Práctica 1. Administración básica de un servidor

Diego Esclarín Fernández

April 1, 2024

Universidad
Rey Juan Carlos

Campus de Móstoles

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 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. Ir a "conectar" en AWS al seleccionar la instancia y copiar el ejemplo y cambiar el nombre de la llave.

2.2 Puesta en marcha de un servidor Linux accesible

- Elegir una imagen de sistema operativo Linux compatible al crear la instancia EC2.

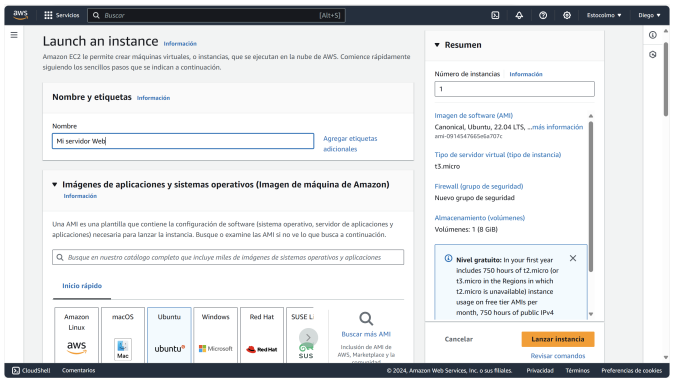

Figure 1: Creación del servidor web

2.3 Securización y programación de script de mantenimiento

2.3.1 Cerrar puertos de entrada y salida en desuso:

1. Identificar el ID del Grupo de Seguridad:
   `curl -s http://169.254.169.254/latest/meta-data/security-groups/`
2. 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`

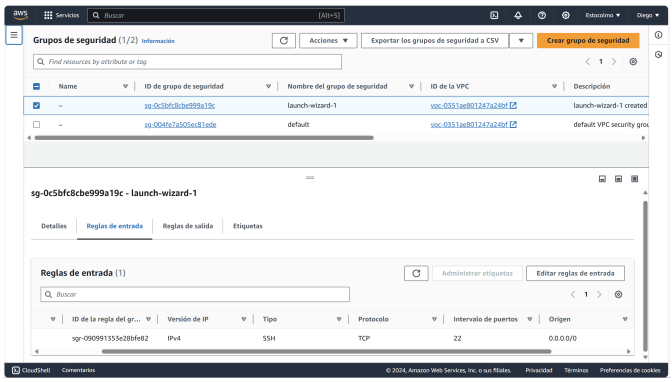

Figure 2: Configuración de Puertos de Entrada

2.3.2 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.

2.3.3 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."
```

Listing 1: Script de mantenimiento

2.3.4 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 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.

```python
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:%M:%S")}'

@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)
```

Listing 2: API básica con Bottle

3. Darle permisos de ejecución al archivo: `chmod +x /home/scripts/api.py`
4. Permitir el acceso de tráfico entrante por el puerto 80 ejecutando:
   `iptables -A INPUT -p tcp --dport 8080 -j ACCEPT`
5. Redirigir el tráfico del puerto 8080 al puerto 80 (donde está la API):
   `iptables -t nat -A PREROUTING -p tcp --dport 8080 -j REDIRECT --to-port 80`
6. Crear el directorio iptables en /etc/: `sudo mkdir -p /etc/iptables/`
7. Guardar las reglas de iptables: `sudo iptables-save > /etc/iptables/rules.v4`

2.5 Instalar y configurar Supervisor:

1. Ejecutar el siguiente comando: `sudo apt-get install supervisor`
2. Crea un archivo de configuración para tu aplicación en Supervisor.
   En este caso, api.conf en /etc/supervisor/conf.d/:

```
[program:my_api]
command=/usr/bin/python3 /home/scripts/api.py  # Ruta absoluta al script api.py
directory=/home/scripts/  # Directorio donde se encuentra api.py
autostart=true
autorestart=true
stderr_logfile=/var/log/api.err.log
stdout_logfile=/var/log/api.out.log

```

Listing 3: API básica con Bottle

3. Actualiza la configuración de Supervisor y comienza a supervisar tu aplicación:
   `sudo supervisorctl reread`
   `sudo supervisorctl update`
   `sudo supervisorctl start my_api`

2.6 Instalar Nginx y configurar un proxy inverso:

1. Instalar Nginx: `sudo apt-get install nginx`
2. Creamos un archivo de configuración para la API en Nginx. En nuestro caso, api.conf en /etc/nginx/sites-available/:

```
server {
    listen 80;
    server_name tu_dominio_o_ip;

    location / {
        proxy_pass http://127.0.0.1:8080;  # Puerto donde se ejecuta la API con Bottle
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

```

Listing 4: API básica con Bottle

3. Habilitamos el sitio y reiniciamos Nginx:
   `sudo ln -s /etc/nginx/sites-available/api.conf /etc/nginx/sites-enabled/`
   `sudo systemctl restart nginx`

3 Conclusión

En esta práctica, se han abordado de manera integral la configuración y administración básica de un servidor Linux, desde su creación en AWS hasta la implementación de medidas de seguridad como el cierre de puertos no utilizados y la configuración de Fail2ban. Además, se han automatizado tareas de mantenimiento mediante scripts y hemos desplegado una API-Rest utilizando Bottle, asegurándonos de su disponibilidad y seguridad con Supervisor y Nginx como proxy inverso. Estos pasos nos permiten tener un servidor funcional, seguro y con capacidades para gestionar servicios web de manera eficiente.