

Comandos Git

Ingeniería de sistemas y computación

Integrantes

Manuel Eduardo Alarcon Aza

Profesor

William Javier Matallana Porras

Universidad de Cundinamarca – UDEC

Chía

2025

## Tabla de contenido

<b>Introducción .....</b>	<b>3</b>
<b>Objetivo.....</b>	<b>3</b>
<b>Configuración nombre de usuario y correo electrónico .....</b>	<b>3</b>
<b>Creación repositorio .....</b>	<b>4</b>
<b>Git Init.....</b>	<b>4</b>
<b>Git Clone.....</b>	<b>5</b>
<b>Git Status .....</b>	<b>5</b>
<b>Git add.....</b>	<b>5</b>
<b>Git Commit.....</b>	<b>6</b>
<b>Git Push.....</b>	<b>7</b>
<b>Git Switch .....</b>	<b>7</b>
<b>Git Branch .....</b>	<b>8</b>
<b>Git Pull .....</b>	<b>9</b>
<b>Como deshacer o reversar un git commit .....</b>	<b>9</b>
<b>Conclusión .....</b>	<b>9</b>
<b>Referencias.....</b>	<b>9</b>

## Introducción

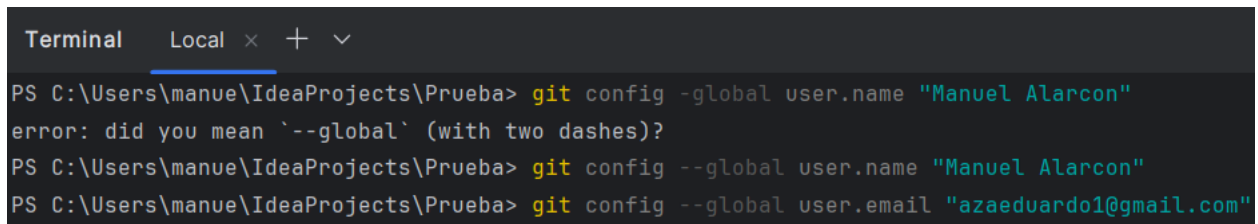
Este manual pretende que identifiquemos y visualizaremos algunos de los comandos más utilizados de Git, facilitando como podemos llegar a utilizarlos en nuestros entornos de trabajo y como estos aportan a la hora de trabajar con repositorios locales y remotos.

## Objetivo

El objetivo de este manual es identificar los principales comandos de git, cómo se utilizan y cómo podemos llegar a aplicarlos en un entorno de trabajo cotidiano.

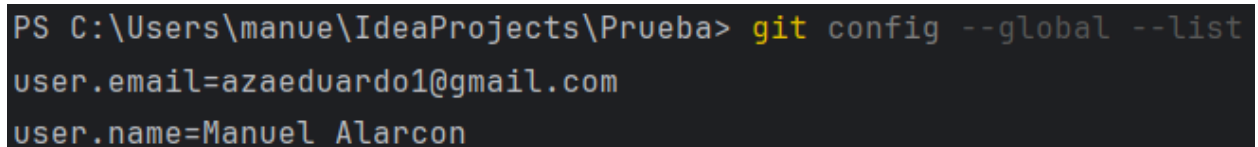
## Configuración nombre de usuario y correo electrónico

Para configurar el nombre de usuario debemos escribir en el terminal el comando `git config - - global user.name "Nombre"` y para el correo escribimos el comando `git config - - global user.email "correoejemplo@gmail.com"`



```
Terminal  Local x + v
PS C:\Users\manue\IdeaProjects\Prueba> git config -global user.name "Manuel Alarcon"
error: did you mean '--global' (with two dashes)?
PS C:\Users\manue\IdeaProjects\Prueba> git config --global user.name "Manuel Alarcon"
PS C:\Users\manue\IdeaProjects\Prueba> git config --global user.email "azaeduardo1@gmail.com"
```

Y comprobamos que haya quedado registrado escribiendo en el terminal `git config --global --list` y nos deberá aparecer el nombre y correo que registramos.



```
PS C:\Users\manue\IdeaProjects\Prueba> git config --global --list
user.email=azaeduardo1@gmail.com
user.name=Manuel Alarcon
```

## Creación repositorio

Para la creación de un repositorio luego de estar creado el proyecto en el local nos dirigimos a Github y creamos un nuevo repositorio dejándolo público y posteriormente copiamos esta información en el terminal

...or create a new repository on the command line

```
echo "# Prueba" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Alarconmanuel/Prueba.git
git push -u origin main
```

```
PS C:\Users\manue\IdeaProjects\Prueba> git add README.md
PS C:\Users\manue\IdeaProjects\Prueba> git commit -m "first commit"
[master (root-commit) 4513bcf] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
PS C:\Users\manue\IdeaProjects\Prueba> git branch -M main
PS C:\Users\manue\IdeaProjects\Prueba> git remote add origin https://github.com/Alarconmanuel/Prueba.git
PS C:\Users\manue\IdeaProjects\Prueba> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 239 bytes | 119.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Alarconmanuel/Prueba.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Hecho esta ya quedarían enlazados nuestro repositorio local y remoto en GitHub.

## Git Init

La función de este comando es el crear un nuevo repositorio de Git, inicializar un nuevo repositorio vacío o para poner un proyecto bajo un control de revisiones

```
PS C:\Users\manue\IdeaProjects\Prueba> git init
Reinitialized existing Git repository in C:/Users/manue/IdeaProjects/Prueba/.git/
```

## Git Clone

Este comando se utiliza para crear una copia o clonar un repositorio remoto y para utilizarlo creamos una carpeta damos clic derecho seleccionamos Open git bash here y escribimos git clone y por último pegamos la url del repositorio.

```
manue@LAPTOP-HOFKARS6 MINGW64 ~/IdeaProjects/Prueba (master)  
$ git clone https://github.com/Alarconmanuel/Ejercicio1Java.git
```

## Git Status

Este comando sirve para ver el estado de los archivos, esto nos ayuda a saber que archivos tenemos, si hemos modificado alguno y si están siendo rastreados o no, en el terminal se escribe git status y nos da esta información.

```
PS C:\Users\manue\IdeaProjects\Prueba> git status  
On branch master  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    .gitignore  
    .idea/  
    Prueba.iml  
    src/  
  
nothing added to commit but untracked files present (use "git add" to track)
```

## Git add

Este comando sirve para agregar todo lo que llevemos trabajado a el área de ensayo además es importante ya que si no lo utilizamos no podremos ejecutar git commit posteriormente.

```
PS C:\Users\manue\IdeaProjects\Prueba> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
    new file:   .idea/.gitignore
    new file:   .idea/misc.xml
    new file:   .idea/modules.xml
    new file:   .idea/vcs.xml
    new file:   Prueba.iml
    new file:   src/Main.java

PS C:\Users\manue\IdeaProjects\Prueba> git add .
```

### Git Commit

Este comando nos permite guardar todos los cambios hechos en un repositorio local, en el terminal se escribe `git commit -m "mensaje"`, el mensaje que va entre comillas debe ser claro y específico.

```
PS C:\Users\manue\IdeaProjects\Prueba> git commit -m "Se agrega clase"
[main 41896d1] Se agrega clase
7 files changed, 70 insertions(+)
create mode 100644 .gitignore
create mode 100644 .idea/.gitignore
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 Prueba.iml
create mode 100644 src/Main.java
```

## Git Push

Este comando se utiliza para enviar los cambios realizados en una rama local de un repositorio a un repositorio remoto, se utiliza escribiendo en el terminal `git push origin "nombre de la rama"` en este caso `main`, y para verificarlo revisamos en Github que hayan sido cargados los cambios.

```
PS C:\Users\manue\IdeaProjects\Prueba> git push origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 1.69 KiB | 577.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Alarconmanuel/Prueba.git
4513bcf..41896d1  main -> main
```

## Git Switch

Este comando se utiliza para crear ramas e ir cambiando entre estas, para crear y pasarnos de una vez a la rama nueva utilizamos `git switch -c "nombre de la nueva rama"` y para solo cambiar de rama es `git switch "nombre de la rama"`.

```
PS C:\Users\manue\IdeaProjects\Prueba> git switch -c RamaPrueba1
Switched to a new branch 'RamaPrueba1'
PS C:\Users\manue\IdeaProjects\Prueba> git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\manue\IdeaProjects\Prueba>
```

## Git Branch

Este comando nos permite crear, modificar, enumerar o eliminar ramas, los comandos para usarlos en el terminal son los siguientes:

- `git Branch` – visualizar las ramas existentes
- `git branch NuevaRama` – crea una nueva rama
- `git branch -d <branch>` - elimina el branch indicado, evitando la eliminación si están presentes commit no fusionados
- `git branch -D <branch>` - elimina el branch indicado sin comprobar la presencia de commit no fusionados
- `git branch -m <branch>` - cambia el nombre del branch actual
- `git branch -a` - enumera branch remotos

```
PS C:\Users\manue\IdeaProjects\Prueba> git branch RamaPrueba2
PS C:\Users\manue\IdeaProjects\Prueba> git branch RamaPrueba3
PS C:\Users\manue\IdeaProjects\Prueba> git branch RamaPrueba4
```

```
PS C:\Users\manue\IdeaProjects\Prueba> git branch
  RamaPrueba1
  RamaPrueba2
  RamaPrueba3
  RamaPrueba4
* main
PS C:\Users\manue\IdeaProjects\Prueba> git branch -m Rama4.0
PS C:\Users\manue\IdeaProjects\Prueba> git branch
* Rama4.0
  RamaPrueba1
  RamaPrueba2
  RamaPrueba3
  main
PS C:\Users\manue\IdeaProjects\Prueba> git branch -d RamaPrueba3
Deleted branch RamaPrueba3 (was 41896d1).
PS C:\Users\manue\IdeaProjects\Prueba> git branch
* Rama4.0
  RamaPrueba1
  RamaPrueba2
  main
```



## Git Pull

Nos sirve para descargar los cambios que se hallan realizado en el repositorio remoto y mantener el local actualizado

```
PS C:\Users\manue\IdeaProjects\Prueba> git pull origin RamaPrueba1
From https://github.com/Alarconmanuel/Prueba
 * branch          RamaPrueba1 -> FETCH_HEAD
Already up to date.
```

## Como deshacer o reversar un git commit

Para deshacer un git commit hay dos formas una si no se ha hecho git push y una cuando ya se haya hecho. Para cuando no se ha realizado git push podemos utilizar git reset, si queremos mantener los cambios escribimos en el terminal git reset --soft HEAD~1; HEAD~1 el programa lo interpreta como que se desea volver a la versión anterior y el parámetro soft lo que no elimina los cambios si no que los mantiene localmente y si no queremos mantener los cambios remplazando soft por hard, quedando git reset --hard HEAD~1.

Cuando ya hemos hecho git push debemos utilizar git revert 74a1092, esto crea un nuevo commit que deshace los cambios hechos por el anterior.

## Conclusión

El uso de comandos de git nos ayuda en el trabajo colaborativo de desarrollo de software, haciendo uso de los comandos principales y el trabajo con ramas además del manejo de repositorios locales y remotos.

## Referencias

- [Configuración de un repositorio de Git](#)
- [Guardar cambios en Git](#)
- [Comando Git Branch](#)
- [Deshacer cambios en Git](#)
- [Cómo deshacer el último commit con Git](#)