COMANDOS GIT

GIT CLONE

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

2. Git branch

Las ramas son muy importantes en el mundo de git. Mediante el uso de ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crear, enumerar y eliminar ramas. Creando una nueva rama:

```
git branch <branch-name>
```

```
git push -u <remote> <branch-name>
```

Para ver las ramas:

```
git branch or git branch --list
```

Para borrar las ramas:

```
git branch -d <branch-name>
```

3. Git checkout

Este es también uno de los comandos Git más utilizados. Para trabajar en una rama, primero debe cambiarse a ella. Usamos git checkout principalmente para cambiar de una rama a otra. También podemos usarlo para verificar archivos y confirmaciones.

```
git checkout <name-of-your-branch>
```

4. Git status

El comando de estado de Git nos brinda toda la información necesaria sobre la rama actual.

```
git status
```

Podemos recopilar información acerca de:

- Si la rama actual está actualizada.
- Si hay algo que necesita un commit, un add, o borrarse.
- Si hay archivos preparados, sin preparar o sin seguimiento
- Si hay archivos creados, modificados o eliminados

5. Git add

Cuando creamos, modificamos o eliminamos un archivo, estos cambios ocurrirán en nuestro local y no se incluirán en la próxima confirmación (a menos que cambiemos las configuraciones).

Para agregar un solo archivo:

```
git add <file>
```

Para añadir todo de una vez:

```
git add -A
```

6. Git commit

Este es quizás el comando más utilizado de Git. Una vez que llegamos a cierto punto en el desarrollo, queremos guardar nuestros cambios (tal vez después de una tarea o problema específico).

Git commit es como establecer un punto de control en el proceso de desarrollo al que puede volver más tarde si es necesario. **También necesitamos escribir un mensaje corto** para explicar lo que hemos desarrollado o cambiado en el código fuente.

```
git commit -m "commit message"
```

Importante: Git commit guarda tus cambios solo localmente.

7. Git push

Después de confirmar los cambios (con git commit), lo siguiente que hay que hacer es enviar estos cambios al servidor remoto. Git push sube tus confirmaciones al repositorio remoto.

```
git push <remote> <branch-name>
```

Sin embargo, si tu rama se creó recientemente, también debes cargar la rama con el siguiente comando:

```
git push --set-upstream <remote> <name-of-your-branch>
```

O bien:

```
git push -u origin <branch_name>
```

Importante: Git push solo carga los cambios que están confirmados.

8. Git pull

El comando git pull se usa para obtener actualizaciones del repositorio remoto. Este comando es una combinación de **git fetch** y **git merge**, lo que significa que, cuando usamos git pull, obtienes las actualizaciones del repositorio remoto (git fetch) e inmediatamente aplica los últimos cambios en su local (git merge). (En simples palabras, sirve para traer el repositorio remoto a tu repositorio local).

```
git pull <remote>
```

Esta operación puede causar conflictos que debes resolver manualmente.

9. Git revert

A veces necesitamos deshacer los cambios que hemos hecho. Hay varias formas de deshacer nuestros cambios de forma local o remota (depende de lo que necesitemos), pero debemos usar estos comandos con cuidado para evitar eliminaciones no deseadas

10. Git merge

Cuando hayas completado el desarrollo en tu rama y todo funcione bien, el paso final es fusionar la rama con la rama principal (dev o master branch). Esto se hace con el comando `git merge`.

Git merge básicamente integra su rama de características (feature branch) con todas sus confirmaciones en la rama dev (o master). Es importante recordar que primero debes estar en la rama específica que deseas fusionar con tu rama de características.