

Google Authenticator & Keys

Noé Abel Vargas López

Juan Alfredo Gómez González

Jonathan Iram Talavera Pardo

Jesus Alberto Ramirez González

¿Qué es Google Authenticator?

Google Authenticator es una aplicación móvil que genera códigos temporales de verificación, usados en la autenticación de dos factores.



Ventajas de usar Google Authenticator

- Autenticación en dos pasos (2FA)
 - Sin dependencia de red
 - Compatibilidad amplia
- Mayor protección ante ataques de fuerza bruta o robo de credenciales

¿Para qué te sirve tener Google Authenticator en tu conexión SSH?

1. Seguridad adicional contra intrusiones.
2. Protección frente a ataques automatizados
3. Control de acceso más fuerte para administradores
4. Cumplimiento de buenas prácticas y políticas de seguridad
5. Códigos locales, sin dependencia de red

Google Authenticator Configuración (Paso a Paso)

Verificar si google-authenticator se encuentra.

```
sudo yum search google-authenticator
```

No se encontraron coincidencias.

El paquete de no está en los repositorios base del sistema operativo. Habilitamos EPEL

```
sudo yum install epel-release -y
```

```
sudo yum repolist
```

Para verificar la instalación.

Ahora si instalamos google-authenticator con qrencode para escanear el código qr en la aplicación.

```
sudo yum install -y google-authenticator qrencode qrencode-libs
```

Iniciamos la configuración desde el usuario con quien vamos a autenticarnos con Google Authenticator.

```
google-authenticator
```

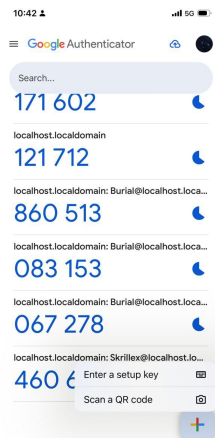
En este paso vienen una serie de preguntas a responder para realizar la configuración.

Preguntas de configuración (Google Authenticator)

¿Quieres que los tokens de autenticación estén basados en tiempo.?

```
Do you want authentication tokens to be time-based (y/n) y
```

Escanear código QR desde la aplicación de Google Authenticator e ingresar un código generado desde la app.



Requisitos previos:

- Instalar la aplicación de Google Authenticator en tu dispositivo móvil.
- Iniciar sesión en la aplicación con una cuenta (De preferencia una de google).

Preguntas de configuración (Google Authenticator)

Guardar códigos de emergencia.

```
Enter code from app (-1 to skip): 402373
Code confirmed
Your emergency scratch codes are:
57066258
78538000
46534530
16842753
86521608
```

Escenarios comunes:

- Pérdida o daño del dispositivo.
- Imposibilidad de generar el código.

Archivo de configuración que contiene el texto oculto de nuestra llave secreta asociada a la cuenta para generar los códigos de un solo uso. ¿Deseas guardar estos cambios en el archivo?

```
Do you want me to update your "/home/nlopez/.google_authenticator" file? (y/n) y
```

Preguntas de configuración (Google Authenticator)

¿Quieres deshabilitar el uso múltiple del mismo token de autenticación? Esto te restringe a un solo inicio de sesión aproximadamente cada 30 segundos, pero incrementa tus posibilidades de notar o incluso prevenir ataques man-in-the-middle?

```
Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y
```

¿Quieres aumentar la ventana de su tamaño predeterminado de 3 códigos permitidos... a 17 códigos permitidos (los 8 códigos anteriores, el código actual y los 8 códigos siguientes)? Esto permitirá una desincronización de hasta 4 minutos entre el cliente y el servidor.

```
By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
Do you want to do so? (y/n) y
```


Preguntas de configuración (Google Authenticator)

¿Quieres deshabilitar el uso múltiple del mismo token de autenticación? Esto te restringe a un solo inicio de sesión aproximadamente cada 30 segundos, pero incrementa tus posibilidades de notar o incluso prevenir ataques man-in-the-middle?

```
Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y
```

¿Quieres aumentar la ventana de su tamaño predeterminado de 3 códigos permitidos... a 17 códigos permitidos (los 8 códigos anteriores, el código actual y los 8 códigos siguientes)? Esto permitirá una desincronización de hasta 4 minutos entre el cliente y el servidor.

```
By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
Do you want to do so? (y/n) y
```

Preguntas de configuración (Google Authenticator)

Si la computadora a la que estás iniciando sesión no está protegida (*hardened*) contra intentos de inicio de sesión por fuerza bruta, puedes habilitar la limitación de tasa para el módulo de autenticación. Por defecto, esto limita a los atacantes a no más de 3 intentos de inicio de sesión cada 30 segundos. ¿Quieres habilitar la limitación de tasa?

```
If the computer that you are logging into isn't hardened against brute-force  
login attempts, you can enable rate-limiting for the authentication module.  
By default, this limits attackers to no more than 3 login attempts every 30s.  
Do you want to enable rate-limiting? (y/n) y
```

Configuración del SSH/PAM

Editar los archivos de configuración de PAM (Pluggable Authentication Modules) y SSH (Secure Shell) para que el sistema le pida el código 2FA al usuario al iniciar sesión. Los archivos típicos son:

`/etc/pam.d/sshd`

`/etc/ssh/sshd_config`

Abrir el archivo de configuración de PAM para SSH:

`sudo nano /etc/pam.d/sshd`

`+auth required pam_google_authenticator.so`

`Ctrl + X`

`y`

`Enter`

```
GNU nano 8.1 /etc/pam.d/sshd
#%PAM-1.0
auth        substack    password-auth
auth        include     postlogin
account     required    pam_sepermit.so
account     required    pam_nologin.so
account     include     password-auth
password    include     password-auth
# pam_selinux.so close should be the first session rule
session     required    pam_selinux.so close
session     required    pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session     required    pam_selinux.so open env_params
session     required    pam_namespace.so
session     optional    pam_keyinit.so force revoke
session     optional    pam_motd.so
session     include     password-auth
session     include     postlogin
```

before

```
GNU nano 5.6.1 /etc/pam.d/sshd
#%PAM-1.0
auth        substack    password-auth
auth        include     postlogin
auth        required    pam_google_authenticator.so nullok
account     required    pam_sepermit.so
account     required    pam_nologin.so
account     include     password-auth
password    include     password-auth
# pam_selinux.so close should be the first session rule
session     required    pam_selinux.so close
session     required    pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session     required    pam_selinux.so open env_params
session     required    pam_namespace.so
session     optional    pam_keyinit.so force revoke
session     optional    pam_motd.so
session     include     password-auth
session     include     postlogin
```

after

Configuración del SSH/PAM

El orden de ejecución es de arriba hacia abajo.

```
GNU nano 5.6.1 /etc/pam.d/sshd
#%PAM-1.0
auth        substack      password-auth
auth        include       postlogin
auth        required pam_google_authenticator.so nullok
account     required      pam_sepermit.so
account     required      pam_nologin.so
account     include       password-auth
password    include       password-auth
# pam_selinux.so close should be the first session rule
session     required      pam_selinux.so close
session     required      pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session     required      pam_selinux.so open env_params
session     required      pam_namespace.so
session     optional      pam_keyinit.so force revoke
session     optional      pam_motd.so
session     include       password-auth
session     include       postlogin
```

***auth required:** Debe pasar esta autenticación obligatoriamente.*

***pam_google_authenticator.so:** Verifica el código de Google Authenticator*

***nullok:** Permite que el acceso por Google Authenticator sea opcional para aquellos usuarios que no lo tengan configurado.*

Configuración del SSHD Rocky Linux 9.6

En Rocky Linux 9.6 en lugar de modificar directamente el archivo se debe de usar archivos drop-in en:

`/etc/ssh/sshd_config.d/`

Crear archivo de configuración personalizado:

`sudo nano /etc/ssh/sshd_config.d/50-2fa.conf`

Agregar las siguientes líneas de configuración:

`KbdInteractiveAuthentication yes`

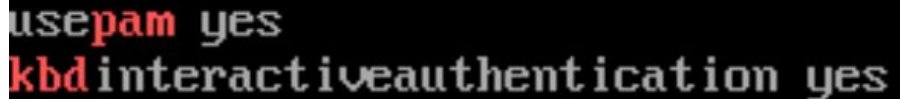
`UsePAM yes`

Ctrl + x

Enter

Verificar la configuración:

`sudo sshd -T | grep -E "kbd|pam|challenge"`



```
usepam yes
kbdinteractiveauthentication yes
```

Verificar el contexto de SELinux (Puedes pasarte al paso de Configuración de SELinux) 9.6

Verificar el contexto de SELinux que determina los accesos es decir que puede acceder a que.

```
ls -laZ /home/<usuario>/google_authenticator
```

Debería de mostrar algo como esto: *unconfined_u:object_r:user_home_t:s0*

Si muestra algo como **auth_home** esto puede ocasionar problemas de permisos:

```
unconfined_u:object_r:auth_home_t:s0
```

Restaurar el contexto correcto de SELinux:

```
sudo semanage fcontext -a -t user_home_t "/home/<usuario>/google_authenticator"
```

```
sudo restorecon -v /home/<usuario>/google_authenticator
```

Reiniciar *sudo systemctl restart sshd*

Configuración de Política de SELinux para Google Authenticator 9.6

Asegurarse de que SELinux esté en enforcing:

```
sudo setenforce 1
```

```
sudo getenforce
```

Debería de mostrar **Enforcing**

Conectarse para generar logs de auditoría desde tu cliente:

```
ssh <user>@<ip> -p <port>
```

Fallara pero generará los logs de auditoría

Generar política desde los logs de auditoría

```
sudo ausearch -m avc -ts recent | audit2allow -M google_auth_policy
```

Nos mostrará un mensaje con el comando para activar la política.

```
***** IMPORTANTE *****  
Para activar este paquete de políticas, ejecute:  
  
semodule -i google_auth_policy.pp
```

Configuración de Política de SELinux para Google Authenticator 9.6

Instalar política:

```
sudo semodule -i google_auth_policy.pp
```

Verificar instalación:

```
sudo semodule -l | grep google_auth
```

Debería de mostrar la política instalada.

google_auth_policy

Reiniciar *sudo systemctl restart sshd*

Configuración del SSHD Rocky Linux 8.1

En Rocky Linux 8.1 se requiere configurar el sshd_config

/etc/ssh/sshd_config

Configurar el sshd:

sudo nano /etc/ssh/sshd_config

Agregar las siguientes líneas de configuración:

ChallengeResponseAuthentication yes

Ctrl + x

Enter

UsePAM yes

Mover archivo a .ssh:

mv ~/.google_authenticator ~/.ssh/.google_authenticator

chmod 600 ~/.ssh/.google_authenticator

Configuración del SSH/PAM 8.1

Editar los archivos de configuración de PAM (Pluggable Authentication Modules) y SSH (Secure Shell) para que el sistema le pida el código 2FA al usuario al iniciar sesión. Los archivos típicos son:

`/etc/pam.d/sshd`

`/etc/ssh/sshd_config`

Abrir el archivo de configuración de PAM para SSH:

`sudo nano /etc/pam.d/sshd`

`+auth required pam_google_authenticator.so secret=${HOME}/.ssh/.google_authenticator` **Ctrl + X** **y** **Enter**

`secret=${HOME}/.ssh/.google_authenticator`: Requiere especificar la ruta.

SELinux en Rocky Linux 8.1

Restaurar contexto:

```
sudo restorecon -Rv ~/.ssh
```

Verificar contexto:

```
ls -laZ ~/.ssh/google_athenticator
```

Debe mostrar: ssh_home_t

Reiniciar SSH

Conexión a través de un cliente SSH

Conexión a través de SSH:

```
ssh <usuario>@<ip> -p <port>
```

```
ssh Skrillex@192.168.1.78 -p 1911
```

Ingresar la contraseña:

```
(Skrillex@192.168.1.78) Password:
```

Ingresar código generado a través de la app de Google Authenticator:

```
(Skrillex@192.168.1.78) Verification code:
```

¡Conexión exitosa!

```
ssh Skrillex@192.168.1.78 -p 1911
(Skrillex@192.168.1.78) Password:
(Skrillex@192.168.1.78) Verification code:

channel 11: open failed: connect failed: open failed
channel 13: open failed: connect failed: open failed
channel 11: open failed: connect failed: open failed
channel 3: open failed: connect failed: open failed
channel 12: open failed: connect failed: open failed
channel 13: open failed: connect failed: open failed
channel 3: open failed: connect failed: open failed
channel 3: open failed: connect failed: open failed

Skrillex@localhost.localdomain ~ (0.022s)
pwd
/home/Skrillex
```

¿Qué son las Google Keys (o llaves de seguridad)?

Las Google Keys son un método de autenticación seguro que sustituye o complementa el uso de contraseñas.

Pueden ser físicas (como una memoria USB o una llave NFC/Bluetooth) o digitales (almacenadas en tu teléfono o en tu cuenta de Google).

¿Para qué sirven?

1. Proteger cuentas contra accesos no autorizados
2. Reemplazar contraseñas tradicionales
3. Evitar el phishing
4. Autenticar usuarios en sistemas, servidores o plataformas web de manera rápida y segura.

Ventajas de usar Google Keys

1. Evitan el robo de contraseñas o ataques de phishing
2. Permiten iniciar sesión solo con una llave o huella digital.
3. Iniciar sesión toma solo unos segundos.
4. Puedes tener una llave USB, NFC o guardarla digitalmente en tu dispositivo.
5. Funciona con Google, GitHub, Microsoft, AWS, y más servicios modernos.

1) En el equipo cliente (Nuestra Laptop)

Generamos una llave ed25519

```
ssh-keygen -t ed25519 -C "Noe@$(hostname)" -f ~/.ssh/id_ed25519
```

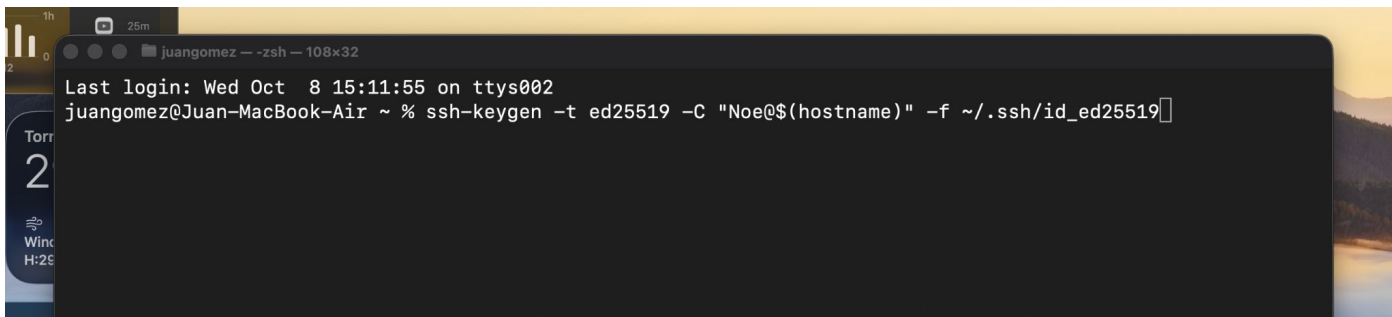
ssh-keygen: Es el comando para generar la llave.

-t ed25519: El tipo de algoritmo con el cuál se genera la llave.

-C "Noe@\$(hostname)": Comentario para identificar la llave.

-f ~/.ssh/id_ed25519: Indica donde se guarda la llave.

si nos pide una frase de paso omitimos dando enter



2) Copiar la llave pública al servidor

`ssh -p 9997 Usuario@ip_servidor 'mkdir -p ~/.ssh && chmod 700 ~/.ssh'`

ssh: Es el comando que nos permite conectarnos a el servidor.

-p 9997: Indica el puerto por el cual nos vamos a conectar.

usuario@ip_servidor: Indicamos el usuario y la ip del servidor.

'mkdir -p ~/.ssh && chmod 700 ~/.ssh': Este comando se va a ejecutar una vez que entremos al servidor.

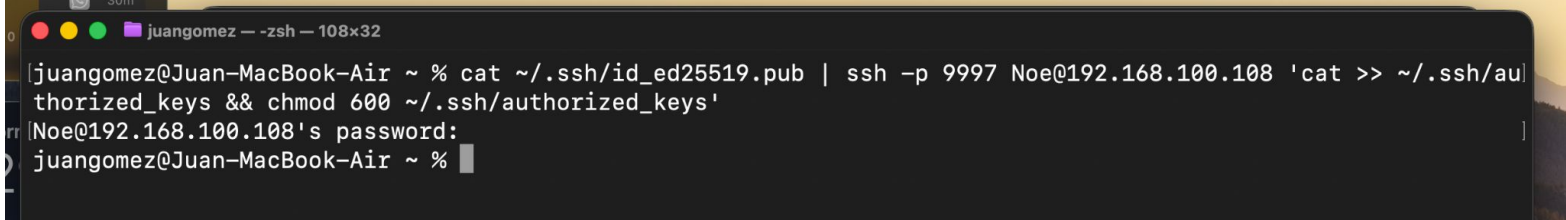
```
[SHA256]
juangomez@Juan-MacBook-Air ~ % ssh -p 9997 Noe@192.168.100.108 'mkdir -p ~/.ssh && chmod 700 ~/.ssh'
The authenticity of host '[192.168.100.108]:9997 ([192.168.100.108]:9997)' can't be established.
ED25519 key fingerprint is SHA256:d4IrwYjk9oMmFuauZE1kfNbK068D1Y/z4QkGNC7ssEU.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:7: 192.168.100.106
  ~/.ssh/known_hosts:10: [192.168.0.19]:9997
  ~/.ssh/known_hosts:11: [192.168.0.66]:9997
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.100.108]:9997' (ED25519) to the list of known hosts.
Noe@192.168.100.108's password:
juangomez@Juan-MacBook-Air ~ %
```

3) Copiar la clave desde el cliente

```
cat ~/.ssh/id_ed25519.pub | ssh -p 9997 Usuario@ip_servidor 'cat >> ~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys'
```

cat ~/.ssh/id_ed25519.pub | ssh -p 9997 Usuario@ip_servidor : Lo que hace este comando es tomar el contenido de la llave de nuestra laptop y la envía al servidor.

cat >> ~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys': Este comando se ejecuta en el servidor y guarda la clave y protege el archivo con permisos seguros.

A screenshot of a macOS terminal window. The title bar shows 'juangomez' and '-zsh - 108x32'. The terminal text shows a user at 'juangomez@Juan-MacBook-Air' running a command to copy a public key to a remote server 'Noe@192.168.100.108' via SSH on port 9997. The command is: `cat ~/.ssh/id_ed25519.pub | ssh -p 9997 Noe@192.168.100.108 'cat >> ~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys'`. The prompt for the password is shown as '[Noe@192.168.100.108's password:]' and the user has entered the password, returning to the prompt 'juangomez@Juan-MacBook-Air ~ %'.

4) Verificar permisos en el servidor (muy importante)

En el servidor (ejecuta como Superusuario o como root):

`sudo chown -R usuario:usuario /home/usuario/.ssh` : Cambiar de manera recursiva el propietario y grupo del archivo de llaves

`sudo chmod 700 /home/usuario/.ssh`: Protege la carpeta donde están las claves SSH.

`sudo chmod 600 /home/usuario/.ssh/authorized_keys`: Asegura que solo el usuario pueda leer o modificar la lista de claves autorizadas.

```
[Noe@localhost root]$ ls -la ~/.ssh
total 4
drwx-----. 2 Noe Noe  29 oct  9 14:56 .
drwx-----. 3 Noe Noe 109 oct  9 14:50 ..
-rw-----. 1 Noe Noe 108 oct  9 14:56 authorized_keys
[Noe@localhost root]$
```

4) Activar PubkeyAuthentication

Vamos al archivo de configuración de ssh

Buscamos la opción:

```
# PubkeyAuthentication yes
```

y la descomentamos.

Reiniciamos el SSH

```
sudo nano /etc/ssh/sshd_config
```

```
#PubkeyAuthentication yes
```

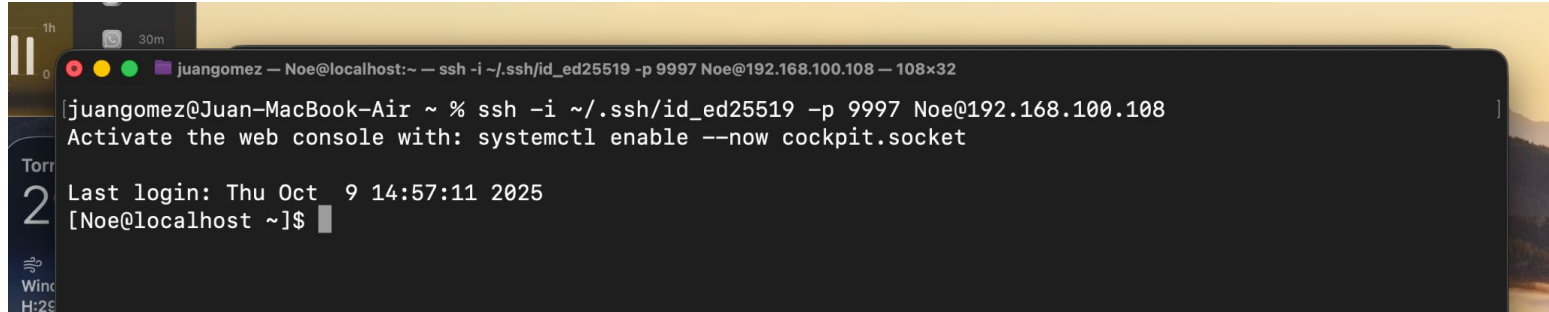
```
PubkeyAuthentication yes
```

5) Accedemos a el servidor desde nuestra computadora local

Para acceder a el servidor

```
ssh -i ~/.ssh/id_ed25519 -p 9997 Usuario@ip_servidor
```

-i ~/.ssh/id_ed25519: Le indica a SSH qué clave privada usar para autenticarse.



```
juangomez — Noe@localhost:~ — ssh -i ~/.ssh/id_ed25519 -p 9997 Noe@192.168.100.108 — 108x32
[juangomez@Juan-MacBook-Air ~ % ssh -i ~/.ssh/id_ed25519 -p 9997 Noe@192.168.100.108
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Oct  9 14:57:11 2025
[Noe@localhost ~]$
```