Que es GIT

Git es un Sistema de Control de Versiones Distribuido (DVCS) utilizado para guardar diferentes versiones de un archivo (o conjunto de archivos) para que cualquier versión sea recuperable cuando lo desee.
Git también facilita el registro y comparación de diferentes versiones de un archivo. Esto significa que los detalles sobre qué cambió, quién cambió qué, o quién ha iniciado una propuesta, se pueden revisar en cualquier momento.

Control de versiones con GIT

Un Sistema de Control de Versiones (VCS) se refiere al método utilizado para guardar las versiones de un archivo para referencia futura.
De manera intuitiva muchas personas ya utilizan control de versiones en sus proyectos al renombrar las distintas versiones de un mismo archivo de varias formas como blogScript.js, blogScript_v2.js, blogScript_v3.js, blogScript_final.js, blogScript_definite_final.js, etcétera. Pero esta forma de abordarlo es propenso a errores y inefectivo para proyectos grupales.

Estados de un archivo en GIT

En Git hay tres etapas primarias (condiciones) en las cuales un archivo puede estar: estado modificado, estado preparado, o estado confirmado.

Estado modificado

Un archivo en el estado modificado es un archivo revisado – pero no acometido (sin registrar).
En otras palabras, archivos en el estado modificado son archivos que has modificado, pero no le has instruido explícitamente a Git que controle.

Estado preparado

Archivos en la etapa preparado son archivos modificados que han sido seleccionados – en su estado (versión) actual – y están siendo preparados para ser guardados (acometidos) al repositorio .git durante la próxima instantánea de confirmación.
Una vez que el archivo está preparado implica que has explícitamente autorizado a Git que controle la versión de ese archivo.

Estado confirmado

Archivos en el estado confirmado son archivos que se guardaron en el repositorio .git exitosamente.
Por lo tanto un archivo confirmado es un archivo en el cual has registrado su versión preparada en el directorio (carpeta) Git.
Nota: El estado de un archivo determina la ubicación donde Git lo colocará.

Como se configura un repositorio

¿Qué es un repositorio de Git?

Un repositorio de Git es un almacenamiento virtual de tu proyecto. Te permite guardar versiones del código a las que puedes acceder cuando lo necesites.

Para crear un nuevo repositorio, usa el comando git init. git init es un comando que se utiliza una sola vez durante la configuración inicial de un repositorio nuevo. Al ejecutar este comando, se creará un nuevo subdirectorio .git en tu directorio de trabajo actual. También se creará una nueva rama principal.

Versión de un proyecto existente con un repositorio de Git nuevo

En este ejemplo, suponemos que ya cuentas con una carpeta de proyecto en la que quieres crear un repositorio. Primero deberás establecer el directorio de la carpeta raíz del proyecto con el comando cd y luego ejecutar git init.

```
---------------------------------
cd /path/to/your/existing/code
git init
---------------------------------
```

Apuntar git init a un directorio de proyecto existente hará que se ejecute la misma configuración de inicio mencionada arriba, pero en el ámbito de ese directorio.

```
---------------------------------
git init <project directory>
---------------------------------
```

Comandos en GIT

git add
Mueve los cambios del directorio de trabajo al área del entorno de ensayo. Así puedes preparar una instantánea antes de confirmar en el historial oficial.

rama de git
Este comando es tu herramienta de administración de ramas de uso general. Permite crear entornos de desarrollo aislados en un solo repositorio.

git checkout
Además de extraer las confirmaciones y las revisiones de archivos antiguas, git checkout también sirve para navegar por las ramas existentes. Combinado con los comandos básicos de Git, es una forma de trabajar en una línea de desarrollo concreta.

git clean
Elimina los archivos sin seguimiento de tu directorio de trabajo. Es la contraparte lógica de git reset, que normalmente solo funciona en archivos con seguimiento.

git clone
Crea una copia de un repositorio de Git existente. La clonación es la forma más habitual de que los desarrolladores obtengan una copia de trabajo de un repositorio central.

git commit
Confirma la instantánea preparada en el historial del proyecto. En combinación con git add, define el flujo de trabajo básico de todos los usuarios de Git.

git commit --amend
Pasar la marca --amend a git commit permite modificar la confirmación más reciente. Es muy práctico si olvidas preparar un archivo u omites información importante en el mensaje de confirmación.

git config
Este comando va bien para establecer las opciones de configuración para instalar Git. Normalmente, solo es necesario usarlo inmediatamente después de instalar Git en un nuevo equipo de desarrollo.

git fetch
Con este comando, se descarga una rama de otro repositorio junto con todas sus confirmaciones y archivos asociados. Sin embargo, no intenta integrar nada en el repositorio local. Esto te permite inspeccionar los cambios antes de fusionarlos en tu proyecto.

git init
Inicializa un nuevo repositorio de Git. Si quieres poner un proyecto bajo un control de revisiones, este es el primer comando que debes aprender.

git log
Permite explorar las revisiones anteriores de un proyecto. Proporciona varias opciones de formato para mostrar las instantáneas confirmadas.

Git merge
Es una forma eficaz de integrar los cambios de ramas divergentes. Después de bifurcar el historial del proyecto con git branch, git merge permite unirlo de nuevo.

git pull
Este comando es la versión automatizada de git fetch. Descarga una rama de un repositorio remoto e inmediatamente la fusiona en la rama actual. Este es el equivalente en Git de svn update.

git push
Enviar (push) es lo opuesto a recuperar (fetch), con algunas salvedades. Permite mover una o varias ramas a otro repositorio, lo que es una buena forma de publicar contribuciones. Es como svn commit, pero envía una serie de confirmaciones en lugar de un solo conjunto de cambios.

Git LFS
git rebase
Un cambio de base con git rebase permite mover las ramas, lo que ayuda a evitar confirmaciones de fusión innecesarias. El historial lineal resultante suele ser mucho más fácil de entender y explorar.

git rebase -i
La marca -i se usa para iniciar una sesión de cambio de base interactivo. Esto ofrece todas las ventajas de un cambio de base normal, pero te da la oportunidad de añadir, editar o eliminar confirmaciones sobre la marcha.

git reflog
Git realiza el seguimiento de las actualizaciones en el extremo de las ramas mediante un mecanismo llamado registro de referencia o reflog. Esto permite volver a los conjuntos de cambios aunque no se haga referencia a ellos en ninguna rama o etiqueta.

git remote
Es un comando útil para administrar conexiones remotas. En lugar de pasar la URL completa a los comandos fetch, pull y push, permite usar un atajo más significativo.

git reset
Deshace los cambios en los archivos del directorio de trabajo. El restablecimiento permite limpiar o eliminar por completo los cambios que no se han enviado a un repositorio público.

git revert
Permite deshacer una instantánea confirmada. Si descubres una confirmación errónea, revertirla es una forma fácil y segura de eliminarla por completo del código base.

git status
Muestra el estado del directorio en el que estás trabajando y la instantánea preparada. Lo mejor es que lo ejecutes junto con git add y git commit para ver exactamente qué se va a incluir en la próxima instantánea.