GA7-220501096-AA1-EV03 IDENTIFICA HERRAMIENTAS DE VERSIONAMIENTO

PRESENTADO POR
UBEIMAR BEDOYA ARANGO

PROFESOR
CRISTIAN ARIAS

SENA
REGIONAL CALDAS
ANÁLISIS Y DESARROLLO DE SOFTWARE 2721447
MEDELLÍN
2024

INTRODUCCIÓN

Git es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Fue creado por Linus Torvalds en 2005 y ha ganado popularidad debido a su eficiencia y flexibilidad en el manejo de proyectos de software. El propósito principal de Git es rastrear los cambios en el código fuente durante el desarrollo de un proyecto, permitiendo a los desarrolladores trabajar de manera colaborativa y mantener un historial completo de todas las modificaciones realizadas.

Algunos conceptos clave en Git incluyen:

- **Repositorio**: Es un lugar donde se almacenan los archivos y carpetas de un proyecto, junto con la información de seguimiento de cambios.
- **Commit**: Un commit (confirmación) representa un conjunto de cambios realizados en los archivos del proyecto en un momento específico. Cada commit tiene un mensaje descriptivo que proporciona información sobre los cambios realizados.
- **Branch (Rama)**: Git permite la creación de ramas independientes que permiten a los desarrolladores trabajar en características o correcciones de errores sin afectar directamente la rama principal del proyecto. Posteriormente, estas ramas pueden fusionarse de nuevo en la rama principal.
- **Clone (Clonar)**: Clonar un repositorio Git significa copiar todo el historial de versiones y archivos de un repositorio remoto a una máquina local.
- **Push (Empujar) y Pull (Tirar)**: Push se refiere a enviar los cambios locales a un repositorio remoto, mientras que pull implica obtener los cambios del repositorio remoto y fusionarlos con la versión local.
- **Tag (Etiqueta)**: Git permite etiquetar puntos específicos en la historia del proyecto, como versiones estables o hitos importantes.

Objetivos de git

1. **Rastreo de Cambios Eficiente:** Git tiene como objetivo proporcionar un sistema eficiente para rastrear los cambios en el código fuente a lo largo del tiempo. Utiliza un modelo de datos basado en instantáneas (snapshots) en lugar de un modelo basado en diferencias, lo que permite una gestión de versiones rápida y eficiente.
2. **Descentralización:** Git es un sistema de control de versiones distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio, incluido el historial completo. Esto facilita el trabajo en entornos descentralizados y permite a los desarrolladores trabajar sin conexión a internet.

3. **Colaboración Efectiva:** Git facilita la colaboración entre desarrolladores al permitir ramificaciones (branches) independientes y fusiones (merges) sencillas. Cada desarrollador puede trabajar en su propia rama sin interferir con el trabajo de otros, y luego fusionar los cambios de manera controlada.
4. **Integridad de Datos:** Git utiliza hash criptográficos para identificar de manera única cada cambio en el repositorio. Esto asegura la integridad de los datos y permite detectar cualquier alteración no autorizada.
5. **Velocidad y Rendimiento:** Git se diseñó para ser rápido y eficiente en operaciones clave como la creación de ramas, confirmaciones (commits) y fusiones (merges). La estructura interna de Git y su enfoque en instantáneas contribuyen a su rendimiento.
6. **Flexibilidad:** Git es muy flexible y se puede adaptar a una variedad de flujos de trabajo de desarrollo. Permite a los equipos adoptar prácticas que se ajusten a sus necesidades específicas.
7. **Compatibilidad y Portabilidad:** Git es compatible con múltiples plataformas y sistemas operativos. Además, es posible utilizar Git en conjunto con servicios de alojamiento en la nube como GitHub, GitLab o Bitbucket.
8. **Historial Completo:** Git mantiene un historial completo de cada cambio realizado en el repositorio, lo que permite a los desarrolladores retroceder en el tiempo y revisar cualquier versión anterior del código.

Tablas comparativas entre git local y git remoto

CARACTERÍSTICAS Y COMANDOS GIT LOCAL	
COMANDO	**DESCRIPCIÓN**
'git init'	Inicia un nuevo repositorio Git en el directorio actual
'git clone [URL]'	Clona un repositorio remoto en tu máquina local
'git add [archivos]'	Agrega cambios al área de preparación para ser confirmados
'git commit -m "Mensaje"'	Confirma los cambios en el repositorio con un mensaje descriptivo
'git status'	Muestra el estado actual de los archivos en el repositorio
'git log'	Muestra el historial de confirmaciones (commits) en el repositorio
'git branch'	Muestra la lista de ramas y resalta la rama actual
'git checkout [rama]'	Cambia a la rama especificada
'git merge [rama]'	Fusiona la rama especificada con la rama actual
'git pull'	Obtiene cambios desde un repositorio remoto y los fusiona con el repositorio local
'git push'	Envía los cambios locales al repositorio remoto
'git remote -v'	Muestra la lista de los repositorios remotos vinculados
'git fetch'	Recupera los cambios desde un repositorio sin fusionarlos
'git diff'	Muestra las diferencias entre los cambios sin confirmar y el último commit
'git reset [archivo]'	Quita los cambios del área de preparación pero los deja sin confirmar
'git rm [archivo]'	Elimina archivos del repositorio y del sistema de archivos

CARACTERÍSTICAS Y COMANDOS GIT REMOTO	
COMANDO	***DESCRIPCIÓN***
'git remote add [nombre] [URL]'	Agrega un nuevo repositorio remoto
'git remote -v'	Muestra la lista de repositorios remotos vinculados
'git push [remoto] [rama]'	Envía los cambios locales de una rama al repositorio remoto
'git pull [remoto] [rama]'	Obtiene cambios desde un repositorio remoto y los fusiona con el repositorio local
'git fetch [remoto]'	Recupera los cambios desde un repositorio remoto sin fusionarlos
'git clone [URL]'	Clona un repositorio remoto en tu máquina local
'git remote rm [nombre]'	Elimina un repositorio remoto
'git remote rename [viejo] [nuevo]'	Cambia el nombre de un repositorio remoto
'git remote show [remoto]'	Muestra información detallada sobre un repositorio remoto
'git branch -r'	Muestra la lista de ramas remotas
'git push --tags'	Envía las etiquetas (tags) al repositorio remoto
'git pull --tags'	Obtiene las etiquetas del repositorio remoto
'git remote prune [remoto]'	Elimina las referencias remotas que ya no existen en el repositorio remoto

Estos son solo algunos de los comandos más comunes. Se puede obtener información detallada sobre cualquier comando utilizando git help [comando] o git [comando] --help.

DIFERENCIAS ENTRE GIT LOCAL Y GIT REMOTO		
CARACTERÍSTICAS	***GIT LOCAL***	***GIT REMOTO***
Ubicación	En el sistema de archivos de tu máquina local	En un servidor remoto o en la nube
Comandos iniciales	'git init' para iniciar un nuevo repositorio local	'git clone [URL]' para clonar un repositorio remoto
Colaboración	Principalmente individual, cambios realizados localmente	Colaborativo, varios desarrolladores trabajan en un mismo repositorio remoto
Confirmaciones (Commit)	Se realizan localmente con 'git commit'	Confirmaciones también se realizan localmente, pero luego se pueden enviar al repositorio remoto
Ramas	Se pueden crear, modificar y fusionar localmente	Las ramas locales pueden ser enviadas y compartidas en el repositorio remoto
Fusiones (Merges)	Se realizan localmente con 'git merge'	Pueden involucrar cambios locales y remotos, fusionando ramas de diferentes colaboradores
Push y Pull	'git push' envía cambios locales al repositorio remoto. 'git pull'	'git push' envía cambios locales al repositorio remoto. 'git pull' obtiene

	obtiene cambios remotos y los fusiona localmente	cambios remotos y los fusiona localmente
Repositorios remotos	No esencialmente dependiente de repositorios remotos	Implica trabajar con repositorios remotos, como GitHub, GitLab o Bitbucket
Visualización del estado	'git status' muestra el estado de los archivos localmente	'git remote -v' muestra información sobre los repositorios remotos vinculados
Flujo de trabajo	Principalmente centrado en el desarrollo local	Incluye colaboración remota, seguimiento de cambios entre varios colaboradores y equipos
Iniciar colaboración	Generalmente no requiere configuración remota inicial	Puede requerir la clonación de un repositorio remoto existente
Etiquetas (Tags)	Se pueden crear y utilizar localmente	Pueden enviarse y compartirse en el repositorio remoto
Administración remota	No requiere manipulación remota directa	Implica comandos como 'git remote', 'git fetch' y 'git push' para gestionar interacciones remotas

Estas diferencias destacan cómo Git local se enfoca en el desarrollo individual y la gestión de versiones local, mientras que Git remoto se centra en la colaboración y la gestión de proyectos compartidos en un entorno distribuido.