UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LAB. SOFTWARE AVANZADO “A”

AUX. DIEGO RENE MOLINA ROLDAN

DOCUMENTACIÓN PRÁCTICA 1 - SA

JONATHAN MARCOS VALIENTE GONZÁLEZ

201931581

07-02-2024

DOCUMENTACIÓN

NOTAS:

- Se realizó la instalación de Terraform en Archlinux
- Las máquinas EC2 se usó Amazon Linux ami-0277155c3f0ab2930

Proceso de instalación de Terraform

1. sudo pacman -S terraform
2. terraform init
3. terraform apply

Configuración del archivo de Terraform

```
# Copyright (c) HashiCorp, Inc.
# SPDX-License-Identifier: MPL-2.0
/*terraform {
   required_providers {
     aws = {
       source = "hashicorp/aws"
       version = "~> 3.5.0"
     }
   }

}*/

provider "aws" {
 region = "us-east-1"
 }

provider "random" {}

# Creacion de la llave pem para la instancia
resource "tls_private_key" "keyRSA" {
 algorithm = "RSA"
 rsa_bits  = 2048
}

#Crea el archivo con la llave
resource "local_file" "private_key" {
 filename = "${path.module}/keyRSA_aws.pem"
 content  = tls_private_key.keyRSA.private_key_pem
}

# Crea el par de llaves
```

```
resource "aws_key_pair" "keyRSA_pair" {
 key_name   = "keyRSA_aws"
 public_key = tls_private_key.keyRSA.public_key_openssh
}


# Para crear nombres aleatorios
resource "random_pet" "name" {}

# Instancia EC2 master
resource "aws_instance" "master" {
   ami           = "ami-0277155c3f0ab2930"
   instance_type = "t2.micro"
   vpc_security_group_ids = [aws_security_group.web-sg.id]
   key_name = aws_key_pair.keyRSA_pair.key_name
   tags = {
       Name = "master-${random_pet.name.id}"
   }

   provisioner "file" {
       source      = "../dist"
       destination = "dist"

       connection {
       type        = "ssh"
       user        = "ec2-user"
       private_key = tls_private_key.keyRSA.private_key_pem
       host        = self.public_ip
       }
   }

   provisioner "file" {
       source      = "../ansible_config"
       destination = "ansible_config"

       connection {
       type        = "ssh"
       user        = "ec2-user"
       private_key = tls_private_key.keyRSA.private_key_pem
       host        = self.public_ip
       }
   }
```

```
   provisioner "remote-exec" {
       inline = [
           "sudo yum update",
           "sudo yum install ansible -y",
           "echo '${tls_private_key.keyRSA.private_key_pem}' >
keyRSA_aws.pem",  # Copiar la clave SSH a la instancia
           "chmod 400 keyRSA_aws.pem && chmod 600 keyRSA_aws.pem",  #
Ajustar los permisos de la clave
           "echo
\"[clients]\n${aws_instance.client.public_dns}\n[clients:vars]\nansible
_ssh_common_args='-o StrictHostKeyChecking=no' \" > inventory.ini",
           "mv ansible_config/* ./",
           "rm -rf ansible_config/",
           "ansible-playbook -i inventory.ini client-ansible.yml
--private-key ./keyRSA_aws.pem",

       ]

       connection {
       type        = "ssh"
       user        = "ec2-user" # Usuario de la instancia EC2 (puede
variar según la AMI)
       private_key = tls_private_key.keyRSA.private_key_pem  # Utiliza
la clave PEM generada por tls_private_key
       host        = self.public_ip
       }
   }
}
# Instancia EC2 cliente
resource "aws_instance" "client" {
   ami           = "ami-0277155c3f0ab2930"
   instance_type = "t2.micro"
   #user_data     = file("init-script.sh")
   vpc_security_group_ids = [aws_security_group.web-sg.id]
   key_name = aws_key_pair.keyRSA_pair.key_name
   tags = {
       Name = "client-${random_pet.name.id}"
   }
}

data "http" "my_ip" {
 url = "http://ipv4.icanhazip.com"
}
```

```
resource "aws_security_group" "web-sg" {
   name = "${random_pet.name.id}-sg"
   ingress {
       from_port = 80
       to_port = 80
       protocol = "tcp"
       cidr_blocks = ["0.0.0.0/0"]
   }

   ingress {
       from_port   = 22
       to_port     = 22
       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"]
   }
}
```

Proceso de instalación de Ansible

1. sudo yum update [actualizamos la paqueteria]
2. sudo yum install ansible -y [instalamos ansible]
3.
```
"echo \"[clients]\n${aws_instance.client.public_dns}\n[clients:vars]\nansible_ssh_common_args='-o StrictHostKeyChecking=no' \" > inventory.ini", #Se establece desde terraform el host de la máquina esclava
```

4.
```
ansible-playbook -i inventory.ini client-ansible.yml --private-key ./keyRSA_aws.pem #Se ejecuta el playbook
```

Configuración del playbook de ansible

```
---
- name: Despliegue de una pagina estatica con Nginx
 hosts: clients
 become: yes
 tasks:
   - name: Actualizar paquetes con yum
     ansible.builtin.yum:
       name: '*'
       state: latest
     become: yes
     become_user: root

   - name: Instalar Nginx
     ansible.builtin.yum:
       name: nginx
       state: present

   - name: Copiar la carpeta de la página estática desde el nodo
controlador al cliente
     ansible.builtin.copy:
       src: ./dist
       dest: /usr/share/nginx/html

   #- name: Copiar la carpeta de la página estática desde el nodo
controlador al cliente
   #  ansible.builtin.synchronize:
   #    src: ./dist
   #    dest: /usr/share/nginx/html/dist
   #    mode: pull
   #  delegate_to: localhost
   #  vars:
   #    ansible_ssh_extra_args: '-o StrictHostKeyChecking=no -i
/home/ec2-user/keyRSA_aws.pem'
   #    ansible_sudo: yes
   #    ansible_sudo_user: root
   #    ansible_sudo_exe: sudo
   #    ansible_rsync_path: /usr/bin/rsync
```

```
   - name: Copiar el archivo de configuración de Nginx desde el nodo
controlador al cliente
     ansible.builtin.template:
       src: ./nginx.conf
       dest: /etc/nginx/nginx.conf
     notify: restart nginx

 handlers:
   - name: restart nginx
     systemd:
       name: nginx
       state: restarted
```

Configuración del nginx

```
worker_processes  1;


events {
   worker_connections  1024;
}


http {
   include       mime.types;
   default_type  application/octet-stream;

   sendfile        on;
   keepalive_timeout  65;

   server {
       listen       80;
       server_name  localhost;

       location / {
           root   /usr/share/nginx/html/dist;
           index  index.html index.htm;
           try_files $uri $uri/ /dist/index.html;
       }

       error_page   500 502 503 504  /50x.html;
```

```
        location = /50x.html {
            root   /usr/share/nginx/html;
        }
    }

}
```

Definir como funciona Terraform

Es un software que permite realizar levantar, configurar y desplegar arquitecturas por medio de un código de alto nivel. Terraform permite establecer recursos (componentes de la arquitectura) y configurarlas desde un archivo de extensión .tf para desplegar estos recursos.

Definir como funciona Ansible

Es una herramienta de software libre que permite orquestar nodos y ejecutar instrucciones por medio de módulos. Se basa en la arquitectura cliente-servidor, donde un nodo maestro le da instrucciones a los nodos clientes y utiliza módulos que son instrucciones guardadas en ansible para realizar cambios en los sistemas de los nodos clientes.

Describir tecnologías a usadas

- AWS
- Ansible
- Terraform
- SSH
- React VITE
- Página Estática
- NGINX
- Amazon Linux