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

Introducción

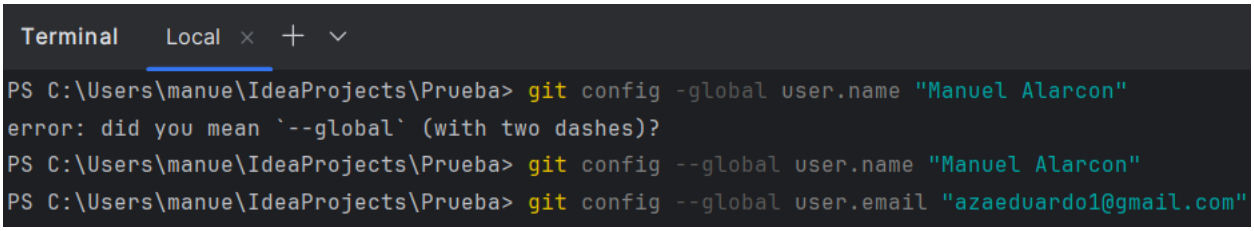
Este manual pretende que identifiquemos y visualizaremos algunos de los comandos más utilizados de Git, facilitando como podemos llegar a utilizarlos en nuestros entornos de trabajo y como estos aportan a la hora de trabajar con repositorios locales y remotos.

Objetivo

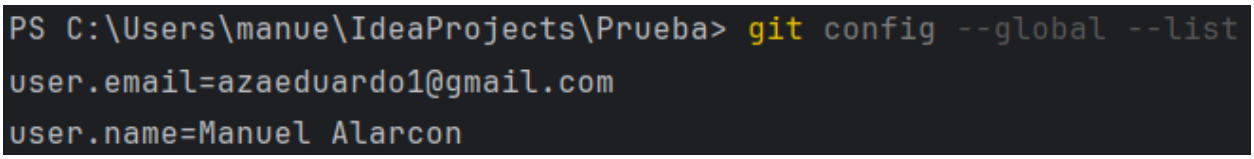
El objetivo de este manual es identificar los principales comandos de git, cómo se utilizan y cómo podemos llegar a aplicarlos en un entorno de trabajo cotidiano.

Configuración nombre de usuario y correo electrónico

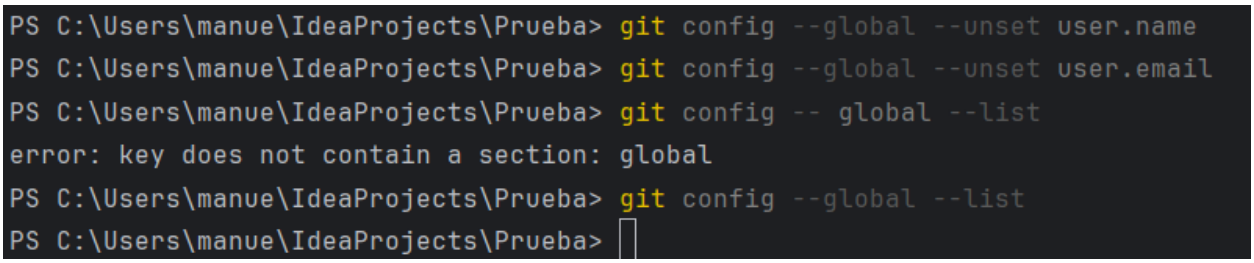
Para configurar el nombre de usuario debemos escribir en el terminal el comando git config - - global user.name “Nombre” y para el correo escribimos el comando git config - - global user.email “correoejemplo@gmail.com”



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

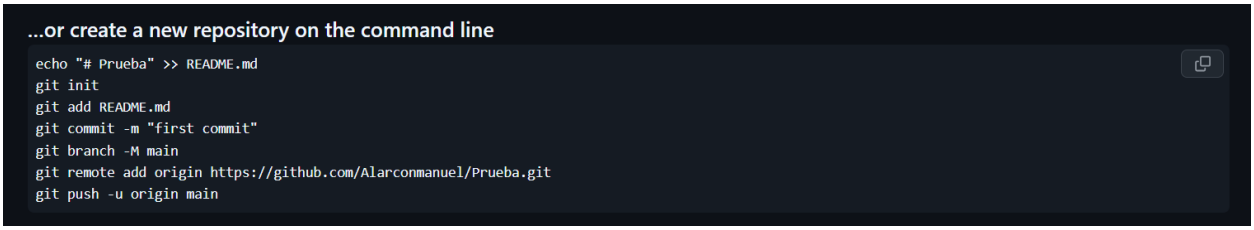


Y para eliminar el nombre registrado utilizamos git config - -unset user.name y para el correo git config - -unset user.email



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




```
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 config - -list

- **Configuración de diff**
  La configuración diff.astextplain.textconv=astextplain permite convertir archivos binarios en texto plano al comparar cambios con git diff.
- **Configuración de filter.lfs (Large File Storage)**
  Git LFS está habilitado con las configuraciones filter.lfs.clean=git-lfs clean -- %f, filter.lfs.smudge=git-lfs smudge -- %f, filter.lfs.process=git-lfs filter-process y filter.lfs.required=true, lo que permite gestionar archivos grandes de manera eficiente en el repositorio.
- **Configuración de http**
  La configuración http.sslbackend=schannel especifica que Git use la biblioteca de seguridad de Windows (Schannel) en lugar de OpenSSL para conexiones HTTPS.
- **Configuración de core (Ajustes Fundamentales)**
  En la configuración central de Git, core.autocrlf=true maneja la conversión de finales de línea entre Windows y Linux, mientras que core.fscache=true mejora el rendimiento mediante la caché del sistema de archivos. Además, core.symlinks=false deshabilita los enlaces simbólicos, core.repositoryformatversion=0 define la versión del repositorio, core.filemode=false ignora cambios en permisos de archivos, core.bare=false indica que

el repositorio no es "bare", core.logallrefupdates=true permite registrar cambios en referencias y core.ignorecase=true hace que Git ignore diferencias entre mayúsculas y minúsculas en nombres de archivos.

- **Configuración de pull**
  La configuración pull.rebase=false establece que Git usará merge en lugar de rebase al realizar git pull, lo que mantiene la historia de los commits sin reescribirlos.
- **Configuración de credential**
  Para la autenticación, credential.helper=manager activa el uso del Administrador de Credenciales de Git, mientras que credential.https://dev.azure.com.usehttppath=true permite que Azure DevOps utilice rutas HTTPS completas.
- **Configuración de init**
  La configuración init.defaultbranch=master define que la rama principal de nuevos repositorios inicializados con git init sea master.
- **Configuración de user**
  La configuración de usuario incluye user.email=azaeduardo1@gmail.com y user.name=Manuel Alarcon, los cuales establecen el correo y nombre que aparecerán en los commits realizados desde este entorno.
- **Configuración de remote.origin**
  La configuración remote.origin.url=https://github.com/Alarconmanuel/Prueba2.git define el repositorio remoto al que está vinculado el repositorio local, mientras que remote.origin.fetch=+refs/heads/*:refs/remotes/origin/* especifica cómo se obtienen las ramas remotas con git fetch.
- **Configuración Duplicada**
  Algunas configuraciones aparecen duplicadas, lo que puede deberse a que están definidas tanto a nivel global como local. Para verificarlo, se pueden usar los comandos git config --global --list y git config --local --list.

```
PS C:\Users\manue\IdeaProjects\Prueba> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=schannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=azaeduardo1@gmail.com
user.name=Manuel Alarcon
core.repositoryformatversion=0
```

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 especifico.

```
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 push origin - -delete
- Git push -u origin rama

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.
```

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
- git Branch -r – nos muestra las ramas remotas que tenemos

```
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
PS C:\Users\manue\IdeaProjects\Prueba> git branch -r
  origin/HEAD -> origin/main
  origin/Rama4.0
  origin/RamaPrueba1
  origin/RamaPrueba2
  origin/main
```

```
PS C:\Users\manue\IdeaProjects\Prueba> git reflog
896cad9 (HEAD -> main, origin/main, origin/HEAD) HEAD@{0}: commit: Archivo2
4b49185 HEAD@{1}: commit: Archivo prueba
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{2}: checkout: moving from RamaPrueba2 to main
23f74bf (RamaPrueba2) HEAD@{3}: commit: Archivo1
4513bcf HEAD@{4}: reset: moving to HEAD~1
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{5}: reset: moving to HEAD~1
8de66c2 HEAD@{6}: commit: Nueva linea
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{7}: checkout: moving from Rama4.0 to RamaPrueba2
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{8}: Branch: renamed refs/heads/rama5 to refs/heads/Rama4.0
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{10}: Branch: renamed refs/heads/RamaPrueba4 to refs/heads/rama5
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{12}: checkout: moving from main to RamaPrueba4
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{13}: checkout: moving from RamaPrueba1 to main
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{14}: checkout: moving from main to RamaPrueba1
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) HEAD@{15}: commit: Se agrega clase
4513bcf HEAD@{16}: Branch: renamed refs/heads/master to refs/heads/main
4513bcf HEAD@{18}: commit (initial): first commit
```

Git fetch - -all

Este comando nos sirve para descargar todos los cambios o archivos nuevos de un repositorio remoto a nuestro repositorio local.

```
PS C:\Users\manue\IdeaProjects\Prueba> git fetch --all
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 916 bytes | 30.00 KiB/s, done.
From https://github.com/Alarconmanuel/Prueba
   4cbb37b..4de2c02  main       -> origin/main
```

Git log

Al usar este comando nos muestra todo el historial de commits que se han realizado, admeas muestra toda la información que se realizó en cada commit.

```
PS C:\Users\manue\IdeaProjects\Prueba> git log
commit 4cbb37be486928586c042c880608780a2415ce83 (HEAD -> main, r)
Author: Manuel ALarcon <azaeduardo1@gmail.com>
Date:   Thu Feb 20 17:47:52 2025 -0500

    Revert "prueba revert"

    This reverts commit 6f12c5a1daa8dcb1190fb922ad56665167156385.
```

Git reflog

Este comando se utiliza para ver todo el registro que hemos hecho en el repositorio, se hacen con la referencia HEAD.

```
PS C:\Users\manue\IdeaProjects\Prueba> git reflog
4cbb37b (HEAD -> main, r) HEAD@{0}: checkout: moving from Rama4.0 to main
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{1}: checkout: moving from main to Rama4.0
4cbb37b (HEAD -> main, r) HEAD@{2}: checkout: moving from RamaPrueba1 to main
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{3}: checkout: moving from main to RamaPrueba1
4cbb37b (HEAD -> main, r) HEAD@{4}: revert: Revert "prueba revert"
6f12c5a HEAD@{5}: commit: prueba revert
896cad9 HEAD@{6}: commit: Archivo2
4b49185 HEAD@{7}: commit: Archivo prueba
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{8}: checkout: moving from RamaPrueba2 to main
23f74bf (RamaPrueba2) HEAD@{9}: commit: Archivo1
4513bcf HEAD@{10}: reset: moving to HEAD~1
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{11}: reset: moving to HEAD~1
8de66c2 HEAD@{12}: commit: Nueva linea
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{13}: checkout: moving from Rama4.0 to RamaPrueba2
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{14}: Branch: renamed refs/heads/rama5 to refs/heads/Rama4.0
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{16}: Branch: renamed refs/heads/RamaPrueba4 to refs/heads/rama5
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{18}: checkout: moving from main to RamaPrueba4
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{19}: checkout: moving from RamaPrueba1 to main
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{20}: checkout: moving from main to RamaPrueba1
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) HEAD@{21}: commit: Se agrega clase
4513bcf HEAD@{22}: Branch: renamed refs/heads/master to refs/heads/main
4513bcf HEAD@{24}: commit (initial): first commit
```

Git log - -one line

Este comando muestra una visualización en una sola línea de cada uno de los commit realizados con su información.

```
PS C:\Users\manue\IdeaProjects\Prueba> git log --oneline
4cbb37b (HEAD -> main, r) Revert "prueba revert"
6f12c5a prueba revert
896cad9 Archivo2
4b49185 Archivo prueba
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1, Rama4.0) Se agrega clase
4513bcf first commit
```

Git merge

Este comando se utiliza principalmente para fusionar ramas.

```
PS C:\Users\manue\IdeaProjects\Prueba> git merge
Updating 4cbb37b..4de2c02
Fast-forward
 Ejemplo fetch | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Ejemplo fetch
```

Git merge –rebase

Este comando permite cambiar el historial de confirmaciones, modificando el historial del repositorio.

```
PS C:\Users\manue\IdeaProjects\Prueba> git rebase main
Current branch main is up to date.
PS C:\Users\manue\IdeaProjects\Prueba>
```

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 “indicador” que lo sabemos utilizando git log - -oneline, esto crea un nuevo commit que deshace los cambios hechos por el anterior.

Ejemplo git revert

```
PS C:\Users\manue\IdeaProjects\Prueba> git log --oneline
6f12c5a (HEAD -> main, origin/main, origin/HEAD) prueba revert
896cad9 Archivo2
4b49185 Archivo prueba
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) Se agrega clase
4513bcf first commit
PS C:\Users\manue\IdeaProjects\Prueba> git revert 6f12
[main 4cbb37b] Revert "prueba revert"
 1 file changed, 1 deletion(-)
 delete mode 100644 revert.txt
PS C:\Users\manue\IdeaProjects\Prueba>
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\manue\IdeaProjects\Prueba> git log --oneline
6f12c5a (HEAD -> main, origin/main, origin/HEAD) prueba revert
896cad9 Archivo2
4b49185 Archivo prueba
41896d1 (origin/RamaPrueba2, origin/RamaPrueba1, origin/Rama4.0, RamaPrueba1) Se agrega clase
4513bcf first commit
PS C:\Users\manue\IdeaProjects\Prueba> git revert 6f12
[main 4cbb37b] Revert "prueba revert"
 1 file changed, 1 deletion(-)
 delete mode 100644 revert.txt
PS C:\Users\manue\IdeaProjects\Prueba>
```

Como asociar un archivo a un repositorio existente

- **Desde Github**: Desde la pagina principal del repositorio damos clic en “Add file” y posteriormente lo arrastramos hasta la ventana en el navegador o damos clic en elegir archivo, en el campo de “Mensaje de Confirmación” escribimos un mensaje corto que describa la modificación, también podemos elegir si se hace en la rama actual y una nueva y por último damos clic en “Proponer cambios”.

- **Utilizando la línea de comando**: En la carpeta donde tenemos el archivo realizamos un git bash pegamos el código de github para enlazarlo con el repositorio remoto hacemos los siguientes comandos: git init para inicializar el repositorio, git add . para que se haga la pre-carga del documento y poder utilizar git commit -m hacemos uso de este último agregando como mensaje carga de archivo por ejemplo y git push origin main para que se cargue al repositorio remoto.

```
manue@LAPTOP-HOFKARS6 MINGW64 ~/OneDrive/Escritorio/Prueba (main)
$ git init
Reinitialized existing Git repository in C:/Users/manue/OneDrive/Escritorio/Prue
ba/.git/

manue@LAPTOP-HOFKARS6 MINGW64 ~/OneDrive/Escritorio/Prueba (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/
        Manual Comandos Git.docx

nothing added to commit but untracked files present (use "git add" to track)

manue@LAPTOP-HOFKARS6 MINGW64 ~/OneDrive/Escritorio/Prueba (main)
$ git add .

manue@LAPTOP-HOFKARS6 MINGW64 ~/OneDrive/Escritorio/Prueba (main)
$ git commit -m "Archivo Manual Comandos Git"
[main dcf6ffa] Archivo Manual Comandos Git
 6 files changed, 32 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/Prueba.iml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 Manual Comandos Git.docx

manue@LAPTOP-HOFKARS6 MINGW64 ~/OneDrive/Escritorio/Prueba (main)
$ Git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 471.86 KiB | 9.63 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Alarconmanuel/Prueba2.git
   719af97..dcf6ffa  main -> main
```

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
- Git fech
- Git log
- Git merge
- Git Rebase
- Agregar archivos al repositorio

Uso de IA: En la actividad utilice un 35% de IA para ver cómo se utilizaban algunos comandos

Link repositorio: https://github.com/Alarconmanuel/Prueba.git