Comandos Principales de GitHub para la Gestión de Repositorios desde la Línea de Comando

Frida Cano Falcón

Academia Java MTY
Agosto 2024

Viernes 16 de agosto de 2024

Introducción

Git es una herramienta esencial en el desarrollo de software para el control de versiones, permitiendo gestionar proyectos de manera eficiente mediante la línea de comando. Este reporte detalla los comandos más relevantes para la configuración inicial, manejo de repositorios, gestión de branches y rebases, y otras operaciones comunes en Git.

Configuración Básica

La configuración básica de Git es el primer paso necesario para comenzar a utilizar esta herramienta de control de versiones. Permite establecer los valores predeterminados, como el nombre y correo electrónico asociados a los commits, así como la configuración de colores en la terminal para una mejor visualización.

Comando	**Descripción y Ejemplo**
Configurar Nombre en los Commits	Establece el nombre que aparecerá en los commits.
`git config --global user.name "dasdo"`	
Configurar Email	Define el correo electrónico asociado a los commits.
`git config --global user.email dasdo1@gmail.com`	
Marco de Colores para los Comandos	Habilita el uso de colores en la terminal para facilitar la lectura de los comandos.
`git config --global color.ui true`	

Iniciando Repositorio

Un repositorio en Git es donde se almacenan todos los cambios realizados en un proyecto. Estos comandos son esenciales para iniciar un nuevo repositorio en una carpeta existente o clonar un repositorio desde un servicio como GitHub o Bitbucket.

Comando	**Descripción y Ejemplo**
Inicializar un Repositorio Git	Inicia un repositorio Git en la carpeta del proyecto actual.
`git init`	
Clonar un Repositorio desde GitHub o Bitbucket	Clona un repositorio existente utilizando la URL proporcionada.
`git clone <url>`	
Añadir Archivos para el Commit	Añade todos los archivos del proyecto al área de staging, preparándolos para ser commiteados.
`git add .`	
Realizar un Commit	Crea un commit con los cambios añadidos, incluyendo un mensaje que describa la razón del commit.
`git commit -m "Texto que identifique por qué se hizo el commit"`	
Subir Cambios al Repositorio Remoto	Sube los cambios realizados al repositorio remoto en la rama principal (master).
`git push origin master`	

GIT CLONE

El comando `git clone` se utiliza para crear una copia local de un repositorio remoto. Es especialmente útil cuando deseas trabajar en un proyecto existente o colaborar con otros desarrolladores.

Comando	**Descripción y Ejemplo**
Clonar un Repositorio desde GitHub o Bitbucket	Clona un repositorio remoto utilizando la URL.
`git clone <url>`	
Clonar y Renombrar un Repositorio	Clona un repositorio remoto y le asigna un nombre local diferente.
`git clone <url> git-demo`	

GIT ADD

El comando `git add` es esencial para preparar los archivos que se incluirán en el próximo commit. Puede utilizarse para añadir archivos específicos, todos los archivos, o archivos que cumplan ciertos criterios.

Comando	**Descripción y Ejemplo**
Añadir Todos los Archivos para el Commit	Añade todos los archivos del proyecto al área de staging.
`git add .`	
Añadir Archivo Específico	Añade un archivo específico al área de staging.
`git add <archivo>`	
Añadir Todos los Archivos Omitiendo los Nuevos	Añade todos los archivos modificados al área de staging, omitiendo los nuevos.
`git add --all`	
Añadir Archivos con una Extensión Específica	Añade todos los archivos con una extensión específica al área de staging.

Comando	Descripción y Ejemplo
`git add *.txt`	
Añadir Archivos dentro de un Directorio Específico	Añade todos los archivos dentro de un directorio específico y que tengan una extensión específica.
`git add docs/*.txt`	
Añadir Archivos dentro de un Directorio	Añade todos los archivos dentro de un directorio.
`git add docs/`	

GIT COMMIT

El comando `git commit` guarda los cambios que se encuentran en el área de staging en el historial de Git. Los commits son puntos en el tiempo que registran la evolución de un proyecto.

Comando	Descripción y Ejemplo
Realizar un Commit con Mensaje	Guarda los cambios en el HEAD con un mensaje que describa la razón del commit.
`git commit -m "Texto que identifique por qué se hizo el commit"`	
Añadir y Realizar un Commit	Añade todos los cambios y realiza el commit en un solo paso.
`git commit -a -m "Texto que identifique por qué se hizo el commit"`	
Mostrar Conflictos en el Commit	Realiza el commit mostrando los conflictos si los hay.
`git commit -a`	

Comando	Descripción y Ejemplo
Modificar el Último Commit	Modifica el último commit sin que aparezca como un nuevo commit en los logs.
`git commit --amend -m "Texto que identifique por qué se hizo el commit"`	

GIT PUSH

El comando `git push` se utiliza para subir los commits realizados en la rama local a un repositorio remoto. Es fundamental para compartir el trabajo con otros desarrolladores o guardar el progreso en un servidor remoto.

Comando	Descripción y Ejemplo
Subir Cambios al Repositorio Remoto	Sube los cambios realizados al repositorio remoto en una rama específica.
`git push <origin> <branch>`	
Subir un Tag	Sube los tags locales al repositorio remoto.
`git push --tags`	

GIT LOG

El comando `git log` permite visualizar el historial de commits realizados en un repositorio. Es útil para rastrear cambios, revisar el historial del proyecto, y entender cómo ha evolucionado el código a lo largo del tiempo.

Comando	Descripción y Ejemplo
Mostrar el Historial de Commits	Muestra los logs de los commits realizados en el proyecto.
`git log`	
Mostrar Cambios en los Commits	Muestra los commits en una línea y los cambios realizados en cada uno.

Comando	Descripción y Ejemplo
`git log --oneline --stat`	
Mostrar Gráficos de los Commits	Muestra los commits en una línea con un gráfico que representa la estructura de ramas.
`git log --oneline --graph`	

GIT DIFF

El comando `git diff` muestra las diferencias entre archivos en diferentes estados. Es útil para ver qué cambios se han realizado en un archivo antes de hacer un commit.

Comando	Descripción y Ejemplo
Mostrar Diferencias en un Archivo	Muestra las diferencias entre el archivo en el área de trabajo y el área de staging.
`git diff`	
Mostrar Diferencias en Archivos Preparados para Commit	Muestra las diferencias en los archivos que están en el área de staging.
`git diff --staged`	

GIT HEAD

El comando `git reset HEAD` y otros relacionados se utilizan para deshacer cambios, ya sea en el área de staging o en los commits realizados. Estos comandos permiten corregir errores y ajustar el historial de commits según sea necesario.

Comando	Descripción y Ejemplo
Sacar un Archivo del Commit	Quita un archivo del área de staging, regresándolo al área de trabajo.
`git reset HEAD <archivo>`	

Comando	Descripción y Ejemplo
Devolver el Último Commit y Poner los Cambios en Staging	Deshace el último commit, pero mantiene los cambios en el área de staging para ser commiteados nuevamente si es necesario.
`git reset --soft HEAD^`	

GIT REMOTE

El comando `git remote` gestiona los repositorios remotos asociados a un proyecto. Permite añadir, cambiar, eliminar y mostrar información sobre los repositorios remotos.

Comando	Descripción y Ejemplo
Agregar Repositorio Remoto	Añade un nuevo repositorio remoto y lo asocia con un nombre local.
`git remote add origin <url>`	
Cambiar Repositorio Remoto	Cambia la URL del repositorio remoto asociado al nombre especificado.
`git remote set-url origin <url>`	
Remover Repositorio Remoto	Elimina el repositorio remoto asociado al nombre especificado.
`git remote rm <name/origin>`	
Mostrar Lista de Repositorios Remotos	Muestra una lista de todos los repositorios remotos asociados al proyecto.
`git remote -v`	
Mostrar Ramas Remotas	Muestra todas las ramas remotas del repositorio asociado.
`git remote show origin`	
Limpiar Ramas Eliminadas	Elimina todas las ramas remotas que ya no existen en el repositorio remoto.

Comando	Descripción y Ejemplo
`git remote prune origin`	

GIT BRANCH

El comando `git branch` se utiliza para gestionar las ramas en un proyecto Git. Las ramas permiten trabajar en diferentes características o versiones de un proyecto de manera aislada, sin afectar la rama principal.

Comando	Descripción y Ejemplo
Crear una Rama	Crea una nueva rama con el nombre especificado.
`git branch <nameBranch>`	
Listar Ramas	Muestra una lista de todas las ramas disponibles en el repositorio local.
`git branch`	
Eliminar una Rama	Elimina una rama específica y fusiona sus cambios en la rama principal (master).
`git branch -d <nameBranch>`	
Eliminar una Rama sin Confirmación	Fuerza la eliminación de una rama sin preguntar confirmación.
`git branch -D <nameBranch>`	

GIT TAG

El comando `git tag` se utiliza para etiquetar puntos específicos en el historial de commits, como versiones de un proyecto. Los tags son útiles para identificar versiones estables y lanzar releases.

Comando	Descripción y Ejemplo
Mostrar Lista de Tags	Muestra una lista de todos los tags creados en el repositorio local.

Comando	Descripción y Ejemplo
`git tag`	
Crear un Nuevo Tag	Crea un nuevo tag anotado con un mensaje descriptivo.
`git tag -a <version> -m "Esta es la versión X"`	

GIT REBASE

El comando `git rebase` se utiliza para reordenar commits, generalmente cuando se está trabajando con ramas. Permite poner una rama al día con la rama principal sin realizar un merge, lo que mantiene un historial más limpio.

Comando	Descripción y Ejemplo
Realizar un Rebase en la Rama Actual	Une la rama actual con la rama principal sin realizar un merge.
`git rebase`	
Continuar Secuencia de Rebase tras Resolver Conflictos	Continúa con el rebase después de haber resuelto conflictos.
`git rebase --continue`	
Omitir Conflicto y Continuar	Omite el conflicto y sigue con el proceso de rebase.
`git rebase --skip`	
Abortar Rebase	Devuelve todo al estado anterior al rebase.
`git rebase --abort`	
Realizar un Rebase a una Rama Específica	Realiza un rebase de la rama actual con otra rama especificada.
`git rebase <nameBranch>`	

Otros Comandos

Además de los comandos principales, Git ofrece una serie de comandos adicionales que facilitan la gestión de archivos, ramas, y la sincronización con repositorios remotos.

Comando	**Descripción y Ejemplo**
Ver Estado Actual del Repositorio	Muestra el estado actual del repositorio, listando archivos modificados, añadidos, o eliminados.
`git status`	
Quitar Archivo de HEAD y Ponerlo en Estado No Trabajado	Restaura un archivo a su estado original, eliminando cualquier cambio no commiteado.
`git checkout -- <file>`	
Crear Rama a partir de una Rama Remota	Crea una nueva rama local a partir de una rama remota especificada.
`git checkout -b newlocalbranchname origin/branch-name`	
Actualizar Repositorio con Cambios Nuevos	Busca cambios en el repositorio remoto y los incorpora en la rama local.
`git pull origin <nameBranch>`	
Cambiar de Rama	Cambia de la rama actual a otra rama especificada.
`git checkout <nameBranch/tagname>`	
Unir la Rama Actual con Otra Específica	Realiza un merge de la rama actual con otra rama especificada.
`git merge <nameBranch>`	
Verificar Cambios en el Repositorio Online	Busca cambios en el repositorio remoto sin incorporarlos a la rama local.
`git fetch`	

Comando	Descripción y Ejemplo
Eliminar un Archivo del Repositorio	Elimina un archivo del repositorio, incluyéndolo en el próximo commit como una eliminación.
`git rm <archivo>`	
Descargar Remote de un Fork	Añade un repositorio remoto de un fork como `upstream` para sincronizar con el repositorio original.
`git remote add upstream <url>`	
Merge con el Master de un Fork	Realiza un merge del master de un fork en la rama principal del repositorio local.
`git fetch upstream`	
`git merge upstream/master`	

Conclusión

Estos comandos cubren una amplia gama de operaciones que se pueden realizar en un repositorio Git, desde la configuración básica hasta la gestión avanzada de ramas y merges. Dominarlos es fundamental para cualquier desarrollador que desee trabajar de manera eficiente y efectiva en proyectos de software.

Referencias

dasdo. (n.d.). *Git cheatsheet*. GitHub. https://gist.github.com/dasdo/9ff71c5c0efa037441b6