



Certified Tech Developer

The Ultimate Degree

Infraestructura II

Actividad obligatoria y grupal

Dificultad: alta

Proceso de despliegue completo

En esta oportunidad vamos a agregar un servidor web a nuestra infraestructura y luego —con otro pipeline— vamos a desplegar un sitio web.

Requisitos

Este trabajo es una continuación de los anteriores, por lo que necesitamos tener completas las prácticas de las clases 25 y 26.

Infraestructura

Para arrancar con esta práctica, vamos a generar una clave SSH para nuestro servidor. Para esto, ejecutamos el comando **ssh-keygen** y cambiamos el nombre y ubicación de la key.

DigitalHouse>

```

root@923bacbf8fee:~/Documents/dh# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): ./ec2
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./ec2
Your public key has been saved in ./ec2.pub
The key fingerprint is:
SHA256:lw87LmP7YaQdGmZbbZw5Ba/Yq9a20cHB9gogiz92ob0 root@923bacbf8fee
The key's randomart image is:
+----[RSA 3072]-----+
|
| .
| . =
| . . ++0|
| . 0 + +0+0|
| . S 0 = +0.|
| + % B . + .
| 0 0 ..+ .
| .+= +..0.|
| ..=E.....|
+-----[SHA256]-----+

```

Para agregar el servidor en nuestro repositorio de Terraform vamos a sumar un archivo que defina un EC2 y las variables para su clave de acceso por SSH.

En el archivo **variables.tf** agregaremos lo siguiente:

```
variable "ec2_sshkey" {
    description = "ssh key para ingresar al servidor"
    type        = string
    default     = "Se pasa por variable de entorno"
}
```

Pasamos el código requerido para nuestra instancia que tendrá el rol de servidor a otro archivo: **servidor.tf**

```
module "key_pair" {
  source = "terraform-aws-modules/key-pair/aws"
  key_name    = "key"
  public_key = var.ec2_sshkey
}

data "aws_ami" "amazon-linux-2" {
  most_recent = true
  owners      = ["amazon"]
  filter {
    name     = "name"
    values   = ["amzn2-ami-hvm*"]
  }
}
```



```
}

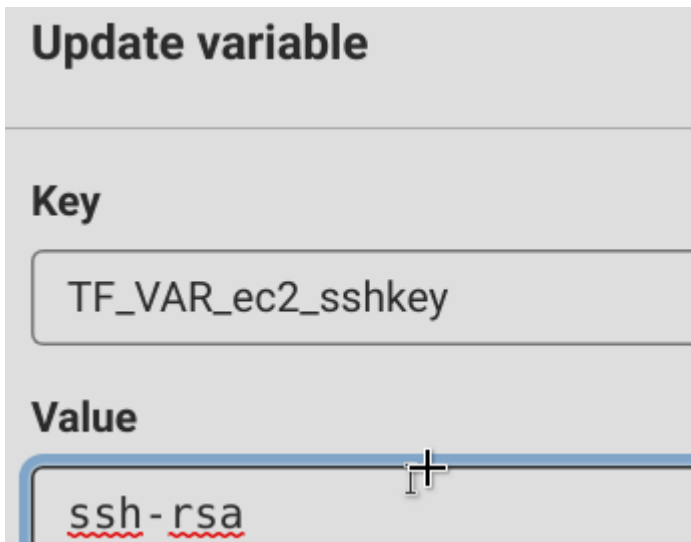
resource "aws_security_group" "server" {
  vpc_id      = aws_vpc.Main.id
  ingress {
    protocol   = "tcp"
    from_port  = 22
    to_port    = 22
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    protocol   = "tcp"
    from_port  = 80
    to_port    = 80
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    protocol   = -1
    from_port  = 0
    to_port    = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "server" {
  ami                  = data.aws_ami.amazon-linux-2.id
  instance_type        = "t2.micro"
  key_name             = module.key_pair.key_pair_key_name
  subnet_id           = aws_subnet.public_subnets.id
  vpc_security_group_ids = [aws_security_group.server.id]
  associate_public_ip_address = true
  user_data            = <<-EOF
                        #!/bin/bash
                        yum -y install httpd
                        sudo systemctl enable httpd
                        sudo systemctl start httpd
                        EOF
}
```

```
output "DNS" {  
  value = aws_instance.server.public_dns  
}
```

Como podés notar, no ponemos explícitamente la clave SSH. Vamos a configurar este valor como variable de entorno en GitLab.

Para hacerlo, nos dirigimos a **Settings >> CI/CD >> variables >> [Add variable]** e insertamos el nombre y el contenido de nuestra clave SSH privada y la guardamos.



Update variable




Key

TF_VAR_ec2_sshkey

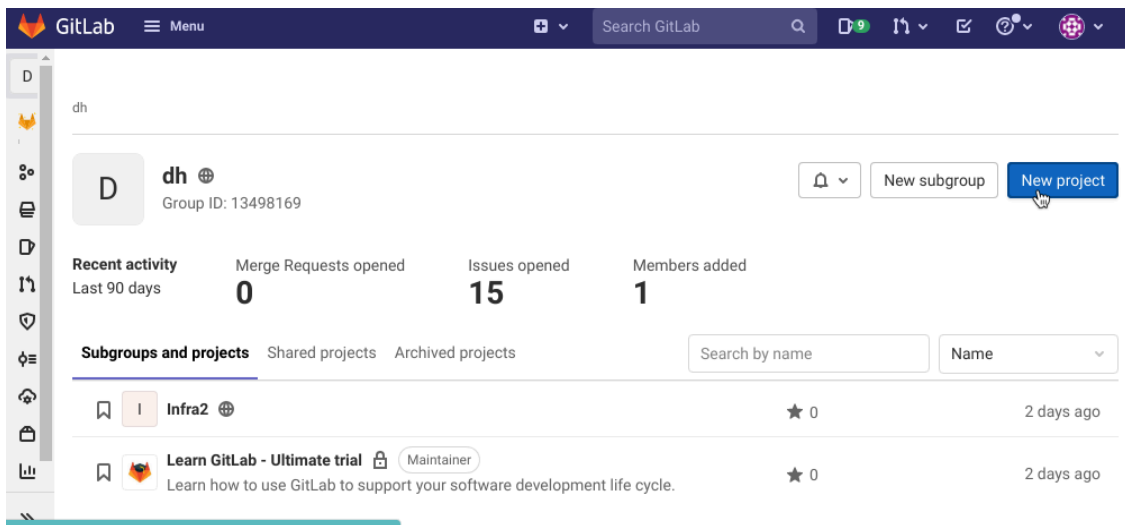
Value

ssh-rsa

¡Ya tenemos lista la infraestructura y podemos correr nuestro pipeline para desplegarla!

Status	Pipeline ID	Triggerer	Commit	Stages	Duration
 passed	#378256777 latest		master → a8927638 out ip		00:04:20 13 minutes ago

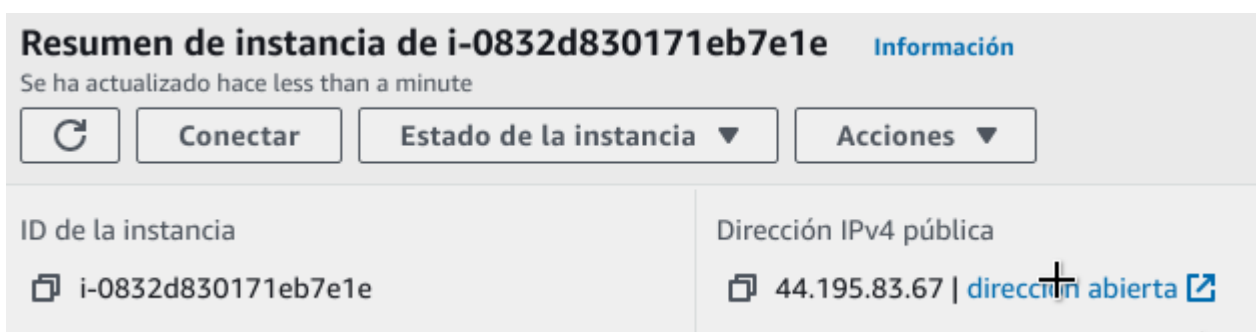
A continuación, vamos a generar un repositorio para desplegar un sitio web en la infraestructura anterior. Primero necesitamos un repositorio nuevo, nos dirigimos a nuestra cuenta y creamos uno.



Agregaremos un archivo **index.html** con el siguiente contenido:

```
<!DOCTYPE html>
<html>
  <body>
    <h1>web</h1>
    <p>Esta es mi web.</p>
  </body>
</html>
```

Vamos a buscar la IP de nuestra máquina virtual. Nos dirigimos a ella en la consola de AWS y la copiamos.



Si abrimos la IP en un explorador web debería mostrar la página de prueba de nuestro servidor.

Test Page

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.


If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting `www.example.com`, you should send e-mail to "`webmaster@example.com`".

If you are the website administrator:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

You are free to use the image below on web sites powered by the Apache HTTP Server:



Para seguir, vamos a crear un par de variables en este nuevo repositorio (vamos a trabajar en la sección CI/CD),

La primera será la clave SSH para poder subir el archivo a la instancia, y la segunda la IP de nuestro servidor.

Variables



Collapse

Variables store information, like passwords and secret keys, that you can use in job scripts. [Learn more.](#)

Variables can be:

- Protected:** Only exposed to protected branches or tags.
- Masked:** Hidden in job logs. Must match masking requirements. [Learn more.](#)

Environment variables are configured by your administrator to be **protected** by default.

Type	↑ Key	Value	Protected	Masked	Environments	
Variable	EC2_IP	44.195.83.67	×	×	All (default)	
Variable	SSHKEY	-----BEGIN OPENS... PRIVATE ...	×	×	All (default)	

Add variable

Hide values

(Esta vez usamos la clave privada)

Luego de completar este paso, agregamos el archivo de nuestro pipeline que subirá nuestro archivo index.

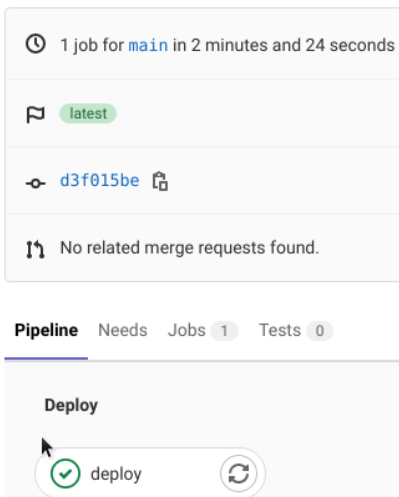
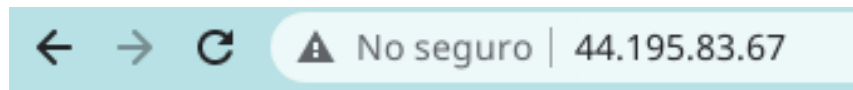
.gitlab-ci.yml

```
image: debian:latest
stages:
  - init
  - deploy
before_script:
  - apt update && apt-get install -qq openssh-client
  - echo "$SSHKEY" > sshkey
  - chmod 400 sshkey
deploy:
  stage: deploy
  script:
    - ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i ./sshkey ec2-user@$EC2_IP
    sudo chmod -R 777 /var/www/html
    - scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i ./sshkey ./index.html
    ec2-user@$EC2_IP:/var/www/html/index.html
```

(Esta es una de las versiones más minimalistas de pipelines que podemos construir)

Una vez corrido nuestro pipeline —lo que puede pasar de forma automática—, veremos los cambios en nuestro sitio web.

Update .gitlab-ci.yml file

web

Esta es mi web.

¡Desarrollamos un CI/CD completo y funcional! Podemos ir a cualquiera de nuestros repositorios, el de infraestructura o el del sitio web, y aplicar los cambios.