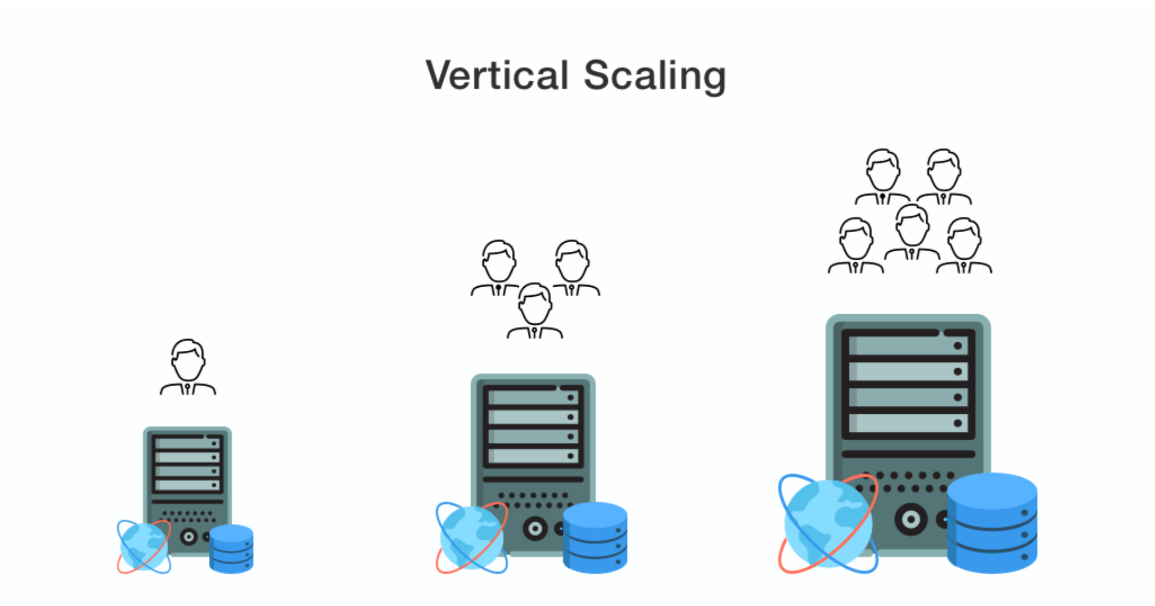
Scaling and balancing ⚖️

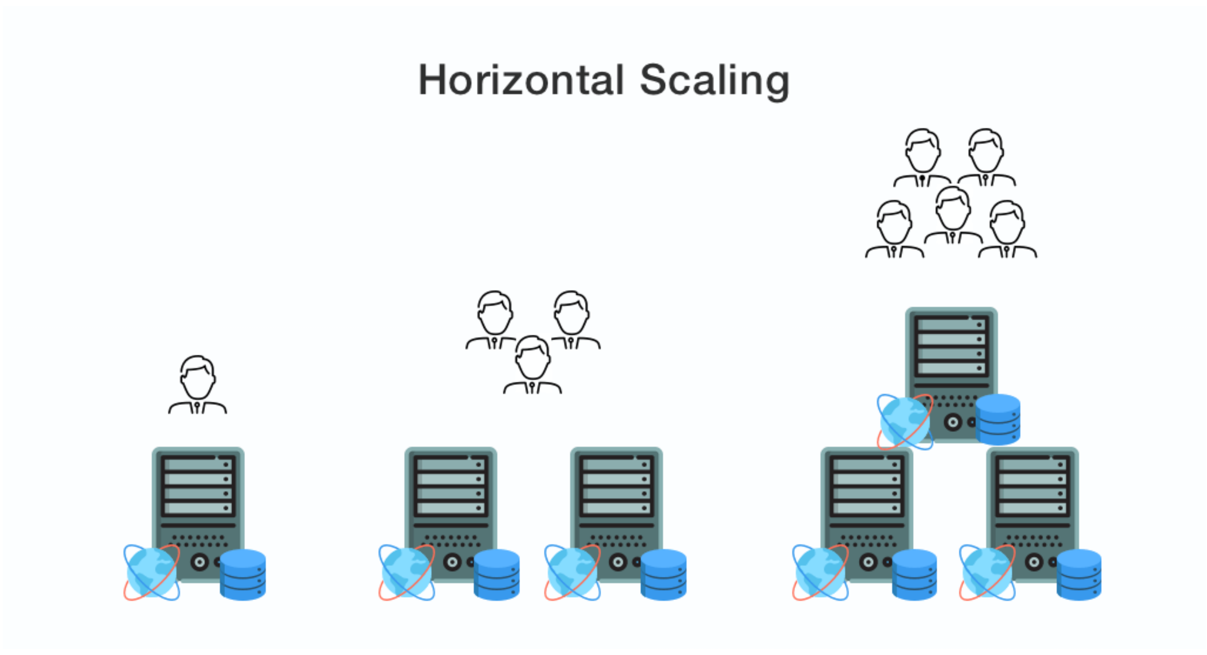
Vertical scaling (scaling up)

- is increasing the computing power of your app and its ability to handle the increasing load
- Horizontal scaling



Horizontal scaling (scaling out)

- Resource increment by the addition of units to the apps cloud architecture, this means adding more units of smaller capacity rather than fewer units of larger capacity
- Requests for units are thus spread over multiple units, thus reducing the excess load on a single machine



Highly available

- reducing the latency ie the time it takes to get a response when you ping a server
- can be done by improving network infrastructure with

Terraform

create a folder named eng.67 terraform

in that folder, create a readme

in that readme, write "Infrastructure as code with terraform"

What is Terraform?

- does it differ to Ansible?
- there are 2 side of IAC
    - 1 is configuration management

- 2 is orchestration

Ansible - configuration management

- provisions the infrastructure we create
- configures the environment to the correct state for the app to run

Terraform - orchestration (IAC)

- provisions the infrastructure we need

Kubernetes - orchestration used in containerisation ie docker, Crio, Rocket (docker is top 1)

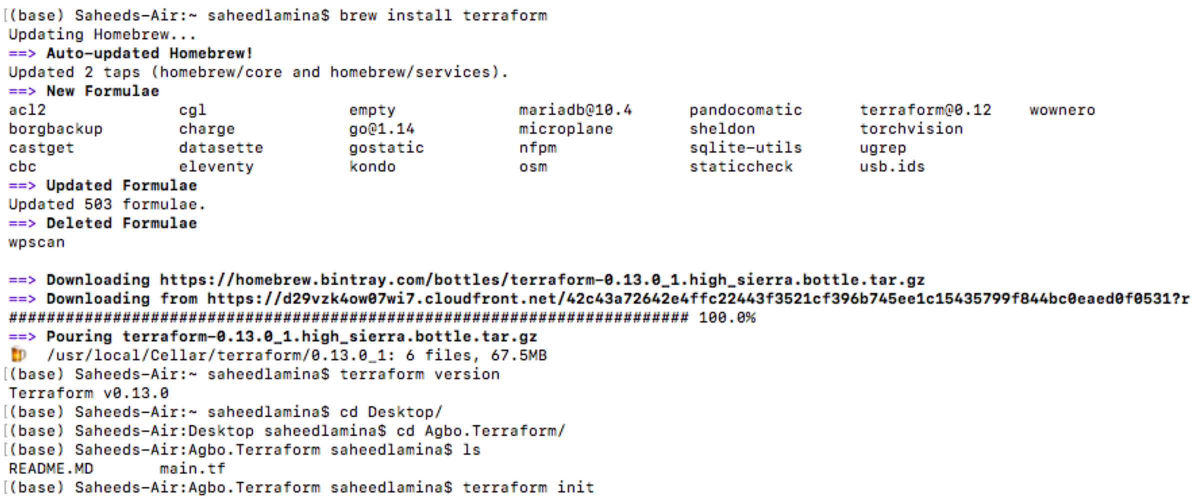
Terraform

can change, update and link with cloud services providers

we need was access and secret key to link terraform with AWS/ cloud provider

Dependencies:

Terraform, for Mac: in your terminal run `brew install terraform`



```
(base) Saheeds-Air:~ saheedlamina$ brew install terraform
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 2 taps (homebrew/core and homebrew/services).
==> New Formulae
acl2            cgl             empty           mariadb@10.4    pandocomatic    terraform@0.12  wownero
borgbackup      charge          go@1.14         microplane      sheldon         torchvision
castget         datasette       gostatic        nfpm            sqlite-utils    ugrep
cbc             eleventy        kondo           osm             staticcheck     usb.ids
==> Updated Formulae
Updated 503 formulae.
==> Deleted Formulae
wpscan

==> Downloading https://homebrew.bintray.com/bottles/terraform-0.13.0_1.high_sierra.bottle.tar.gz
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/42c43a72642e4ffc22443f3521cf396b745ee1c15435799f844bc0eaed0f0531?r
######################################################################## 100.0%
==> Pouring terraform-0.13.0_1.high_sierra.bottle.tar.gz
🍺  /usr/local/Cellar/terraform/0.13.0_1: 6 files, 67.5MB
(base) Saheeds-Air:~ saheedlamina$ terraform version
Terraform v0.13.0
(base) Saheeds-Air:~ saheedlamina$ cd Desktop/
(base) Saheeds-Air:Desktop saheedlamina$ cd Agbo.Terraform/
(base) Saheeds-Air:Agbo.Terraform saheedlamina$ ls
README.MD       main.tf
(base) Saheeds-Air:Agbo.Terraform saheedlamina$ terraform init
```

nano `variables.tf`

```
variable "vpc_id" {
    type = string
    default = "vpc-xxxxxxxxx"
}

# using vpc_id variable in main.tf
vpc_id - var.vpc_id
variable "name" {
    type - string
    default = "agbo.terraform.ec2"
}

variable "app_ami_id" {
    type = string
    default = " ami-xxxxx"
}
```

`nano main.tf`

This is to launch an AMI on AWS

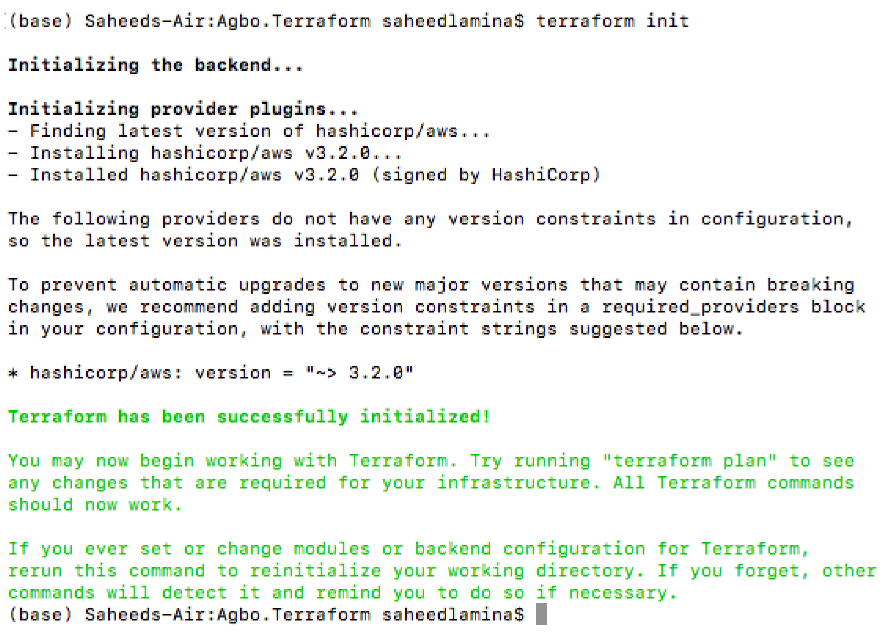
main.tf file is used in terraform

syntax:

```
provider "aws" {
# which region do we have ami available (in my case it's ireland)
   region = "eu-west-1"


}
# create an instance - launch an instace from AMI
resource "aws_instance" "app_instance" {
          ami          = "ami-xxxxxxxx"
# what type of ec2 instance we want to create t2micro
          instance_type = "t2.micro"

# do we want public IP
          associate_public_ip_ip_address = true

          tags = {
              Name = "agbo.terraform.ec2"
          }

}
```

exit the file

`terraform init` - to initialise (in the same location) initialise it after specifying provider and region



```
(base) Saheeds-Air:Agbo.Terraform saheedlamina$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.2.0...
- Installed hashicorp/aws v3.2.0 (signed by HashiCorp)

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* hashicorp/aws: version = "~> 3.2.0"

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.
(base) Saheeds-Air:Agbo.Terraform saheedlamina$
```

after entering the additional code, run `terraform plan` s got a no changes up to date message

then run `terraform apply` can get an error about credentials, this is fixed by setting up access keys

Terraform commands

- `terraform init`
- `terraform plan` - checks the steps inside the code and lists success or errors
- `terraform apply`- will implement the code, or deploy the infrastructure (we can run this cmd directly but we should run plan first)

Next step

- code a VPC using terraform

- they used student VPC

Task

- create a subnet block of code
- attach this subnet to your VPC
- create a security group, attach it to VPC
- create ingress (inbound traffic) block of code to allow port 80 and 0.0.0.0/0
- create Egress (outbound traffic) block of code to allow all