QUÉ ES GITHUB Y TODOS SUS COMANDOS PARA USAR EN CONSOLA

GitHub es una plataforma de desarrollo colaborativo que aloja proyectos en la nube utilizando el sistema de control de versiones llamado Git. Ayuda a los desarrolladores a almacenar y administrar el código llevando un registro de cambios. Generalmente el código es abierto, lo que permite realizar proyectos compartidos y mantener el seguimiento detallado de su progreso. La plataforma GitHub también funciona como red social conectando a los desarrolladores con los usuarios. Como usuario puedes descargar programas o aplicaciones, y de la misma manera puedes aportar a su desarrollo ofreciendo mejoras y discutir las cuestiones que te interesan en foros temáticos.

COMANDOS

Git clone

Git clone es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo). En otras palabras, Git clone básicamente realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador.

Hay un par de formas de descargar el código fuente, pero principalmente yo prefiero clonar de la forma con https:

git clone <https://link-con-nombre-del-repositorio>

Por ejemplo, si queremos descargar un proyecto desde Github, todo lo que necesitamos es hacer clic sobre el botón verde (clonar o descargar), copiar la URL de la caja y pegarla después del comando git clone que he mostrado más arriba.

bootstrap-github-1

Código fuente de Bootstrap en Github

Esto hará una copia del proyecto en tu espacio de trabajo local y así podrás empezar a trabajar con él.

2. Git branch

Las ramas (branch) son altamente importantes en el mundo de Git. Usando ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crearlas, listarlas y eliminarlas.

Creando una nueva rama:

git branch <nombre-de-la-rama>

Este comando creará una rama en local. Para enviar (push) la nueva rama al repositorio remoto, necesitarás usar el siguiente comando:

git push <nombre-remoto> <nombre-rama>

Visualización de ramas:

git branch

git branch --list

Borrar una rama:

git branch -d <nombre-de-la-rama>

3. Git checkout

Este es también uno de los comandos más utilizados en Git. Para trabajar en una rama, primero tienes que cambiarte a ella. Usaremos git checkout principalmente para cambiarte de una rama a otra. También lo podemos usar para chequear archivos y commits.

git checkout <nombre-de-la-rama>

Hay algunos pasos que debes seguir para cambiarte exitosamente entre ramas:

Los cambios en tu rama actual tienen que ser confirmados o almacenados en el guardado rápido (stash) antes de que cambies de rama.

La rama a la que te quieras cambiar debe existir en local.

Hay también un comando de acceso directo que te permite crear y cambiarte a esa rama al mismo tiempo:

git checkout -b <nombre-de-tu-rama>

Este comando crea una nueva rama en local (-b viene de rama (branch)) y te cambia a la rama que acabas de crear.

4. Git status

El comando de git status nos da toda la información necesaria sobre la rama actual.

git status

Podemos encontrar información como:

Si la rama actual está actualizada

Si hay algo para confirmar, enviar o recibir (pull).

Si hay archivos en preparación (staged), sin preparación(unstaged) o que no están recibiendo seguimiento (untracked)

Si hay archivos creados, modificados o eliminados

5. Git add

Cuando creamos, modificamos o eliminamos un archivo, estos cambios suceden en local y no se incluirán en el siguiente commit (a menos que cambiemos la configuración).

Necesitamos usar el comando git add para incluir los cambios del o de los archivos en tu siguiente commit.

Añadir un único archivo:

git add <archivo>

Añadir todo de una vez:

git add -A

Si revisas la captura de pantalla que he dejado en la sección 4, verás que hay nombres de archivos en rojo - esto significa que los archivos sin preparación. Estos archivos no serán incluidos en tus commits hasta que no los añadas.

Para añadirlos, necesitas usar el git add:

git-add

Los archivos en verde han sido añadidos a la preparación gracias al git add

Importante: El comando git add no cambia el repositorio y los cambios que no han sido guardados hasta que no utilicemos el comando de confirmación git commit.

6. Git commit

Este sea quizás el comando más utilizado de Git. Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizás después de una tarea o asunto específico).

Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario.

También necesitamos escribir un mensaje corto para explicar qué hemos desarrollado o modificado en el código fuente.

git commit -m "mensaje de confirmación"

Importante: Git commit guarda tus cambios únicamente en local.

7. Git push

Después de haber confirmado tus cambios, el siguiente paso que quieres dar es enviar tus cambios al servidor remoto. Git push envía tus commits al repositorio remoto.