

4.5

Aprovisionamiento de infraestructura: Terraform

Aprovisionamiento de infraestructura: Terraform

- Aprovisionamiento de infraestructura: Terraform



<https://www.terraform.io/>



Aprovisionamiento de infraestructura: ~~Terraform~~

- Aprovisionamiento de infraestructura: Terraform

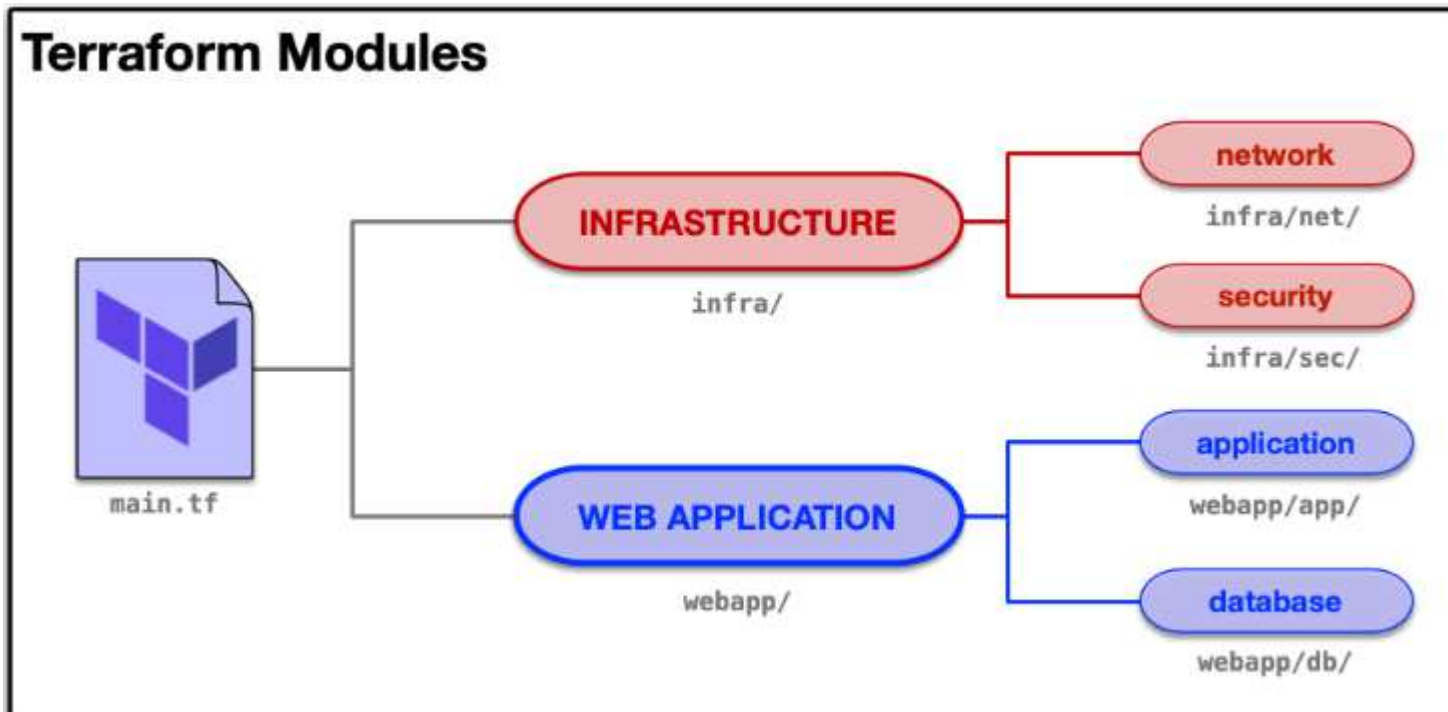


<https://www.terraform.io/>



Aprovisionamiento de infraestructura: Terraform

- Aprovisionamiento de infraestructura: Terraform
- Para introducirnos a Terraform realizaremos un ejercicio donde crearemos
 - Infraestructura AWS VPC
 - Infraestructura AWS EC2 y RDS



Aprovisionamiento de infraestructura: ~~Terraform~~

- Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

- Aprovisionamiento Infraestructura
 - VPC
 - VPC Internet Gateway
 - VPC subredes
 - VPC Routing Tables
 - VPC security Groups
- Aprovisionamiento Aplicación WEB
 - EC2
 - RDS
- Se utilizara
 - AWS CLI
 - Terraform

Aprovisionamiento de infraestructura: Terraform

- Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

- Configuración AWS CLI
- ~/.aws/credentials ~/.aws/config

```
[default]
aws_access_key_id = REDACTED
aws_secret_access_key = REDACTED

[learning]
aws_access_key_id = REDACTED
aws_secret_access_key = REDACTED
```

```
[default]
region = us-west-2
output = json

[profile learning]
region = us-east-2
output = json
```

aws configure --profile learning

Aprovisionamiento de infraestructura: Terraform

- Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

- Instalación Terraform

wget https://releases.hashicorp.com/terraform/0.12.12/terraform_0.12.12_linux_386.zip
unzip terraform_0.12.12_linux_386.zip

```
vagrant@manager:~/terraform$ wget https://releases.hashicorp.com/terraform/0.12.12/terraform_0.12.12_linux_386.zip
--2019-10-25 14:20:55-- https://releases.hashicorp.com/terraform/0.12.12/terraform_0.12.12_linux_386.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 151.101.133.183, 2a04:4e42:1f::439
Connecting to releases.hashicorp.com (releases.hashicorp.com)|151.101.133.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15334684 (15M) [application/zip]
Saving to: 'terraform_0.12.12_linux_386.zip'

terraform_0.12.12_linux_386.z 100%[=====>] 14.62M 14.0MB/s in 1.0s

2019-10-25 14:20:57 (14.0 MB/s) - 'terraform_0.12.12_linux_386.zip' saved [15334684/15334684]

vagrant@manager:~/terraform$ |
```

```
vagrant@manager:~/terraform$ ls -ls
total 57044
42068 -rwxr-xr-x 1 vagrant vagrant 43076160 Oct 18 18:44 terraform
14976 -rw-rw-r-- 1 vagrant vagrant 15334684 Oct 18 18:44 terraform_0.12.12_linux_386.zip
vagrant@manager:~/terraform$ |
```

Aprovisionamiento de infraestructura: Terraform

- Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

- Instalación Terraform
- Una vez descargado y descomprimido agregar al PATH del sistema
 - Para ello utilizar /home/vagrant/.profile y agregar el directorio donde esta el binario de terraform
 - Actualizar el bash
 - source .profile
 - Y probar a ejecutar terraform
 - \$terraform -v
- Para obtener ayuda
 - \$terraform help
 - \$terraform help plan

```
vagrant@manager:~$ terraform -v
Terraform v0.12.12
vagrant@manager:~$ |
```

```
# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin directories
PATH="$HOME/bin:$HOME/.local/bin:$PATH"
export PATH="$PATH:/home/vagrant/terraform"
```

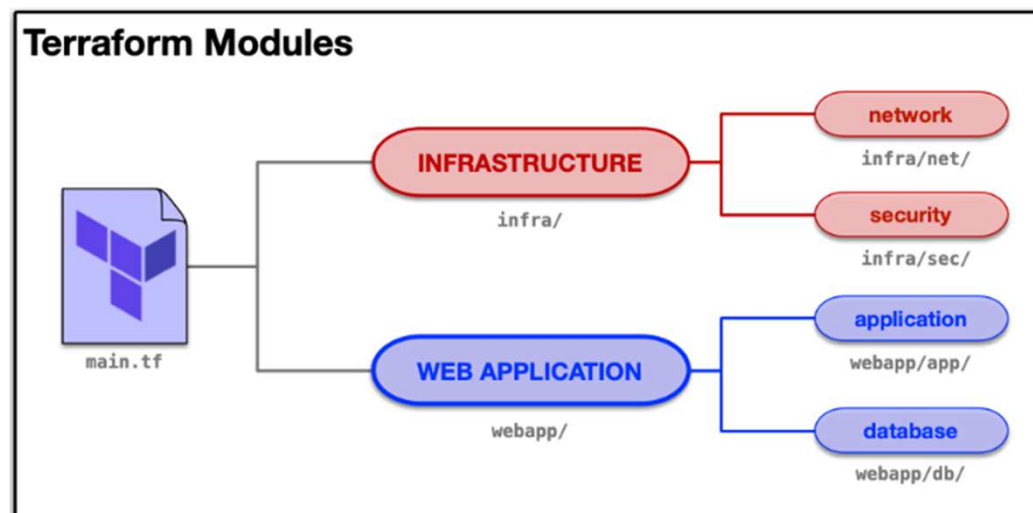

Aprovisionamiento de infraestructura: Terraform

- Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Creación de estructura y ficheros para un proyecto
TERRAFORM

- Los ficheros Terraform tienen extensión.tf
- Un ejemplo de proyecto puede ser el siguiente



```
i tf-projects/  
├── infra/  
│   ├── aws.tf  
│   ├── net/  
│   └── sec/  
└── webapp/  
    ├── app/  
    ├── aws.tf  
    └── db/
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de estructura y ficheros ficheros

```
├── tf-projects/  
│   ├── infra/  
│   │   ├── aws.tf  
│   │   ├── net/  
│   │   └── sec/  
│   └── webapp/  
│       ├── app/  
│       ├── aws.tf  
│       └── db/
```

```
mkdir -p ~/tf-ejercicio1/{infra/{net,sec},webapp/{app,db}}  
touch ~/tf-projects/{infra,webapp}/aws.tf
```

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Configurando los módulos AWS

- Describir en el fichero `./infra/aws.tf` que se va a utilizar el proveedor/modulo AWS y crear una variable denominada **región**

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

- Hacer lo mismo con el modulo `webapp/aws.tf`
- Terraform no accede a la región de nuestra credenciales por defecto ,por lo tanto habrá que especificarlo
 - Al exportar la variable `TF_VAR_región` el valor esta accesibe en los modulos Terraform

```
export TF_VAR_región= us-east-1
```

```
export TF_VAR_region=$(  
    awk -F'= ' '/region/{print $2}' <(  
        grep -A1 "\[.*$AWS_PROFILE\]" ~/.aws/config)  
)
```

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Descargar el plugin AWS de terraform a los dos proyecto (ficheros .tf)

- Para instalar el plugin de AWS ir a la carpeta donde están los .tf y ejecutar : *terraform init*
- Este comando crea la carpeta *.terraform/* e instala los plugins utilizando la información de los ficheros .tf

```
vagrant@manager:~/terraform$ pushd ~/terraform/tf-ejercicio1/infra/ && terraform init && popd
~/terraform/tf-ejercicio1/infra ~/terraform
Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 2.33.0...

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, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.33"

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,
commands will detect it and remind you to do so if necessary.

vagrant@manager:~/terraform$
```

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

```
vagrant@manager:~/terraform$ ls -la tf-ejercicio1/infra/
total 24
drwxrwxr-x 5 vagrant vagrant 4096 Oct 25 14:51 .
drwxrwxr-x 4 vagrant vagrant 4096 Oct 25 14:39 ..
-rw-rw-r-- 1 vagrant vagrant  71 Oct 25 14:44 aws.tf
drwxrwxr-x 2 vagrant vagrant 4096 Oct 25 14:39 net
drwxrwxr-x 2 vagrant vagrant 4096 Oct 25 14:39 sec
drwxr-xr-x 3 vagrant vagrant 4096 Oct 25 14:51 .terraform
vagrant@manager:~/terraform$ ls -la tf-ejercicio1/infra/.terraform/
total 12
drwxr-xr-x 3 vagrant vagrant 4096 Oct 25 14:51 .
drwxrwxr-x 5 vagrant vagrant 4096 Oct 25 14:51 ..
drwxr-xr-x 3 vagrant vagrant 4096 Oct 25 14:51 plugins
```

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Creación de par de claves e importar a AWS EC2 KeyPair

- Crean una carpeta donde generar el par de claves “.secrets”
- Generar las claves

```
KEYPATH=/home/vagrant/.secrets
```

```
KEYNAME="deploy-aws"
```

```
openssl genrsa -out "$KEYPATH/aws.pem" 4096
```

```
openssl rsa -in "$KEYPATH/aws.pem" -pubout > "$KEYPATH/a
```

```
chmod 400 "$KEYPATH/aws.pem"
```

- Subir a AWS

```
aws ec2 import-key-pair \  
--key-name $KEYNAME \  
--public-key-material \  
"$$(grep -v PUBLIC $KEYPATH/aws.pub | tr -d '\n')"
```

```
vagrant@manager:~/secrets$ aws ec2 import-key-pair --key-name $KEYNAME --public-key-material "$$(grep -v  
PUBLIC $KEYPATH/aws.pub |  
tr -d '\n')";  
{  
  "KeyName": "deploy-aws",  
  "KeyFingerprint": "56:57:3a:ad:5b:36:07:77:65:e9:20:90:aa:8d:81:4a"  
}  
vagrant@manager:~/secrets$ |
```

- Copiar claves SSH a .ssh

```
cp $KEYPATH/aws.pem $HOME/.ssh/$KEYNAME.pem
```

```
cp $KEYPATH/aws.pub $HOME/.ssh/$KEYNAME.pub
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Verificar la creación en ASWS del par de claves

```
aws ec2 describe-key-pairs \  
  --query 'KeyPairs[*].[KeyName]' \  
  --output text | grep $KEYNAME
```

```
vagrant@manager:~/secrets$ aws ec2 describe-key-pairs \  
> --query 'KeyPairs[*].[KeyName]' \  
> --output text | grep $KEYNAME;  
deploy-aws  
vagrant@manager:~/secrets$ |
```

Ahora y apodemos utilizra para conectarnos a maquinas
EC2 el siguiente comando

```
KEYNAME="deploy-aws"  
ssh -i ~/.ssh/$KEYNAME.pem $AWS_HOST_IP
```

Aprovisionamiento de infraestructura: ~~de~~ Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- Conceptos Terraform a trabajar
 - Variables de entrada y salida
 - Definición de proveedores
 - Definición de recursos
 - Fuentes de datos
 - División en Modulos
- Estos conceptos los crearemos en la carpeta “infra”
 - La carpeta se divide en fods subcarpetas
 - Net : igw, routing, subredes
 - Sec: Grupos de seguridad

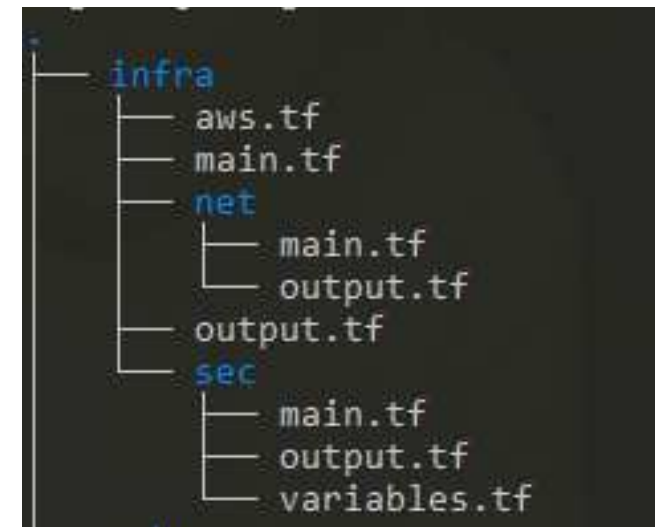


Aprovisionamiento de infraestructura: ~~Infra~~ Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- Se crearán los siguientes ficheros
 - Infra/main.tf
 - Infra/output.tf
 - Infra/net/main.tf
 - Infra/net/output.tf
 - infra/sec/main.tf
 - Infra/sec/output.tf
 - Infra/sec/variables.tf



```
cd ~/tf-projects/infra
```

```
touch {.,net,sec}/{main.tf,output.tf} sec/variables.tf
```


Aprovisionamiento de infraestructura:

Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- En una organización podríamos tener subdivisiones de red en base a la región, por ejemplo

- `net/us-west-1`
- `net/us-east-1`
- `sec/us-west-1`
- `sec/us-east-1`

Aprovisionamiento de infraestructura:

Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/main.tf** : División de las tareas de creación en módulos
 - Módulo de red
 - Módulo de seguridad

```
module "network" {  
  source = "./net"  
}  
module "security" {  
  source = "./sec"  
  vpc_id = "${module.network.vpc}"  
}
```

- El grupo de seguridad debe de enlazar el VPC sobre el cual debe de trabajar

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de VPC y tags

```
resource "aws_vpc" "my-main" {  
  cidr_block      = "10.0.0.0/16"  
  enable_dns_hostnames = false  
  enable_dns_support = true  
  instance_tenancy  = "default"  
  tags = {  
    Site = "my-web-site"  
    Name = "my-vpc"  
  }  
}
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Obtención de datos de AWS
 - Zonas de disponibilidad

```
data "aws_availability_zones" "available" {}
```

Aprovisionamiento de infraestructura: ~~De~~ Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de subredes
 - Cada subred en una zona de disponibilidad

```
resource "aws_subnet" "my-public1" {  
  vpc_id          = "${aws_vpc.my-main.id}"  
  cidr_block      = "10.0.2.0/24"  
  availability_zone =  
    "${data.aws_availability_zones.available.names[1]}"  
  map_public_ip_on_launch = true  
  tags = {  
    Name = "my-public2"  
    Site = "my-web-site"  
  }  
}
```

Aprovisionamiento de infraestructura: ~~De~~ Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de subredes públicas
 - Cada subred en una zona de disponibilidad

```
resource "aws_subnet" "my-public2" {  
  vpc_id          = "${aws_vpc.my-main.id}"  
  cidr_block      = "10.0.1.0/24"  
  availability_zone = "${data.aws_availability_zones.available.names[0]}"  
  map_public_ip_on_launch = true  
  tags = {  
    Name = "my-public1"  
    Site = "my-web-site"  
  }  
}
```

Aprovisionamiento de infraestructura: ~~de~~ Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de subredes públicas
 - Cada subred en una zona de disponibilidad

```
resource "aws_subnet" "my-public1" {  
  vpc_id          = "${aws_vpc.my-main.id}"  
  cidr_block      = "10.0.2.0/24"  
  availability_zone = "${data.aws_availability_zones.available.names[1]}"  
  map_public_ip_on_launch = true  
  tags = {  
    Name = "my-public2"  
    Site = "my-web-site"  
  }  
}
```

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de subredes privadas
 - Cada subred en una zona de disponibilidad

```
resource "aws_subnet" "my-private1" {  
  vpc_id          = "${aws_vpc.my-main.id}"  
  cidr_block      = "10.0.3.0/24"  
  availability_zone = "${data.aws_availability_zones.available.names[1]}"  
  map_public_ip_on_launch = true  
  tags = {  
    Name = "my-private1"  
    Site = "my-web-site"  
  }  
}
```


Aprovisionamiento de infraestructura: ~~Ansible~~ Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de subredes privadas
 - Cada subred en una zona de disponibilidad

```
resource "aws_subnet" "my-private1" {  
  vpc_id          = "${aws_vpc.my-main.id}"  
  cidr_block      = "10.0.3.0/24"  
  availability_zone = "${data.aws_availability_zones.available.names[1]}"  
  map_public_ip_on_launch = true  
  tags = {  
    Name = "my-private1"  
    Site = "my-web-site"  
  }  
}
```

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de subredes privadas
 - Cada subred en una zona de disponibilidad

```
resource "aws_subnet" "my-private2" {  
  vpc_id          = "${aws_vpc.my-main.id}"  
  cidr_block      = "10.0.4.0/24"  
  availability_zone = "${data.aws_availability_zones.available.names[0]}"  
  map_public_ip_on_launch = true  
  tags = {  
    Name = "my-private2"  
    Site = "my-web-site"  
  }  
}
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación del IGW

```
resource "aws_internet_gateway" "my-igw" {  
  vpc_id = "${aws_vpc.my-main.id}"  
  tags = {  
    Name = "my-igw"  
    Site = "my-web-site"  
  }  
}
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/main.tf** : Creación de tablas de rutado

```
resource "aws_route_table" "my-rt" {  
  vpc_id = "${aws_vpc.my-main.id}"  
  route {  
    cidr_block = "0.0.0.0/0"  
    gateway_id = "${aws_internet_gateway.my-igw.id}"  
  }  
  tags = {  
    Site = "my-web-site"  
    Name = "my-rt"  
  }  
}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: VPC, subredes y grupos de seguridad, tablas de rutado y IGW

- **Infra/net/output.tf** : Información de salida

```
output "vpc" {  
  value = "${aws_vpc.my-main.id}"  
}  
output "sn_pub1" {  
  value = "${aws_subnet.my-public1.id}"  
}  
output "sn_pub2" {  
  value = "${aws_subnet.my-public2.id}"  
}  
output "sn_priv1" {  
  value = "${aws_subnet.my-private1.id}"  
}  
output "sn_priv2" {  
  value = "${aws_subnet.my-private2.id}"  
}
```

Aprovisionamiento de infraestructura:

Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: grupos de seguridad

- **Infra/sec/variable.tf** : Variable para el id del VPC a obtener del otro modulo

```
variable "vpc_id" {}
```

- Esta variable se establece en el infrea/main.tf

```
module "network" {  
  source = "../net"  
}  
  
module "security" {  
  source = "../sec"  
  vpc_id = "${module.network.vpc}"  
}
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: grupos de seguridad

- **Infra/sec/main.tf** : Seguridad para servidores WEB

```
resource "aws_security_group" "my-webserver" {  
  name      = "webserver"  
  description = "Allow HTTP from Anywhere"  
  vpc_id    = "${var.vpc_id}"  
  ingress {  
    from_port = 80  
    to_port   = 80  
    protocol  = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
  egress {  
    from_port = 0  
    to_port   = 0  
    protocol  = "-1"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
  tags = { ...}  
}
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: grupos de seguridad

- **Infra/sec/main.tf** : Seguridad para servidores BBDD

```
resource "aws_security_group" "my-database" {
  name      = "database"
  description = "Allow MySQL/Aurora from WebService"
  vpc_id    = "${var.vpc_id}"
  ingress {
    from_port    = 3306
    to_port      = 3306
    protocol     = "tcp"
    security_groups = ["${aws_security_group.my-
webserver.id}"]
    self         = false
  }
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {...}
}
```


Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: grupos de seguridad

- **Infra/sec/output.tf** : Salidas grupos de Seguridad

```
output "sg_web" {  
  value = "${aws_security_group.my-webserver.id}"  
}  
output "sg_db" {  
  value = "${aws_security_group.my-database.id}"  
}
```

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: grupos de seguridad

- **Infra/infra/output.tf** : Para utilizar las variables de salida desde el modulo Infra

```
# Net module output
output "vpc" {
  value = "${module.network.vpc}"
}
output "sn_pub1" {
  value = "${module.network.sn_pub1}"
}
output "sn_pub2" {
  value = "${module.network.sn_pub2}"
}
output "sn_priv1" {
  value = "${module.network.sn_priv1}"
}
output "sn_priv2" {
  value = "${module.network.sn_priv2}"
}
```

```
# Sec module output
output "sg_web" {
  value = "${module.security.sg_web}"
}
output "sg_db" {
  value = "${module.security.sg_db}"
}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: Desplegando el VPC

Infra/infra/maint.tf

- Export TF_VAR_REGION=us-east-1
- # initialize modules and see changes
- cd ~/tf-projects/infra
- terraform init
- terraform plan
- # create infrastructure
- terraform apply
- # cleanup infrastructure
- terraform destroy

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: Desplegando el VPC

Infra/infra/maint.tf -> *terraform apply*

The image shows a Terraform CLI session on the left and the AWS Management Console on the right. The CLI output displays the configuration for a VPC and its associated resources, followed by the execution of `terraform apply`. The AWS console shows the newly created VPC and subnets.

```
terraform apply -auto-approve
```

```
Plan: 11 to add, 0 to change, 0 to destroy.
```

```
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes
```

```
Apply complete! Resources: 11 added, 0 changed, 0 destroyed.
```

```
Outputs:
sg_db = sg-0984f223d2fa9cb5a
sg_web = sg-0f9c4e7cbba8c1baa
sn_priv1 = subnet-066b0b0b2de286ae7
sn_priv2 = subnet-04f185892a4955b7a
sn_pub1 = subnet-06180c246fbb0aac
sn_pub2 = subnet-05bcd5805a53060cb
vpc = vpc-06c56047f9c57ad10
```

The AWS console shows the following resources:

Name	VPC ID
dev-vpc	vpc-048ddc107c8370
my-vpc	vpc-06c56047f9c57ad10
	vpc-948ee7ee

Name	Subnet ID
dev-public	subnet-02969cefcadc7acba
my-private2	subnet-04f185892a4955b7a
my-public1	subnet-05bcd5805a53060cb
my-public2	subnet-06180c246fbb0aac

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0

Aprovisionamiento de infraestructura: ~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de infraestructura: Desplegando el VPC

Infra/infra/maint.tf -> *terraform destroy*

```
    },
  ] -> null
  name           = "webserver" -> null
  owner_id       = "238854161131" -> null
  revoke_rules_on_delete = false -> null
  tags           = {
    "Name" = "my-webserver"
    "Site" = "my-web-site"
  } -> null
  vpc_id       = "vpc-06c56047f9c57ad10" -> null
}

Plan: 0 to add, 0 to change, 11 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

module.network.aws_internet_gateway.my-igw: Still destroying... [id=vpc-06c56047f9c57ad10]
module.network.aws_internet_gateway.my-igw: Destruction complete after 1s
module.network.aws_vpc.my-main: Destroying... [id=vpc-06c56047f9c57ad10]
module.network.aws_vpc.my-main: Destruction complete after 1s

Destroy complete! Resources: 11 destroyed.
vagrant@manager:~/terraform/tf-ejercicio1/infra$
```

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Web server y Database server

- En esta iteración crearemos
 - Instancias Ec2 para alojar el servidor web
 - Mysql via AWS RDS
 - La aplicación web
- Crear estrucutra y ficheros

```
├── app
│   ├── main.tf
│   ├── user_data.sh
│   └── variables.tf
├── aws.tf
├── db
│   ├── main.tf
│   └── variables.tf
├── main.tf
└── variables.tf
```

```
touch {.,app,db}/{main.tf,variables.tf}
touch app/user_data.sh
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Web server y Database server

/webapp/variables.tf

- En este modulo crearemos las variables que inicializaremos con valores de salida del modulo de infraestructura

variable "profile" {}	# security groups	# config artifact
variable "region" {}	variable "sg_web" {}	variable "database_name" {}
	variable "sg_db" {}	variable "database_user" {}
	# subnets	# secrets artifact
	variable "sn_web" {}	variable "database_password" {}
	variable "sn_db1" {}	# instance key pair
	variable "sn_db2" {}	variable "key_name" {}

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: variables globales

./db.tfvars -> En este fichero se especificarán valores propias del nivel de aplicación (en .gitignore)

```
database_name    = "webdb"  
database_user    = "admin"  
database_password =  
"@U1bO8s$^&GkUAz*I$$@BG87"
```


Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Fichero principal

/webapp/main.tf

- Se definen los modulos en los que se subdivide el proyecto y la ubicacuión de sus carpetas
 - Instances -> ../app
 - db -> ../db
- También se definen las variables a utilizar

```
module "instances" {  
  source = "../app"  
  sg_web = "${var.sg_web}"  
  sn_web = "${var.sn_web}"  
  key_name = "${var.key_name}"  
}
```

```
module "db" {  
  source = "../db"  
  sg_db = "${var.sg_db}"  
  sn_db1 = "${var.sn_db1}"  
  sn_db2 = "${var.sn_db2}"  
  database_name = "${var.database_name}"  
  database_user = "${var.database_user}"  
  database_password = "${var.database_password}"  
}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Variables modulo app

/webapp/app/variables.tf

- Variables a exportar del modulo principal a este submodulo

```
variable "sg_web" {}  
variable "sn_web" {}  
variable "key_name" {}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Modulo app

/webapp/app/main.tf -> Creación de servidores

- Búsqueda de ami -> data.aws_ami.amazon-linux-2.id

```
data "aws_ami" "amazon-linux-2" {
  most_recent = true
  filter {
    name     = "virtualization-type"
    values   = ["hvm"]
  }
  filter {
    name     = "architecture"
    values   = ["x86_64"]
  }
  filter {
    name     = "name"
    values   = ["amzn2-ami-hvm-2.0*"]
  }
  owners = ["137112412989"] # Amazon
}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: USER_DATA

/webapp/app/user_data.sh -> USER_DATA servidores

- Instalación LAMP y aplicación

```
#!/bin/bash -ex
yum -y update
yum -y install httpd php mysql php-mysql
chkconfig httpd on
service httpd start
cd /var/www/html
S3_HOST=s3-us-west-2.amazonaws.com
APP_PATH=us-west-2-aws-training/awsu-spl/spl-13/scripts/app.tgz
wget https://${S3_HOST}/${APP_PATH}
tar xvfz app.tgz
chown apache:root /var/www/html/rds.conf.php
```

Aprovisionamiento de infraestructura:

Terraform

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Modulo app

/webapp/app/main.tf -> Creación de servidores

- Instancia Ec2 : Con USER_DATA ,AMI_id y configuración

```
resource "aws_instance" "my-webserver" {  
  ami          = "${data.aws_ami.amazon-linux-2.id}"  
  instance_type = "t2.micro"  
  key_name     = "${var.key_name}"  
  user_data    = "${file("${path.module}/user_data.sh")}"  
  subnet_id    = "${var.sn_web}"  
  associate_public_ip_address = true  
  vpc_security_group_ids = [  
    "${var.sg_web}",  
  ]  
  tags = {  
    "Name" = "my-webserver"  
    "Site" = "my-web-site"  
  }  
}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Base de datos

/webapp/db/variables.tf -> Parametros para BBDD

```
variable "sg_db" {}  
variable "sn_db1" {}  
variable "sn_db2" {}  
variable "database_name" {}  
variable "database_user" {}  
variable "database_password" {}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Base de datos

/webapp/db/main.tf -> Creación BBDD

- Selección de subredes
- Selección de base de datos

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Base de datos

/webapp/db/main.tf -> Creación BBDD

- Selección de subredes

```
resource "aws_db_subnet_group" "my-dbsg" {  
  name      = "my-dbsg"  
  description = "my-dbsg"  
  subnet_ids = ["${var.sn_db1}", "${var.sn_db2}"]  
  tags = {  
    "Name" = "my-dbsg"  
    "Site" = "my-web-site"  
  }  
}
```


Aprovisionamiento de infraestructura:

Terraform

Ejercicio : Aprovisionamiento AWS

Creación de servidores: Base de datos

/webapp/db/main.tf -> Creación BBDD

```
resource "aws_db_instance" "my-db" {
  identifier      = "my-db"
  allocated_storage = 20
  storage_type    = "gp2"
  engine          = "mysql"
  engine_version  = "5.6.40"
  instance_class  = "db.t2.micro"
  name            = "${var.database_name}"
  username        = "${var.database_user}"
  password        = "${var.database_password}"
  parameter_group_name = "default.mysql5.6"
  db_subnet_group_name = "${aws_db_subnet_group.my-dbsg.id}"
  vpc_security_group_ids = ["${var.sg_db}"]
  # set these for dev db
  backup_retention_period = 0
  # required for deleting
  skip_final_snapshot      = true
  final_snapshot_identifier = "Ignore"
  tags { ... }
}
```

Aprovisionamiento de infraestructura:

~~Terraform~~

Ejercicio : Aprovisionamiento AWS

Script Principal Terraform

- Conectar variables de los dos modulos **infra<->webapp**

```
#### VARIABLES
variable "profile" {}
variable "region" {}
variable "database_name" {}
variable "database_user" {}
variable "database_password" {}
variable "key_name" {
  default = "deploy-aws"
}
```

Aprovisionamiento de infraestructura:

Terraform

Ejercicio : Aprovisionamiento AWS

Script Principal Terraform -> **Definir los dos subproyectos**

```
#### CALL MDOULES
module "core_infra" {
  source = "./infra"
  profile = "${var.profile}"
  region = "${var.region}"
}
```

```
module "webapp" {
  source = "./webapp"
  profile = "${var.profile}"
  region = "${var.region}"
  key_name = "${var.key_name}"
  # pass web security group and public networks
  sg_web = "${module.core_infra.sg_web}"
  sn_web = "${module.core_infra.sn_pub1}"
  # pass database security group and private networks
  sg_db = "${module.core_infra.sg_db}"
  sn_db1 = "${module.core_infra.sn_priv1}"
  sn_db2 = "${module.core_infra.sn_priv2}"
  # database parameters
  database_name = "${var.database_name}"
  database_user = "${var.database_user}"
  database_password = "${var.database_password}"
}
```

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Script Principal Terraform -> Ejecutar

```
# show changes required (using db variables file)
```

```
terraform plan -var-file="db.tfvars"
```

```
# apply changes required (using db variables file)
```

```
terraform apply -var-file="db.tfvars"
```

Validar aplicación

```
terraform show | grep -o 'public_ip = .*$'
```

```
terraform show | grep -o 'endpoint = .*$'
```

my-db.cknof0oc3nnn.us-east-1.rds.amazonaws.com:3306

[illegible]

Address Book

Last name	First name	Phone	Email	Admin	
				Add Contact	
Doe	Jane	010-110-1101	janed@someotheraddress.org	Edit	Remove
Johnson	Roberto	123-456-7890	roberto@someaddress.com	Edit	Remove

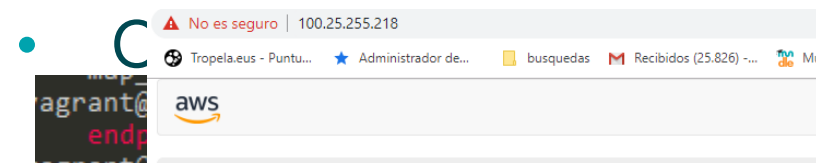
Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Script Principal Terraform -> **Ejecutar**

- Validar aplicación
- Buscar la IP pública del servidor WEB

```
vagrant@manager:~/terraform/tf-ejercicio1$ terraform show | grep 'public_ip'
  associate_public_ip_address = true
  public_ip                   = "100.25.255.218"
  map_public_ip_on_launch     = true
  map_public_ip_on_launch     = true
  map_public_ip_on_launch     = true
  map_public_ip_on_launch     = true
```



```
rm show | grep 'endpoint'
my-db.cer6jmsndi2p.us-east-1.rds.amazonaws.com:3306"
```

- A **configurar Endpoint BBDD**

Endpoint	my-db.cer6jmsndi2p.us-east-1.rds.amazonaws.com:3306
Database	webdb
Username	admin
Password
	<input type="button" value="Submit"/>

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Aprovisionamiento AWS

Script Principal Terraform -> **Ejecutar**

- Abrir aplicación WEB y configurar Endpoint BBDD

No es seguro | 100.25.255.218

Tropela.eus - Puntu... Administrador de... busquedas Recibidos (25.826) -... Misiones

aws

Endpoint my-db.cer6jmsndi2p.us-east-1.rds.amazonaws.com:3306

Database webdb

Username admin

Password

Submit

No es seguro | 100.25.255.218/index.php

Tropela.eus - Puntu... Administrador de... busquedas Recibidos (25.826) -... Mudle

aws

Address Book

Last name	First name	Phone	Email	Admin
				Add Contact
Doe	Jane	010-110-1101	janed@someotheraddress.org	Edit Remove
Johnson	Roberto	123-456-7890	roberto@someaddress.com	Edit Remove

Aprovisionamiento de infraestructura:



Terraform

Ejercicio : Aprovisionamiento AWS

Script Principal Terraform -> **Eliminar recursos**

- terraform destroy -var-file="db.tfvars"



```
}  
- root_block_device {  
  - delete_on_termination = true -> null  
  - encrypted              = false -> null  
  - iops                   = 100 -> null  
  - volume_id              = "vol-09b42a5b4  
  - volume_size            = 8 -> null  
  - volume_type            = "gp2" -> null  
}  
}  
Plan: 0 to add, 0 to change, 14 to destroy.  
module.webapp.module.instances.aws_instance.my-webserver: Still destroying... [id=i-0d277622890edefa0]  
module.webapp.module.instances.aws_instance.my-webserver: Destruction complete after 33s  
module.core_infra.module.network.aws_subnet.my-public1: Destroying... [id=subnet-03362fc4b90]  
module.core_infra.module.network.aws_subnet.my-public1: Destruction complete after 2s  
module.webapp.module.db.aws_db_instance.my-db: Still destroying... [id=my-db, 40s elapsed]  
module.webapp.module.db.aws_db_instance.my-db: Still destroying... [id=my-db, 50s elapsed]  
module.webapp.module.db.aws_db_instance.my-db: Still destroying... [id=my-db, 1m0s elapsed]  
module.core_infra.module.network.aws_subnet.my-private: Still destroying... [id=subnet-03362fc4b90]  
module.core_infra.module.network.aws_subnet.my-private: Destruction complete after 1m40s elapsed  
module.core_infra.module.security.aws_security_group.my-private: Still destroying... [id=sg-03362fc4b90]  
module.core_infra.module.network.aws_vpc.my-main: Destroying... [id=vpc-03362fc4b90]  
module.core_infra.module.network.aws_vpc.my-main: Destruction complete after 1m40s elapsed  
Destroy complete! Resources: 14 destroyed.
```

	my-webserver	i-0d277622890edefa0	t2.micro	us-east-1b	 terminated
---	--------------	---------------------	----------	------------	--

Aprovisionamiento de infraestructura: Terraform

Ejercicio : Ejemplo Wordpress HA con Terraform”

<https://github.com/geass/aws-terraform-workshop>

	my-webserver	i-0d277622890edefa0	t2.micro	us-east-1b	 terminated
---	--------------	---------------------	----------	------------	--