

COMANDOS DE GIT

git --version: Muestra la versión utilizada de git.

```
~ > git --version  
git version 2.25.1
```

git init: Inicia un repositorio en la carpeta actual.

```
~ > md git_adsi  
~ > cd git_adsi/  
~/git_adsi > git init  
Inicializado repositorio Git vacío en /home/alejo/git_adsi/.git/  
~/git_adsi master > |
```

Necesitaremos configurar un nombre y un correo para acciones futuras. Esto lo hacemos con **git config --global user.name** y **git config --global user.email**:

```
~/git_adsi master > git config --global user.name "Alejo"  
~/git_adsi master > git config --global user.email "alejito23001@gmail.com"  
~/git_adsi master > |
```

Listando los elementos ocultos, vemos el directorio `.git` donde se almacenan los identificadores de las versiones, cambios, remotos y ramas.

```
~/git_adsi master > ls -a  
.  ..  .git
```

Se crea un archivo de texto llamado `adsi.txt` con el siguiente contenido:

```
1 Código original  
2
```

En este punto el directorio “git adsi” contiene lo siguiente:

```
~/git_adsi master ?1 > la
total 8,0K
-rw-r--r-- 1 alejo alejo 17 feb 23 07:10 adsi.txt
drwxrwxr-x 7 alejo alejo 4,0K feb 23 07:03 .git
~/git_adsi master ?1 > |
```

git add: Añade los archivos modificados o nuevos al área de ‘staged’ o en espera. Para añadir los archivos se usa **git add <archivo>** o **git add .**
Ejemplo: **git add adsi.txt** o **git add .**

```
~/git_adsi master ?1 > git add .
~/git_adsi master +1 > |
```

git status: Lista el estado de los cambios.

```
~/git_adsi master +1 > git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
  (usa "git rm --cached <archivo>..." para sacar del área de stage)
nuevo archivo: adsi.txt
~/git_adsi master +1 > |
```

El comando “git commit” acepta comentarios usando la opción “-m” así: “git commit -m “. Para aplicar los cambios al ejercicio en curso se procede así:

```
~/git_adsi master +1 > git commit -m "Versión Original"
[master (commit-raíz) 3c438fa] Versión Original
1 file changed, 1 insertion(+)
create mode 100644 adsi.txt
~/git_adsi master > |
```

Se puede en este momento volver a revisar el estado o “status” así:

```
~/git_adsi master > git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
~/git_adsi master > |
```

En este punto se puede revisar el historial de cambios o log, con **git log**:

```
~/git_adsi master > git log|
```

```
commit 5de88b82bc59a6296c3ed42d488ca27b746e0204 (HEAD -> master)
Author: Alejo <alejito23001@gmail.com>
Date:   Wed Mar 9 21:41:07 2022 -0500
```

```
    Versión Original
(END)
```

Aquí veremos todos los commits, con su autor, su fecha, su hash y su mensaje.

Editamos el archivo ads.txt con el siguiente contenido:

```
1 Código Original.
2 Primer cambio|
3
```

```
ads.txt (2,14) |
Saved ads.txt
```

Comprobemos el estado del repositorio:

```
~/git_adsi master !1 > git status 21:47:27
En la rama master
Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directo
rio de trabajo)
      modificado:      ads_i.txt

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
~/git_adsi master !1 > 21:47:28
```

Podemos observar que git detectó un nuevo cambio en el archivo y lo listó como modificado. Añadámoslo stagin area:

```
~/git_adsi master !1 > git add . 21:48:35
~/git_adsi master +1 > git status 21:48:36
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
      modificado:      ads_i.txt

~/git_adsi master +1 > | 21:48:48
```

Ahora agreguemos los cambios a un commit:

```
~/git_adsi master +1 > git commit -m "Primer cambio"
[master 39607fe] Primer cambio
 1 file changed, 1 insertion(+)
~/git_adsi master > |
```

Por último revisemos el historial de cambios, allí podremos observar los dos commits con sus respectivos datos.

```
~/git_adsi master > git log
```

```
commit 39607fe4f16e0f8e9b6593010381be3b5ea47fd5 (HEAD -> master)
Author: Alejo <alejito23001@gmail.com>
Date: Wed Mar 9 21:50:02 2022 -0500
```

Primer cambio

```
commit 5de88b82bc59a6296c3ed42d488ca27b746e0204
Author: Alejo <alejito23001@gmail.com>
Date: Wed Mar 9 21:41:07 2022 -0500
```

Versión Original

Una de las características más importantes de Git es que permite regresar a un estado previo del proyecto, algo similar a un “viaje al pasado” que permite analizar los cambios o definitivamente regresar el proyecto a ese punto de restauración.

Como observamos en la imagen anterior el último commit (con mensaje de “Primer cambio”) tiene un ID de 39607fe4f16e0f8e9b6593010381be3b5ea47fd5 y el primer commit (con mensaje “Versión Original”) tiene un ID

5de88b82bc59a6296c3ed42d488ca27b746e0204 podemos volver a este primer commit con el comando **git checkout <al menos 7 caracteres del ID del commit>**:

```
~/git_adsi master > git checkout 5de88b8
Nota: cambiando a '5de88b8'.

Te encuentras en estado 'detached HEAD'. Puedes revisar por aquí, hacer
cambios experimentales y hacer commits, y puedes descartar cualquier
commit que hayas hecho en este estado sin impactar a tu rama realizando
otro checkout.

Si quieres crear una nueva rama para mantener los commits que has creado,
puedes hacerlo (ahora o después) usando -c con el comando checkout. Ejemplo:

    git switch -c <nombre-de-nueva-rama>

O deshacer la operación con:

    git switch -

Desactiva este aviso poniendo la variable de config advice.detachedHead en false

HEAD está ahora en 5de88b8 Versión Original
~/git_adsi @5de88b82 > |
```

Ahora nos encontramos en el primer commit, lo podemos comprobar viendo el contenido del archivo adsi.txt:

```
1 Código Original
2
adsi.txt (1,1) |
```

Para volver a la rama principal podemos usar:

- **git checkout <rama>**
- **git checkout -**
- **git switch -**
- **git switch <rama>**

```
~/git_adsi @5de88b82 > git checkout master
La posición previa de HEAD era 5de88b8 Versión Original
Cambiado a rama 'master'
~/git_adsi master > |
```

Si revisamos de nuevo adsi.txt veremos que el archivo sigue igual a como lo dejamos en el último commit:

```
1 Código Original
2 Primer cambio
3
adsi.txt (1,1) |
```

Aparte de trabajar con repositorios locales, podemos trabajar con repositorios remotos hospedados en lugares como www.github.com, www.gitlab.com, etc. Para almacenar repositorios en estos lugares, debemos de tener una cuenta, la cual es gratuita de registrar y muy sencilla. Para esta guía se tomará un repositorio ya creado con un contenido básico.

El repositorio se encuentra en <https://github.com/xAGH/HolaMundo>. Podemos traerlo a nuestra máquina local con el comando **git clone <url>**:

```
~ > git clone https://github.com/xAGH/HolaMundo.git
Clonando en 'HolaMundo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Desempaquetando objetos: 100% (3/3), 267 bytes | 267.00 KiB/s, listo.
~ > cd HolaMundo
~/HolaMundo main > |
```

Ya tenemos el repositorio remoto en nuestra máquina local.

Este repositorio sigue enlazado al repositorio remoto, esto lo podemos ver con el comando **git remote -v**:

```
~/HolaMundo main > git remote -v
origin https://github.com/xAGH/HolaMundo.git (fetch)
origin https://github.com/xAGH/HolaMundo.git (push)
~/HolaMundo main > |
```

Listamos el contenido de este repositorio:

```
~/HolaMundo main > ls
archivo.txt
~/HolaMundo main > |
```

Vamos a ver el contenido del archivo archivo.txt:

```
~/HolaMundo main > cat archivo.txt
¡Este archivo ha
sido traído desde
un repositorio
en GitHub!
~/HolaMundo main > |
```

Si tenemos los permisos necesarios, podremos hacer cambios en el repositorio local y subirlos estos cambios al repositorio remoto.

En este caso, como el repositorio remoto es mío, no tendré problemas. Haré un cambio y le haré un commit.

```
1 ¡Este archivo ha
2 sido traído desde
3 un repositorio
4 en GitHub y ha
5 sido modificado
6 desde el repositorio
7 local!
8 |
archivo.txt (8,1) |
```



```
~/HolaMundo main !1 > git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
      modificado:      archivo.txt

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
~/HolaMundo main !1 > git add archivo.txt
~/HolaMundo main +1 > git status
En la rama main
Tu rama está actualizada con 'origin/main'.


Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
      modificado:      archivo.txt

~/HolaMundo main +1 > git commit -m "Cambio al repo remoto"
[main 2b98906] Cambio al repo remoto
 1 file changed, 5 insertions(+), 2 deletions(-)
~/HolaMundo main +1 > |
```

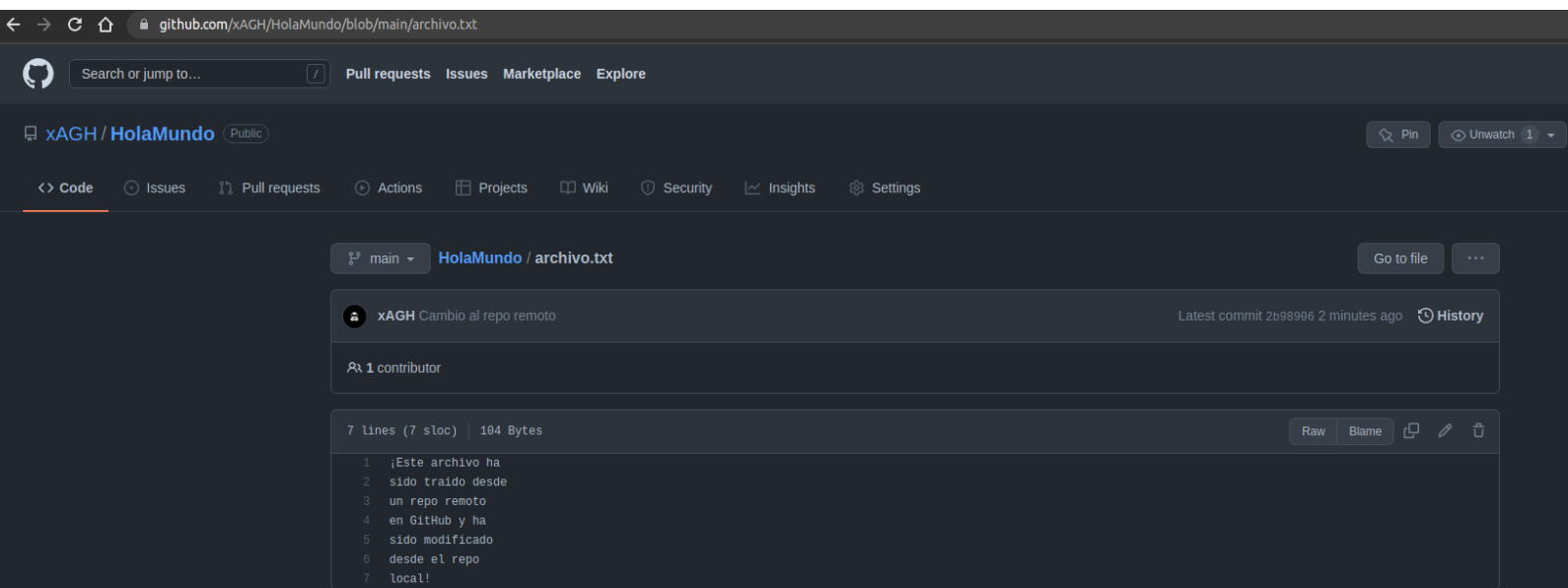


Ya con el commit, subiré los cambios al repositorio remoto con el comando **git push** **<alias del repo remoto>** **<rama>**:

```
~/HolaMundo main +1 > git push origin main
Enumerando objetos: 5, listo.
Contando objetos: 100% (5/5), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 332 bytes | 332.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
To https://github.com/xAGH/HolaMundo.git
   f90b936..2b98906  main -> main
~/HolaMundo main > |
```



Y en Github vemos el contenido modificado:



Con esto terminamos la guía básica sobre comandos de git.