GA7-220501096-AA1-EV03 identifica herramientas de versionamiento

Edwin De Jesús Restrepo
Aprendiz

Felipe Martínez Leiva
Instructor

Tecnología En Análisis Y Desarrollo De Software ficha 2721402
2024

Introducción

El control de versiones es fundamental en el mundo del desarrollo de software contemporáneo, no se trata solo de rastrear cambios en el código, sino de una práctica esencial que permite a los equipos trabajar de manera colaborativa y mantener un registro detallado de la evolución del proyecto, teniendo en cuenta todo esto, Git es una herramienta imprescindible, destacada por su versatilidad y extenso conjunto de funcionalidades, sin embargo, Git no es solo una herramienta estática, puede ser utilizada de dos maneras principales: localmente y de manera remota, cada enfoque tiene sus propias características, ventajas y casos de uso específicos, en este trabajo, exploraremos las diferencias entre Git local y Git remoto, así como su importancia y utilidad en el desarrollo de software.

Objetivo

El objetivo de esta evidencia es demostrar una comprensión profunda de las características y diferencias entre los comandos de Git que se utilizan en entornos locales y los que se utilizan para interactuar con repositorios remotos, al profundizar en estas diferencias, se busca no solo entender cómo funcionan los comandos en cada entorno, sino también comprender la importancia de su correcta aplicación en el desarrollo de software colaborativo.

¿Qué es git? git es un sistema de control de versiones distribuido y de código abierto que se utiliza principalmente para el seguimiento de cambios en el código fuente durante el desarrollo de software. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux y desde entonces ha ganado una amplia popularidad en la comunidad de desarrollo de software.

Características principales de Git:

Distribuido: Cada desarrollador tiene una copia local completa del repositorio de Git, lo que permite un flujo de trabajo descentralizado y la capacidad de trabajar offline.

Rápido y Eficiente: Git es rápido en la mayoría de las operaciones, gracias a su diseño eficiente y a la capacidad de trabajar con instantáneas de archivos en lugar de deltas.

Gestión de Ramas: Git facilita la gestión de múltiples ramas de desarrollo, lo que permite a los equipos trabajar en paralelo en diferentes características o versiones del software.

Integridad de Datos: Utiliza algoritmos criptográficos para garantizar la integridad de los datos almacenados en el repositorio, lo que lo hace altamente confiable en la detección de corrupción o manipulación de archivos.

Amplia Adopción: Es utilizado por una amplia comunidad de desarrolladores y empresas en todo el mundo, lo que significa que hay una gran cantidad de recursos, herramientas y servicios disponibles para trabajar con Git.

GitLab, por otro lado, es una plataforma completa para el ciclo de vida del desarrollo de software que se basa en Git como su sistema de control de versiones subyacente. Ofrece una variedad de herramientas para la gestión de proyectos, la colaboración en equipo, la integración continua y la entrega continua (CI/CD), la supervisión del rendimiento y más.

Características principales de GitLab:

Gestión de Proyectos: Proporciona herramientas para planificar, rastrear y organizar tareas y problemas en un tablero de proyecto.

Colaboración en Equipo: Permite a los equipos trabajar juntos de manera eficiente, revisar y comentar el código, y gestionar las solicitudes de incorporación de cambios (pull requests).

Integración Continúa y Entrega Continua (CI/CD): Ofrece un pipeline de CI/CD integrado que automatiza la construcción, las pruebas y la implementación del software.

Control de Acceso: Permite la configuración de permisos granulares para controlar quién tiene acceso a qué recursos dentro del repositorio.

Escalabilidad y Personalización: Es altamente escalable y puede ser instalado en infraestructuras locales o en la nube pública, y ofrece una amplia gama de opciones de personalización y extensibilidad.

Tabla de diferencias

Características	Git Local	Git Remoto
Ubicación	Repositorio en tu máquina local.	Repositorio en un servidor remoto.
Acceso	Accedido solo localmente.	Accedido por múltiples colaboradores.
Colaboración	Adecuado para trabajo individual o en equipo pequeño.	Facilita el trabajo en equipo y la colaboración remota.
Control de versiones	Controla la versión de tus archivos localmente.	Proporciona un historial centralizado de cambios.
Sincronización	Los cambios no se comparten automáticamente.	Los cambios se sincronizan entre repositorios remotos.
Privacidad	Archivos son privados por defecto.	Pueden ser públicos o privados, según la configuración.
Comandos Principales	git init, git add, git commit, git checkout, etc.	git clone, git push, git pull, git fetch, etc.
Respaldo	Depende de copias de seguridad locales.	Proporciona un respaldo centralizado de los cambios.
Seguridad	Depende de la seguridad local y medidas de protección.	Utiliza medidas de seguridad en el servidor remoto.
Alcance del historial	Historial limitado al repositorio local.	Historial accesible a todos los colaboradores remotos.

Comandos básicos de Git Local y Git Remoto

Git local

Comando	Descripción
Git init	Inicializa un nuevo repositorio local.
Git add [archivo]	Agrega cambios al área de preparación.
Git clone [URL]	Clona un repositorio Git existente en tu directorio local.
Git status	Visualiza el estado actual de tus archivos indicando si están o no rastreados.
Git commit -m "Mensaje"	Confirma los cambios con un mensaje descriptivo.
Git add .	Se utiliza para agregar todos los archivos modificados y nuevos en el directorio de trabajo.
Git branch [nombre]	Crea una nueva rama.
Git checkout [rama]	Cambia a una rama existente.
Git merge [rama]	Fusiona los cambios de una rama con la rama actual.
Git log	Muestra el historial de commits.
Git reset [archivo]	Deshace los cambios en un archivo específico.
Git reset --hard	Deshace todos los cambios locales en el directorio de trabajo.

Git remoto

Comando	Descripción
Git remote add [nombre-remoto] [url]	Sirve paradefinir un repositorio remoto y asociarlo a un nombre para su referenciación.
Git push [remoto] [rama]	Envía los cambios locales al repositorio remoto.
Git pull [remoto] [rama]	Obtiene y fusiona los cambios del repositorio remoto con el local.
Git remote -v	Muestra los repositorios remotos asociados.
Git branch -r	Muestra las ramas remotas.
Git fetch [nombre-remoto]	Obtiene los cambios del repositorio remoto sin fusionarlos con tu rama local.
Git remote remove [nombre]	Elimina un repositorio remoto de tu configuración local.