Práctica 1. Administración básica de un servidor

Tú Nombre

April 1, 2024

1 Introducción

Los objetivos de esta práctica son:

- Poner en marcha un servidor Linux accesible.
- Securizarlo y programar un script de mantenimiento.
- Instalar y mantener levantada una API-Rest en Python.

2 Paso a paso

2.1 1. Administración básica de un servidor

- Crear una instancia de EC2 en AWS (Amazon Elastic Compute Cloud) utilizando la consola de AWS.
- Conectar a la instancia utilizando SSH desde tu terminal o cliente SSH favorito.

2.1.1 Conexión SSH:

1. Entrar en *Visual Studio Code* y conectarse a la máquina virtual.
2. Moverse hasta la carpeta de la clave de entrada al servidor.
3. Ejecutar el comando: `chmod 400 "Nombre_de_la_clave.pem"` en terminal.
4. Conectar mediante SSH utilizando:
   `ssh -i "Llave-Mi_servidor_web.pem" ubuntu@ec2-13-53-136-17.eu-north-1.compute.amazonaws`

2.2 2. Puesta en marcha de un servidor Linux accesible

- Elegir una imagen de sistema operativo Linux compatible al crear la instancia EC2.
- Configurar reglas de seguridad en el grupo de seguridad de la instancia EC2.

2.2.1 Cerrar puertos de entrada y salida en desuso:

Identificar el ID del Grupo de Seguridad:
`curl -s http://169.254.169.254/latest/meta-data/security-groups/`
Revocar acceso a todos los puertos excepto el 22:
`aws ec2 revoke-security-group-ingress --group-id tu-id-de-grupo --protocol all --cidr 0.0.0.0/0`

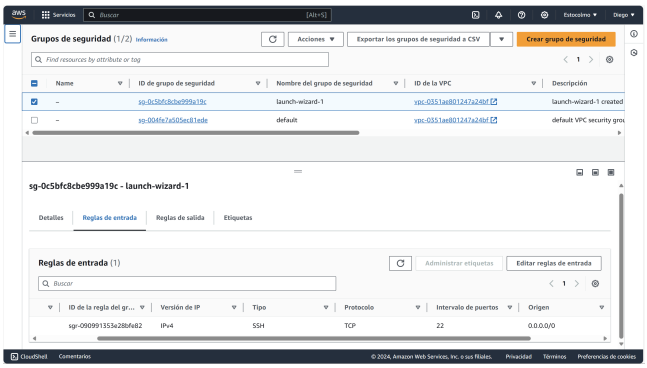

Figure 1: Puertos de entrada

2.3 3. Securización y programación de script de mantenimiento

2.3.1 Pasos para implementar Fail2ban:

1. Descargar Python (versión mayor o igual a 3.5)
2. Actualizar el sistema: `sudo apt update`
3. Instalar Python: `sudo apt install python3 python3-pip`
4. Verificar la instalación: `pip3 --version`
5. Ejecutar comando de instalación: `sudo apt install fail2ban`
6. Configurar *fail2ban* según los requisitos.

2.3.2 Script de mantenimiento:

Listing 1: Script de mantenimiento

```
#!/bin/bash

directorios_respaldar="/etc /var/lib /home/*"
directorio_backup="/home/directorio_backup"
```

```
fecha=$(date +"%Y-%m-%d")
tar_file="$directorio_backup/backup_$fecha.tar.gz"
tar -czvf "$tar_file" $directorios_respaldar

find /ruta/a/logs -type f -name "*.log" -mtime +7 -exec rm {} \;

echo "Copia de seguridad realizada en $tar_file"
echo "Archivos de log m s antiguos de 7 d as eliminados."
```

2.3.3 Automatización:

Para automatizar la ejecución del script, modificar el cron con el comando:
`crontab -e`
Agregar la siguiente línea al final del archivo:

```
0 3 * * * /home/scripts/script_backup.sh > /home/directorio_backup/log_backup.log 2>&1
```

2.4 4. Servicios

2.4.1 Crear la API mínima con Bottle

1. Instalar Bottle: `pip install bottle`
2. Crear un archivo python que implemente los recursos requeridos.

Listing 2: API básica con Bottle

```
from bottle import Bottle, run
import datetime

app = Bottle()

@app.route('/hi')
def hello():
    now = datetime.datetime.now()
    return f'Hola, hoy es {now.strftime("%d/%m/%Y")} y son las {now.strftime("%H

@app.route('/status')
def status():
    # Implementar l gica para listar servicios en ejecuci n
    return "Servicios en ejecuci n: Servicio1, Servicio2, ..."

if __name__ == '__main__':
    run(app, host='0.0.0.0', port=8080)
```