**Inicializar y clonar repositorios**

* git init: Inicializa un repositorio de Git en el directorio actual
* git clone <url>: Clona un repositorio remoto de Git en tu sistema

**Configuración**

Configuracion nombre de usuario y tu correo electrónico

* git config --local user.name "Tu nombre": Configura tu nombre de usuario de forma local (sólo en el repositorio actual)
* git config --local user.email "tu@email.com": Configura tu correo electrónico de forma local (sólo en el repositorio actual)
* git config --global user.name "Tu nombre": Configura tu nombre de usuario de forma global (para todos los repositorios en tu sistema)
* git config --global user.email "tu@email.com": Configura tu correo electrónico de forma global (para todos los repositorios en tu sistema)

**Trabajar con ramas**

* git branch: Muestra una lista de todas las ramas del repositorio
* git branch <nombre\_rama>: Crea una nueva rama con el nombre especificado
* git checkout <nombre\_rama>: Cambia a la rama especificada
* git merge <nombre\_rama>: Fusiona la rama especificada con la rama actual
* git branch -d <nombre\_rama>: Borra la rama especificada
* git branch -m <nuevo\_nombre>: Renombra la rama actual

**Añadir y confirmar cambios**

* git add <archivo>: Añade un archivo al área de “staging” (o “commit” actual)
* git add .: Añade todos los archivos modificados al área de “staging” (o “commit” actual)
* git rm <archivo>: Elimina un archivo del área de “staging” (o “commit” actual)
* git mv <archivo> <nueva\_ruta>: Actualiza la ruta de un archivo en el área de “staging” (o “commit” actual)
* git commit -m "Mensaje del commit": Confirma los cambios en el área de “staging” (o “commit” actual) con un mensaje de commit
* git commit -am "Mensaje del commit": Añade y confirma todos los archivos modificados en un solo paso
* git log: Muestra una lista de todos los commits realizados en el repositorio

**Deshacer cambios**

* git checkout -- <archivo>: Deshace los cambios de un archivo específico
* git revert <commit-hash>: Deshace los cambios de un commit específico
* git reset --soft <commit-hash>: Deshace los cambios, pero deja los archivos modificados en el área de “staging” (o “commit” actual)
* git reset --mixed <commit-hash>: Deshace los cambios y elimina los archivos modificados del área de “staging” (o “commit” actual)
* git reset --hard <commit-hash>: Deshace los cambios y elimina completamente los archivos modificados del área de “staging” (o “commit” actual) y del directorio de trabajo
* git rebase <commit-hash>: Aplica los cambios realizados en el código desde del commit especificado que no estén en la rama actual

**Ver el estado y la historia de un repositorio**

* git status: Muestra el estado actual del repositorio (qué archivos están en el área de “staging” (o “commit” actual), qué archivos han sido modificados, etc.)
* git log: Muestra una lista de todos los commits realizados en el repositorio
* git log --graph: Muestra un gráfico ASCII de la historia de commits y ramas
* git show <commit-hash>: Muestra los cambios realizados en un commit específico
* git grep <texto>: Busca el texto especificado en todos los archivos del repositorio
* git blame <archivo>: Muestra quién ha realizado cada cambio en un archivo específico
* git diff <nombre\_rama\_2>...<nombre\_rama\_1>: Muestra los cambios realizados en el código de la rama 1 que no están en la rama 2
* git diff --staged: Muestra los cambios realizados en el código que se han añadido al área de “staging” (o “commit” actual)

**Sincronizar cambios con un repositorio remoto**

Sincronizar los cambios es importante porque nos permite asegurarnos de que todos los miembros del equipo tengan acceso a la última versión del código. Si varios miembros del equipo están trabajando en el mismo proyecto, es probable que cada uno realice cambios diferentes en el código. Si no sincronizamos los cambios, es posible que terminemos con varias versiones diferentes del mismo código, lo que puede llevar a confusiones y problemas.

* git fetch <nombre\_remoto>: Descarga los cambios del repositorio remoto especificado, pero no los fusiona con la rama actual
* git pull <nombre\_remoto>: Descarga los cambios del repositorio remoto especificado y los fusiona con la rama actual
* git push <nombre\_remoto> <nombre\_rama>: Sube los cambios de la rama especificada al repositorio remoto
* `git merge

**Etiquetas**

Los “tags” en Git son etiquetas que podemos asignar a commits específicos para marcarlos de alguna manera. Por ejemplo, podemos usar tags para marcar versiones específicas de nuestro código, como versiones estables o lanzamientos. Los tags son útiles porque nos permiten acceder fácilmente a ciertos commits sin tener que recordar sus identificadores de commit.

* git tag: Muestra una lista de todos los tags del repositorio
* git tag <nombre\_tag>: Crea un nuevo tag con el nombre especificado
* git tag -a <nombre\_tag> -m "Mensaje del tag": Crea un nuevo tag anotado con el nombre y el mensaje especificados
* git tag -d <nombre\_tag>: Elimina el tag especificado
* git push <nombre\_remoto> <nombre\_tag>: Sube el tag especificado al repositorio remoto
* git push <nombre\_remoto> --tags: Sube todos los tags al repositorio remoto
* git push <nombre\_remoto> <nombre\_tag> --delete: Elimina el tag especificado del repositorio remoto
* git show <nombre\_tag>: Muestra los cambios realizados en el commit al que apunta el tag especificado

**Stash**

“Stash” es una característica de Git que nos permite guardar de forma temporal los cambios realizados en nuestro repositorio sin tener que hacer un commit. Esto es útil cuando queremos cambiar de rama o hacer un pull de un repositorio remoto, pero no queremos perder los cambios que hemos realizado.

* git stash: Guarda los cambios en un stash temporal
* git stash list: Muestra una lista de todos los stashes
* git stash apply: Aplica los cambios del stash más reciente
* git stash drop: Elimina el stash más reciente
* git stash pop: Aplica los cambios del stash más reciente y lo elimina
* git stash apply stash@{n}: Aplica los cambios del stash con el índice n (el número del “stash” en la lista)
* git stash apply stash@{n} --onto <rama>: Aplica los cambios del stash con el índice n (el número del “stash” en la lista) y los fusiona con la rama especificada
* git stash drop stash@{n}: Elimina el stash con el índice n (el número del “stash” en la lista)
* git stash pop stash@{n}: Aplica los cambios del stash con el índice n (el número del “stash” en la lista) y lo elimina