Material descargable

egg

GIT: Control de Versiones

¿Por qué Git es tan importante?

Los desarrolladores de software frecuentemente se enfrentan a la necesidad de revisar o revertir cambios, comparar versiones anteriores del código, o trabajar en múltiples características o funciones al mismo tiempo. Estos procesos, si se manejan de manera inadecuada, pueden llevar a una gestión desorganizada y propensa a errores, especialmente en proyectos complejos con múltiples colaboradores.

Sin un sistema como Git, los desarrolladores a menudo recurren a la creación de múltiples copias de archivos para cada cambio o nueva característica. Esto no solo consume un valioso espacio de almacenamiento, sino que también complica el seguimiento de las versiones y aumenta el riesgo de confusión y errores. Además, cuando se trabaja en equipo, coordinar cambios y fusionar diferentes versiones del código puede convertirse en un proceso tedioso y propenso a conflictos.

Git es un sistema de control de versiones que permite realizar un seguimiento de los cambios en el código fuente u otros archivos de texto. Funciona como un diario de bitácora, registrando todas las modificaciones que tú y tu equipo realicen.

> 💡Si has trabajado con herramientas de colaboración como Google Drive tendrás una idea básica de lo que Git ofrece ya que puedes ver los cambios realizados por otros usuarios. Sin embargo, Git es mucho más potente y específico para el código, proporcionando un control más granular y funciones especializadas para el desarrollo de software.

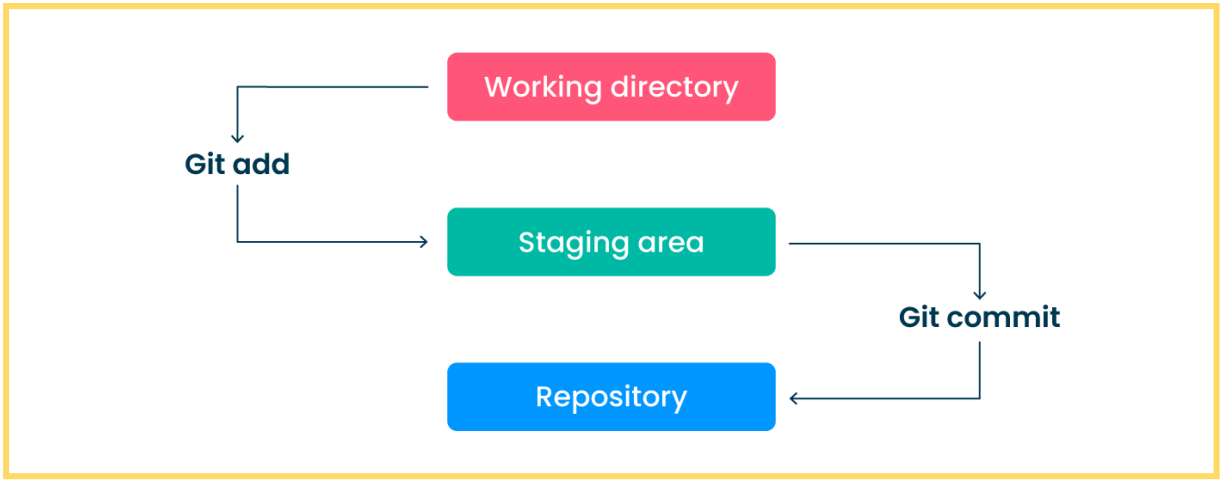
Es importante destacar que la mayoría de las interacciones con Git se realizan a

través de la **consola de comandos**. Puede parecer intimidante al principio, pero el uso de la consola para trabajar con Git ofrece un control preciso y una flexibilidad que son esenciales para el manejo eficiente de versiones de código. Aunque existen interfaces gráficas para Git, la consola es universal y permite un **acceso directo a todas las funcionalidades de Git sin restricciones**.

¿Cuáles son las áreas de trabajo de Git?

Antes de comenzar a utilizar Git en un proyecto, debes inicializar un nuevo repositorio en la carpeta de tu proyecto. Esta acción permite que Git comience a rastrear los cambios en los archivos y almacenar, organizar y mantener la información de los mismos.

El sistema de control de Git está estructurado en **tres áreas clave**: el Working Directory (Directorio de trabajo), el Staging Area (Área de Preparación) y el "Local" Repository (Repositorio local). Cada una de estas áreas juega un rol vital en el proceso de control de versiones, facilitando la organización y gestión de tus cambios en el código.



¿Cómo trabajan estas áreas?

Utilizaremos la siguiente metáfora para comprender mejor estas áreas. Imagina que estás escribiendo un libro:

- **Working Directory:** imagina este espacio como tu escritorio personal, lleno de notas, borradores, y elementos de tu proyecto actual. Este espacio representa el entorno en vivo donde realizas cambios activos en tus archivos. Aquí es donde editas y modificas tus archivos. Estos cambios aún no están listos para

ser parte permanente del historial de tu proyecto; son como borradores en tu escritorio.

Comandos:

- Para inicializar un nuevo repositorio, navega en la consola de comandos hasta la carpeta donde está tu proyecto y ejecuta el comando **“git init**”. Esto creará una nueva subcarpeta llamada .git en tu proyecto, que contiene todos los archivos necesarios para el control de versiones.
- Utiliza “**git status**“ para obtener una visión general de los cambios que has hecho pero que aún no has preparado para un commit. Este comando te mostrará una lista de archivos modificados y su estado.

- **Staging Area**: piensa este espacio como una carpeta donde organizas los capítulos de tu libro antes de enviarlos a tu editor. Aquí seleccionas y preparas qué cambios específicos quieres incluir en tu próximo “envío”. Sirve como una zona intermedia entre el trabajo en progreso y el historial de versiones.

Comandos:

- Para mover todos los archivos del Working Directory al Staging Area, usas “**git add .**”
- Puedes agregar un archivo específico con “**git add nombre.extensión**“.

- **Local Repository**: este espacio es como la editorial donde se publica tu libro. Una vez que tu editor aprueba y publica un capítulo, se convierte en parte oficial de tu libro.Es el corazón de tu proyecto Git. Aquí se almacenan los envíos a través de los commits, que son capturas instantáneas de tus archivos en momentos específicos. Cada commit tiene un mensaje descriptivo y forma parte del historial de tu proyecto.

Comandos:

- Para crear un envío, utiliza “**git commit -m "mensaje descriptivo"**”. Esto tomará los cambios que están en el Staging Area y los añadirá al repositorio de tu computadora (o repositorio local) con un mensaje que

resume los cambios realizados.

- Además con "**git log**", puedes explorar el historial de tus commits, viendo una lista cronológica inversa con detalles como el autor, la fecha, y el mensaje descriptivo de cada commit.
- Con "**git log --oneline**", podrás ver una lista más condensada con cada commit en una sola línea.
- Si prefieres una opción gráfica, con "**git log --graph**" verás la representación de las ramificaciones y fusiones en el historial de commits.

Además de tu repositorio local, puedes usar servicios en la nube como **GitHub** o **GitLab** para alojar una copia de tu repositorio en un repositorio remoto. Esto facilita la colaboración y el intercambio de información con otros desarrolladores.

¿Cuáles son los comandos esenciales de Git?

Entonces, en resumen:

git init: inicia un proyecto de git.

git status: muestra los archivos que han cambiado desde el último commit y los cambios preparados para el próximo commit.

git add: mueve los cambios desde el Working Directory al Staging Area, preparándolos para el próximo commit.

git commit: mueve los cambios del Staging Area al Repository, actualizando el historial de tu proyecto.

git log: permite ver el historial de commits.

Estos conceptos de Git son fundamentales para comprender cómo manejar tus cambios. A partir de los mismos, podrás explorar características más avanzadas con mayor facilidad.

¿Qué es MarkDown?

El formato Markdown (extensión .md) permite aplicar formatos básicos de texto (como negritas, cursivas), crear listas, enlazar imágenes, insertar enlaces, y más, utilizando una sintaxis sencilla.

Markdown es un lenguaje de marcado ligero diseñado para facilitar la lectura y escritura. Markdown es popular en el mundo del desarrollo de software debido a su simplicidad y eficacia para crear documentos formateados, como archivos **README**, documentación de proyectos, notas, entre otros.

¿Cómo implemento la sintaxis de MarkDown?

Para definir encabezados o títulos::

para títulos de nivel 1 (de máxima jerarquía).
para títulos de nivel 2 (equivalente a <h2> en HTML).
Así sucesivamente hasta ###### para encabezados de nivel 6.

Para dar énfasis en un texto:

Texto en cursiva o _Texto en cursiva_ para cursiva.
Texto en negrita o __Texto en negrita__ para negrita.
~~Texto tachado~~ para texto tachado.

Para generar listas:

- o * para listas sin orden (bullets).
1. para listas ordenadas. Los números se incrementan automáticamente.

¿Para qué sirve el README de un repositorio?

El archivo README.md de un repositorio es esencialmente la "cara frontal" o la primera página de documentación de tu proyecto. Es uno de los archivos más importantes en cualquier repositorio y sirve para varios propósitos cruciales:

- **Descripción del proyecto**: Breve explicación del propósito y función del proyecto.
- **Instrucciones de instalación**: Pasos para configurar y comenzar a usar el proyecto.
- **Uso y ejemplos**: Ejemplos prácticos de cómo se utiliza el proyecto.
- **Enlaces a documentación adicional**: Referencias a documentación más detallada.
- **Información para contribuyentes**: Instrucciones para aquellos interesados en contribuir al proyecto.
- **Licencia**: Detalles sobre la licencia bajo la cual se distribuye el proyecto.
- **Contacto y reconocimientos**: Información de contacto del autor y agradecimientos a colaboradores.

- **Badges**: Indicadores visuales que muestran el estado y métricas del proyecto.