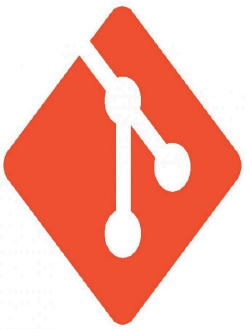
Cristian Alessandro Popoca Trujillo. 26/04/2024

INTRODUCCIÓN A GIT :



Git es un sistema de control de versiones (VCS, por sus siglas en inglés) es una herramienta que registra los cambios realizados en un conjunto de archivos a lo largo del tiempo. Estos sistemas son comúnmente utilizados en el desarrollo de software, aunque también pueden ser aplicados en otros tipos de proyectos que involucren la edición y modificación de archivos.

- Registra cada cambio en el proyecto o repositorio, quién y cuándo lo hace, en una base de datos.
- Permite volver a estados previos del desarrollo.
- Permite gestionar diferentes versiones del proyecto (ramas) para trabajar en paralelo y luego fundirlas.
- Permite colaborar entre diferentes usuarios en un mismo repositorio, facilitando la resolución de conflictos.
- Se utiliza principalmente en proyectos de desarrollo de software, pero sirve para cualquier otro tipo de proyecto.

Configuración de Git :

- Establecer el nombre de usuario: git config --global user.name "Your-Full-Name"
- Establecer el correo del usuario : git config --global user.email "your-email-address"
- Activar el coloreado de la salida : git config --global color.ui auto
- Mostrar la configuración : git config --list

Repositorios :

Un repositorio en Git es básicamente un espacio donde se almacenan todos los archivos y carpetas de tu proyecto, junto con un historial de cambios que registra todas las modificaciones realizadas en esos archivos a lo largo del tiempo. Esto te permite realizar un seguimiento de cada modificación, quién la hizo y cuándo se realizó.

- ejecuta el siguiente comando para inicializar un repositorio Git en ese directorio : git init
- Una vez que tienes un repositorio Git creado, puedes agregar archivos a él. Puedes hacerlo manualmente copiando los archivos al directorio del repositorio, o puedes utilizar el comando git add para agregar

archivos específicos al área de preparación (staging area) : git add archivo.txt archivo2.txt

- Después de agregar los archivos que deseas incluir en el repositorio, debes confirmar estos cambios utilizando el comando git commit. Esto guarda permanentemente los cambios en el repositorio con un mensaje descriptivo que explica qué cambios se realizaron. Por ejemplo: git commit -m “Mensaje descriptivo de los cambios”

Con estos pasos básicos, has creado un repositorio Git en tu proyecto y has guardado los archivos iniciales en él.

git show :

Muestra el usuario, el día, la hora y el mensaje del último commit, así como las diferencias con el anterior.
El git show <commit> muestra el usuario, el día, la hora y el mensaje del commit indicado, así como las diferencias con el anterior.

Copia de Repositorios(git clone):

El git clone crea una copia local del repositorio ubicado en la dirección .

A partir de que se hace la copia, los dos repositorios, el original y la copia, son independientes, es decir, cualquier cambio en uno de ellos no se verá reflejado en el otro.

git clone + url del repositorio que deseas clonar.
git clone https://github.com/usuario/repositorio.git

Mostrar el estado de un repositorio(git status):

muestra el estado de los cambios en el repositorio desde la última versión guardada. En particular, muestra los ficheros con cambios en el directorio de trabajo que no se han añadido a la zona de intercambio temporal que no se han añadido al repositorio.

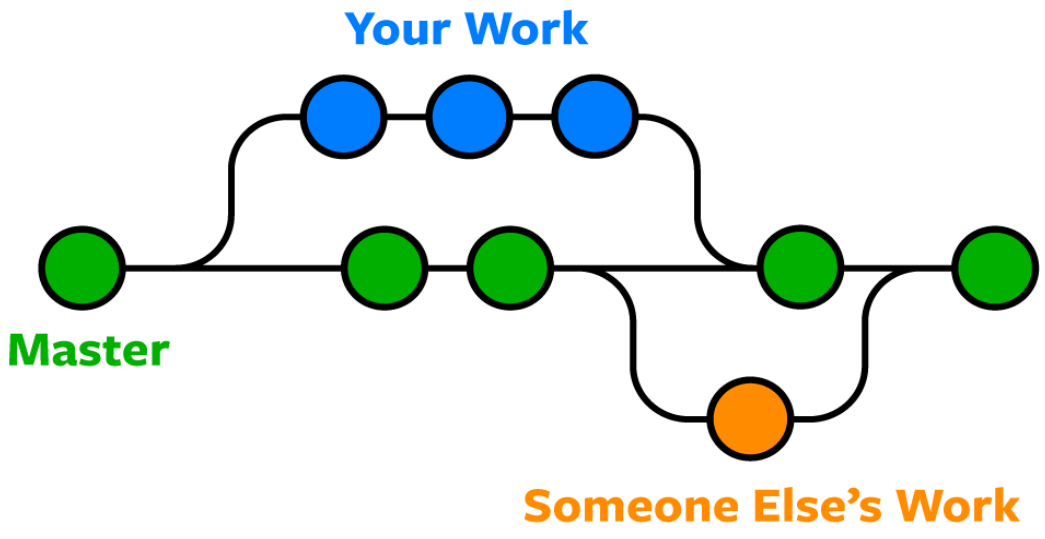
Ramas :

Las ramas en Git son líneas de desarrollo independientes que permiten trabajar en diferentes versiones de un proyecto de forma simultánea. Cada rama representa una línea de desarrollo separada que puede contener cambios únicos en el código sin afectar directamente otras ramas. Las ramas en Git son una de las características más poderosas y flexibles de la herramienta, y permiten una gestión eficiente del desarrollo colaborativo y del control de versiones.

características y conceptos clave sobre las ramas en Git:

1. Divergencia: Cuando creas una nueva rama a partir de una rama existente, ambas ramas comparten el mismo historial de commits hasta el momento de la bifurcación. A partir de ese punto, los commits en una rama pueden divergir de los de la otra, lo que permite el desarrollo independiente de características o experimentos.
2. Paralelismo: Puedes tener múltiples ramas activas al mismo tiempo. Cada rama puede representar una función nueva, una corrección de errores, una versión experimental, entre otras posibilidades.
3. Merge (Fusión): Después de trabajar en una rama y completar tus cambios, puedes fusionar esos cambios de vuelta a otra rama (como la rama principal) utilizando el comando git merge. Esto combina los cambios realizados en la rama actual con la rama especificada.
4. Conflictos: A veces, al fusionar dos ramas, Git puede encontrar conflictos, que son situaciones en las que el mismo archivo ha sido modificado de diferentes maneras en cada rama. En estos casos, es necesario resolver manualmente los conflictos antes de completar la fusión.
5. Creación y eliminación: Puedes crear una nueva rama en cualquier momento utilizando el comando git branch nombre_de_la_rama. Para eliminar una rama, puedes utilizar git branch -d nombre_de_la_rama.

La Siguiente Imagen Muestra cómo funcionan las Ramas(Branches), cada rama en en este caso representadas en verde, azul y naranja muestran una línea de trabajo diferente, la rama main o rama master representa la rama a partir de la cual parten todas las ramas que se pueden crear, la línea azul representa una nueva rama en la cual un usuario agrega tres nuevas funcionalidades y dicha rama se vuelve a incorporar en la rama master, así mismo se puede apreciar como otro usuario de manera simultánea agrega una nueva funcionalidad desde otra rama.



Cristian Alessandro Popoca Trujillo. 26/04/2024

Pull Request :

- Función: Es una solicitud que un colaborador hace a los dueños del repositorio para que revisen y fusionen sus cambios en el proyecto principal.
- Uso: Se crea generalmente en plataformas de alojamiento de código como GitHub, GitLab o Bitbucket. Después de hacer cambios en tu fork del repositorio, puedes enviar un pull request desde tu fork hacia el repositorio original para que los dueños del proyecto revisen tus cambios.

Fork :

- Función: Hace una copia independiente de un repositorio en tu propia cuenta de usuario. Esto permite contribuir a un proyecto sin tener permisos de escritura en el repositorio original.
- Uso: En plataformas como GitHub, puedes hacer clic en el botón "Fork" en el repositorio que deseas contribuir. Esto creará una copia del repositorio en tu cuenta que puedes modificar y luego enviar pull requests para proponer cambios al proyecto original.

Rebase :

- Función: Reescribe el historial de commits de una rama, moviendo los cambios de una rama a otra.
- Uso: Se utiliza para mantener un historial de commits limpio y lineal, especialmente al colaborar en proyectos en equipo. Permite aplicar los cambios de una rama sobre otra y resolver conflictos de fusión de forma individual para cada commit.

Clean :

- Función: Elimina archivos no rastreados y/o ignorados del directorio de trabajo.
- Uso: git clean -m muestra los archivos que serían eliminados en seco (sin realmente eliminarlos), y git clean -f realiza la eliminación real.

cherry pick :

- Función: Permite seleccionar y aplicar un único commit de una rama a otra, sin necesidad de fusionar ramas completas.
- Uso: git cherry-pick <commit> aplicará los cambios del commit especificado a la rama actual.

Stash :

- Función: Guarda temporalmente los cambios en un área de almacenamiento temporal, dejando el directorio de trabajo limpio.
- Uso: Útil cuando necesitas cambiar de branch o realizar otra tarea sin hacer commit de tus cambios actuales. Puedes usar git stash save para guardar tus cambios, git stash list para ver la lista de stashes y git stash apply para aplicar los cambios guardados.

- Trabajo en branches: Una vez que estás en un branch, puedes hacer cambios en los archivos sin afectar otras ramas. Esto te permite trabajar de forma aislada en una función o característica específica.

Conflicts:

- ¿Qué son?: Los conflictos ocurren cuando Git no puede automáticamente fusionar los cambios de diferentes ramas. Esto sucede cuando dos ramas editan la misma parte de un archivo de formas que no se pueden combinar automáticamente.
- Resolución de conflictos: Cuando ocurre un conflicto, Git te notificará y marcará los archivos con conflictos. Debes abrir estos archivos en un editor de texto y resolver manualmente los conflictos. Una vez resueltos, debes agregar los archivos modificados y realizar un commit para completar la fusión.

Merge(Fusión):

- ¿Qué es?: La fusión (merge) es el proceso de combinar los cambios de dos ramas diferentes en una sola. Se utiliza comúnmente para integrar los cambios de una rama secundaria en la rama principal (por ejemplo, master o main).
- Ejecución de merge: Para fusionar cambios de un branch en otro, utiliza el comando git merge nombre_del_branch. Esto combinará los cambios del branch especificado en el branch actual.
- Fusiones automáticas y manuales: Si Git puede fusionar los cambios automáticamente sin conflictos, lo hará. Si hay conflictos, deberás resolverlos manualmente antes de completar la fusión.

Repositorios Remotos :

Los repositorios remotos son versiones de tu proyecto que se encuentran alojadas en un servidor en línea o en otro lugar accesible a través de Internet. Estos repositorios remotos actúan como puntos de conexión centralizados donde

múltiples colaboradores pueden contribuir y sincronizar su trabajo en un proyecto común. Algunos servicios populares que proporcionan alojamiento para repositorios remotos incluyen GitHub, GitLab y Bitbucket.

conceptos clave sobre los repositorios remotos en Git:

1. Colaboración: Los repositorios remotos facilitan la colaboración entre múltiples desarrolladores en un proyecto. Cada desarrollador puede clonar el repositorio remoto en su máquina local, realizar cambios y enviar esos cambios de vuelta al repositorio remoto para que otros los vean y revisen.
2. Copia exacta: Cuando clonas un repositorio remoto en tu máquina local, obtienes una copia exacta del historial de cambios y todos los archivos del proyecto. Esto te permite trabajar de forma independiente en tu propia versión del proyecto.
3. Sincronización: Puedes sincronizar tu trabajo local con el repositorio remoto utilizando comandos como git push para enviar tus cambios al repositorio remoto y git pull para obtener los cambios más recientes desde el repositorio remoto a tu máquina local.
4. Gestión de versiones: Los repositorios remotos actúan como una copia de seguridad centralizada de tu proyecto y su historial de cambios. Esto es útil en caso de pérdida de datos en tu máquina local o si necesitas acceder al proyecto desde múltiples ubicaciones.
5. Seguridad: Los servicios de alojamiento de repositorios remotos suelen proporcionar funciones de seguridad avanzadas, como control de acceso basado en roles, autenticación de dos factores y auditorías de actividad, para proteger el código y los datos del proyecto.

¿Qué es Github?

GitHub es una plataforma de desarrollo colaborativo basada en la nube, que utiliza el sistema de control de versiones Git. Fue lanzada en 2008 y se ha convertido en una de las plataformas más populares para alojar proyectos de software, colaborar en código, revisar y discutir cambios, y gestionar proyectos de desarrollo de software de forma remota.

Cristian Alessandro Popoca Trujillo. 26/04/2024

git push:

- Función: Este comando se utiliza para enviar (push) los cambios locales realizados en tu repositorio hacia un repositorio remoto.
- Uso: La sintaxis general es git push <nombre remoto> <nombre_rama_local>. Por ejemplo, git push origin main enviaría los cambios de la rama `main` al repositorio remoto llamado origin.
-

git remote:

- Función: Este comando permite ver los repositorios remotos configurados en tu repositorio local.
- Uso: Puedes usar git remote -v para ver los nombres y las URL de los repositorios remotos configurados. También puedes usar otros subcomandos, como git remote add para agregar un nuevo repositorio remoto o git remote rm para eliminar un repositorio remoto.
-

git remote add :

- Función: Se utiliza para agregar un nuevo repositorio remoto a tu repositorio local.
- Uso: La sintaxis es git remote add <nombre_remoto> <URL del repositorio>. Por ejemplo, git remote add origin https://github.com/usuario/repositorio.git agregaría un nuevo repositorio remoto con el nombre origin.