**CONFIGURACION DE USUARIO**

$ git config --global user.name "[name]" # Genera un nombre de usuario global.

$ git config --global user.email "[email address]" # Genera un nombre de usuario global.

$ git config user.name "[name]" # Genera un nombre de usuario para un repositorio particular.

$ git config user.email "[email address]" # Genera un nombre de usuario para un repositorio particular.

REPOSITORIOS

$ git init [project-name] # Inicia un nuevo repositorio.

$ git clone [url del proyecto] # Para descargar y sincronizar por primera vez un proyecto.

$ git remote add origin [url del repositorio remoto] # Sirve para sincronizar nuestro repositorio local con un repositorio remoto de Github.

$ git remote -v # Para comprobar si nuestro repositorio local está sincronizado con algún repositorio remoto creado en Github.

HACER CAMBIOS Local

$ git status # Permite ver el estado de nuestros archivos, de las ediciones y cambios.

$ git reset (file) # Mueve el archivo del área de preparación, pero preserva su contenido.

$ git add . # Envía todos los archivos de la carpeta en la que estamos posicionados y de sus subdirectorios al área de preparación.

$ git add (nombre del archivo.extensión) Envía solo el archivo referenciado al área de preparación.

$ git commit -m "[descriptive message]" # Guarda un parque con los cambios que hemos hecho, con un mensaje.

$ git rm # Elimina el archivo de tu directorio de trabajo y no elimina su historial del sistema de versiones.

$ git rm -r # (-r: modo recursivo) Elimina el directorio de destino y todo el contenido de dicho directorio.

REHACER COMMITS

$ git reset (commit) # Deshace todos los commits después de (commit), preservando los cambios localmente

$ git reset --hard (commit) # Desecha todo el historial y regresa al commit especificado.

USO DE LAS BRANCH

$ git branch [branch-name] # Crea una rama.

$ git branch # Enumera todas las ramas de tu repositorio, es similar a git branch –list.

$ git checkout [branch-name] # Te ubicas en esa rama. Generalmente, Git solo permitirá que nos movamos a otra rama si no tenemos cambios. Si tenemos cambios, para cambiarnos de rama, debemos: 1. eliminarlos (deshaciendo los cambios); 2. conﬁrmarlos (haciendo un git commit).

$ git checkout -b (branch-name) # crea una rama y nos cambia a ella con un solo comando.

$ git merge [branch] # Mergea una rama. Combina el historial de la rama especificada con la rama actual.

$ git branch -d [branch-name] # Borra la rama, Git evita que eliminemos la rama si tiene cambios que aún no se han fusionado con la rama Main.

$ git branch -D [branch-name] # Fuerza la eliminación de la rama especiﬁcada, incluso si tiene cambios sin fusionar.

HISTORIAL

$ git log # Muestra el historial de confirmaciones en la rama actual.

$ git log --oneline # Mostrar información en una sola línea.

$ git log -[n] # Ver un número limitado de commits.

$ git log -p # Ver información extendida del commit.

$ git log --all --decorate --oneline –graph # Ver información de los logs a color y en una sola línea.

$ git diff [first-branch]...[second-branch] # Muestra las diferencias entre las ramas.

$ git diff # Muestra las diferencias entre archivos que no se han enviado aún al área de preparación.

$ git diff –staged # Muestra las diferencias del archivo entre el área de preparación y la última versión del archivo.

SINCRONIZAR CAMBIOS

$ git push origin [branch] # Envia los commits a la rama que nombremos.

$ git pull # Descarga, actualiza y fusiona tu repositorio local con todo lo que está en el repositorio remoto de Github.

$ git pull origin “nombre rama” # Descarga y actualiza la rama especificada.

$ git fetch # actualizamos el repositorio local según el repositorio remoto sin fusionar.

$ git rev-parse --show-toplevel.

OTROS

$ git config --system core.longpaths true # Si nos sale un mensaje que nos informa que la ruta de los archivos que queremos descargar del repositorio local es muy larga, este comando nos permite cambiar la configuración de Git para que soporte rutas largas.

$ git tag (tag) (commit ID) # crea una etiqueta.

$ git rm –cached # elimina el archivo del área de preparación.