

# Tarea 5: Validación automática con GitHub Actions

## PASO 1: Crear el repositorio y el script .sh

1. En GitHub, creamos un nuevo repositorio (por ejemplo, **validacion-shell**).
2. Lo clonamos en nuestra máquina
3. Creamos el script **mi\_script.sh** con contenido básico:

```
root@daniel-MS-7E28:/home/validacion-shell# cat mi_script.sh
#!/bin/bash
echo "Hola Mundo"
root@daniel-MS-7E28:/home/validacion-shell# git add mi_script.sh
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Añadir script mi_script.sh"
[master (commit-raíz) af28953] Añadir script mi_script.sh
 1 file changed, 2 insertions(+)
 create mode 100644 mi_script.sh
root@daniel-MS-7E28:/home/validacion-shell# git branch -M main
root@daniel-MS-7E28:/home/validacion-shell# git push -u origin main
Enter passphrase for key '/root/.ssh/id_ed25519':
Enter passphrase for key '/root/.ssh/id_ed25519':
Enumerando objetos: 3, listo.
Contando objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (3/3), 270 bytes | 270.00 KiB/s, listo.
Total 3 (delta 0), reusados 0 (delta 0), pack-reusados 0
To github.com:Danimartinez-1997/validacion-shell.git
 * [new branch]      main -> main
rama 'main' configurada para rastrear 'origin/main'.
root@daniel-MS-7E28:/home/validacion-shell#
```

## PASO 2: Crear el flujo de trabajo en **.github/workflows/validate.yml**

Creamos la carpeta y el archivo:

```
root@daniel-MS-7E28:/home/validacion-shell# mkdir -p .github/workflows
root@daniel-MS-7E28:/home/validacion-shell# touch .github/workflows/validate.yml
root@daniel-MS-7E28:/home/validacion-shell#
```

```
! validate.yml x
validacion-shell > .github > workflows > ! validate.yml > name
GitHub Workflow - YAML GitHub Workflow (github-workflow.json)
1 name: Validar Shell Script
2
3 on:
4   push:
5     branches: [ "main" ]
6
7 jobs:
8   shellcheck:
9     runs-on: ubuntu-latest
10
11   steps:
12     - name: Checkout del repositorio
13       uses: actions/checkout@v3
14
15     - name: Instalar ShellCheck
16       run: sudo apt-get install -y shellcheck
17
18     - name: Ejecutar ShellCheck
19       run: shellcheck mi_script.sh
```

Y commiteamos y pusheamos los cambios en nuestro repositorio.

```
root@daniel-MS-7E28:/home/validacion-shell# git add .github/workflows/validate.yml
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Configurar GitHub Actions para validar el script con ShellCheck"
[main 6a7dc0b] Configurar GitHub Actions para validar el script con ShellCheck
1 file changed, 19 insertions(+)
create mode 100644 .github/workflows/validate.yml
root@daniel-MS-7E28:/home/validacion-shell# git push
Enter passphrase for key '/root/.ssh/id_ed25519':
Enumerando objetos: 6, listo.
Contando objetos: 100% (6/6), listo.
Compresión delta usando hasta 12 hilos
Comprimiendo objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (5/5), 624 bytes | 624.00 KiB/s, listo.
Total 5 (delta 0), reusados 0 (delta 0), pack-reusados 0
To github.com:Danimartinez-1997/validacion-shell.git
   af28953..6a7dc0b  main -> main
root@daniel-MS-7E28:/home/validacion-shell#
```

**PASO 3: Añadir el badge de estado en [README.md](#)**

1. Abrimos el archivo **README.md**.
2. Añadimos la siguiente línea:

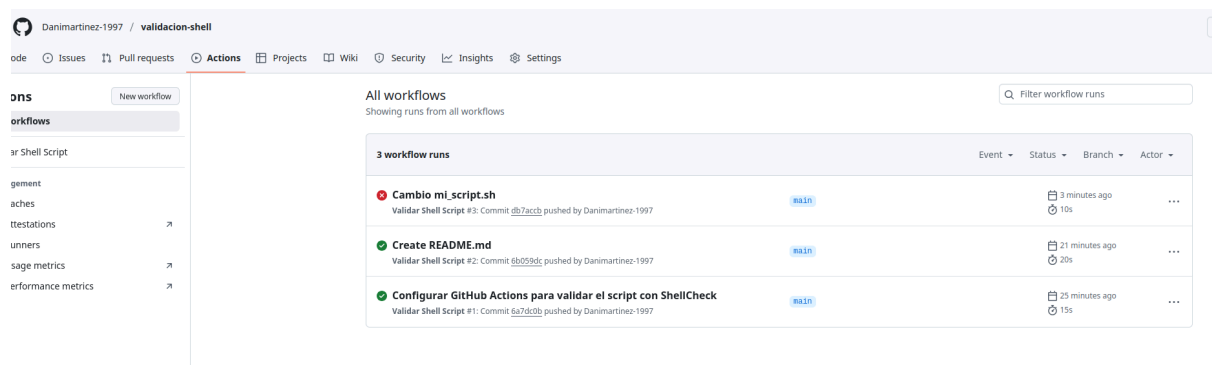
![ShellCheck](<https://github.com/Danimartinez-1997/validacion-shell/actions/workflows/s/validate.yml/badge.svg>)

## COMPROBACIÓN

Hacemos un pequeño cambio en el archivo **mi\_script.sh** (quitarle una comilla al echo en este caso) y lo subimos al repositorio.

```
$ mi_script.sh x
validacion-shell > $ mi_script.sh
1  #!/bin/bash
2  echo Hola Mundo"
3
```

Comprobamos en la pestaña “Actions” dentro de GitHub si detecta el error en el archivo



The screenshot shows the GitHub Actions interface for the repository 'Danimartinez-1997 / validacion-shell'. The 'Actions' tab is active, displaying a list of workflow runs. The first run, titled 'Cambio mi\_script.sh', is marked with a red circle and a failed status, indicating that the ShellCheck action detected an error in the script. The second run, 'Create README.md', and the third run, 'Configurar GitHub Actions para validar el script con ShellCheck', are marked with green circles and a successful status.

Workflow	Status	Event	Branch	Actor	Time
Cambio mi_script.sh	Failed	Push	main	Danimartinez-1997	3 minutes ago
Create README.md	Success	Push	main	Danimartinez-1997	21 minutes ago
Configurar GitHub Actions para validar el script con ShellCheck	Success	Push	main	Danimartinez-1997	25 minutes ago

# Tarea 7: Limpieza de commits

## PASO 1: Crear commits "sucios"

Vamos a crear varios commits en el repositorio cada uno insertando una línea en un archivo.

```
root@daniel-MS-7E28:/home/validacion-shell# echo "Primera línea" > limpieza.txt
root@daniel-MS-7E28:/home/validacion-shell# git add limpieza.txt
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Cambios"
[main 9741ba8] Cambios
 1 file changed, 1 insertion(+)
 create mode 100644 limpieza.txt
root@daniel-MS-7E28:/home/validacion-shell# echo "Segunda línea" >> limpieza.txt
root@daniel-MS-7E28:/home/validacion-shell# git add limpieza.txt
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Cambios2"
[main 3b0b66c] Cambios2
 1 file changed, 1 insertion(+)
root@daniel-MS-7E28:/home/validacion-shell# echo "Tercera línea" >> limpieza.txt
root@daniel-MS-7E28:/home/validacion-shell# git add limpieza.txt
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Cambios3"
[main aac071d] Cambios3
 1 file changed, 1 insertion(+)
root@daniel-MS-7E28:/home/validacion-shell# █
```

## PASO 2: Reescribir el historial con git rebase -i

Con ese comando se nos abrirá un editor con la siguiente pinta.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

GNU nano 7.2
pick 9741ba8 Cambios
pick 3b0b66c Cambios2
pick aac071d Cambios3

# Rebase db7accb..aac071d en db7accb (3 comandos)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                               commit's log message, unless -C is used, in which case
#                               keep only this commit's message; -c is same as -C but
#                               opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#       create a merge commit using the original merge commit's
#       message (or the oneline, if no original merge commit was
#       specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                       to this position in the new commits. The <ref> is
#                       updated at the end of the rebase
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# Si eliminas una línea aquí EL COMMIT SE PERDERÁ.
#
# Como sea, si quieres borrar todo, el rebase será abortado.
#
```

Debemos hacer los siguientes cambios, para convertir el primero en el único commit y los otros dos en dos líneas a mayores.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

GNU nano 7.2
reword 9741ba8 NuevoCommit
squash 3b0b66c Añadir segunda línea
squash aac071d Añadir tercera línea

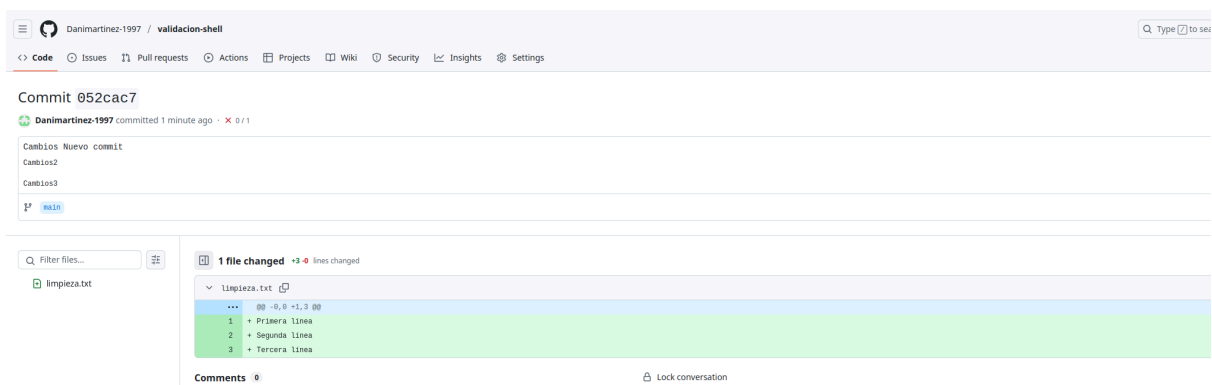
# Rebase db7accb..aac071d en db7accb (3 comandos)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                               commit's log message, unless -C is used, in which case
#                               keep only this commit's message; -c is same as -C but
#                               opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#       create a merge commit using the original merge commit's
#       message (or the oneline, if no original merge commit was
#       specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                       to this position in the new commits. The <ref> is
#                       updated at the end of the rebase
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# Si eliminas una línea aquí EL COMMIT SE PERDERÁ.
#
# Como sea, si quieres borrar todo, el rebase será abortado.
#
```

Finalmente forzamos un push con los cambios realizados.


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

root@daniel-MS-7E28:/home/validacion-shell# git rebase -i HEAD~3
Rebase aplicado satisfactoriamente y actualizado refs/heads/main.
root@daniel-MS-7E28:/home/validacion-shell# git rebase -i HEAD~3
[HEAD desacoplado 4a7e18c] Cambios Nuevo commit
Date: Thu Jun 26 08:59:00 2025 +0200
1 file changed, 1 insertion(+)
create mode 100644 limpieza.txt
[HEAD desacoplado 052cac7] Cambios Nuevo commit
Date: Thu Jun 26 08:59:00 2025 +0200
1 file changed, 3 insertions(+)
create mode 100644 limpieza.txt
Rebase aplicado satisfactoriamente y actualizado refs/heads/main.
root@daniel-MS-7E28:/home/validacion-shell# git push --force
Enter passphrase for key '/root/.ssh/id_ed25519':
Enumerando objetos: 4, listo.
Contando objetos: 100% (4/4), listo.
Compresión delta usando hasta 12 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 403 bytes | 403.00 KiB/s, listo.
Total 3 (delta 0), reusados 0 (delta 0), pack-reusados 0
To github.com:Danimartinez-1997/validacion-shell.git
+ aac071d...052cac7 main -> main (forced update)
root@daniel-MS-7E28:/home/validacion-shell#
```

Y comprobamos en Github que los cambios se han realizado con un solo commit.



## PASO 3: Crear [REBASE.md](#)



The screenshot shows a GitHub repository page for 'validation-shell' by user 'Danimartinez-1997'. The file 'REBASE.md' is selected, showing its content. The file has 22 lines, 16 loc, and 602 bytes. The content of the file is as follows:

### Reescritura del historial con git rebase -i

#### Objetivo

Limpiar el historial de commits y dejar mensajes claros y significativos.

#### Commits originales

- "Cambios"
- "Arreglos"
- "Actualización de cosas"

#### Pasos realizados

1. Ejecuté `git rebase -i HEAD~3` para reescribir los últimos 3 commits.
2. Cambié el primer commit a `reword` y los otros dos a `squash`.
3. Unifiqué los tres commits en uno solo con el mensaje:  
| "Mejorar limpieza.txt con varias lineas nuevas"
4. Guardé y cerré el editor.
5. Finalmente, actualicé el repositorio remoto con:  
`git push --force`

## Tarea 8: Versionado con tags y releases

### PASO 1: Simular un proyecto con cambios importantes

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

root@daniel-MS-7E28:/home/validacion-shell# echo "echo 'Versión 1.0'" > version.sh
root@daniel-MS-7E28:/home/validacion-shell# git add version.sh
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Versión inicial del script"
[main 424c0ce] Versión inicial del script
1 file changed, 1 insertion(+)
create mode 100644 version.sh
root@daniel-MS-7E28:/home/validacion-shell#
```

### PASO 2: Crea un tag para marcar esta versión

Creamos un tag para los cambios realizados anteriormente. También vamos a seguir simulando cambios en el proyecto y creando un tag por cada uno de ellos.



```

root@daniel-MS-7E28:/home/validacion-shell# git tag 1.0
root@daniel-MS-7E28:/home/validacion-shell# echo "echo 'Versión 1.1 - Añadido saludo'" >> version.sh
root@daniel-MS-7E28:/home/validacion-shell# git add version.sh
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Añadido saludo al script"
[main cfec7bb] Añadido saludo al script
1 file changed, 1 insertion(+)
root@daniel-MS-7E28:/home/validacion-shell# git tag v1.1
root@daniel-MS-7E28:/home/validacion-shell# echo "# Comentario de mantenimiento" >> version.sh
root@daniel-MS-7E28:/home/validacion-shell# git add version.sh
root@daniel-MS-7E28:/home/validacion-shell# git commit -m "Limpieza de código y comentarios"
[main b828eaf] Limpieza de código y comentarios
1 file changed, 1 insertion(+)
root@daniel-MS-7E28:/home/validacion-shell# git tag v1.2
root@daniel-MS-7E28:/home/validacion-shell# █

```

### PASO 3: Sube los cambios y los tags a GitHub

Primero subimos todos los commits realizados. Luego, con el segundo comando subimos todos los tags que hemos creado.

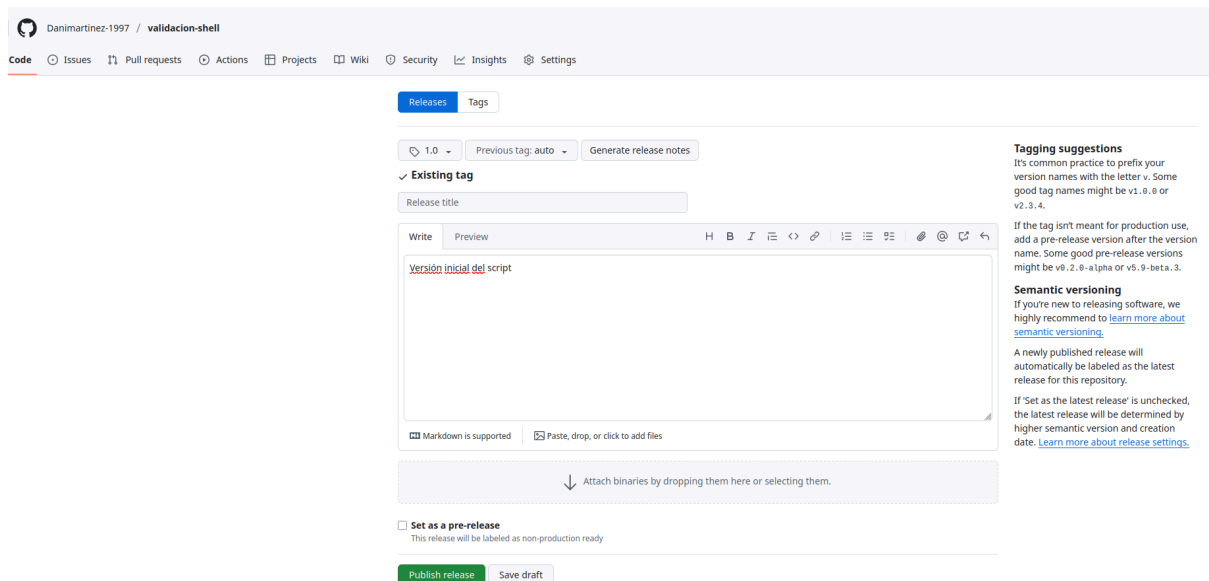
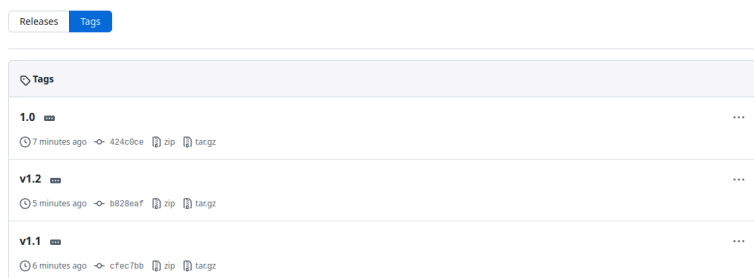
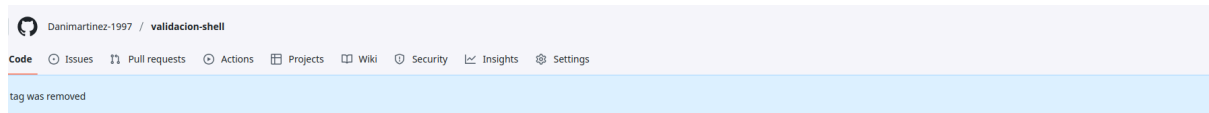
```

root@daniel-MS-7E28:/home/validacion-shell# git push
Enter passphrase for key '/root/.ssh/id_ed25519':
Enumerando objetos: 10, listo.
Contando objetos: 100% (10/10), listo.
Compresión delta usando hasta 12 hilos
Comprimiendo objetos: 100% (8/8), listo.
Escribiendo objetos: 100% (9/9), 856 bytes | 856.00 KiB/s, listo.
Total 9 (delta 4), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:Danimartinez-1997/validacion-shell.git
   052cac7..b828eaf  main -> main
root@daniel-MS-7E28:/home/validacion-shell# git push --tags
Enter passphrase for key '/root/.ssh/id_ed25519':
Enter passphrase for key '/root/.ssh/id_ed25519':
Total 0 (delta 0), reusados 0 (delta 0), pack-reusados 0
To github.com:Danimartinez-1997/validacion-shell.git
 * [new tag]          1.0 -> 1.0
 * [new tag]          1.0+ -> 1.0+
 * [new tag]          v1.1 -> v1.1
 * [new tag]          v1.2 -> v1.2
root@daniel-MS-7E28:/home/validacion-shell# █

```

# PASO 4: Crea Releases en GitHub

1. Ve a tu repositorio en GitHub.
2. Haz clic en la pestaña "Releases".
3. Crea una nueva release para cada tag (v1 .0, v1 .1, v1 .2).
4. Escribe una descripción para cada una explicando los cambios.



Danimartinez-1997 / validacion-shell

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Releases / 1.0

1.0

Latest

Compare

Danimartinez-1997

 released this now · 2 commits to main since this release · 1.0 · 424c8ce

Versión inicial del script

Assets

2

Source code (zip)

9 minutes ago

Source code (tar.gz)

9 minutes ago

© 2025 GitHub, Inc. · [Terms](#) · [Privacy](#) · [Security](#) · [Status](#) · [Docs](#) · [Contact](#) · [Manage cookies](#) · [Do not share my personal information](#)

Danimartinez-1997 / validacion-shell

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Releases

Tags

Draft a new release

Find a release

now

Danimartinez-1997

V1.2

b828eaf

Compare

v1.2

Latest

Tercera versión del script

Assets

2

Source code (zip)

7 minutes ago

Source code (tar.gz)

7 minutes ago

now

Danimartinez-1997

V1.1

cfec7bb

Compare

v1.1

Segunda versión del script

Assets

2

1 minute ago

Danimartinez-1997

1.0

424c8ce

Compare


1.0

Versión inicial del script

Assets

2

# PASO 5: Crea el archivo [CHANGELOG.md](#)



Danimartinez-1997 / validacion-shell

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

main

validacion-shell / CHANGELOG.md

Danimartinez-1997

Create CHANGELOG.md

review

Code

Blame

22 lines (15 loc) · 487 Bytes

## CHANGELOG

Todas las versiones importantes de este proyecto serán listadas aquí.

---

### [v1.2] - 2025-06-25

#### Changed

- Añadido comentario de mantenimiento al script.
- Limpeza menor de código.

---

### [v1.1] - 2025-06-25

#### Added

- Nueva línea de saludo añadida al script `version.sh`.

---

### [v1.0] - 2025-06-25

#### Added

- Versión inicial del script con mensaje básico "Versión 1.0".