| **1.Git Clone** |
| --- |
| Realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador. |
| **git clone <https://link-con-nombre-del-repositorio>** |

| **2.Git branch** |
| --- |
| Las ramas (branch) son altamente importantes en el mundo de Git. Usando ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crearlas, listarlas y eliminarlas. |
| **git branch <nombre-de-la-rama>** |
| Este comando creará una rama en local. Para enviar (push) la nueva rama al repositorio remoto, necesitarás usar el siguiente comando:<br><br>**git push <nombre-remoto> <nombre-rama>**<br><br>**Visualización de ramas:**<br><br>**git branch**<br>**git branch --list**<br><br>**Borrar una rama:**<br><br>**git branch -d <nombre-de-la-rama>** |

**3. Git checkout**

Este es también uno de los comandos más utilizados en Git. Para trabajar en una rama, primero tienes que cambiarte a ella. Usaremos git checkout principalmente para cambiarte de una rama a otra. También lo podemos usar para chequear archivos y commits.

**git checkout <nombre-de-la-rama>**

Hay algunos pasos que debes seguir para cambiarte exitosamente entre ramas:

Los cambios en tu rama actual tienen que ser confirmados o almacenados en el guardado rápido (stash) antes de que cambies de rama.
La rama a la que quieras cambiar debe existir en el local.
Hay también un comando de acceso directo que te permite crear y cambiarte a esa rama al mismo tiempo:

**git checkout -b <nombre-de-tu-rama>**
Este comando crea una nueva rama en local (-b viene de rama (branch)) y te cambia a la rama que acabas de crear.

**4. Git status**
El comando de git status nos da toda la información necesaria sobre la rama actual.

**git status**
Podemos encontrar información como:

Si la rama actual está actualizada
Si hay algo para confirmar, enviar o recibir (pull).
Si hay archivos en preparación (staged), sin preparación(unstaged) o que no están recibiendo seguimiento (untracked)
Si hay archivos creados, modificados o eliminados

git status nos da información acerca del archivo y las ramas
**5. Git add**
Cuando creamos, modificamos o eliminamos un archivo, estos cambios suceden en local y no se incluirán en el siguiente commit (a menos que cambiemos la configuración).

Necesitamos usar el comando git add para incluir los cambios del o de los archivos en tu siguiente commit.

Añadir un único archivo:

**git add <archivo>**

Añadir todo de una vez:

**git add -A**

Si revisas la captura de pantalla que he dejado en la sección 4, verás que hay nombres de archivos en rojo - esto significa que los archivos sin preparación. Estos archivos no serán incluidos en tus commits hasta que no los añadas.

Para añadirlos, necesitas usar el git add:

Los archivos en verde han sido añadidos a la preparación gracias al git add
Importante: El comando git add no cambia el repositorio y los cambios que no han sido guardados hasta que no utilicemos el comando de confirmación git commit.

**6. Git commit**

Este sea quizás el comando más utilizado de Git. Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizás después de una tarea o asunto específico).

Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario.

También necesitamos escribir un mensaje corto para explicar qué hemos desarrollado o modificado en el código fuente.

**git commit -m "mensaje de confirmación"**

Importante: Git commit guarda tus cambios únicamente en local.

**7. Git push**

Después de haber confirmado tus cambios, el siguiente paso que quieres dar es enviar tus cambios al servidor remoto. Git push envía tus commits al repositorio remoto.

**git push <nombre-remoto> <nombre-de-tu-rama>**

De todas formas, si tu rama ha sido creada recientemente, puede que tengas que cargar y subir tu rama con el siguiente comando:

git push --set-upstream <nombre-remoto> <nombre-de-tu-rama>
or

git push -u origin <nombre-de-tu-rama>
Importante: Git push solamente carga los cambios que han sido confirmados.

**8. Git pull**

El comando git pull se utiliza para recibir actualizaciones del repositorio remoto. Este comando es una combinación del git fetch y del git merge lo cual significa que cundo usemos el git pull recogeremos actualizaciones del repositorio remoto (git fetch) e inmediatamente aplicamos estos últimos cambios en local (git merge).

**git pull <nombre-remoto>**

Esta operación puede generar conflictos que tengamos que resolver manualmente.

**9. Git revert**

A veces, necesitaremos deshacer los cambios que hemos hecho. Hay varias maneras para deshacer nuestros cambios en local y/o en remoto (dependiendo de lo que necesitemos), pero necesitaremos utilizar cuidadosamente estos comandos para evitar borrados no deseados.

Una manera segura para deshacer nuestras commits es utilizar git revert. Para ver nuestro historial de commits, primero necesitamos utilizar el  git log--oneline:

histórico de git en mi rama master
Entonces, solo necesitamos especificar el código de comprobación que encontrarás junto al commit que queremos deshacer:

git revert 3321844
Después de esto, verás una pantalla como la de abajo -tan solo presiona shift + q para salir:

El comando git revert deshará el commit que le hemos indicado, pero creará un nuevo commit deshaciendo la anterior:

commit generado con el git revert
La ventaja de utilizar git revert es que no afecta al commit histórico. Esto significa que puedes seguir viendo todos los commits en tu histórico, incluso los revertidos.

Otra medida de seguridad es que todo sucede en local a no ser que los enviemos al repositorio remoto. Por esto es que git revert es más seguro de usar y es la manera preferida para deshacer los commits.

**10. Git merge**
Cuando ya hayas completado el desarrollo de tu proyecto en tu rama y todo funcione correctamente, el último paso es fusionar la rama con su rama padre (dev o master). Esto se hace con el comando git merge.

Git merge básicamente integra las características de tu rama con todos los commits realizados a las ramas dev (o master).  Es importante que recuerdes que tienes que estar en esa rama específica que quieres fusionar  con tu rama de características.

Por ejemplo, cuando quieres fusionar tu rama de características en la rama dev:

Primero, debes cambiarte a la rama dev:

git checkout dev
Antes de fusionar, debes actualizar tu rama dev local:

git fetch

Por último, puedes fusionar tu rama de características en la rama dev:

**git merge <nombre-de-la-rama>**

Pista: Asegúrate de que tu rama dev tiene la última versión antes de fusionar otras ramas, si no, te enfrentarás a conflictos u otros problemas no deseados.