**Git**

Git es un sistema de control de versiones. El control de versiones, también conocido como "control de código fuente", es la práctica de rastrear y gestionar los cambios en el código de software.

Realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir hacia atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error, al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

Permite:

* **Historial de cambios** a largo plazo de todos los archivos. Esto quiere decir todos los cambios realizados por muchas personas a lo largo de los años.
* **Creación de ramas y fusiones**. Si se tiene a miembros del equipo trabajando al mismo tiempo, es algo evidente; pero incluso las personas que trabajan solas pueden beneficiarse de la capacidad de trabajar en flujos independientes de cambios.
* **Trazabilidad**. Ser capaz de trazar cada cambio que se hace en el software y conectarlo con un software de gestión de proyectos y seguimiento de errores

**Funcionamiento**

Inicialización del repositorio central

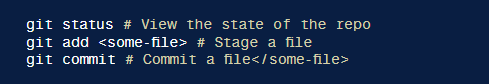
En primer lugar, hace falta crear el repositorio central en un servidor. Si se trata de un proyecto nuevo, puedes inicializar un repositorio vacío. Si no es así, tendrás que importar un repositorio Git existente.

Clonación de un repositorio

cada desarrollador crea una copia local de todo el proyecto. Esto se consigue mediante el comando git clone. Cuando clonas un repositorio, Git añade automáticamente un acceso rápido llamado origin que apunta al repositorio "principal", ya que da por sentado que querrás interactuar con él más adelante.

Aplicación de cambios y confirmaciones

Cuando el repositorio se clona de forma local, el desarrollador puede aplicar cambios mediante el proceso de confirmación estándar de Git: editar, preparar y confirmar.



Envío de nuevas confirmaciones al repositorio central

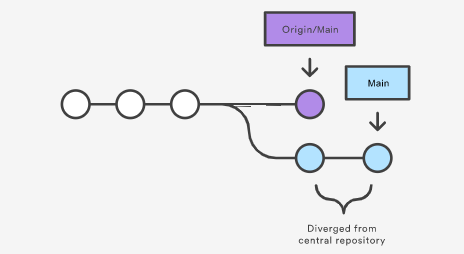
Una vez que se hayan aplicado cambios nuevos al repositorio local, se deberán enviar estos cambios para compartirlos con otros desarrolladores del proyecto.

https://lh4.googleusercontent.com/c4eqchjfQA4-UWlWdK_k9ztjDv5FJFGemQs4btg9Tq1DcKgfcCxwXICc9nMxvP2cFKbGGwG7pIXwtSsI1oBK0haLgy4uHZo25sG-Mtl3SfE2tHjBUXwJZlJv112q1j27RBs2M31_w4uSPlzFudIxJ9t7pyqSNayh0-YF16pi003tPyG8v4m_1oKdiA

Este comando enviará los nuevos cambios confirmados al repositorio central. Cuando envías cambios al repositorio central, es posible que las actualizaciones enviadas anteriormente por otros desarrolladores contengan código que entra en conflicto con las actualizaciones que tú quieres enviar. Git mostrará un mensaje en el que se indica este problema. En esta situación, tendrá que ejecutarse el comando git pull en primer lugar. Esta situación de conflicto se explica en profundidad en la sección siguiente.

Gestión de conflictos

El repositorio central representa el proyecto oficial, por lo que su historial de confirmación se debe tratar como algo sagrado e inmutable. Si las confirmaciones locales de un desarrollador difieren de las del repositorio central, Git rechazará el envío de cambios porque estos sobrescribirían las confirmaciones oficiales.



Si los cambios locales entran en conflicto directo con las confirmaciones en niveles superiores, Git pondrá en pausa el proceso de fusión mediante cambio de base y te dará la oportunidad de resolver los conflictos de forma manual. Lo bueno de Git es que usa los mismos comandos, git status y git add, tanto para generar confirmaciones como para resolver conflictos de fusión. Esto facilita a los nuevos desarrolladores la gestión de sus propias fusiones. Además, si tienen problemas, Git hace que sea muy fácil anular por completo la fusión mediante cambio de base y volver a intentarlo (o ir a pedir ayuda).

Sincronización de Git

* git remote: te permite crear, ver y eliminar conexiones con otros repositorios.
* git fetch: descarga commits, archivos y referencias de un repositorio remoto a tu repositorio local. Esta acción la llevas a cabo cuando quieres ver en qué han estado trabajando los demás.
* git push se usa para cargar contenido del repositorio local a un repositorio remoto.Es el equivalente a git fetch, pero mientras que al recuperar se importan las confirmaciones a ramas locales, al enviar estas se exportan a ramas remotas.
* git pull se emplea para extraer y descargar contenido desde un repositorio remoto y actualizar al instante el repositorio local para reflejar ese contenido.

Uso de ramas

El comando git branch te permite crear, enumerar y eliminar ramas, así como cambiar su nombre. No te permite cambiar entre ramas o volver a unir un historial bifurcado. Por este motivo, git branch está estrechamente integrado con los comandos git checkout y git merge.

* git branch:Enumera todas las ramas de tu repositorio. Es similar a git branch --list.
* git branch <branch>:Crea una nueva rama llamada ＜branch＞. Este comando no extrae la nueva rama.
* git branch -d <branch>:Elimina la rama especificada. Esta es una operación segura, ya que Git evita que elimines la rama si tiene cambios que aún no se han fusionado.
* git branch -D <branch>:Fuerza la eliminación de la rama especificada, incluso si tiene cambios sin fusionar.
* git branch -m <branch>:Cambia el nombre de la rama actual a ＜branch＞.

git branch -a: Enumera todas las ramas remotas.