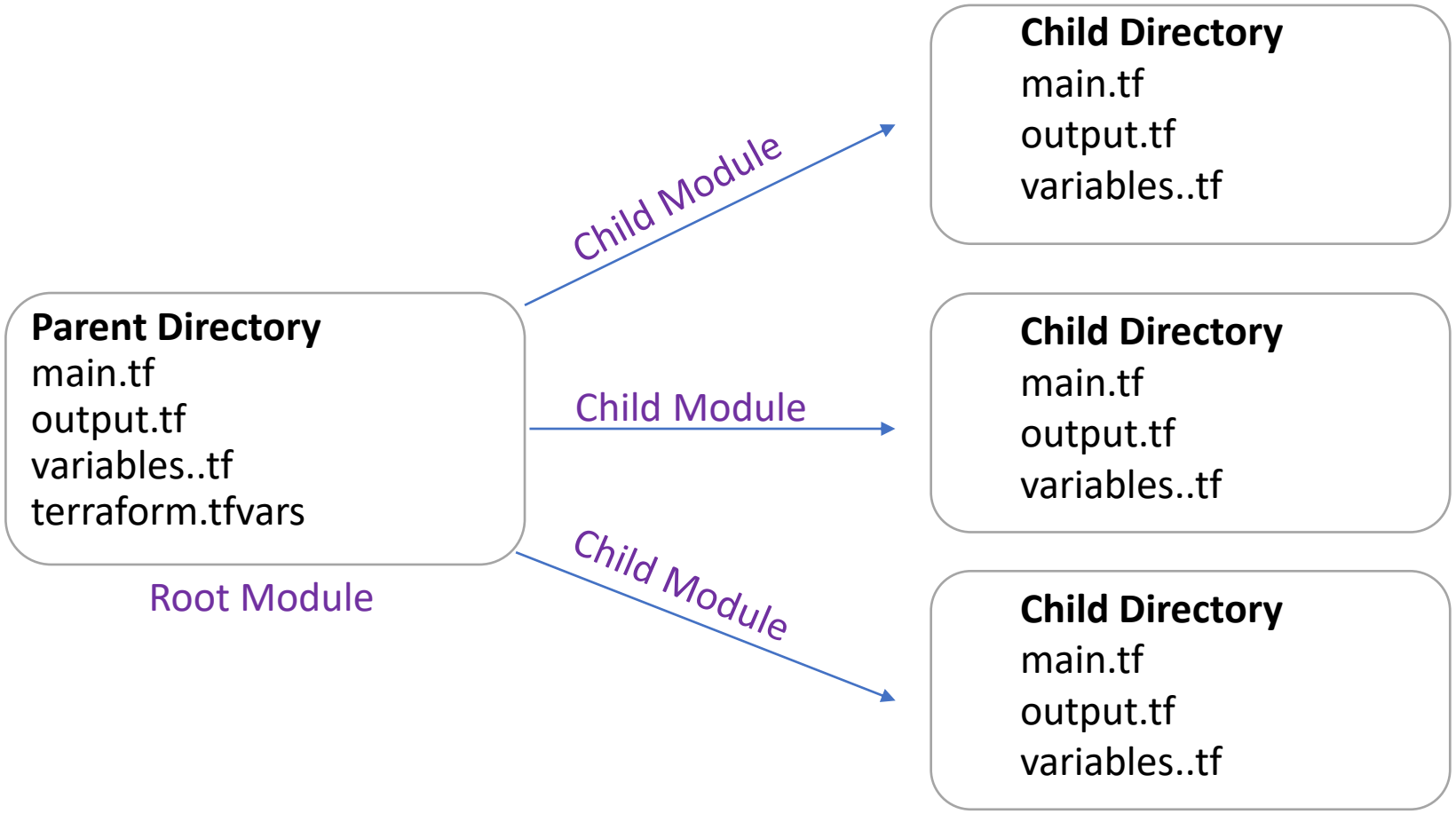
- HomeBrew on OS X
  - brew tap hashicorp/tap :add repo for terraform
  - brew install hashicorp/tap/terraform :Install terraform
- Chocolatey on Windows
  - choco install terraform :install terraform
- Linux
  - sudo apt-get update && sudo apt-get install -y gnupg software-properties-common curl
  - curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add – ::add the Hashicorp GPG key
  - sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main“ :add repo
  - sudo apt-get update && sudo apt-get install terraform :install terraform

Authentication

- Authentication to the platform
- Cloud Platforms behave differently for AuthN
- Cloud Platforms require different levels of AuthZ

HashiCorp
Terraform

Terraform files and modules

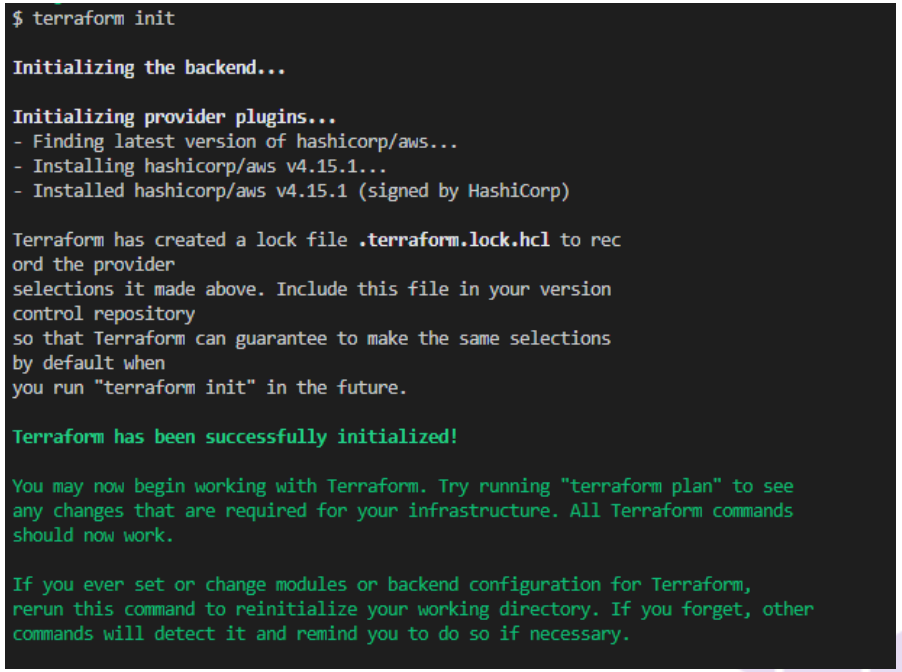


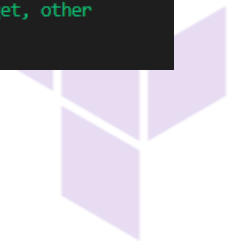
HashiCorp
Terraform

Terraform Init, Plan and Apply

terraform init

- First command run after writing new configuration or after cloning an existing configuration from version control
- Can safely be run multiple times
- Used to initialise a working directory & initialise chosen backend
- Parses the configuration, identifies the providers and installs the plug-ins
- Module blocks are identified from ‘source’ arguments.





HashiCorp Terraform

Terraform Init, Plan and Apply

terraform plan

- Inspects the current configuration proposed
- Inspects the current state of existing objects
- Proposes a set of changes to be made

```
    listener.3238714211.lb_port:                "" => "443"
    listener.3238714211.lb_protocol:            "" => "https"
    listener.3238714211.ssl_certificate_id:     "" => "arn:aws:ia

~ aws_iam_policy.user-my-test
    policy:   }
                    },
                    {
                      "Effect": "Allow",
                      "Resource": [
                -       "arn:aws:s3:::my-test-development",
                -       "arn:aws:s3:::my-test-development/*"
                +       "arn:aws:s3:::my-test-development"
                      ],
                      "Action": [
                        "s3:*"
```


HashiCorp Terraform

Terraform Init, Plan and Apply

terraform apply

- Executes the actions described within the plan.

  [Automatic plan mode] *apply*, without further arguments will run *plan* first

  ~ you will be prompted for approval ~

  [Saved plan mode] *apply,* with a previously saved plan – *terraform apply [plan filename]*

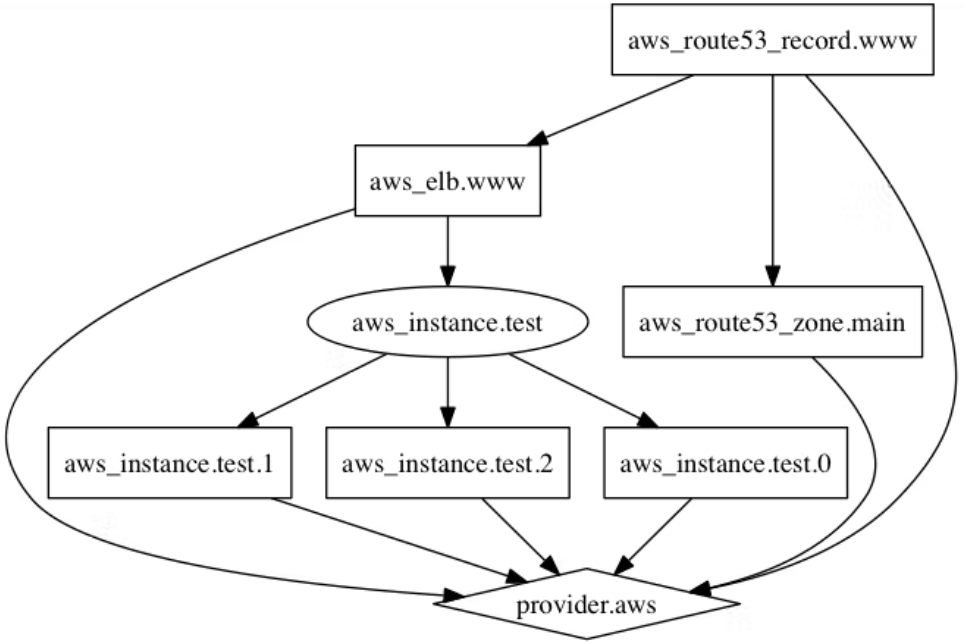
  ~ you will not be prompted for approval ~

- You can inspect saved plan before applying using ‘terraform show’

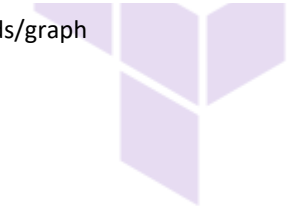
HashiCorp Terraform

Parallelism

- Terraform builds a dependency graph from the terraform configurations, and walks this graph to generate plans, refresh state, and more
- Greater parallelism potentially increases the speed of deployment but may overwhelm the resources of the machine running Terraform
- Can be used as an option *–parallelism* with *init, plan* and *apply (*Default 10)



https://developer.hashicorp.com/terraform/cli/commands/graph



HashiCorp Terraform

Implicit vs explicit dependencies

Implicit dependences are known and calculated by Terraform
Explicit dependencies are defined by the configuration author

- Implicit dependencies

  An elastic IP references an AWS instance. Terraform therefore knows that the IP is dependent on the instance existing so it will create the instance before the IP.

- Explicit dependencies

  Two instances could be created at the same time (parallelism) but the DB instance needs to be created and its configuration exported before creating a web instance

```
resource "aws_instance" “my_ec2" {
  ami = data.aws_ami.amazon_linux.id
  instance_type = "t2.micro" }

resource "aws_eip" “staticip" {
   vpc = true
   instance = aws_instance.my_ec2.id
}
```

```
.....
resource "aws_instance" “my_web" {
   ami = data.aws_ami.amazon_linux.id
   instance_type = "t2.micro"

depends_on = [aws_instance.my_rds ]
}
```


HashiCorp Terraform

Lab Group Discussion & Explore

Scoping

- What Type of Resources are you going to create ?
- What specific Properties do we need to consider for these resources ?

Workflow

- Are there any Implicit or Explicit dependencies ?
- What TF *Provider*, *Resource,* and *Modules* are required ?

Research

- Identify the TF documentation that will assist in developing the solution

HashiCorp Terraform

Lab 2

Create an EC2 instance

In this lab you will deploy an EC2 instance in AWS

HashiCorp Terraform

Any questions…

Terraform