Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

Actividad Abierta Laboratorio: despliegue de MEAN multicapa mediante Terraform

Contenido

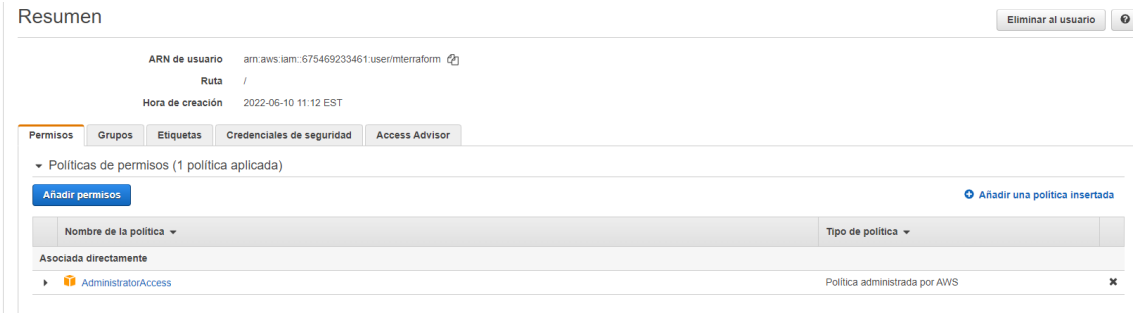
Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

Objetivos

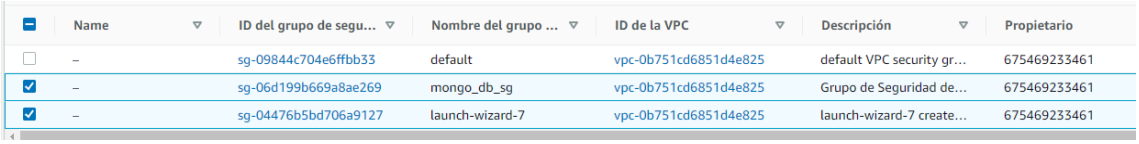
En esta actividad aprenderás a utilizar Terraform con un ejemplo sencillo pero completo. Para ello, usaremos un *stack* MEAN con dos o más máquinas. A través de esta actividad conseguirás familiarizarte con la herramienta Terraform para desplegar conjuntos de componentes que se comuniquen entre ellos.

Configuraciones AWS previas.

Creación de usuario: Se debe crear un usuario con permisos para conectarse a la consola de AWS

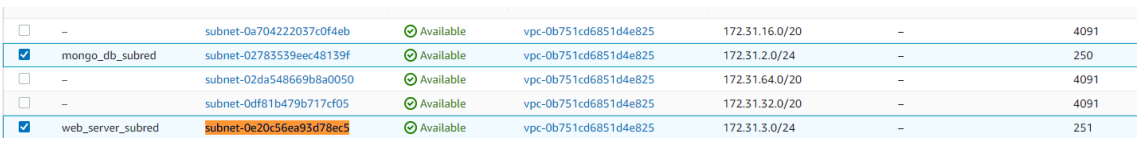


Creación de Security Group: Se creó un security group tanto para el servidor web como para mongodb



Name	ID del grupo de segu...	Nombre del grupo ...	ID de la VPC	Descripción	Propietario
–	sg-09844c704e6ffbb33	default	vpc-0b751cd6851d4e825	default VPC security gr...	675469233461
–	sg-06d199b669a8ae269	mongo_db_sg	vpc-0b751cd6851d4e825	Grupo de Seguridad de...	675469233461
–	sg-04476b5bd706a9127	launch-wizard-7	vpc-0b751cd6851d4e825	launch-wizard-7 create...	675469233461

Creación de Subredes: Una subred utilizará el Servidor Web y la otra la utilizará MongoDB. Las direcciones ip serán **172.31.2.0/24** para mongo y **172.31.3.0/24** para el servidor Web



–	subnet-0a704222037c0f4eb	Available	vpc-0b751cd6851d4e825	172.31.16.0/20	–	4091
mongo_db_subred	subnet-02783539eec48139f	Available	vpc-0b751cd6851d4e825	172.31.2.0/24	–	250
–	subnet-02da548669b8a0050	Available	vpc-0b751cd6851d4e825	172.31.64.0/20	–	4091
–	subnet-0df81b479b717cf05	Available	vpc-0b751cd6851d4e825	172.31.32.0/20	–	4091
web_server_subred	subnet-0e20c56ea93d78ec5	Available	vpc-0b751cd6851d4e825	172.31.3.0/24	–	251

Se tiene la siguiente configuración:

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

USUARIO CONSOLA AWS: **mterraform**

USUARIO ACCESO EC2 SSH

KeyName: **mromero**

Claves PEM: **mromero.pem**

SECURITY GROUP

mongo_sg: sg-06d199b669a8ae269

app_sg: sg-04476b5bd706a9127

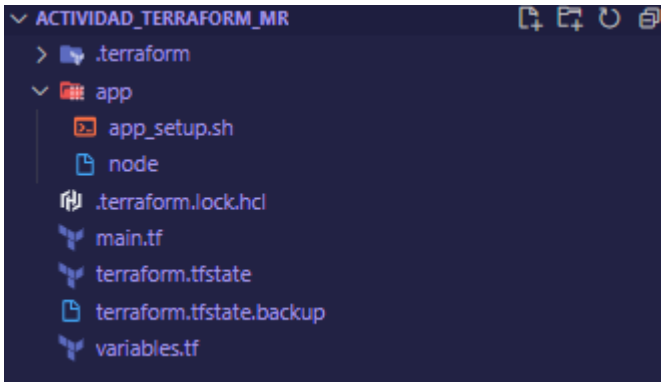
SUBREDES

mongo_subnet: subnet-02783539eec48139f

app_subnet: subnet-0e20c56ea93d78ec5

Instancias y Despliegue.

A continuación, se muestran los ficheros utilizados para terraform.



Ficheros de configuración

main.tf: Este es el archivo de configuración principal que define y despliega los recursos.

Definir la versión requerida tanto del provider como de terraform.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
```

```
  }
  required_version = ">= 0.14.9"
}
```

Configuración del provider de “aws”

```
#Credenciales para conectarse a AWS
provider "aws" {
  profile    = "default"
  region     = "us-east-1"
  access_key = var.aws_access_key
  secret_key = var.aws_secret_key
}
```

Recurso de configuración de MongoDB. Se utilizan las variables de ami, las security group, las subnets y he definido una ip privada.

```
#Desplegando la instancia de MONGO DB
resource "aws_instance" "mongodb" {
  ami           = var.mongo_ami
  instance_type = "t2.micro"

  vpc_security_group_ids = var.mongo_sg
  subnet_id              = var.mongo_subnet
  private_ip             = var.mongo_priv_ip
  tags = {
    Name = "MongoDB"
  }
}
```

Datasource para obtener la versión más reciente de la IAM (AMI) de Ubuntu que fue la que se utilizó en esta práctica

```
#Usamos una AMI del catalogo de AWS
data "aws_ami" "ubuntu" {
  #Usamos la version reciente
  most_recent = true
  #Colocamos el ID de la AMI
  owners = ["099720109477"]

  #Un filtro ya que el owner puede tener varias versiones de ubuntu.
  filter {
```

```
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-*"]
  }
}
```

Recurso de Web Server: Las configuraciones iniciales son primero, referenciamos a la ami de Ubuntu creada. Seteamos las variabls del key_name, la subnet, la ip privada, el security group y pedimos una ip pública a la instancia

```
#Despliegue del servidor web
resource "aws_instance" "app_server" {
  ami                         = data.aws_ami.ubuntu.id
  instance_type               = "t2.micro"
  key_name                    = var.key_name
  subnet_id                   = var.app_subnet
  private_ip                  = var.app_priv_ip
  vpc_security_group_ids      = var.app_sg
  associate_public_ip_address = true
  tags = {
    Name = "WebServer"
  }
```

Provisioners:

Primer Provisioner: La conexión se realiza mediante SSH a la consola AWS con la clave .pem ya definida. Este provisioner es de tipo “file”, donde enviará el contenido de nuestro fichero **hello.js** a la carpeta de destino /tmp de la instancia creada.

Además contiene el contenido que queremos mostrar en el servidor y utiliza la ip privada de nuestra instancia de MongoDB para colocarlo en el Connection string.

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

```
# hello.js paso como parametro la ip privada para el conecction string
hacia mongoDb
  provisioner "file" {
    content = <<-EOT
     const http = require('http');

     const hostname = 'localhost';
     const port = 8080;

     const server = http.createServer((req, res) => {
      res.statusCode = 200;
      res.setHeader('Content-Type', 'text/plain');
      res.end("HOLA UNIR! Soy Marcelo Romero! \nConnection string to
MongoDb: mongodb://${aws_instance.mongodb.private_ip}:27017");
     });

     server.listen(port, hostname, () => {
      console.log("Server running at http://"+hostname+":"+port+"/");
     });
    EOT
    #Destino del fichero
    destination = "/tmp/hello.js"

    connection {
      type        = "ssh"
      user        = "ubuntu"
      private_key = file("~/Downloads/credentials/mromero.pem")
      host        = self.public_ip
    }
  }
```

Segundo Provisioner: Este provisioner toma y copia todo el contenido del script app_setup.sh a la ruta de la instancia /tmp

Este archivo app_setup.sh contiene los pasos necesarios para configurar nuestra app de node.js

```
 #En la instancia desplegada copiamos el fichero app_setup que tiene los
comandos necesarios para desplegar la aplicacion
  provisioner "file" {
    source      = "app/app_setup.sh"
```

```
    destination = "/tmp/app_setup.sh"

    connection {
      type        = "ssh"
      user        = "ubuntu"
      private_key = file("~/Downloads/credentials/mromero.pem")
      host        = self.public_ip
    }
  }
```

Tercer Provisioner: En este provisioner se copia el contenido del archivo de node a la ruta /tmp de la instancia, además este archivo contiene la configuración de puertos de escucha que la aplicación necesita para mostrarse y se coloca en nuestro servidor nginx para su despliegue

```
  #copiando el archivo node en la configuracion de nginx
  provisioner "file" {
    source      = "app/node"
    destination = "/tmp/node"

    connection {
      type        = "ssh"
      user        = "ubuntu"
      private_key = file("~/Downloads/credentials/mromero.pem")
      host        = self.public_ip
    }
  }
```

Cuarto Provisioner: Este provisioner, realiza la ejecución remota de comandos en la instancia, primero da permisos al fichero app_setup.sh y luego lo ejecuta

```
#modifcando los permisos del script y ejecutamos
  provisioner "remote-exec" {
    inline = ["chmod +x /tmp/app_setup.sh", "/tmp/app_setup.sh", ]

    connection {
      type        = "ssh"
      user        = "ubuntu"
      private_key = file("~/Downloads/credentials/mromero.pem")
```

```
      host        = self.public_ip
    }
  }
```

Archivo variables.tf: El siguiente archivo tiene definidas todas las variables que se utilizan en el archivo principal

```
# Credenciales AWS
variable "aws_access_key" {
  type = string
  default = "AKIAZ2RI2RU2TPQNIFHE"
}

variable "aws_secret_key" {
  type = string
  default = "5sbo4MKbDc/cipeQsLIfZKcq7DP1Am9+BX73S31Q"
}


variable "key_name" {
  type = string
  default = "mromero"
}
# VARIABLES DE MONGODB

variable "mongo_ami" {
  type    = string
  default = "ami-0019f1e85386a77e1"
}

variable "mongo_sg" {
  type    = list(string)
  default = ["sg-06d199b669a8ae269"]
}
variable "mongo_subnet" {
  type    = string
  default = "subnet-02783539eec48139f"
}

variable "mongo_priv_ip" {
  type    = string
  default = "172.31.2.25"
}
```

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

```
# Application variables
variable "app_priv_ip" {
  type    = string
  default = "172.31.3.15"
}

variable "app_sg" {
  type    = list(string)
  default = ["sg-04476b5bd706a9127"]
}
variable "app_subnet" {
  type    = string
  default = "subnet-0e20c56ea93d78ec5"
}
```

Contenido del archivo node

```
server {
   listen 80;
   server_name example.com;

   location / {
      proxy_set_header   X-Forwarded-For $remote_addr;
      proxy_set_header   Host $http_host;
      proxy_pass         http://127.0.0.1:8080;
   }
}
```

Archivo app_setup.sh: Este archivo script contiene los pasos necesarios para configurar el servidor y desplegar nuestra aplicación

```
#!/usr/bin/env bash
sleep 30

# Install node js
curl -fsSL https://deb.nodesource.com/setup_15.x | sudo -E bash -
sudo apt-get install -y nodejs
sudo apt install build-essential -y
```

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

```
# Install pm2
sudo npm install -g pm2

# Install NGINX
sudo apt update -y
sudo apt-get install nginx -y
systemctl enable nginx
sudo rm /etc/nginx/sites-enabled/default
sudo mv /tmp/node /etc/nginx/sites-available/node
sudo ln -s /etc/nginx/sites-available/node /etc/nginx/sites-enabled/node
# sudo systemctl restart nginx
service nginx restart

# Setup firewall
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
ufw enable


# Configure pm2 to run hello on startup
mkdir -p ~/code/app-dist
mv /tmp/hello.js ~/code/app-dist/hello.js
cd  ~/code/app-dist/
sudo pm2 start hello.js
sudo pm2 startup systemd
sudo pm2 save
sudo pm2 list

sudo shutdown -r now
```

Logs

El primer comando que se utiliza para inicializar nuestro proyecto es **terraform init**

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

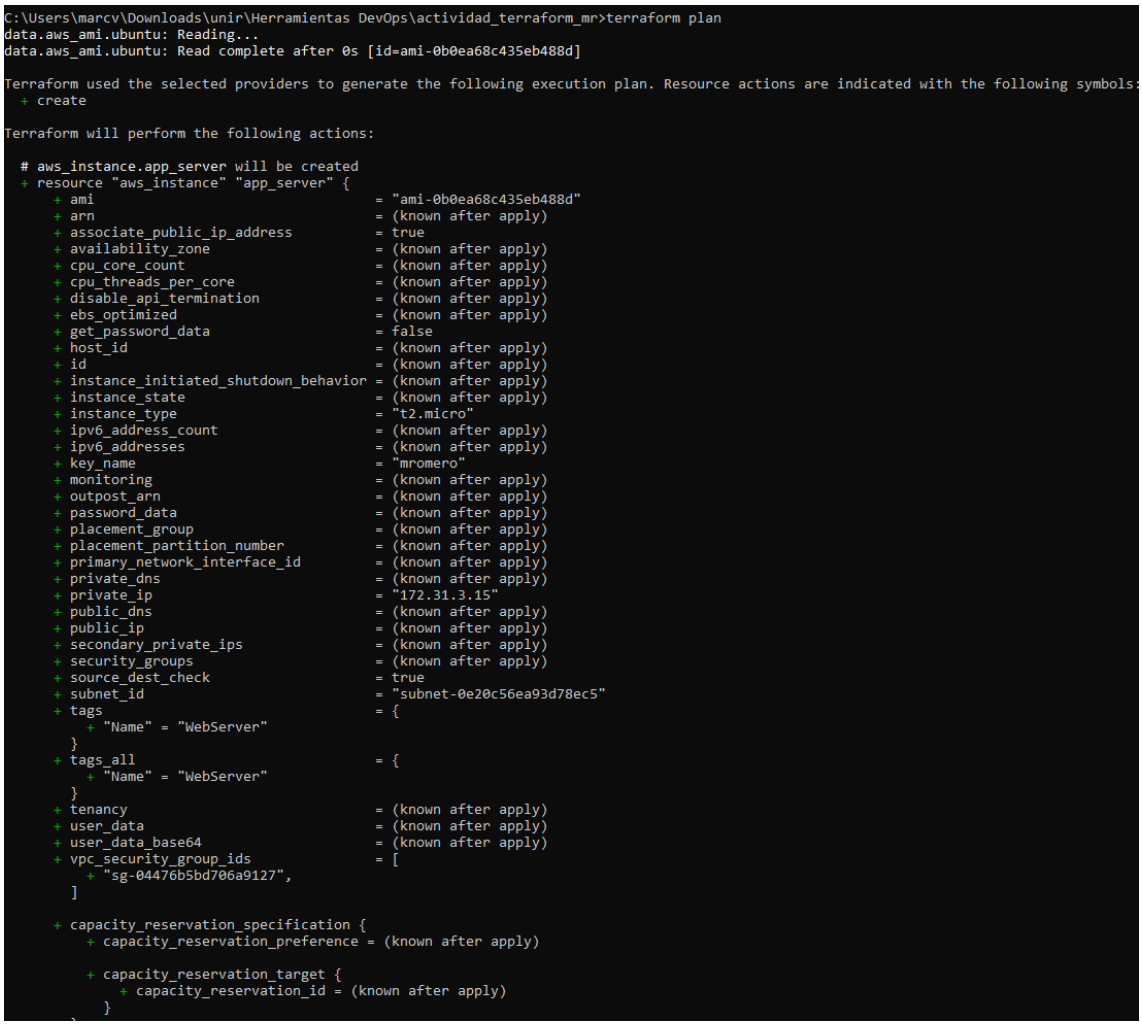
```
C:\Users\marcv\Downloads\unir\Herramientas DevOps\actividad_terraform_mr>terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v3.75.2

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.
```

El siguiente comando que se debe hacer por buena práctica es **terraform validate** el cual revisa que todo lo de nuestro archivo main.tf esté correcto.

```
C:\Users\marcv\Downloads\unir\Herramientas DevOps\actividad_terraform_mr>terraform validate
Success! The configuration is valid.
```

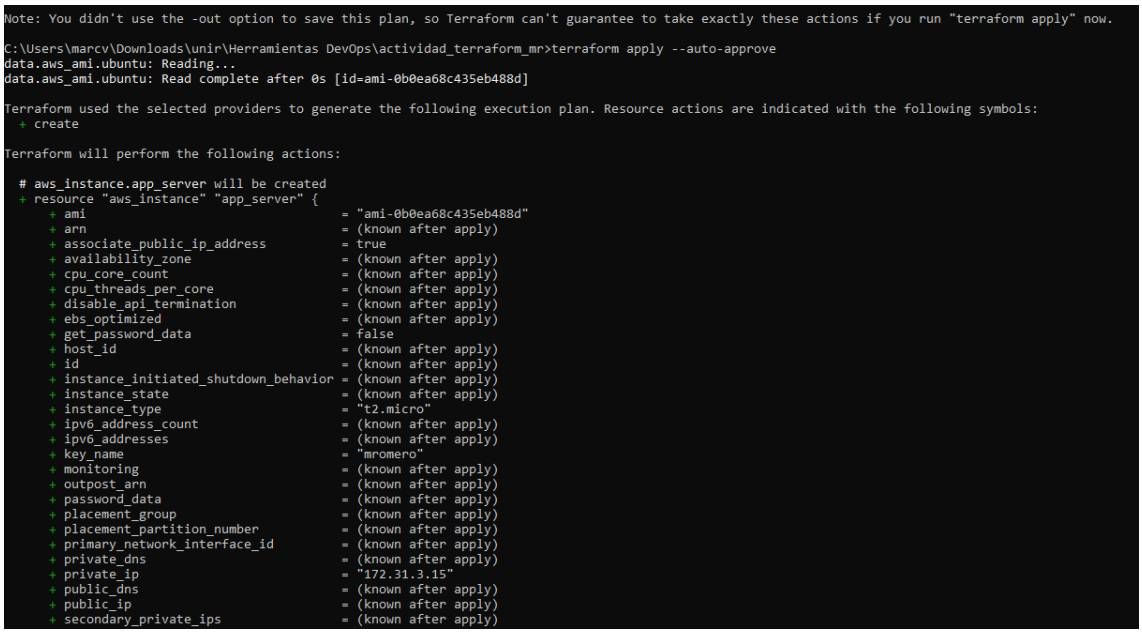
El siguiente comando que se debe realizar es **terraform plan** para comprobar lo que vamos a desplegar en AWS, además que permite comprobar que todos los recursos tengan los parámetros necesarios para crearlos. Indica que se va a crear una instancia llamada app_server y una llamada mongodb

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

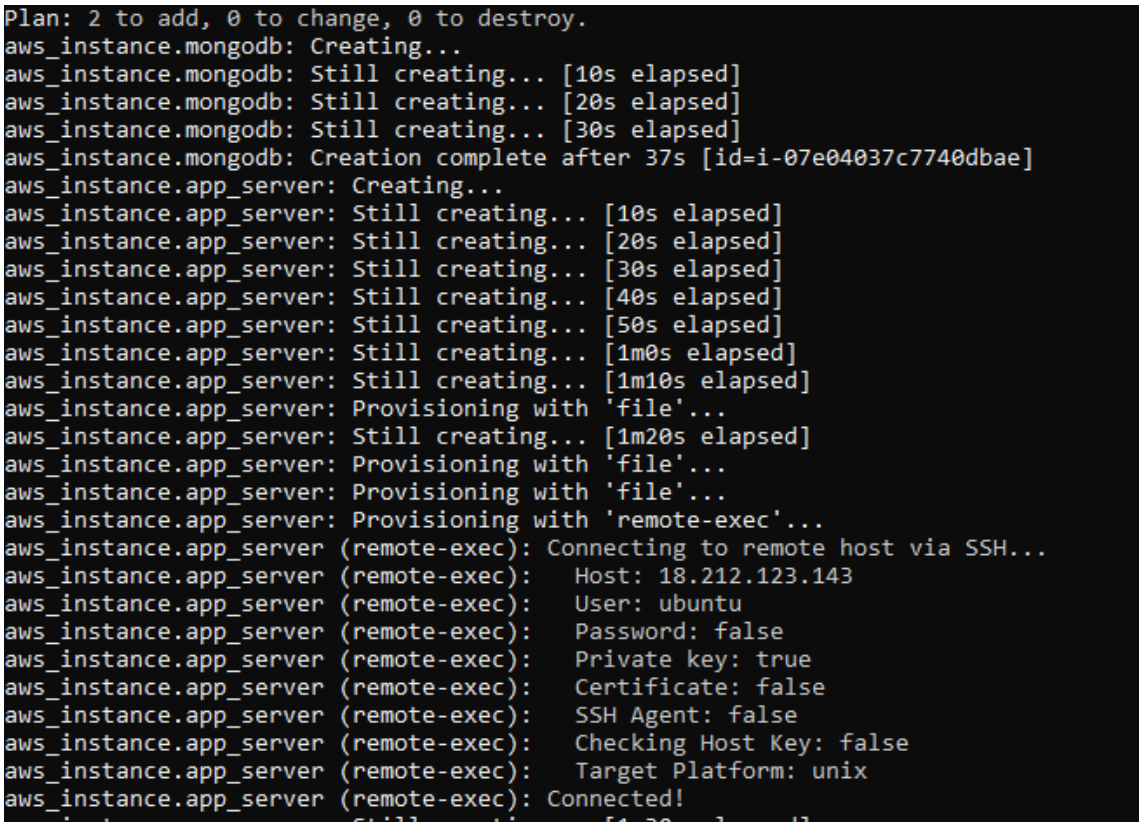


Al momento de aplicar el comando **terraform apply –auto-approve** el despliegue de nuestra app iniciará y no necesita pedir cofirmación

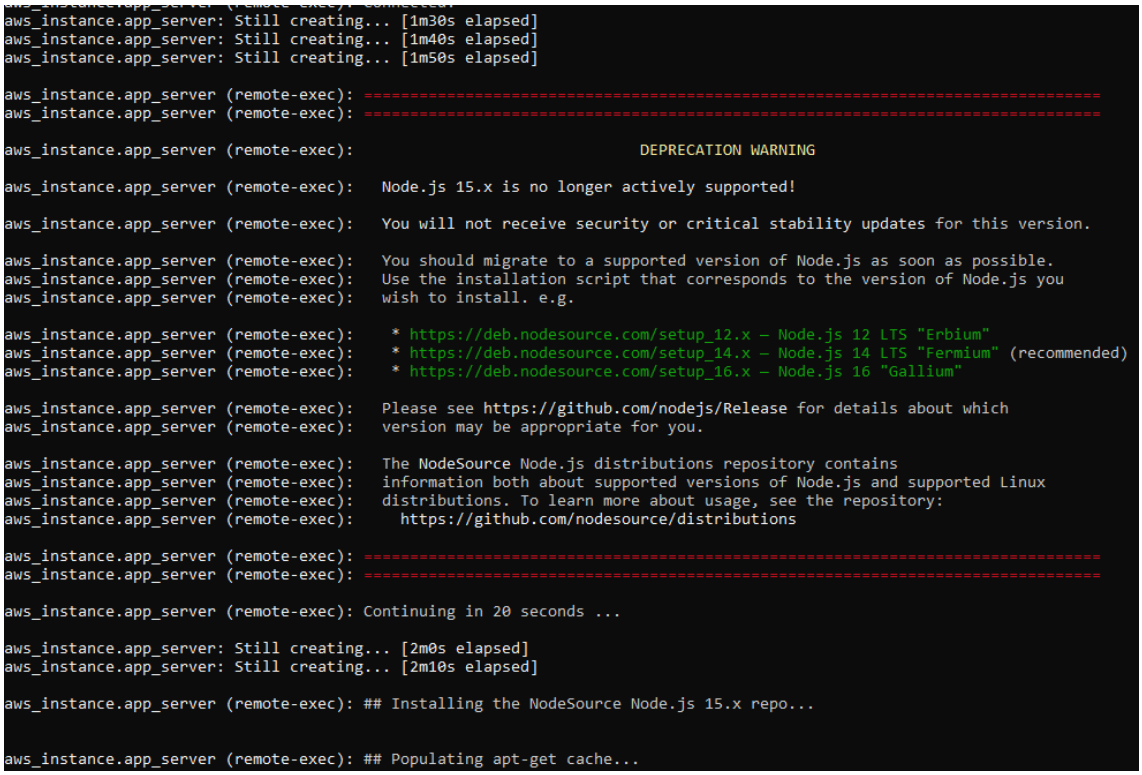
Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

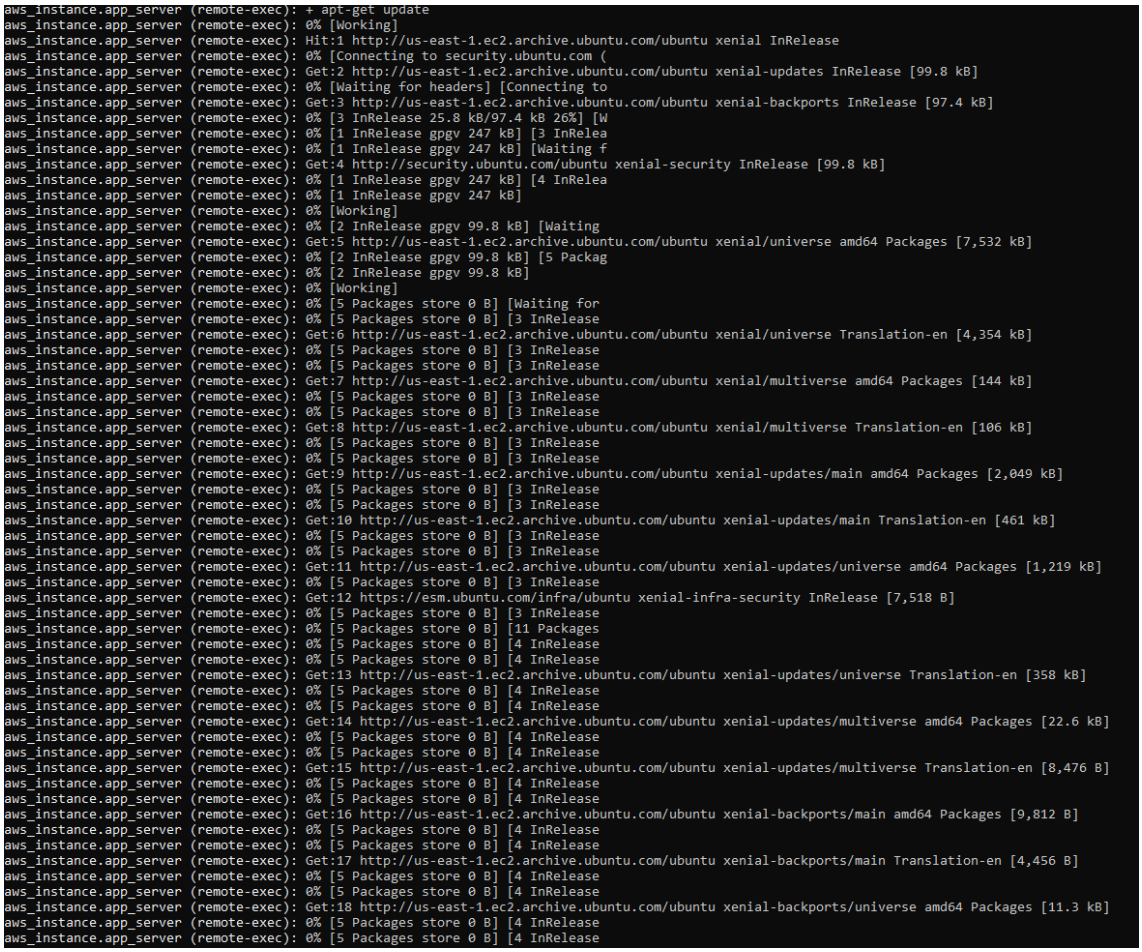


LOGS DE DESPLIEGUE



Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	



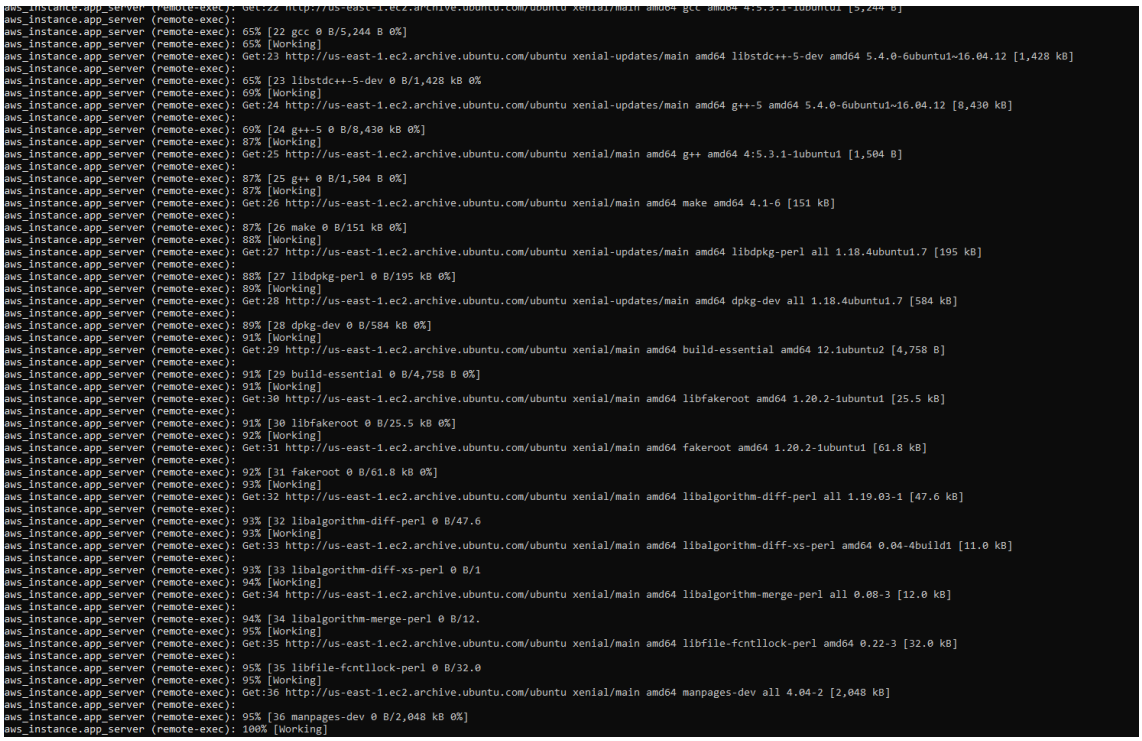


Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo Nombre: Marcelo Vicente	10/06/2022

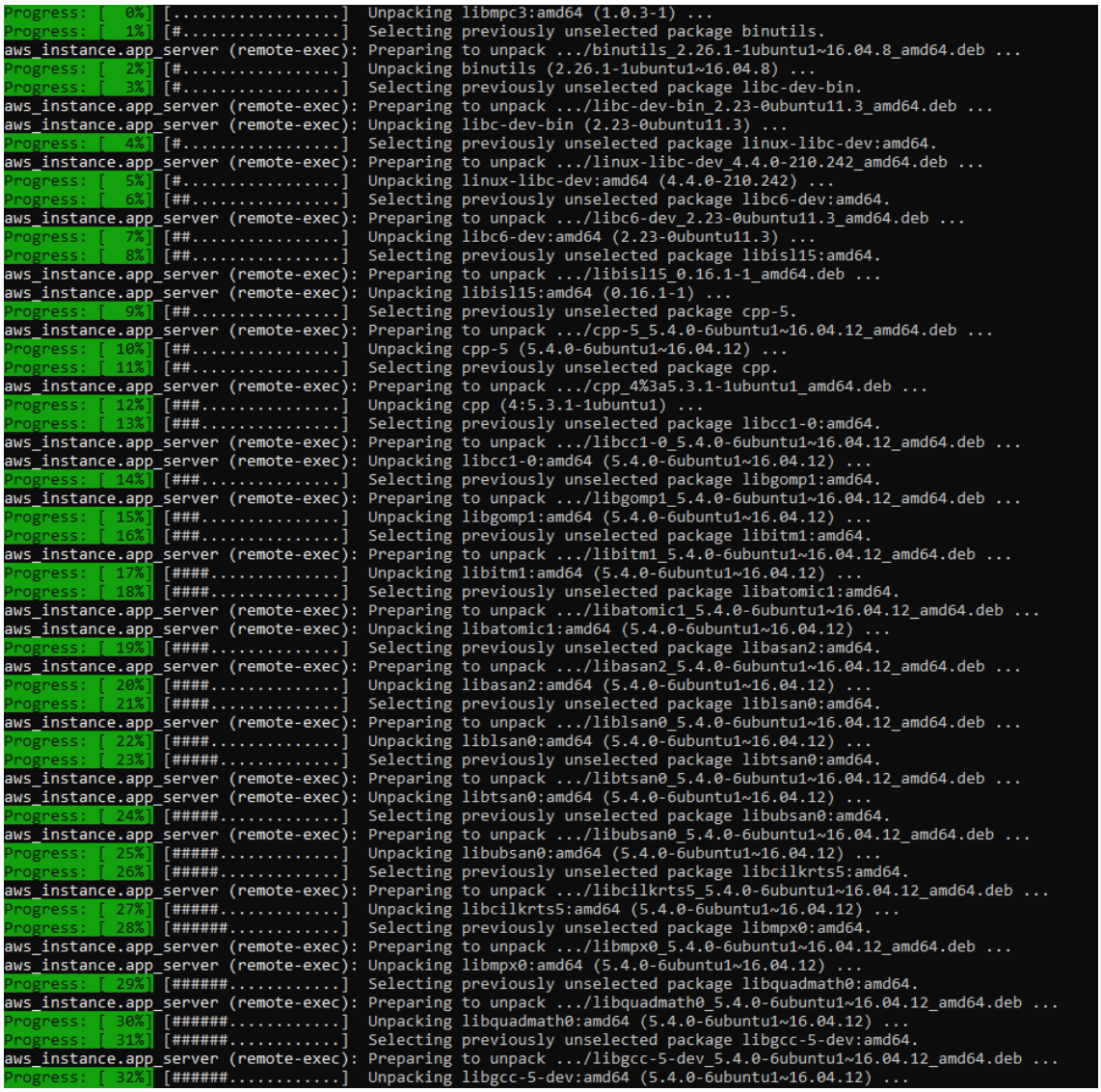
```
aws_instance.app_server (remote-exec): Get:20 https://esm.ubuntu.com/infra/ubuntu xenial-infra-updates InRelease [7,475 B]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [4 InRelease
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): Get:21 http://security.ubuntu.com/ubuntu xenial-security/main Translation-en [360 kB]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): Get:22 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 Packages [785 kB]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): Get:23 http://security.ubuntu.com/ubuntu xenial-security/universe Translation-en [225 kB]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): Get:24 http://security.ubuntu.com/ubuntu xenial-security/multiverse amd64 Packages [7,864 B]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): Get:25 http://security.ubuntu.com/ubuntu xenial-security/multiverse Translation-en [2,672 B]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [12 InRelease
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [20 InRelease
aws_instance.app_server (remote-exec): Get:26 https://esm.ubuntu.com/infra/ubuntu xenial-infra-security/main amd64 Packages [450 kB]
aws_instance.app_server (remote-exec): 0% [5 Packages store 0 B] [20 InRelease
aws_instance.app_server (remote-exec): 93% [5 Packages store 0 B] [26 Packages
aws_instance.app_server (remote-exec): 95% [5 Packages store 0 B]
aws_instance.app_server (remote-exec): 95% [5 Packages store 0 B]
aws_instance.app_server (remote-exec): 95% [Working]
aws_instance.app_server (remote-exec): 95% [6 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 95% [6 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 95% [Working]
aws_instance.app_server (remote-exec): 95% [7 Packages store 0 B]
aws_instance.app_server (remote-exec): 96% [Working]
aws_instance.app_server (remote-exec): 96% [8 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 96% [Working]
aws_instance.app_server (remote-exec): 96% [9 Packages store 0 B]
aws_instance.app_server (remote-exec): 96% [Working]
aws_instance.app_server (remote-exec): 96% [10 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 96% [Working]
aws_instance.app_server (remote-exec): 96% [11 Packages store 0 B]
aws_instance.app_server (remote-exec): 97% [Working]
aws_instance.app_server (remote-exec): 97% [13 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 97% [Working]
aws_instance.app_server (remote-exec): 97% [14 Packages store 0 B]
aws_instance.app_server (remote-exec): 97% [Working]
aws_instance.app_server (remote-exec): 97% [15 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 97% [Working]
aws_instance.app_server (remote-exec): 97% [16 Packages store 0 B]
aws_instance.app_server (remote-exec): 98% [Working]
aws_instance.app_server (remote-exec): 98% [17 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 98% [Working]
aws_instance.app_server (remote-exec): 98% [18 Packages store 0 B]
aws_instance.app_server (remote-exec): 98% [Working]
aws_instance.app_server (remote-exec): 98% [19 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 98% [Working]
aws_instance.app_server (remote-exec): 98% [21 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 99% [Working]
aws_instance.app_server (remote-exec): 99% [22 Packages store 0 B]
aws_instance.app_server (remote-exec): 99% [Working]
aws_instance.app_server (remote-exec): 99% [23 Translation-en store 0 B]
aws_instance.app_server (remote-exec): 99% [Working]
aws_instance.app_server (remote-exec): 99% [24 Packages store 0 B]
aws_instance.app_server (remote-exec): 99% [Working]
aws_instance.app_server (remote-exec): 99% [25 Translation-en store 0 B]
```

```
aws_instance.app_server (remote-exec): Reading package lists... Done

aws_instance.app_server (remote-exec): ## Confirming "xenial" is supported...

aws_instance.app_server (remote-exec): + curl -sLf -o /dev/null 'https://deb.nodesource.com/node_15.x/dists/xenial/Release'

aws_instance.app_server (remote-exec): ## Adding the NodeSource signing key to your keyring...

aws_instance.app_server (remote-exec): + curl -s https://deb.nodesource.com/gpgkey/nodesource.gpg.key | gpg --dearmor | tee /usr/share/keyrings/nodesource.gpg >/dev/null

aws_instance.app_server (remote-exec): ## Creating apt sources list file for the NodeSource Node.js 15.x repo...

aws_instance.app_server (remote-exec): + echo 'deb [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_15.x xenial main' > /etc/apt/sources.list.d/nodesource.list
aws_instance.app_server (remote-exec): + echo 'deb-src [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_15.x xenial main' >> /etc/apt/sources.list.d/nodesource.list

aws_instance.app_server (remote-exec): ## Running `apt-get update` for you...

aws_instance.app_server (remote-exec): + apt-get update
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
aws_instance.app_server (remote-exec): 0% [Waiting for headers] [Connecting to
aws_instance.app_server (remote-exec): Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
aws_instance.app_server (remote-exec): Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
aws_instance.app_server (remote-exec): Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): 0% [1 InRelease gpgv 247 kB]
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): Get:5 https://deb.nodesource.com/node_15.x xenial InRelease [4,584 B]
aws_instance.app_server (remote-exec): Hit:6 https://esm.ubuntu.com/infra/ubuntu xenial-infra-security InRelease
aws_instance.app_server (remote-exec): Hit:7 https://esm.ubuntu.com/infra/ubuntu xenial-infra-updates InRelease
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): 0% [2 InRelease gpgv 99.8 kB]
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): 0% [3 InRelease gpgv 97.4 kB]
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): 0% [4 InRelease gpgv 99.8 kB]
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): 0% [5 InRelease gpgv 4,584 B]
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): 0% [6 InRelease gpgv 7,518 B]
aws_instance.app_server (remote-exec): 0% [Working]
aws_instance.app_server (remote-exec): 0% [7 InRelease gpgv 7,475 B]
aws_instance.app_server (remote-exec): 88% [Working]
aws_instance.app_server (remote-exec): Get:8 https://deb.nodesource.com/node_15.x xenial/main amd64 Packages [762 B]
aws_instance.app_server (remote-exec): 100% [Working]
aws_instance.app_server (remote-exec): 100% [8 Packages store 0 B]
aws_instance.app_server (remote-exec): 100% [Working]
```

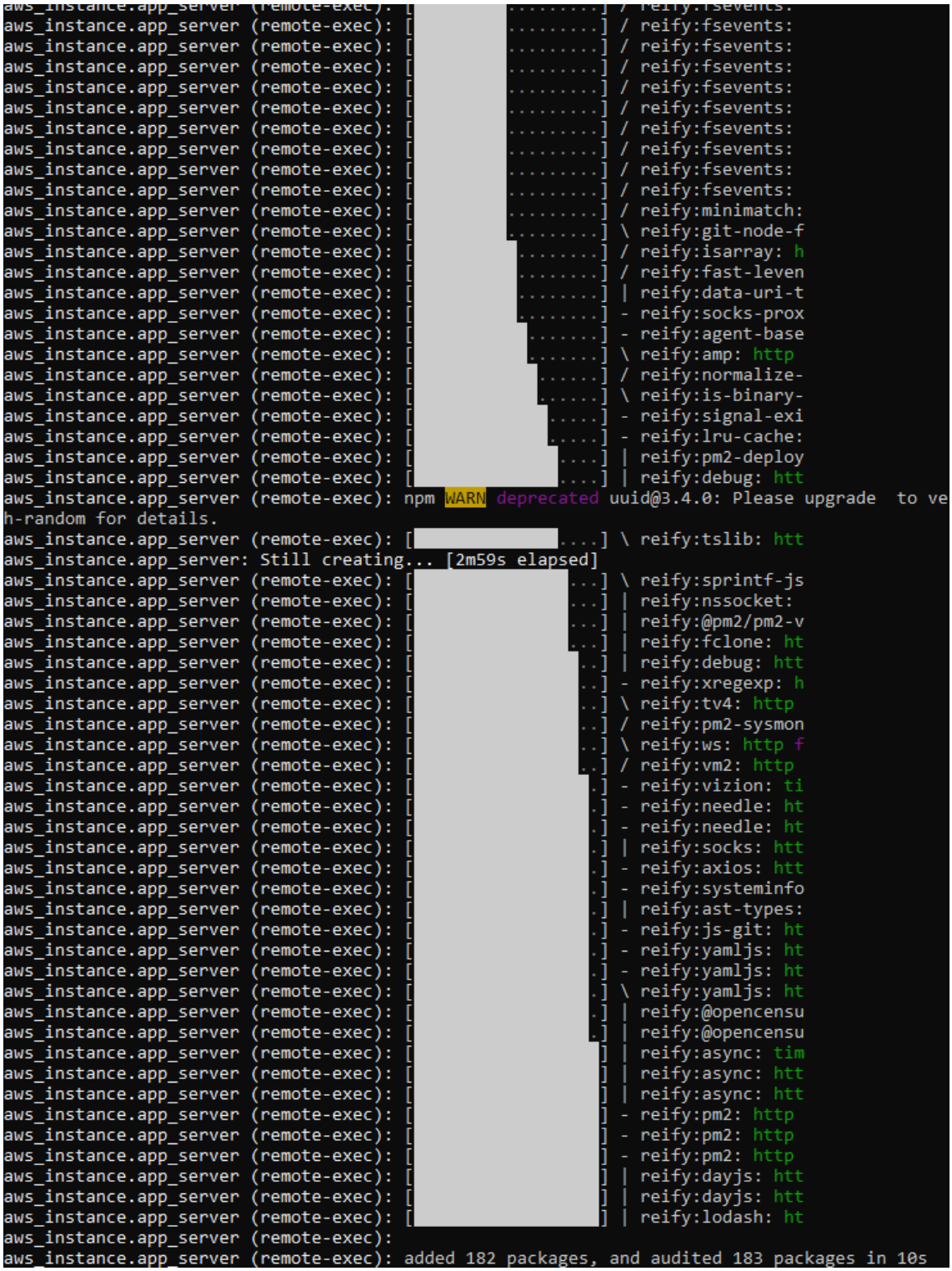
Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	



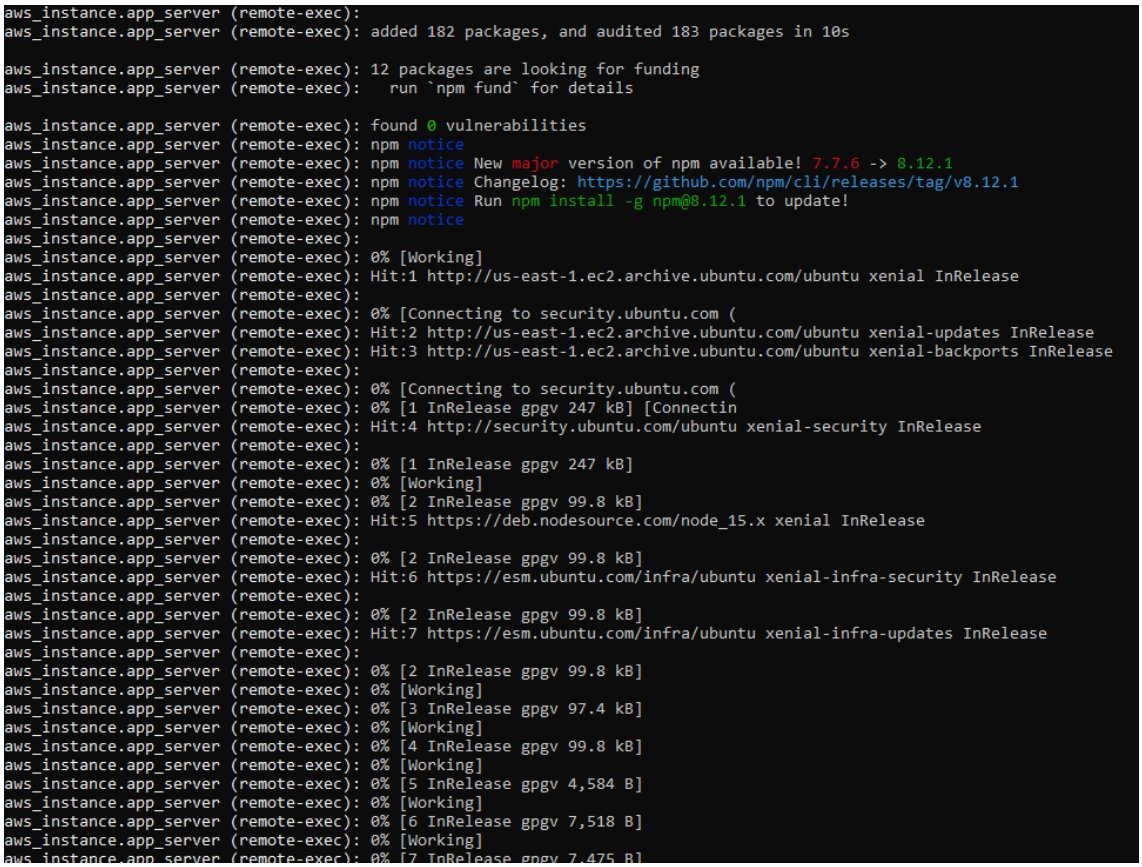
Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	



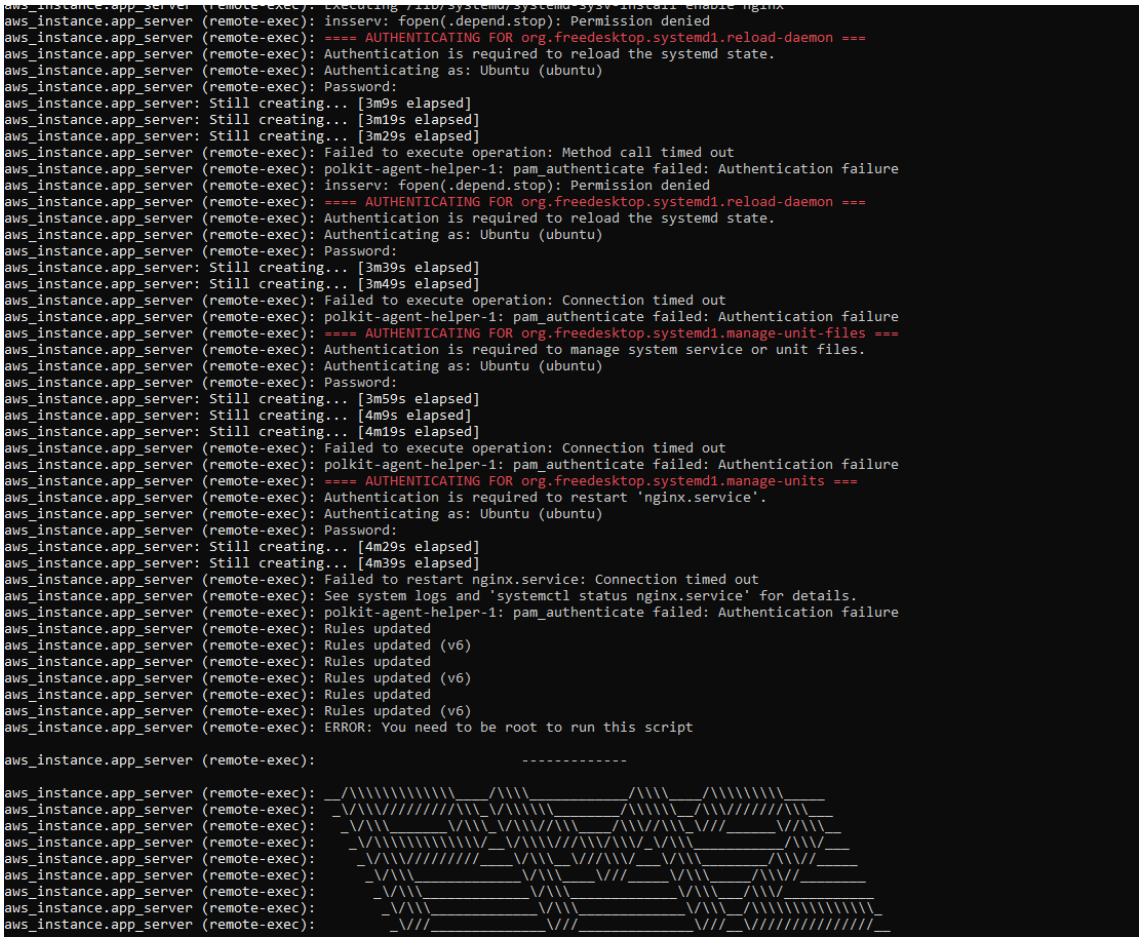
Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

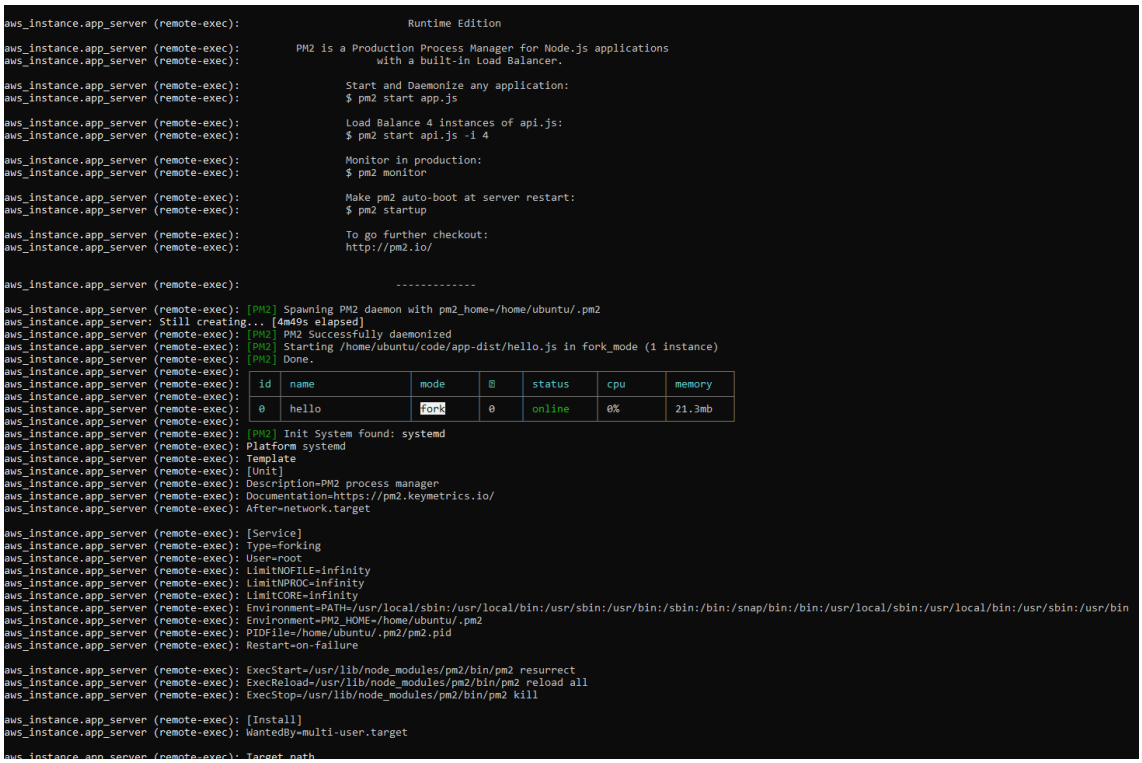


Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	



Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	



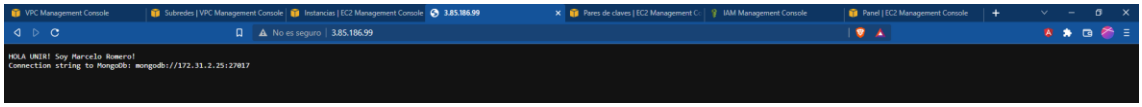


Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Romero Astudillo	10/06/2022
	Nombre: Marcelo Vicente	

```
aws_instance.app_server (remote-exec): Description=PM2 process manager
aws_instance.app_server (remote-exec): Documentation=https://pm2.keymetrics.io/
aws_instance.app_server (remote-exec): After=network.target

aws_instance.app_server (remote-exec): [Service]
aws_instance.app_server (remote-exec): Type=forking
aws_instance.app_server (remote-exec): User=root
aws_instance.app_server (remote-exec): LimitNOFILE=infinity
aws_instance.app_server (remote-exec): LimitNPROC=infinity
aws_instance.app_server (remote-exec): LimitCORE=infinity
aws_instance.app_server (remote-exec): Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/bin:/usr/local/sbin:/usr/
aws_instance.app_server (remote-exec): Environment=PM2_HOME=/home/ubuntu/.pm2
aws_instance.app_server (remote-exec): PIDFile=/home/ubuntu/.pm2/pm2.pid
aws_instance.app_server (remote-exec): Restart=on-failure

aws_instance.app_server (remote-exec): ExecStart=/usr/lib/node_modules/pm2/bin/pm2 resurrect
aws_instance.app_server (remote-exec): ExecReload=/usr/lib/node_modules/pm2/bin/pm2 reload all
aws_instance.app_server (remote-exec): ExecStop=/usr/lib/node_modules/pm2/bin/pm2 kill

aws_instance.app_server (remote-exec): [Install]
aws_instance.app_server (remote-exec): WantedBy=multi-user.target

aws_instance.app_server (remote-exec): Target path
aws_instance.app_server (remote-exec): /etc/systemd/system/pm2-root.service
aws_instance.app_server (remote-exec): Command list
aws_instance.app_server (remote-exec): [ 'systemctl enable pm2-root' ]
aws_instance.app_server (remote-exec): [PM2] Writing init configuration in /etc/systemd/system/pm2-root.service
aws_instance.app_server (remote-exec): [PM2] Making script booting at startup...
aws_instance.app_server (remote-exec): [PM2] [-] Executing: systemctl enable pm2-root...
aws_instance.app_server (remote-exec): Created symlink from /etc/systemd/system/multi-user.target.wants/pm2-root.service to /etc/systemd/system/pm2-root.
aws_instance.app_server (remote-exec): [PM2] [v] Command successfully executed.
aws_instance.app_server (remote-exec): +---------------------------------------+
aws_instance.app_server (remote-exec): [PM2] Freeze a process list on reboot via:
aws_instance.app_server (remote-exec): $ pm2 save

aws_instance.app_server (remote-exec): [PM2] Remove init script via:
aws_instance.app_server (remote-exec): $ pm2 unstartup systemd
aws_instance.app_server (remote-exec): [PM2] Saving current process list...
aws_instance.app_server (remote-exec): [PM2] Successfully saved in /home/ubuntu/.pm2/dump.pm2
aws_instance.app_server (remote-exec):
aws_instance.app_server (remote-exec):
aws_instance.app_server (remote-exec):
aws_instance.app_server (remote-exec):
```

id	name	mode	↺	status	cpu	memory
0	hello	fork	0	online	0%	40.4mb

Resultados.



Video.

El siguiente enlace lleva al video de youtube de la actividad.

https://www.youtube.com/watch?v=LhgJEZL7i1I

Conclusiones.

- Terraform es una herramienta de aprovisionamiento de infraestructura de código muy completa, y además fácil de utilizar ya que su código es declarativo, permite que sin tener conocimientos previos puedas trabajar sin muchos problemas
- Se logró comprender y validar como funciona la herramienta de terraform sin muchos problemas, pudiendo conectar las dos instancias mediante código.
- Se debe tener una buena configuración previa de AWS ya que esto puede ocasionar problemas al momento de realizar el despliegue con terraform.