# ¿Qué son Git y GitHub?

Git es un sistema de control de versiones. Realiza un seguimiento de los cambios en los archivos y permite que las personas colaboren en proyectos. Git se utiliza a menudo para proyectos de software, pero se puede utilizar para cualquier proyecto y es especialmente potente para proyectos basados ​​en archivos de texto. Un proyecto en Git se denomina "repositorio" (o repositorio para abreviar). Un repositorio es un conjunto de archivos y directorios. Cada repositorio incluye un directorio .git que contiene información sobre cada cambio realizado en el repositorio.

GitHub es un servicio de alojamiento en línea para Git. Si bien la mayor parte de su trabajo con Git se lleva a cabo en su propia computadora, a menudo es útil sincronizar su repositorio local con un repositorio "remoto" (en GitHub), lo que le permite trabajar en varias computadoras o colaborar con otros en un proyecto. Hay muchas alternativas a GitHub, como Stash, GitLab o el alojamiento propio de sus repositorios remotos.

## Init

Para crear un nuevo repositorio, cree un directorio y use git init dentro de el.

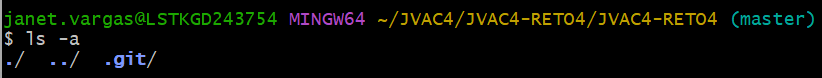
* mkdir ISpractica
* cd ISpractica
* git init

A screenshot of a computer program

AI-generated content may be incorrect.

Eso es todo, muy simple. Deberías ver un .git directorio dentro.

* ls -a # los directorios con inicial . son ocultos, por lo que se usa -a para mostrarlos



No utilizaremos este nuevo repositorio, así que puedes continuar y eliminarlo.

* rm -rf ISpractica

Bifurca este repositorio en tu cuenta de GitHub presionando el botón Bifurcar en GitHub (arriba a la derecha de la página de GitHub para este repositorio). Esto creará una bifurcación (copia) de este repositorio en tu cuenta de GitHub, creando un nuevo repositorio remoto que tú controlas. La bifurcación es útil cuando quieres usar un repositorio existente como base para tu propio proyecto. Un repositorio bifurcado también se puede usar para proponer cambios al repositorio original.

Si no estás usando GitHub para administrar tu repositorio remoto, entonces crea tu propia copia remota de este repositorio en cualquier servicio de alojamiento remoto que estés usando.

## Clone

clone Crea una copia local de un repositorio remoto. Clonemos el repositorio bifurcado del paso anterior.

* git clone https://github.com/rgfernan/git-practice.git

## Add y Commit

Ahora que tenemos un clon local del repositorio, haremos un cambio en él.

* echo "Este es un ejercicio de prueba de Git" > crawl.txt
* git status

A screen shot of a computer program

AI-generated content may be incorrect.

El comando git status muestra que tenemos un cambio no preparado. Para registrar un cambio en Git, primero tenemos que preparar ese cambio en el índice y luego confirmar todos los cambios preparados.

* git add crawl.txt # preparar el cambio
* git status # demuestra que nuestro cambio está preparado y listo para commit
* git commit -m "Hice un cambio" # Confirmar los cambios programados con un mensaje

A screen shot of a computer program

AI-generated content may be incorrect.

El paso de preparación parece innecesario en este caso, pero es útil en proyectos complejos en los que se modifican varios archivos. Probemos con un ejemplo

* echo "Las naves espaciales rebeldes, que atacan desde una base oculta, han conseguido su primera victoria contra el malvado Imperio Galáctico." >> crawl.txt # modificamos el primer archivo
* echo "Me gustan los tomates" > newFile.txt # creamos un archivo nuevo
* git status

A computer screen with text on it

AI-generated content may be incorrect.

Ahora tenemos dos cambios y queremos confirmarlos por separado, lo que podemos hacer con el comando staging. Prueba a ejecutar git status después de cada comando a continuación para ver los cambios que estás realizando.

* git add crawl.txt
* git commit -m "Iniciando crawl"
* git add newFile.txt
* git commit -m "iniciando blog de comida"

A computer screen with text and numbers

AI-generated content may be incorrect.

## HEAD y el git tree

En Git, el HEAD es el puntero a la referencia de la rama actual. Eso significa que apunta a la confirmación actual en la rama actual. Mover el encabezado es la forma en que nos movemos a confirmaciones más nuevas o más antiguas o nos movemos entre ramas. Puedes pensar en el encabezado como un marcador de "estás aquí". Echemos un vistazo a dónde estamos

* git log

A screenshot of a computer code

AI-generated content may be incorrect.

Puedes ver "HEAD" en la parte superior, pero no es una visualización particularmente clara del árbol de Git. Si tienes un programa gráfico de Git, puedes usarlo para examinar el árbol de Git, pero yo prefiero quedarme en la línea de comandos.

* git log --graph --decorate --pretty=oneline --abbrev-commit --all # Muestra un registro mucho más bonito.

A computer screen shot of a program

AI-generated content may be incorrect.

¡Eso está mucho mejor! Podemos ver HEAD claramente en la parte superior, apuntando a la rama master. También se ven otras ramas, pero ignórelas por ahora.

* git config --global alias.lola 'log --graph --decorate --pretty=oneline --abbrev-commit --all'
* git lola

A screen shot of a computer program

AI-generated content may be incorrect.

## iff

diff te permite comparar dos commits

## Branches

Como viste en git lola, nuestro árbol git tiene múltiples ramas. Vamos a enumerarlas todas:

* git branch -l

A computer screen with numbers and letters

AI-generated content may be incorrect.

Cada rama es un puntero a una instantánea de su proyecto. Si está trabajando en varias funciones diferentes, utilice siempre una rama nueva para cada una.

## New branch

Vamos a crear una nueva rama

* git branch myFirstBranch # crear una nueva rama
* git status

A screen shot of a computer program

AI-generated content may be incorrect.

¿Qué ha sucedido? No mucho. Hemos creado una nueva rama (una instantánea de nuestro código), pero eso es todo. HEAD todavía apunta a la rama master, por lo que cualquier cambio de código que hagamos se realizará en master, no en nuestra nueva rama. Para mover HEAD a una rama diferente, use el comando checkout.

* git checkout myFirstBranch
* git status

A screen shot of a computer program

AI-generated content may be incorrect.

Ahora, podemos ver que HEAD apunta a nuestra nueva rama. A menudo, querrás ir a una rama inmediatamente después de crearla. Puedes hacerlo con un solo comando:

* git checkout -b "foodBlog" #Crea una nueva rama y cámbiate a ella inmediatamente
* git status

A screen shot of a computer program

AI-generated content may be incorrect.

Si confirmamos los cambios en nuestra nueva rama, veremos que se vuelve diferente a nuestra rama maestra:

* echo "Me has fallado por última vez." >> crawl.txt
* git add crawl.txt
* git commit -m "Se agregó una cita de Vader"
* git lola
* git diff master # compara dos ramas

A screen shot of a computer

AI-generated content may be incorrect.

## reset

### restablecer un commit

Ahora, intentemos modificar el árbol con reset. Este comando mueve la rama a la que apunta HEAD (y mueve HEAD junto con ella). El comando se llama "reset" porque "restablece" o "deshace" los cambios.

El comando reset tiene varios modos, así que comencemos con --soft. Restableceremos el padre de la confirmación actual usando el atajo HEAD~. También puedes moverte a una confirmación específica especificando el hash de la confirmación (visible en git log o git lola) en lugar de HEAD~.

* git checkout master
* echo "limpiar" >> mess.txt
* git add tamagotchi.txt
* git commit -m "error agregado"
* git reset --soft HEAD~
* git lola

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

Puedes ver que la rama maestra se ha movido a la confirmación anterior, y HEAD se ha movido junto con ella. Sin embargo, ninguno de los archivos ha cambiado, y si ejecutamos git status vemos que mess.txt está preparado para ser confirmado. git reset --soft HEAD~ básicamente ha deshecho nuestra última confirmación.

A continuación, se muestra el comportamiento predeterminado para restablecer: --mixed. Esto no cambiará ningún archivo, pero deshará los cambios realizados y deshará las confirmaciones. Ejecute git lola y git status después de cada comando a continuación para ver los cambios que está realizando.

* git commit -m "Confirmar los cambios que acabamos de deshacer"
* git reset HEAD~

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

* git add . # preparamos todo lo que acabamos de deshacer
* git commit -m "Confirmar los cambios que acabamos de deshacer"
* git reset --hard HEAD~

A computer screen with text and images

AI-generated content may be incorrect.

* git checkout -b butternut
* touch pumpkin.txt
* git add pumpkin.txt
* git commit -m "vegetariano"
* echo "querido diario" >> pumpkin.txt
* git add pumpkin.txt
* git commit -m "si"
* echo "Hoy comi un rabano" >> pumpkin.txt
* git add pumpkin.txt
* git commit -m "oops"
* echo "O sea, me comi una calabaza. Que dia" >> pumpkin.txt
* git add pumpkin.txt
* git commit -m "max"
* git lola

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Esos mensajes de confirmación son inútiles. Reemplácelos con un único mensaje de confirmación.

* git reset HEAD~4
* git add pumpkin.txt
* git commit -m "Comence un registro de mis habitos alimentarios."

Ahora volvamos a la rama master para el resto de nuestro trabajo.

* git checkout master

A screen shot of a computer code

AI-generated content may be incorrect.

* git checkout master
* echo "El imperio no hizo nada malo" > protest.txt # haz un cambio
* git add protest.txt # prepara el archivo
* git reset protest.txt # quita archivo de staging

A screen shot of a computer code

AI-generated content may be incorrect.

* git checkout HEAD~
* git lola

A computer screen with text and images

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

* git checkout master
* git lola

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

* echo "alto" > crawl.txt
* git status
* git checkout HEAD~

A screenshot of a computer program

AI-generated content may be incorrect.

## checking out a file

También puedes usar checkout en una ruta de archivo. Esto no moverá HEAD, pero desmarcará cualquier cambio en ese archivo en el índice y también sobrescribirá ese archivo. git checkout [branch] file es equivalente a git reset --hard [branch] file, excepto que el último comando no existe.

Usemos checkout para eliminar nuestros cambios no deseados para rastrear

* git checkout master crawl

A black background with white text

AI-generated content may be incorrect.

## Reflog

Cuando dije que reset --hard eliminaría un commit para siempre, no estaba diciendo la verdad. git reflog está disponible como una opción de último recurso. Este comando muestra un reflog (historial) de cada vez que se actualizó la punta de una rama. Si tienes buena memoria y buenos mensajes de commit, puedes usar reflog para recuperar commits que ya no están en el árbol git.

* git reflog # muestra reflog para HEAD
* git reflog show --all # muestra reflog para todas las ramas
* git checkout blizzard
* git reset --hard HEAD~ # acabo de perder un commit que quería
* git reflog show blizzard # muestra registro de referencias de la rama de Blizzard y busca commits perdidos
* git reset <commit-hash> # Coloque el hash de commit aquí, no <>
* git status # comprobar que los cambios sean los esperados
* git reset --hard <commit-hash> # Coloque el hash de commit aquí

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

## Stash and pop

A veces, desea almacenar temporalmente algún trabajo pero no confirmarlo (por ejemplo, si desea verificar rápidamente otra rama o si se da cuenta de que comenzó a trabajar en la rama incorrecta). git stash le permitirá hacer esto. git stash pop recuperará los cambios almacenados más recientemente (los cambios almacenados se almacenan usando una pila).

* git checkout master
* echo "mcTempFace temporal" >> crawl.txt
* git stash # Almacenamos temporalmente nuestro trabajo porque esta es la rama incorrecta.
* git checkout blizzard # pasar a la rama correcta
* git stash pop # recuperar trabajo almacenado
* git add .
* git commit -m "agrego algo temporal"

A screenshot of a computer program

AI-generated content may be incorrect.

Nunca utilices stash para almacenar a largo plazo, porque te garantizo que olvidarás lo que habías almacenado y en qué rama debía ir. Si quieres almacenar algo durante más de un día y no lo quieres en ninguna de tus ramas, simplemente colócalo en una nueva rama en el lugar correspondiente.

## Merging

Cuando quieres incorporar código de una rama a otra, es momento de realizar un "merge".

## Auto-resolved merge

La mayoría de los merges son asuntos sencillos y Git las gestiona de forma completamente automática.

* git checkout merge-target
* git merge simple-feature
* git lola

A screen shot of a computer program

AI-generated content may be incorrect.

## Merge conflicts

### Resolucion manual

En ocasiones, un merge requiere intervencion manual

* git checkout merge-target
* git merge complex-feature
* git status

A computer screen with text on it

AI-generated content may be incorrect.

No hay problema. El merge ha generado algunos conflictos y Git no está seguro de cómo resolverlos. git status nos dice que estos conflictos están en crawl.txt, así que abre ese archivo en tu editor favorito.

Cuando abras crawl.txt, verás X e Y. X es de la rama X e Y de Y. Para resolver la fusión, edita el archivo como creas conveniente y luego elimina los marcadores. En este caso, deberíamos hacer Z. Cuando hayas terminado, guarda tu trabajo y regresa a la terminal.

* git add crawl.txt
* git merge --continue

A black screen with colorful text

AI-generated content may be incorrect.

## Ours vs theirs

A veces, cuando tienes conflictos de merge, simplemente querrás tomar todos los cambios de una rama específica siempre que haya un conflicto.

* git checkout merge-target
* git merge complex-feature-2

A screen shot of a computer program

AI-generated content may be incorrect.

Al fusionar, "ours" significa la rama en la que se está fusionando, mientras que "theirs" significa la rama de características desde la que se está fusionando. En nuestro caso, estamos fusionando en merge-target y estamos fusionando desde complex-feature-2. Hemos decidido que solo queremos los cambios de complex-feature-2, también conocido como "theirs", para que podamos evitar hacer un merge manual completo.

* git checkout --theirs funky.txt # toma los cambios de la rama feature-2
* git add funky.txt
* git merge --continue

A screen shot of a computer program

AI-generated content may be incorrect.

## Remotes

Un repositorio remoto es una copia remota de un repositorio. En la mayoría de los proyectos, existe un único repositorio remoto que se utiliza como fuente central de información.

## Pull

pull sincroniza tu rama local con una rama remota. Funciona realizando una fetch (descarga los ultimos datos del remoto) y despues merges los datos obtenidos en la rama actual. En su caso, no habrá nada que actualizar, ya que no se han realizado cambios en el remoto.

* git checkout master
* git pull origin master

A screen shot of a computer code

AI-generated content may be incorrect.

## Push

push sincroniza la rama remota con su rama local.

* git checkout master
* echo "porgos" >> crawl.txt
* git add crawl.txt
* git commit -m "agrega algo lindo de animal."
* git push origin master

A screenshot of a computer program

AI-generated content may be incorrect.

## checkout branch from remote

Cuando clonaste este repositorio, se creó con un único control remoto, llamado "origin". Probemos eso ahora.

* git fetch # actualizar información remota
* git checkout -b origin/check-me-out # Crea una nueva rama local para realizar un seguimiento de la rama remota

A black screen with white text

AI-generated content may be incorrect.

## Adding remotes

Dado que ha bifurcado este repositorio, desea mantenerlo actualizado con el repositorio original del que lo bifurcó. Para ello, agregue el repositorio original como un nuevo repositorio remoto llamado "upstream:

* git remote add upstream https://github.com/rgfernan/git-practice.git# agregar nuevo remoto

A black background with white text

AI-generated content may be incorrect.

## Sync upstream (fetch and merge)

Ahora que ha agregado un nuevo upstream, puede sincronizar las ramas upstream con las ramas locales

* git checkout master
* git fetch upstream
* git merge upstream/master

A computer screen shot of text

AI-generated content may be incorrect.

Nuevamente, este ejemplo específico no hará nada a menos que haya actualizado el repositorio original.

## Rebasing

El rebase mueve uno o más commits a una nueva confirmación base. Si desea incorporar cambios de una rama principal en su rama de trabajo (pero no desea volver a fusionarla con la rama principal), entonces la rebase es lo que necesita. Advertencia: [no rebase ramas públicas](https://benmarshall.me/git-rebase/)

## Auto-resolved rebase

Al igual que con el merge, la mayoría de los rebase son triviales.

* git checkout rebel-alliance
* git lola
* git rebase hoth
* git lola

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Al examinar el árbol git, puedes ver que el inicio de la rama rebel-alliance se ha movido al final de la rama hoth.

## Merge conflicts while rebasing

Alguns rebase provocarán conflictos de merge

* git checkout yoda
* git lola
* git rebase hoth
* git lola

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Como hiciste en la sección de merge, abre tu editor favorito y resuelve los conflictos de merge. Usa git status para ver los archivos conflictivos. Cuando hayas terminado, agrega los archivos y continúa con el rebase.

* git add .
* git rebase –continue

A screenshot of a computer program

AI-generated content may be incorrect.

## ours vs theirs

* git checkout palp
* git rebase hoth

A computer screen with yellow and white text

AI-generated content may be incorrect.

Durante un rebase, Git comprueba la rama en la que estás rebasando y reproduce los cambios de tu rama sobre ella. Por lo tanto, "ours" significa la rama en la que estás rebasando y "theirs" significa la rama que se está rebasando. En este caso, estamos rebasando palp ("theirs") en hoth ("ours"). Al igual que con el merge, podemos hacer uso de estos términos para evitar un merge complejo. Digamos que queremos mantener los cambios en la rama hoth siempre que haya un conflicto

* git checkout --ours padawan.txt
* git add padawan.txt
* git rebase --continue

A screen shot of a computer code

AI-generated content may be incorrect.

## Cherry-picking

Cherry pick le permite elegir un commit de una rama y aplicarlo a otra. Digamos que queremos aplicar el commit con el mensaje "PICK ME" de la rama cherries a la rama farm.

* git checkout farm
* git cherry-pick <commit-hash> # inserta el commit correcto
* git lola

A screen shot of a computer program

AI-generated content may be incorrect.