Informe de Instalación y Configuración de dos Servidores SSH en Docker

Objetivo

Implementar dos servidores SSH en contenedores Docker, uno con autenticación por clave pública (sin contraseña) y otro con autenticación en dos factores (2FA) mediante Google Authenticator.

Arquitectura General

El sistema se compone de dos contenedores Docker:

1. ssh-server: servidor SSH clásico con acceso por clave pública y sin contraseña.
2. ssh2fa: servidor SSH con autenticación de doble factor (contraseña + Google Authenticator).

Ambos están desplegados con Docker Compose en redes separadas y direcciones IP fijas.

Configuración del Contenedor ssh-server

Docker Compose

Se usó la siguiente configuración para `ssh-server`:

```
services:
 ssh-server:
  image: linuxserver/openssh-server
  container_name: ssh_server
  environment:
   - PUID=1000
   - PGID=1000
   - TZ=Etc/UTC
   - PUBLIC_KEY=ssh-rsa AAAA[...] daniel@gmail.com
   - SUDO_ACCESS=false
   - PASSWORD_ACCESS=false
   - USER_NAME=usuario
  ports:
   - "2222:2222"
  restart: unless-stopped
```

```
    networks:
      services_net:
        ipv4_address: 172.20.0.2
```

Decisiones tomadas:

- Imagen base: linuxserver/openssh-server ya viene preconfigurada.
- PUBLIC_KEY: se configura para autenticación sin contraseña.
- PASSWORD_ACCESS=false: fuerza uso de clave pública.
- SUDO_ACCESS=false: mayor seguridad.
- Red separada con IP fija.

Configuración del Contenedor ssh2fa (2FA)

Docker Compose

Configuración del contenedor con autenticación en dos factores:

```
ssh2fa:
  build:
    context: ./production
    args:
      SSH_USER: usuario
      SSH_PASSWORD: clave123
  container_name: ssh2fa
  ports:
    - "2223:22"
  networks:
    production_net:
      ipv4_address: 172.30.0.2
  restart: unless-stopped
```

Dockerfile personalizado

```
FROM debian:bullseye

RUN apt-get update && \
    apt-get install -y openssh-server libpam-google-authenticator qrencode && \
    mkdir /var/run/sshd

ARG SSH_USER
ARG SSH_PASSWORD
```

```
RUN useradd -m -s /bin/bash $SSH_USER && \
   echo "$SSH_USER:$SSH_PASSWORD" | chpasswd && \
   mkdir -p /home/$SSH_USER/.ssh && \
   chown -R $SSH_USER:$SSH_USER /home/$SSH_USER/.ssh

COPY sshd_config /etc/ssh/sshd_config

RUN echo "auth required pam_google_authenticator.so" >> /etc/pam.d/sshd

EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]
```

sshd_config personalizado

```
Port 22
PermitRootLogin no
ChallengeResponseAuthentication yes
UsePAM yes
PasswordAuthentication yes
AuthenticationMethods password,keyboard-interactive
```

Decisiones tomadas:

- Imagen base estable: debian:bullseye.
- Autenticación combinada: contraseña + código OTP.
- Configuración de PAM para Google Authenticator.
- Uso de argumentos de construcción para flexibilidad.

Configuración dinámica del OTP

Una vez desplegado el contenedor, se configura el 2FA ejecutando google-authenticator como el usuario creado. Se escanea el código QR, se configuran parámetros de seguridad como protección contra reuse de tokens y se guarda el archivo .google_authenticator.

Pruebas de Conexión

Conexión al servidor SSH sin 2FA:

```
ssh -p 2222 usuario@localhost
```

Conexión al servidor con 2FA:

ssh -p 2223 usuario@localhost

Salida esperada:
usuario@localhost's password: ←Aquí ingresamos clave123
Verification code: ← Aquí ingresamos el código de Google authenticator generado

Redes Docker utilizadas

```
networks:
 services_net:
   driver: bridge
   ipam:
     config:
       - subnet: 172.20.0.0/24

 production_net:
   driver: bridge
   ipam:
     config:
       - subnet: 172.30.0.0/24
```

Conclusión

Se implementaron dos servidores SSH en contenedores Docker con configuraciones distintas:

- Uno seguro sin contraseña, basado en clave pública.
- Otro con autenticación en dos pasos, utilizando Google Authenticator.

Ambos entornos están aislados, con redes separadas y direcciones IP fijas, han sido probados y funcionan

Comprobación del resultado del servidor con Google authenticator tras introducir la contraseña (clave123) y el código de Google authenticator:

```
exit
daniel@daniel-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~/Escritorio/AS/PRACTICASSH$ service ssh restart
daniel@daniel-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~/Escritorio/AS/PRACTICASSH$ ssh -p 2223 usuario@localhost
(usuario@localhost) Password:
(usuario@localhost) Verification code:
Linux c50fcbcce72b 6.8.0-57-generic #59-Ubuntu SMP PREEMPT_DYNAMIC Sat Mar 15 17:40:59 UTC 2025 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 17 18:15:32 2025 from 172.30.0.1

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
usuario@c50fcbcce72b:~$
```

Comprobación del resultado de conexión con el servidor que no posee Google authenticator:

```
daniel@daniel-ASUS-TUF-Dash-F15-FX517ZM-FX517ZM:~/Escritorio/AS/PRACTICASSH$ ssh -p 2222 usuario@localhost
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:IKFsDQUhzf46ugKg4U9kKs2mjm6EsZ1ttHPX2Etwa/o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of known hosts.
Welcome to OpenSSH Server
eec5af11c361:~$
```