Para revertir una funcionalidad pusheada primero debemos hacer git log, para poder identificar el commit que queremos revertir.

```
Rayxr@DESKTOP-ASG1QLG MINGW64 /d/Giovanni/Proyectos/Trabajo GCBA/3 - GIT (dev)
$ git log
commit ec0f761a8b682e1203b9a2ad4399d762a1945f25 (HEAD -> dev)
Author: Giovanni-Martini <giovanni.martini@estudiantes.unahur.edu.ar>
Date:   Thu Nov 6 14:25:01 2025 -0300

    Agregada funcionalidad

commit 99029f63dfc734a4ae3c9b63e71651e326ca9849 (master)
Author: Giovanni-Martini <giovanni.martini@estudiantes.unahur.edu.ar>
Date:   Thu Nov 6 14:24:16 2025 -0300

    Primer commit
```

Luego, hacemos git revert junto con el hash del commit para revertir los cambios.

```
Rayxr@DESKTOP-ASG1QLG MINGW64 /d/Giovanni/Proyectos/Trabajo GCBA/3 - GIT (dev)
$ git revert ec0f761a8b682e1203b9a2ad4399d762a1945f25
```

Ahora nos muestra esta pestaña para ver que se hizo, para salir, hay que apretar la tecla ESC y escribir :wq

```
Revert "Agregada funcionalidad"

This reverts commit ec0f761a8b682e1203b9a2ad4399d762a1945f25.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch dev
# Changes to be committed:
#       deleted:    Ejemplo 2.txt
#
~
~
~
~
~
~
~
~
~
~
~
~
.git/COMMIT_EDITMSG [unix] (14:26 06/11/2025)                 1,1 All
<royectos/Trabajo GCBA/3 - GIT/.git/COMMIT_EDITMSG" [unix] 11L, 306B
```

Luego volvemos a escribir git log. Acá podemos ver que los cambios fueron revertidos.

```
Rayxr@DESKTOP-ASG1QLG MINGW64 /d/Giovanni/Proyectos/Trabajo GCBA/3 - GIT (dev)
$ git log
commit c290ca016061b8405cfaa9937a8f2af3faeea9fe (HEAD -> dev)
Author: Giovanni-Martini <giovanni.martini@estudiantes.unahur.edu.ar>
Date:   Thu Nov 6 14:26:20 2025 -0300

    Revert "Agregada funcionalidad"

    This reverts commit ec0f761a8b682e1203b9a2ad4399d762a1945f25.

commit ec0f761a8b682e1203b9a2ad4399d762a1945f25
Author: Giovanni-Martini <giovanni.martini@estudiantes.unahur.edu.ar>
Date:   Thu Nov 6 14:25:01 2025 -0300

    Agregada funcionalidad

commit 99029f63dfc734a4ae3c9b63e71651e326ca9849 (master)
Author: Giovanni-Martini <giovanni.martini@estudiantes.unahur.edu.ar>
Date:   Thu Nov 6 14:24:16 2025 -0300

    Primer commit
```

Para hacer deploy, de una versión llamada v1.2.0 lo primero que tenemos que hacer es estar en la rama en la que queramos hacerlo usando git checkout <Nombre de la branch>.

Luego, es conveniente usar el comando git pull origin <Nombre de la branch> para obtener los cambios que se hayan realizado en esa Branch.

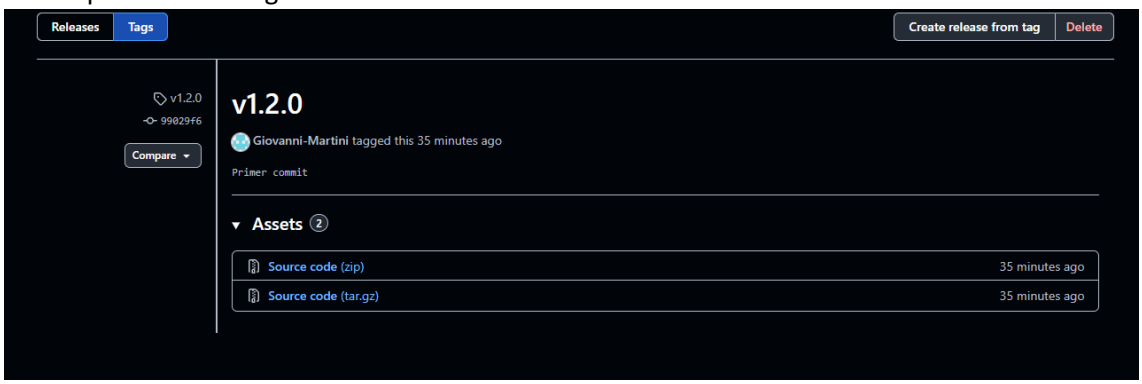
Para renombrar la versión a v1.2.0 usamos este comando:

```
Rayxr@DESKTOP-ASG1QLG MINGW64 /d/Giovanni/Proyectos/Trabajo GCBA/3 - GIT (master
)
$ git tag v1.2.0
```

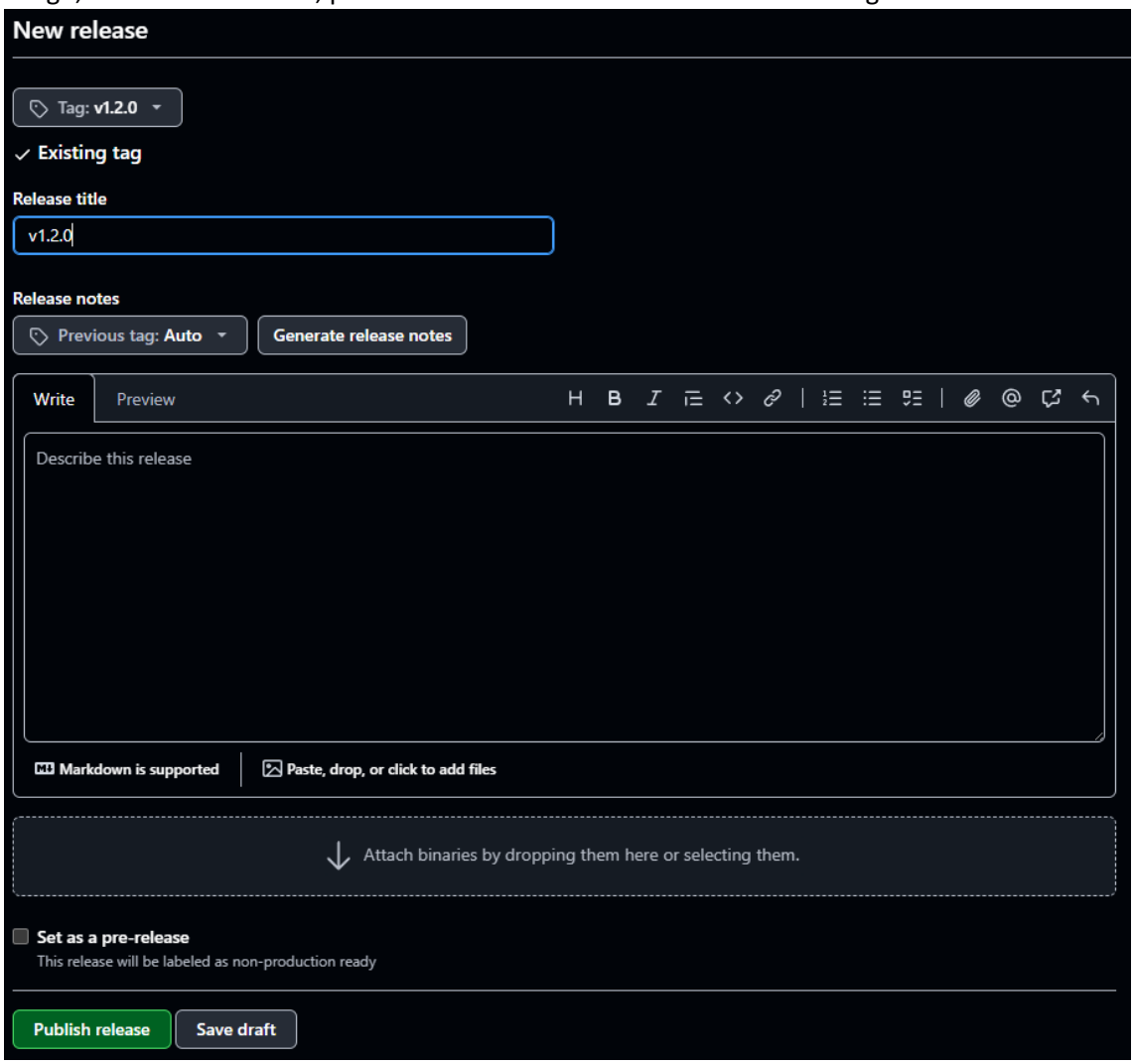
Después, subimos la versión a un repositorio remoto. En este caso GitHub.

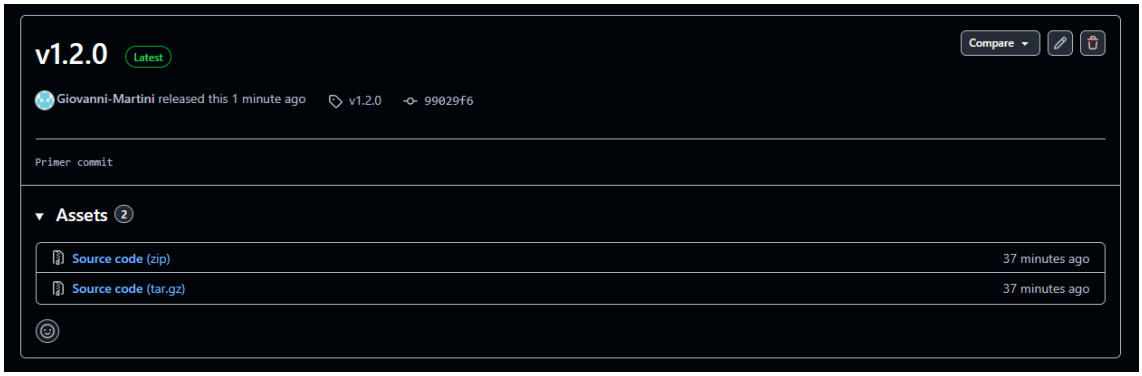
```
Rayxr@DESKTOP-ASG1QLG MINGW64 /d/Giovanni/Proyectos/Trabajo GCBA/3 - GIT (master
)
$ git push origin v1.2.0
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 377 bytes | 188.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Giovanni-Martini/ejemplo.git
 * [new tag]         v1.2.0 -> v1.2.0
```

Acá se puede ver la tag subida a GitHub



Luego, si vamos a releases, podemos hacer una release utilizando esta tag.





Y finalmente, podremos hacer el deploy, el como lo hacermos dependerá del entorno que estemos utilizando, este puede ser tanto manual como automático, lo que se conoce como CI/CD.

Por ejemplo, para hacer un deploy automático tenemos que subir la versión al repositorio, luego, el sistema CI/CD que estemos utilizando detecta los cambios realizados y procede a descargar el código, instalar las dependencias que sean necesarias, realizar tests y compilar la aplicación si hace falta. En el caso de estar todo correcto, se suben los archivos al entorno, donde se actualiza la aplicación para que los usuarios tengan la nueva versión.