

Sistema de Control de Versiones Git Hooks en Ubuntu

Gabriel Carbajal Carrasco

2024-09-03

1. Introducción a los Git Hooks

Los Git Hooks son scripts personalizados que se ejecutan automáticamente en respuesta a eventos en un repositorio de Git. Puedes usarlos para validar código antes de hacer commits, ejecutar pruebas automáticas antes de realizar un push, o enviar notificaciones después de ciertos eventos.

2. Ubicación de los Hooks en un Repositorio

Cada repositorio de Git tiene una carpeta llamada hooks dentro del directorio .git que contiene ejemplos de hooks predefinidos. Estos ejemplos están desactivados por defecto (los archivos terminan en .sample).

Para activar un hook, simplemente elimina la extensión .sample y edita el archivo según tus necesidades.

```
gabriel@gabriel-VirtualBox:~$ cd New_Repositorio/.git/hooks
gabriel@gabriel-VirtualBox:~/New_Repositorio/.git/hooks$ ls
applypatch-msg.sample      pre-commit.sample          pre-receive.sample
commit-msg.sample          pre-merge-commit.sample    push-to-checkout.sample
fsmonitor-watchman.sample  prepare-commit-msg.sample  sendemail-validate.sample
post-update.sample         pre-push.sample            update.sample
pre-applypatch.sample      pre-rebase.sample
```

3. Creación de un Hook Pre-Commit

Un hook pre-commit se ejecuta antes de que se complete un commit. Puede usarse para revisar el código o prevenir commits que no cumplen con ciertas reglas.

3.1 Crear el Hook

Crea un archivo llamado pre-commit en la carpeta .git/hooks/ y hazlo ejecutable.

```
gabriel@gabriel-VirtualBox:~/New_Repositorio/.git/hooks$ touch pre-commit
gabriel@gabriel-VirtualBox:~/New_Repositorio/.git/hooks$ chmod +x pre-commit
```

3.2 Escribir el Script del Hook

Por ejemplo, este script impide que se realicen commits si hay archivos con código Python que no están formateados según black.

```
pre-commit
/home/gabriel/New_Repositorio/.git/hooks/pre-commit

#!/bin/bash
files=$(git diff --cached --name-only --diff-filter=ACM | grep '\.py$')
if [ "$files" != "" ]; then
    unformatted=$(black --check $files)
    if [ $? -ne 0 ]; then
        echo "Hay archivos sin formatear. Formatea el código antes de hacer commit."
        exit 1
    fi
fi
```

3.3 Probar el Hook

Intenta hacer un commit con un archivo Python que no esté formateado correctamente. El hook evitará el commit hasta que el archivo esté formateado. Instalando Python, pip y black previamente.

```
gabriel@gabriel-VirtualBox: ~/New_Repositorio$ nano test_script.py
gabriel@gabriel-VirtualBox: ~/New_Repositorio$ git add test_script.py
gabriel@gabriel-VirtualBox: ~/New_Repositorio$ git commit -m "prueba de hook pre-commit"
.git/hooks/pre-commit: línea 4: black: orden no encontrada
Hay archivos sin formatear. Formatea el código antes de hacer commit.
gabriel@gabriel-VirtualBox: ~/New_Repositorio$

GNU nano 7.2 test_script.py *
def example_function(x,y): return x+y
```

4. Creación de un Hook Pre-Push

Un hook pre-push se ejecuta antes de enviar los cambios a un repositorio remoto. Puedes usarlo para ejecutar pruebas automáticas o asegurarte de que el código pase ciertas validaciones.

4.1 Crear el Hook

Crea un archivo llamado pre-push en la carpeta .git/hooks/ y hazlo ejecutable.

```
gabriel@gabriel-VirtualBox: ~/New_Repositorio$ touch .git/hooks/pre-push
gabriel@gabriel-VirtualBox: ~/New_Repositorio$ chmod +x .git/hooks/pre-push
```

4.2 Escribir el Script del Hook

Este ejemplo de script ejecuta pruebas unitarias antes de hacer un push:

```
pre-push
~/New_Repositorio/.git/hooks

echo "Ejecutando pruebas antes del push..."
if ! make test; then
    echo "Las pruebas fallaron. Push abortado."
    exit 1
fi
```

4.3 Probar el Hook

Intenta hacer un push. Si las pruebas fallan, el hook evitará que los cambios se envíen al servidor remoto.

```
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git remote add origin https://github.com/gabriel-carbajal/Repositorio.git
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git remote -v
origin https://github.com/gabriel-carbajal/Repositorio.git (fetch)
origin https://github.com/gabriel-carbajal/Repositorio.git (push)
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git add Archivo1.txt
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git add origin master
fatal: ruta especificada 'origin' no concordó con ningún archivo
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git push origin master
Username for 'https://github.com': gabriel-carbajal
Password for 'https://gabriel-carbajal@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Autenticación falló para 'https://github.com/gabriel-carbajal/Repositorio.git/'
```

5. Creación de un Hook Post-Merge

Un hook post-merge se ejecuta después de fusionar una rama. Puedes usarlo para ejecutar scripts que configuran el entorno de desarrollo o realicen acciones posteriores a la fusión.

5.1 Crear el Hook

Crea un archivo llamado post-merge en la carpeta .git/hooks/ y hazlo ejecutable.

```
gabriel@gabriel-VirtualBox:~/New_Repositorio$ touch .git/hooks/post-merge
gabriel@gabriel-VirtualBox:~/New_Repositorio$ chmod +x .git/hooks/post-merge
```

5.2 Escribir el Script del Hook

Por ejemplo, este script reinstala las dependencias del proyecto si el archivo package.json cambio durante la fusión:

```
post-merge
~/New_Repositorio/.git/hooks

#!/bin/bash
if git diff-tree -r --name-only --no-commit-id ORIG_HEAD HEAD | grep -q 'package.json'; then
    echo "package.json ha cambiado. Instalando dependencias..."
    npm install
fi
```

5.3 Probar el Hook

Fusiona una rama que haya cambiado el archivo package.json. El hook ejecutará automáticamente npm install.

```
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git checkout rama3
M       Archivo1.txt
A       test_script.py
Cambiado a rama 'rama3'
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git add package.json
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git commit -m "Modificar package.json"
.git/hooks/pre-commit: línea 4: black: orden no encontrada
Hay archivos sin formatear. Formatea el código antes de hacer commit.
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git commit -m "Modificar package.json"
[rama3 7aa8335] Modificar package.json
 3 files changed, 3 insertions(+)
 create mode 100644 package.json
 create mode 100644 test_script.py
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git checkout master
Cambiado a rama 'master'
gabriel@gabriel-VirtualBox:~/New_Repositorio$ git merge rama3
Actualizando 9ddcb60..7aa8335
Fast-forward
 Archivo1.txt   | 1 +
 package.json   | 1 +
 test_script.py | 1 +
 3 files changed, 3 insertions(+)
 create mode 100644 package.json
 create mode 100644 test_script.py
package.json ha cambiado. Instalando dependencias...
.git/hooks/post-merge: línea 4: npm: orden no encontrada
gabriel@gabriel-VirtualBox:~/New_Repositorio$
```