GIT y un Poquito de GITHUB

Si estamos en Windows, no usemos interfaces gráficas ni el cmd, dan muchos problemas. Los desarrolladores usan GIT BASH:

Para que nuestro proyecto lleve nuestro nombre, esto es útil para saber luego quien ha hecho las distintas partes de un proyecto:

Git config --global user.name "Eliezer Cabrales"

Git config --global user.mail cabrales.eliezer@correo.com

Para ver el archivo de configuración global:

Git config --global core.editor "code --wait"

Git config --global -e

En Windows cada salto de línea, se meten DOS caracteres, CR y LF, mientras que en Linux/MAC, solo mete LF. Por eso, al programar código desde Windows, el usuario debe eliminar CR si lo va a subir al repositorio, o, si lo va a descargar, para que funcione debe funcionar este caracter.

Para eso la propiedad auto_CRLF debe estar en "input":

Git config --global core.autocrlf true/ input (pondremos true en Windows e Input en Linux/MAC)

Podemos ver más opciones de git con:

git config -h

TRABAJANDO CON GIT

Para inicializar el directorio de Git, usamos:

git init

Dentro del directorio que se crea (.git) es donde se van a guardar las versiones de todo nuestro código, detalles de implementación y todo está optimizado. La carpeta es ignorada por todos los repositorios y no pasa ni se comparte entre los distintos desarrolladores o Hubs.

git add (archivo)

Selecciona los archivos a una etapa (stage). Estamos verificando todos los cambios que finalmente acabarán en nuestro repositorio. Esto no modifica directamente el repositorio, sino que va apuntando los cambios que seleccionemos al directorio. Antes de hacer el commit, podemos eliminar.

git commit

Es la siguiente etapa. Es comprometer los cambios, asegurarlos (control de versiones).

git push

Subir nuestro repositorio al servidor.

Si nosotros eliminamos un archivos debemos ejecutar el mismo comando a ese archivo eliminado para eliminarlo, ya sea en la etapa del stage, el commit o el push.

git pull

Actualizar el repositorio local por el que tenemos en el servidor.

git status

Nos muestra el estado actual del repositorio, números de commits. También los tracked files. Por defecto recuerda que hace falta hacer git add para todos los archivos que queramos que Git controle

git add .

Añade todos los archivos. Esto es una **MALA** práctica, pues podríamos añadir al repositorio archivos grandes, inútiles o hasta confidenciales.

Lo mejor es seleccionar el/los archivo/s específicamente:

git add (archivos separados por espacio)

Ahora para hacer commit, es decir, confirmar los cambios:

git commit -m "Commit Inicial"

Si no se añade el comentario (-m), se abrirá el editor de texto para añadir un mensaje. Este mensaje sirve para especificar los cambios realizados.

ELIMINAR, RECUPERAR y RENOMBRAR ARCHIVOS EN GIT

Para eliminar un archivo del stage:

git rm (archivo)

Para recuperarlo en el stage usamos:

git restore --staged (archivo)

Y para simplemente recuperar un archivo borrado usamos:

git restore (archivo)

Esto recuperará el archivo incluso en una etapa antes de hacer el commit.

Para renombrar archivos, podemos hacerlo manualmente, y luego haciendo un add al nombre viejo y otro al nombre nuevo o usar el comando:

git mv (archivo) (archivo nombre nuevo)

IGNORAR ARCHIVOS Y CARPETAS

Creamos un documento llamado .gitignore

Dentro añadimos en cada linea el nombre de los archivos que queremos que Git ignore. Luego, agregamos el archivo .gitignore con:

git add .gitignore

git commit -m "agregando gitignore"

MEJOR CONTROL DE CAMBIOS

Con **git status -s** en vez de hacer un verbose grande, nos da un resumen de lo que está pasando en nuestro stage.

La **A** significa que está agregándose. En **verde** significa que está listo, en **rojo** que falta.

La **M** es que se está modificando. En **verde** significa que está listo, en **rojo** que falta.

Con **git diff** muestra los cambios entre el stage actual y el commit. Muestra los cambios de todas las líneas. Las rojas significan que han sido eliminadas y las verdes, agregadas. Arriba, muestra entre **@@ -X +X,Y @@**, con **-** las líneas por donde se modifican/eliminan y con **+** las líneas que se agregan. **Recuerda que los saltos de línea son un carácter que modifica una línea completa**, no hay que asustarse por ver una línea en rojo si después le hemos agregado un intro. Pulsa **Q** para salir.

Con **git diff --staged**

Aquí podemos ver los cambios que se encuentran en nuestra etapa staged. Con la primera muestra los cambios que todavía no han pasado a la etapa del staged, pero con staged vemos los cambios con el commit, lo cual es más útil.

git log

Muestra quien y cuando ha realizado los distintos commits.

git logg --oneline Nos permite verlo más fácilmente.

RAMAS (BRANCHES)

Las ramas permiten bifurcar el trabajo para los distintos desarrolladores. De tal forma que cada cual tenga sus propios commit. Lo contrario es un **Merge**, que es cuando la rama vuelve a la Branch principal.

Con **git branch** podemos saber la rama en la que trabajamos. **Main** es la rama principal en todos los proyectos por defecto.

Para crear una rama nueva, usamos:

git checkout - b feature/nombre-de-la-funcionalidad

Esto también nos hará cambiar de rama. Simplemente para cambiar entre ramas existentes:

git chekcout (rama)

CONECTARSE A UN REPOSITORIO REMOTO:

git remote add origin https://github.com/Eliezer-Cabrales/ejemplo.git

git push -u origin main

Nos permite subir los cambios. Por defecto en GitHub no se encuentra creada la rama Main por defecto, por eso, con la opción **-u** la estamos creando.

Por defecto pide nombre de usuario y luego una contraseña (TOKEN).

Esta contraseña no es nuestra contraseña, sino una key, un **Personal Access Token**. Debemos generarlo en GitHub (Mi cuenta -> Developer Setting -> Generar Token de seguridad clásico).

Al token le ponemos un nombre, una fecha de expiración y el alcance de privilegios del token. Se genera (es imporante guardarlo porque no se vuelve a generar de nuevo).

Una vez se haya configurado, solo necesita hacer un **git add (arhivo)** - **git commit -m "comentario"** - **git push**

Con eso ya subiría el nuevo archivo/actualización.