TEC | Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Documentación Proyecto 1

Redes Grupo 2

Profesor: Nereo Campos

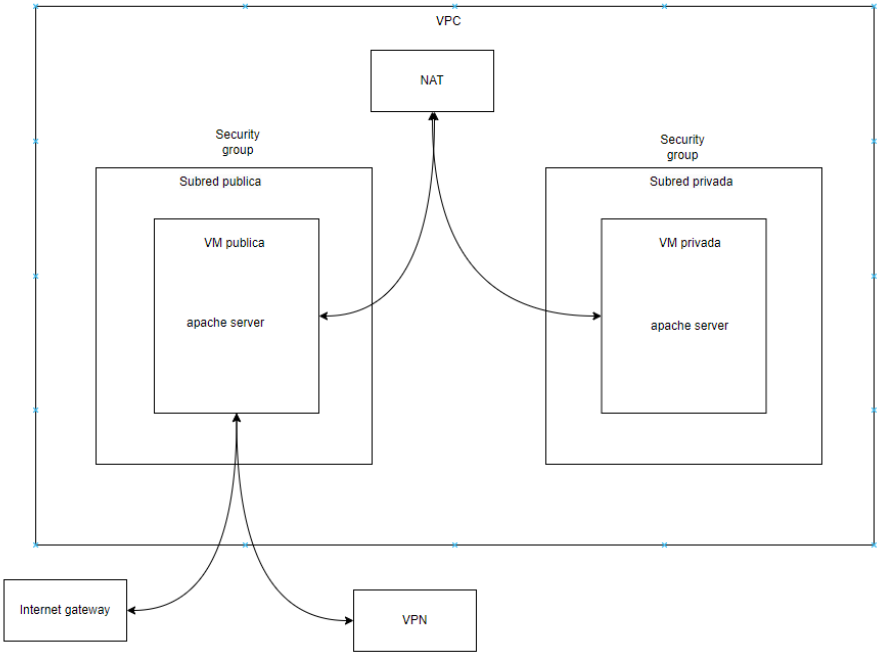
Estudiante:
Juan Pablo Soto Rojas - 2017113948

Fecha de entrega: Viernes 20 de Octubre del 2023

Contenidos

Objetivos

El objetivo principal para este proyecto es crear una red con la siguiente forma



Dependencias

Para ejecutar este proyecto solo se requiere de:

- Cuenta de AWS
- Session local de AWS
- SSH keys en AWS llamadas testKey
- Instalación de Terraform
- Codigo de terraform

Pruebas

Server Apache2 publico

Se obtiene el IP de la máquina pública, cuyo server de apache escucha en el puerto 443

```
Outputs:

private_ip = "10.0.1.190"
public_ip = "3.140.195.116"
vpn_ip = "3.142.197.104"
```

Al hacer curl al apache VM público se pueden obtener los contenidos del

```
PS C:\Users\juanp\OneDrive\Desktop\temp\ii23\redews\Proyecto 1> curl 3.131.38.183:443


StatusCode        : 200
StatusDescription : OK
Content           : <h1>Hello world!</h1><h2>Mi apache publico!</h2>

RawContent        : HTTP/1.1 200 OK
                    Keep-Alive: timeout=5, max=100
                    Connection: Keep-Alive
                    Accept-Ranges: bytes
                    Content-Length: 49
                    Content-Type: text/html
                    Date: Fri, 13 Oct 2023 08:54:39 GMT
                    ETag: "31-60795167db745...
Forms             : {}
Headers           : {[Keep-Alive, timeout=5, max=100], [Connection, Keep-Alive], [Accept-Ranges, bytes],
                    [Content-Length, 49]...}
Images            : {}
InputFields       : {}
Links             : {}
```

Resultado: exitoso.

Server Apache2 privado

En la iteración anterior no se podía tener salida a internet por medio de la máquina privada por un error en la configuración del nat gateway. Por lo cual no fue posible instalar apache, en esta iteración eso se corrigió y podemos verificar que el apache de la VM privada está vivo por el puerto 80, al llamarlo a su IP privada desde la VM pública.

```
ubuntu@ip-10-0-0-61:~$ curl 10.0.1.139
<h1>Hello world!</h1><h2>Servidor privado!</h2>
```

Resultado: exitoso.

VPN

Se instaló en una nueva instancia de EC2, aunque se pudo haber instalado en el la VM pública con unas instrucciones en userdata, se decidió mantener este aspecto modularizado.
Para esta instancia se creó un script que automatiza lo más posible el proceso de configuración del servidor. Por el momento se tiene que inicializar el servidor manualmente.
Para conectarnos al servidor podemos usar el usuario openvpnas

```
> ssh -i testKey.pem openvpnas@3.145.196.140
```

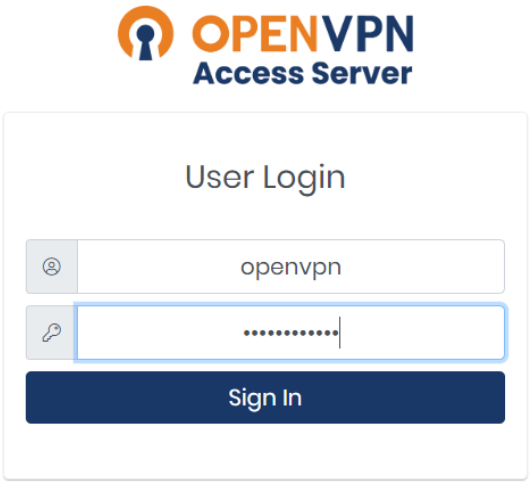
Al finalizar el cuestionario (todo en default) se puede observar que se genera una contraseña, la cual es usada para acceder a los servicios del VPN. Si se pudiera automatizar el seteo de esta contraseña se podría automatizar el resto, pero no se alcanzó a completar esta tarea.

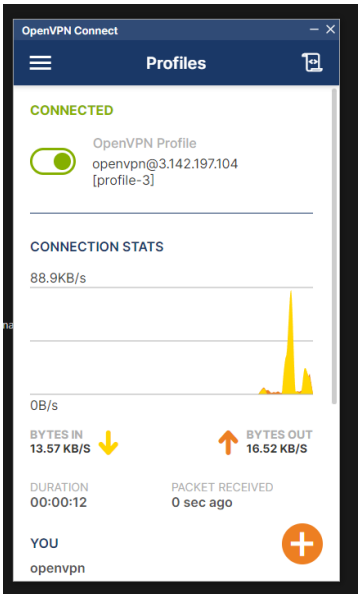
```
https://3.142.197.104:943/admin

During normal operation, OpenVPN AS can be accessed via these URLs:
Admin  UI: https://3.142.197.104:943/admin
Client UI: https://3.142.197.104:943/
To login please use the "openvpn" account with "gbV39zD47cXA" password.

See the Release Notes for this release at:
   https://openvpn.net/vpn-server-resources/release-notes/
```

Después de haber configurado el servidor se puede seguir con el uso del vpn
https://openvpn.net/quick-start-guide/





Resultado: parcial

Instrucciones

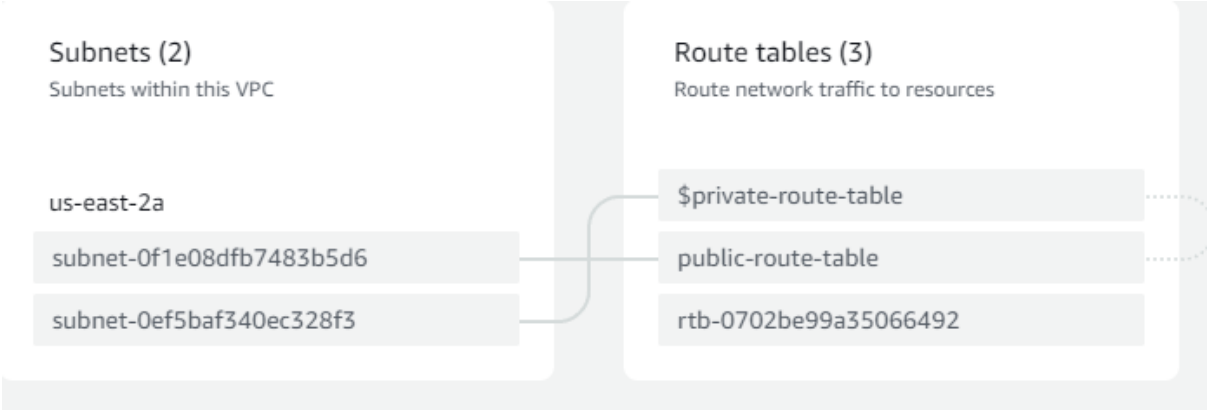
La ejecución del programa solo requiere llamar a la instrucción **terraform apply**
Esta va a crear la infraestructura en la cuenta de AWS enlazada, esta infraestructura cuenta con dos subredes, cada una con una VM que hospeda un Apache2.

El programa da como output la IP pública de la VM pública y la IP privada de la VM privada, ambas pueden usarse para acceder a ellas por medio del comando
ssh -i testKey ubuntu@{ip}

Para el módulo de VPN su configuración de cliente es explicado en la sección de pruebas.

Recomendaciones

1. Usar la consola de AWS para conectarse a las Virtual Machines y ejecutar los comandos ahí antes de escribir los scripts de user data
2. Buscar varias fuentes y ejemplos de las tecnologías usadas
3. Tener un dominio completo del tema antes de intentar solucionar el reto presentado por este proyecto
4. La herramienta de user_data es poderosa, es bueno tenerla en cuenta para automatizar procesos.
5. Tener las direcciones y puertos claros antes de crear las reglas para IPTABLES
6. Utilizar los diagramas generados por la consola de AWS para verificar el mapa de conexiones de la VPC



7. Hacer pruebas unitarias de cada una de las tecnologías antes de aplicarlas al proyecto.
8. Explorar las opciones del marketplace antes de implementar algo manualmente, o inspirarse en los recursos encontrados antes de iniciar desde cero.
9. Utilizar *terraform destroy* cuando se termina de trabajar para no desperdiciar los recursos disponibles.
10. Utilizar otros comandos como *terraform fmt* o -auto-approve para agilizar y simplificar el desarrollo y legibilidad.

Conclusiones

1. Las herramientas de infrastructure as code permiten la estandarización y automatización del proceso de creación de infraestructura en la nube.
2. La herramienta de security rules de terraform permite que por defecto la mayoría de los puertos no sean accesibles, esto da una seguridad en caso de que no se le preste atención a este aspecto.
3. El uso de AWS permite que las operaciones sean escalables, el costo de instancias muy pequeñas da un mínimo poder con un mínimo costo, excelente para hacer pruebas de concepto.
4. Las herramientas de IaC permiten el uso de comentarios dentro del código, esto puede facilitar el entendimiento de la infraestructura y del proceso de desarrollo.
5. La herramienta de VPN permite que la ip a la cual los servidores asignan al usuario sea el IP en donde se hospeda el servicio de VPN.

6. IaC permite que la gestión de recursos esté en un único proyecto/lugar, lo que agiliza el proceso de desarrollo.