CONFIGURAR GIT

Establece el nombre que desea que esté anexado a las transacciones

```
$ git config --global user.name "nombre usuario"
```

Establece el email que desea que esté anexado a las transacciones

```
$ git config --global user.email "usuario@mail.com"
```

Habilita la colorización de la salida de los comandos por la terminal

```
$ git config --global color.ui auto
```

Establece el editor predeterminado y espera a que el editor se cierre

```
$ git config --global core.editor "[editor] --wait"
```

Establece un alias para un comando más largos de GIT

```
$ git config --global alias.as [comando]
```

```
$ git config --system  → [path]/etc/gitconfig
$ git config --global  → ~/.gitconfig
$ git config --local   → .git/config
```

CONFIGURAR REMOTOS

Añadir una conexión remota y nombramos la conexión como 'origin'

```
$ git remote add origin [url.repo.git]
```

Elimina una conexión remota especificando su nombre. Ej: 'origin'

```
$ git remote remove [nombre-conexion]
```

SINCRONIZAR CAMBIOS

Descarga los últimos cambios de remoto y rama especificada

```
$ git fetch [alias-remoto]/[rama]
```

Fusiona la rama remota con la rama local actual

```
$ git merge [alias-remoto]/[rama]
```

CLONAR REPOSITORIOS

Clonar un repositorio vía https

```
$ git clone https://<host>/<user>/<repo>.git
```

Clonar un repositorio vía ssh

```
$ git clone ssh://<host>/<user>/<repo>.git
```

Clonar una rama específica

```
$ git clone --branch <nombre-rama> <remote-repo-url>
```

Clonar una rama específica usando la abreviatura

```
$ git clone -b <nombre-rama> <remote-repo-url>
```

OPERACIONES BÁSICAS EN LOCAL

Crea un nuevo repositorio local en la ruta especificada.

```
$ git init [path]
```

Crea un nuevo repositorio local con el nombre especificado en el directorio actual

```
$ git init "nombre proyecto"
```

Agrega archivos al área de preparación (*staging*)

```
$ git add [archivos]
```

Agrega todos los archivos para preparación (*staging*)

```
$ git add -A # equivalente a: $git add .
```

Enumera todos los archivos nuevos o modificados que se deben modificar. El flag -s viene de *short status*

```
$ git status [-s]
```

Toma todos los archivos en el (*staging*), crea un nuevo objeto de confirmación y le añade un mensaje adjunto (si se omite el flag -m, se abre el editor por defecto, en la mayoría de los casos vim)

```
$ git commit -m "mensaje descriptivo"
```

Muestra las diferencias del archivo entre el área de preparación y la última versión del archivo

```
$ git diff
```

REGISTRO HISTORIAL

Listar el historial de commit, con sus respectivos ID. Con gráfico de bifurcaciones

```
$ git log --graph --oneline --decorate
```

Listar el historial de commit, con sus respectivos ID. Con el hash abreviado y lo leemos la salida con el programa cat

```
$ git log --oneline | cat
```

TRABAJAR CON RAMAS

Crea una nueva rama y **NO** se cambia a la rama creada, sólo la crea

```
$ git branch <nombre-rama>
```

Listar todas las ramas existentes

```
$ git branch # o también : $git branch --list
```

Elimina una rama local con el respectivo nombre (*se debe estar en otra rama para poder eliminar rama*).

```
$ git branch -d <nombre-rama>
```

Crea una nueva rama y se cambia a ella (*en un solo comando*).

```
$ git checkout -b <nombre-rama>
```