Centro Concertado
por el Gobierno
de La Rioja

APLICACIONES WEB

APUNTES DE GIT

Asier Vicente Reinares
Jose Adriel Marcos Sánchez

ÍNDICE

DÍA 1: 02/12/2025

Git: Control de versiones

git config –global user.name “<user>” → este comando cambia nuestro nombre de usuario para que, cuando hagamos cambios desde este dispositivo y cuenta, se guarde con nuestro nombre seleccionado.

```
Alumno@PC07 MINGW64 ~
$ git config --global user.name "Asier"
```

git config --global user.email <email> → parecido a el de usuario, pero asocia nuestro correo con nuestra identidad en este caso.

```
Alumno@PC07 MINGW64 ~
$ git config --global user.email asier.vicente@salesianoslosboscos.com
```

git config --global push.default simple → hace la forma de pushear simple

```
Alumno@PC07 MINGW64 ~
$ git config --global push.default simple
```

git init → inicia una nueva línea en la que podemos empezar a trabajar.

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT
$ git init
Initialized empty Git repository in C:/Users/Alumno/Desktop/PROGRAMAS/GIT/.git/

Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (master)
```

git branch -m <n> → cambia el nombre de nuestra línea a lo que elijamos en <n>

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (master)
$ git branch -m main

Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
```

git status

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

git add → añade los cambios que hemos hecho a la “staging area”.

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.txt

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

Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git add hello.txt

Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.txt
```

git commit -m “<nombre>”
Los primeros caracteres de un commit:

```
[main (root-commit) 9f7122e]
```

git log

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git log
commit 9f7122e8179dcaa23f1d3eda740fc158b8974eee (HEAD -> main)
Author: Asier <asier.vicente@salesianoslosboscos.com>
Date:   Tue Dec 2 09:44:15 2025 +0100

    Este es mi primer commit
```

2 commits después de hacer cambios:

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git log
commit 3c4dfaf8c428225b3191b76a52dce7837b72699e (HEAD -> main)
Author: Asier <asier.vicente@salesianoslosboscos.com>
Date:   Tue Dec 2 09:46:53 2025 +0100

    Este es mi segundo commit!

commit 9f7122e8179dcaa23f1d3eda740fc158b8974eee
Author: Asier <asier.vicente@salesianoslosboscos.com>
Date:   Tue Dec 2 09:44:15 2025 +0100

    Este es mi primer commit
```

Creación de un alias:

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git config --global alias.tree "log --graph --decorate --all --oneline"
```

Prueba del alias:

```
Alumno@PC07 MINGW64 /c/users/alumno/desktop/PROGRAMAS/GIT (main)
$ git tree
* 3c4dfaf (HEAD -> main) Este es mi segundo commit!
* 9f7122e Este es mi primer commit
```

Ejemplo secundario:

```
Alumno@PC07 MINGW64 /c/GIT/practica7-2 (main)
$ git config --global alias.oak "log --graph --all --oneline"

Alumno@PC07 MINGW64 /c/GIT/practica7-2 (main)
$ git oak
* ded0bd5 (HEAD -> main) Tercer commit en Main
| *   8633aeb (f2) Merge branch 'f1' into f2
| |\
| | * 7399b66 (f1) Tercer commit en Feature 1
| | * 051fb2f Segundo commit en Feature 1
| | * f8e5881 Primer commit en Feature 1
| |/
|/|
| * 1699d2b Commit innecesario en Feature 2
| * 42878b0 Primer commit en Feature 2
|/
* ce9053f Segundo commit en Main
* 47d753e Primer commit en Main
```

DÍA 2: 03/12/2025

WORKING DIRECTORY → **ADD** → STAGING AREA → **COMMIT** → REPOSITORY
git checkout <archive> → Hace volver a la versión anterior del archivo cuando esta en el staging area.
git checkout <hash> → mueve la HEAD a el commit que acabamos de elegir

git checkout <b-name> → lleva el HEAD a la última versión de la rama
git diff → enseña la diferencia entre nuestro archivo actualizado y su versión anterior antes de commitear los cambios

DÍA 3: 10/12/2025

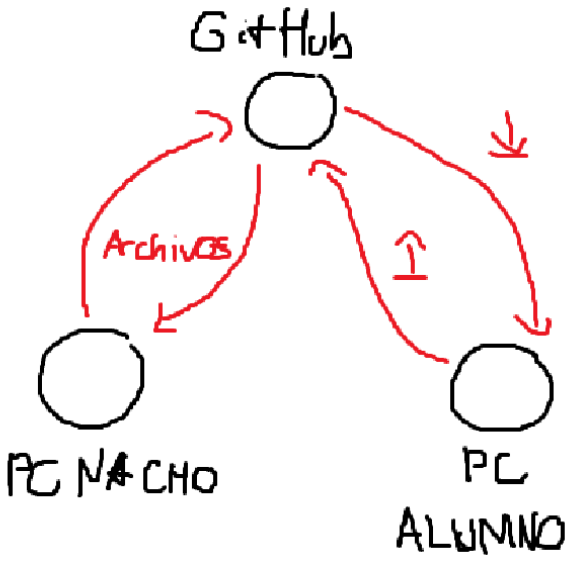
git ignore → primero hay que crear el archivo ".gitignore" para que funcione, en el que incluimos los ficheros que queremos que nos ignore. (puede ser carpeta, archivo, extensión)
git reset --hard <hash> → hace que nuestra rama vuelva a la versión del hash que hemos elegido
git reflog → nos enseña todas las acciones que hemos realizado
git reset --hard <hashToReattach> → nos vuelve a añadir una parte de nuestra rama que de la que hayamos eliminado sus cambios
git branch <name> → crea una nueva rama con el nombre escrito
git switch <name> → cambia a la rama elegida
git merge <ramaSecundaria> → juntamos una de nuestras ramas a la que estemos actualmente
Con git restore <Nombre del archivo>→ Se restauran los cambios que no se han añadido al staging area
Con git reset HEAD → sacas todo lo que hay en el staging area
Con git reset HEAD <archivo> → para sacar un archivo en especifico, para este tambien se puede utilizar git restore --staged <archivo>
Con git branch -d <nombre-rama> → Eliminas ramas

DÍA 4: 12/01/2026

git stash nos sirve para guardar los cambios en staging area sin necesidad de crear un commit (dejar algo a medias sin commit-earlo porque está inacabado o no funciona)
git stash pop se usa para recuperar los cambios guardados en el stash y volver a poder utilizarlos
git stash drop se usa para eliminar los cambios guardados en el stash y volver al archivo como estaba antes del stash
git branch -d <rama> se usa para eliminar una rama, manteniendo los commits disponibles en el log
Esta hoja se permite usar en el exámen:
https://training.github.com/downloads/es_ES/github-git-cheat-sheet.pdf
git tag "<nombre>" nos permite añadir una nota por separado a un commit. Se les puede hacer **checkout** a estos tags

GITHUB

Hay dos formas de crear repositorios: directamente en github o importando uno de nuestros repositorios locales.
Creando nuestro primer repositorio en GitHub: gitignore [AL], README [incluido]



13/01/2026 GITHUB

Conectando nuestro dispositivo a GitHub mediante SSH.
https://docs.github.com/es/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
https://docs.github.com/es/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account
Siguiendo los tutoriales de GitHub, podemos crear nuestras claves ssh para después añadirlas a GitHub. Empezamos usando este comando:
ssh-keygen -t ed25519 -C "correo@completo.com"
El cual nos creará una clave en el directorio *C:\Users\<usuario>\.ssh* que se llamará **id_ed25519** por el algoritmo que hemos utilizado. (No funciona si se le cambia el nombre a este documento, por lo que lo mejor es dejarlo por predeterminado)
Después de crear esta clave, abrimos una ventana de PowerShell con permisos de administrador e introducimos los siguientes comandos:
Get-Service -Name ssh-agent | Set-Service -StartupType Manual
Start-Service ssh-agent
Después de ejecutar estos comandos, vamos a una ventana CMD sin permisos de administrador y ejecutamos el siguiente comando:
ssh-add C:/Users/<usuario>/.ssh/id_ed25519
Y dejamos esta ventana abierta por un momento más !!!!!
Habiendo hecho esto, pasamos a GitHub en la Web y entramos a la configuración de nuestro perfil, en la que iremos al apartado en *Access* llamado *SSH and GPG keys*, donde vamos a añadir una clave nueva de ssh. Le podemos introducir el título que queramos a esta llave, por ejemplo "Portátil Clase".
Ahora vamos a volver a nuestra ventana de CMD y ejecutamos el siguiente comando:
clip < ~/.ssh/id_ed25519.pub
Y si este no funciona:
clip < C:/Users/<usuario>/.ssh/id_ed25519.pub

Y copiamos esto en el apartado de *Key* en nuestra entrada. Si el comando no ha funcionado, también podemos ir a el mismo documento .pub y abrirlo, copiando la sección que aparece antes de nuestro correo.
ssh -T git@github.com → comprobamos que nuestra llave funciona correctamente:

```
Alumno@PC07 MINGW64 ~
$ ssh -T git@github.com
Hi asiervr07! You've successfully authenticated, but GitHub does not provide
shell access.
```

14/01/2026

Con git fetch → baja los archivos pero no los aplica a tu rama
Con git push -u origin main → se suben los cambios al remoto
Con git pull → los cambios hechos en remoto se descargan a local y se aplican a la rama
Con git clone <lo que copiemos de ssh> → clonamos el repositorio de remoto a local para trabajar con las ramas, los commit y todo lo demás tal cual.