Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Estudios de Postgrado
Maestría en Ingeniería para la Industria con
Especialización en Ciencias de la Computación
Curso: Fundamentos de Programación y Scripting
Estudiante: German Omar Chiguichón Sibrian

Investigación de Git

¿Qué es Git?

Git es un sistema de control de versiones distribuido y de código abierto que se utiliza para el seguimiento de cambios en archivos y proyectos de software, uno de sus propósitos es llevar registro de las modificaciones de archivos y coordinar el trabajo que las personas realizan sobre archivos compartidos en repositorios de código. Originalmente fue creado para el desarrollo del kernel de Linux, pero desde entonces se ha convertido en una herramienta popular y ampliamente utilizada por desarrolladores de todo el mundo.

Con Git, los desarrolladores pueden mantener un registro de todos los cambios realizados en un proyecto de software y colaborar con otros desarrolladores de manera eficiente. Asimismo, permite que varias personas trabajen en diferentes ramas de un proyecto de manera simultánea y fusionar las diferentes versiones para crear una versión final en un mismo proyecto de forma simultánea, resolviendo los conflictos que puedan surgir al combinar sus cambios.

Git también ofrece una amplia gama de herramientas y características para facilitar el desarrollo de software, como el control de acceso, las etiquetas, las bifurcaciones, la reversión de cambios y la gestión de conflictos. Hoy en día es una herramienta muy popular y ampliamente utilizada en la industria de la programación.

Control de versiones con GIT

El control de versiones con Git permite a los desarrolladores administrar el historial de cambios en su código de manera eficiente. Cada vez que se realiza una modificación en un archivo, Git registra los cambios realizados y almacena esa información en su base de datos. Esto permite que los desarrolladores puedan realizar un seguimiento de los cambios realizados en el código a lo largo del tiempo y puedan volver a versiones anteriores si es necesario.

Para comenzar a utilizar Git, se debe crear un repositorio de Git en el directorio que contiene los archivos del proyecto. Los desarrolladores pueden agregar, eliminar y modificar archivos en el repositorio, y luego realizar confirmaciones (commits) para registrar los cambios en la base de datos de Git. Cada confirmación incluye un mensaje que describe los cambios realizados en esa versión del archivo.

Git también permite a los desarrolladores trabajar en diferentes ramas (branches) del proyecto de manera simultánea, lo que facilita la colaboración en equipo y el desarrollo de nuevas funcionalidades sin afectar la versión principal del código. Cuando se completa una nueva funcionalidad o una corrección de errores en una rama, se puede fusionar (merge) esa rama en la versión principal del código.

Otras características importantes de Git incluyen la capacidad de etiquetar (tag) versiones específicas del código, revertir cambios no deseados, resolver conflictos de fusión y colaborar con otros desarrolladores utilizando servicios como GitHub o GitLab. En general, Git es una herramienta esencial para el control de versiones en proyectos de programación y es ampliamente utilizada por la comunidad de desarrolladores en todo el mundo.

Estados de un archivo en GIT

En Git, los archivos pueden estar en tres estados diferentes:

- Untracked (No rastreado): Los archivos que están en el directorio de trabajo pero que no han sido añadidos al repositorio de Git se consideran no rastreados. Esto significa que Git no está haciendo un seguimiento de los cambios realizados en estos archivos.
- Staged (Preparado): Cuando se añade un archivo al área de preparación (staging area) utilizando el comando git add, el archivo pasa al estado preparado. Esto significa que Git está haciendo un seguimiento de los cambios realizados en el archivo y se prepara para hacer una confirmación (commit).
- Committed (Confirmado): Cuando se realiza una confirmación utilizando el comando git commit, los archivos en el área de preparación se convierten en confirmados. Esto significa que Git ha registrado los cambios realizados en el archivo en la base de datos del repositorio y que se puede volver a esta versión del archivo en cualquier momento.

Es importante tener en cuenta que los archivos pueden estar en diferentes estados en diferentes momentos. Por ejemplo, un archivo puede estar no rastreado al principio, pero después de agregarlo al área de preparación, se convierte en preparado. Luego, después de la confirmación, se convierte en confirmado. Además, los archivos pueden tener diferentes estados en diferentes ramas del repositorio, lo que puede causar confusiones en el control de versiones si no se manejan adecuadamente.

Como configurar un repositorio en GIT

Para configurar un repositorio en Git, se deben seguir los siguientes pasos:

- Crear una cuenta en GitHub (opcional): Si desea alojar su repositorio de Git en GitHub, debe crear una cuenta en la plataforma.
- Inicializar un repositorio local de Git: Abra una terminal o consola de comandos en la ubicación donde desea crear su repositorio local de Git y ejecute el siguiente comando: git init. Este comando inicializa un repositorio local de Git vacío.
- Agregar archivos al repositorio: Agregue los archivos que desea incluir en el repositorio utilizando el comando git add. Por ejemplo, si desea agregar todos los archivos en el directorio actual, ejecute el comando git add.
- Realizar la confirmación inicial: Después de agregar los archivos, realice una confirmación inicial utilizando el comando git commit. Este comando confirma los cambios realizados en los archivos agregados y los almacena en el repositorio local de Git. Por ejemplo, puede ejecutar el comando git commit -m "Commit inicial" para realizar la confirmación inicial con un mensaje de confirmación que describe los cambios realizados.
- Configurar un repositorio remoto (opcional): Si desea alojar su repositorio de Git en GitHub u otra plataforma de alojamiento de Git, debe configurar un repositorio remoto. Para hacerlo, ejecute el comando git remote add <nombre-remoto> <url-remoto>, donde <nombre-remoto> es un nombre fácil de recordar para el repositorio remoto (como "origin") y <url-remoto> es la URL del repositorio remoto en la plataforma de alojamiento de Git.

- Empujar el repositorio local a un repositorio remoto (opcional): Si ha configurado un repositorio remoto, puede empujar su repositorio local a ese repositorio remoto utilizando el comando git push. Por ejemplo, puede ejecutar el comando git push origin master para empujar el repositorio local a un repositorio remoto llamado "origin" en la rama principal "master".

Con estos pasos, ha configurado correctamente un repositorio de Git. Puede seguir agregando archivos y realizando confirmaciones para controlar las versiones de su proyecto y, si lo desea, empujar los cambios a un repositorio remoto para colaborar con otros desarrolladores.

Comandos en GIT

A continuación, se presentan algunos de los comandos más utilizados en Git:

- git init: Inicializa un repositorio local de Git vacío.
- git add: Agrega archivos al área de preparación (staging) para la próxima confirmación. Por ejemplo, git add archivo.txt agrega el archivo "archivo.txt" al área de preparación.
- git commit: Confirma los cambios realizados en los archivos agregados al área de preparación y los almacena en el repositorio local de Git. Por ejemplo, git commit -m "Mensaje de confirmación" realiza una confirmación con un mensaje de confirmación que describe los cambios realizados.
- git status: Muestra el estado actual del repositorio local de Git. Por ejemplo, git status muestra los archivos que se han modificado, agregado o eliminado desde la última confirmación.
- git log: Muestra el historial de confirmaciones realizadas en el repositorio local de Git. Por ejemplo, git log muestra una lista de todas las confirmaciones realizadas, junto con los mensajes de confirmación y los detalles de autoría.
- git diff: Muestra las diferencias entre dos versiones de un archivo. Por ejemplo, git diff archivo.txt muestra las diferencias entre la versión actual del archivo "archivo.txt" y la última confirmación realizada.
- git branch: Muestra la lista de ramas del repositorio local de Git y muestra en cuál está actualmente trabajando. Por ejemplo, git branch muestra una lista de todas las ramas y resalta la rama actual con un asterisco.
- git checkout: Cambia a una rama diferente o a una versión anterior de un archivo. Por ejemplo, git checkout otra-rama cambia a la rama "otra-rama", mientras que git checkout HEAD~1 archivo.txt cambia a la versión anterior del archivo "archivo.txt".
- git merge: Combina dos ramas diferentes del repositorio local de Git. Por ejemplo, git merge otra-rama combina la rama actual con la rama "otra-rama".
- git pull: Descarga los cambios desde un repositorio remoto y los fusiona con el repositorio local de Git. Por ejemplo, git pull origin master descarga los cambios desde el repositorio remoto "origin" en la rama principal "master" y los fusiona con el repositorio local de Git.
- git push: Empuja los cambios realizados en el repositorio local de Git a un repositorio remoto. Por ejemplo, git push origin master empuja los cambios realizados en el repositorio local a un repositorio remoto llamado "origin" en la rama principal "master".

Estos son solo algunos de los comandos más comunes en Git. Hay muchos más comandos disponibles, pero estos son los que se utilizan con mayor frecuencia en el flujo de trabajo diario de un desarrollador.