# **COMANDOS DE GIT EN CONSOLA:**

### **1º Configuración y primeros pasos:**

**git init <nombre\_proyecto>:** Inicia un repositorio local con el nombre.

**git init** : Dentro de la carpeta del proyecto, inicia un repositorio en esta carpeta.

**git clone <url>** : Clona el repositorio que haya en la url.

**git config --global user.name "Nombre Usuario"**

**git config --global user.email "Email Usuario"**

**git config --global credential.helper 'cache --timeout=3600'**: Recuerda usuario y contraseña temporalmente, en este caso una hora

**git config --global credential.helper cache**: Recuerda usuario y contraseña durante un tiempo default de 15 min.

### **2º Trabajo con repositorios:**

**git status** : Hace un listado de los archivos nuevos o a insertar en un commit.

**git add <nombre de archivo>** : Añade al stagin área (o comienza el seguimiento)

**git add .** : Añade TODOS los archivos al stagin area (o comienza el seguimiento)

**git reset <archivo>** : Saca de stagin area el archivo, poniendo -- . saca todos.

**git commit -m "Comentario de commit"** : Realiza un commit. Guarda una copia del estado del archivo/s en ese momento y añade el código SHA a la cabecera del repositorio.

**Deshacer un commit**:

**git reset --soft HEAD~1** : Deshace el último commit, pero deja los cambios en local, fuera de stagin área.

**git reset --hard HEAD~1** : Deshace el último commit y elimina los cambios realizados

**git checkout -- <nombre\_archivo>** :elimina los cambios en el archivo. Si se pone 'git checkout -- .' se pierden TODOS los cambios y no se pueden recuperar por ningún medio.

### **3º Trabajo con ramas:**

**git checkout -b <nombre\_de\_rama>:** Crea una nueva rama en el repositorio, con los archivos en el estado en el que se encuentran en el momento que se crea y desde la rama en la que se crea y nos coloca en esa rama.

**git branch <nombre\_de\_rama>** : Crea una rama a partir de donde estamos y con el estado de los archivos en el que nos encontramos pero no nos lleva a ella.

**git checkout <nombre\_de\_rama>** : Nos lleva a la rama que mencionamos, no se puede hacer un cambio de rama si tenemos archivos en la rama que nos encontramos sin un commit, puesto que el cambio de rama haría que se perdiesen.

**git branch -d <nombre\_de\_rama>** Eliminar una rama en local.

En el caso de que esa rama contenga trabajos sin fusionar, el comando anterior nos devolverá el siguiente error:

error: The branch 'nombre-rama' is not an ancestor of your current HEAD.

If you are sure you want to delete it, run 'git branch -D nombre-rama'.

**git branch -D <nombre\_de\_rama>**: Fuerza la eliminación de la rama mencionada.

En el caso de querer eliminar una rama del repositorio remoto, la sintaxis será la siguiente:

**git push origin :nombre-rama:** De esta forma, desaparecerá la rama nombre-rama del servidor.

### **4º Sincronización:**

**git fetch** : Descarga el historial del repo remoto.

**git merge <nombre\_de\_rama>** : Trae los cambios de la rama mencionada a la rama en la que nos encontramos.

**git pull** : Baja el historial del repo remoto e incorpora los cambios. Es necesario hacer esto antes de subir nada.

**git push <nombre\_de\_rama>** : Sube los commits que tengamos en la rama local a la rama con ese nombre del repositorio remoto. Puede ser necesaria la palabra origin.

### **Errores:**

**git pull origin master --allow-unrelated-histories**: Cuando has creado un repo remoto y añadido el remoto en local pero no has clonado, y lanza un error al hacer pull