TEMA: Qué es GitHub y todos sus comandos para usar en consola

Github es un repositorio online gratuito que permite gestionar proyectos y controlar versiones de código. Es muy utilizado por desarrolladores para almacenar sus trabajos dando así la oportunidad a millones de personas de todo el mundo a cooperar en ellos.

Se podría hablar de Github como la red social pensada para desarrolladores, siendo este repositorio uno de los más usados a nivel mundial.

Podemos seguir e interactuar con personas interesadas en un tipo de proyecto en concreto, dando a conocer los nuestros o cooperando en el proyecto de terceros.
Git es un sistema de control de versiones distribuido de código abierto desarrollado por Linus Torvalds, el creador de Linux.

El control de versiones distribuido permite a los desarrolladores descargar un software, realizar cambios y subir la versión que han modificado. Todas las modificaciones subidas se guardan en versiones independientes, no sobrescribiendo en el archivo original.

La diferencia entre el control de versiones y Git, es que en Git cada desarrollador tendrá en el ordenador una copia del código fuente original y de las versiones disponibles del proyecto, permitiendo la ramificación y fusión.

De esta forma todos los desarrolladores interesados en el proyecto podrán ver las modificaciones realizadas y contribuir mejorando el código del Software.

Cuando nos referimos a la parte del hub hablamos de lo que hace especial a este Git, la comunidad e interacción con otros usuarios.

Comandos

1° 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 usuario.nombre "Nombre Usuario"

git config --global user.email "Usuario de correo electrónico"

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 predeterminado de 15 min.

2º Trabajo con repositorios:

git status : Hace un listado de los archivos nuevos oa insertar en un commit.

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

git agregar. : Agregue 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. Guarde una copia del estado del archivo/s en ese momento y agregue el código SHA a la cabecera del repositorio.

Deshacer un comité :

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

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 rama <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 repositorio 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 repositorio 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 origen.

5° Errores:

git pull origin master --allow-un related-histories : Cuando ha creado un repositorio remoto y añadido el remoto en local pero no ha clonado, y lanza un error al hacer pull