

Git clone

Git clone es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo). En otras palabras, Git clone básicamente realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador.

Hay un par de formas de descargar el código fuente, pero principalmente yo prefiero clonar de la forma con https:

```
git clone <https://link-con-nombre-del-repositorio>
```

Por ejemplo, si queremos descargar un proyecto desde Github, todo lo que necesitamos es hacer clic sobre el botón verde (clonar o descargar), copiar la URL de la caja y pegarla después del comando git clone que he mostrado más arriba.

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.

Creando una nueva rama:

```
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:

```
git push <nombre-remoto> <nombre-rama>
```

Visualización de ramas:

```
git branch
```

```
git branch -list
```

Borrar una rama:

```
git branch -d <nombre-de-la-rama>
```

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 te quieras cambiar debe existir en 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.

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

```
Cem-MacBook-Pro:my-new-app cem$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/App.js

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

        src/components/
```

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:

```
Cem-MacBook-Pro:my-new-app cem$ git add -A
Cem-MacBook-Pro:my-new-app cem$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   src/App.js
    new file:   src/components/myFirstComponent.js
```

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"
```

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>
```

O

```
git push -u origin <nombre-de-tu-rama>
```

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 cuando 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>
```

