Nombre y Apellido: Nehuen Parra
Curso: 5to 2da
Fecha de entrega: 12/12/24

T.P. Git

Origen de Git

- 2005: Git fue creado por Linus Torvalds, el creador del núcleo Linux, en respuesta a un problema crítico relacionado con el sistema de control de versiones que se usaba para gestionar el desarrollo de Linux en ese momento: BitKeeper.
    1. BitKeeper era una herramienta propietaria que la comunidad de Linux había estado utilizando desde 2002. Sin embargo, en 2005, surgieron tensiones entre los desarrolladores de Linux y los propietarios de BitKeeper, lo que llevó a que la herramienta dejara de estar disponible para su uso gratuito.
- Linus Torvalds decidió crear su propia solución para garantizar que el proyecto Linux tuviera un sistema de control de versiones libre y eficiente. Así nació Git, con tres objetivos principales:
    1. Rapidez.
    2. Diseño distribuido (cada desarrollador tiene una copia completa del repositorio).
    3. Integridad (los datos nunca deben corromperse o perderse).

Desarrollo inicial

- En tan solo unas semanas, Linus diseñó y lanzó la primera versión de Git. Su arquitectura se centró en el manejo eficiente de grandes proyectos como el núcleo Linux.
- Poco después, Junio Hamano, otro destacado desarrollador, asumió el liderazgo del proyecto y contribuyó a convertir a Git en la herramienta poderosa y versátil que conocemos hoy.

Evolución y adopción

- Con el tiempo, Git se refinó y se hizo más fácil de usar, lo que facilitó su adopción por una comunidad más amplia.
- En 2008, Git recibió un gran impulso con el lanzamiento de **GitHub**, una plataforma que ofrecía alojamiento de repositorios Git y herramientas de colaboración. GitHub se convirtió en el estándar para compartir y colaborar en proyectos de código abierto, ayudando a popularizar Git en todo el mundo.

Características clave de Git

- Es distribuido: Cada copia del repositorio contiene el historial completo del proyecto, lo que lo hace robusto frente a fallos.
- Es rápido: Diseñado para operaciones eficientes incluso en proyectos de gran escala.
- Proporciona seguridad e integridad: Utiliza hash SHA-1 para garantizar que los datos no se alteren sin que el cambio sea detectable.

Impacto actual

Hoy en día, Git es el sistema de control de versiones más utilizado del mundo. No solo es una herramienta esencial para desarrolladores de software, sino que también ha influido en la forma en que otros sistemas de control de versiones han sido diseñados. Su versatilidad y eficiencia han hecho que sea un estándar tanto en el desarrollo de código abierto como en el desarrollo de software comercial.

¿Y qué usos se le puede dar a git?

1. Gestión de código fuente

- Propósito principal: Git permite rastrear y gestionar los cambios en el código fuente de un proyecto. Los desarrolladores pueden trabajar en diferentes características o correcciones sin interferir entre sí, gracias a las ramas.
- Ejemplo: Un equipo de desarrolladores puede colaborar en un proyecto de software, mantener el historial de cambios y revertir errores fácilmente.

2. Colaboración en equipo

- Uso**:** Git facilita la colaboración entre desarrolladores, ya que cada miembro puede trabajar en su propia copia del repositorio y fusionar los cambios posteriormente.
- Ejemplo: Plataformas como GitHub, GitLab o Bitbucket utilizan Git para ofrecer funcionalidades adicionales, como revisiones de código, seguimiento de tareas y gestión de proyectos.

3. Documentación y escritura colaborativa

- Aunque es más común en el desarrollo de software, Git también puede usarse para trabajar en documentos de texto o proyectos de escritura donde los cambios y versiones sean importantes.
- Ejemplo: Gestionar libros, manuales técnicos o documentos científicos.

4. Desarrollo de proyectos open-source

- Git es la base de la mayoría de los proyectos de código abierto. Permite a los colaboradores de todo el mundo contribuir a un proyecto, enviar cambios mediante "pull requests" y mantener un registro de quién hizo qué.
- Ejemplo: Colaborar en bibliotecas o herramientas públicas como Node.js, React o Python.

5. Proyectos de diseño gráfico

- Aunque menos común, Git puede usarse para rastrear cambios en archivos binarios como imágenes o diseños. Existen complementos y herramientas para integrar Git con programas de diseño.
- Ejemplo: Diseñadores gráficos que trabajan en un equipo pueden gestionar diferentes versiones de logos o diseños.

6. Automatización y DevOps

- Git se utiliza en pipelines de integración y entrega continua (**CI/CD**) para automatizar pruebas, despliegues y otros procesos relacionados con el desarrollo.
- Ejemplo: Al actualizar el código en un repositorio, un sistema como Jenkins o GitHub Actions puede compilar y desplegar automáticamente el software en producción.

7. Gestión de configuraciones

- Equipos de infraestructura y DevOps usan Git para gestionar configuraciones y scripts que automatizan servidores o servicios.
- Ejemplo: Mantener un repositorio con scripts para configurar servidores o gestionar infraestructura como código (IaC) con herramientas como Terraform.

8. Seguimiento de proyectos personales

- Muchas personas usan Git para gestionar proyectos no relacionados con el software, como la creación de blogs, portafolios o incluso el seguimiento de sus finanzas personales.
- Ejemplo: Guardar un historial de cambios en un blog escrito en Markdown o en los archivos de un portafolio.

9. Educación y aprendizaje

- Los estudiantes y profesores utilizan Git para rastrear el progreso en proyectos educativos, distribuir código fuente para prácticas y facilitar la colaboración.
- Ejemplo: Un profesor puede compartir código base con los estudiantes y recibir sus contribuciones mediante "pull requests".

¿Qué es GitHub?

GitHub es un servicio que combina:

1. Almacenamiento de repositorios Git: Permite alojar y gestionar proyectos utilizando Git.
2. Herramientas de colaboración: Incluye funciones como seguimiento de tareas, revisiones de código, discusiones y más.
3. Una red social para desarrolladores: Los usuarios pueden seguir proyectos, contribuir a repositorios públicos y conectar con otros desarrolladores.

Fue lanzado en 2008 por Tom Preston-Werner, Chris Wanstrath y PJ Hyett. En 2018, fue adquirida por Microsoft.

Características principales

1. Repositorio remoto

- GitHub permite a los desarrolladores alojar sus proyectos de Git en la nube. Esto facilita el acceso desde cualquier lugar y la colaboración con otros.

2. Colaboración

- Soporta herramientas como:
    - Pull Requests: Permiten a los colaboradores proponer cambios en el código.
    - Revisiones de código: Los miembros del equipo pueden revisar y comentar los cambios antes de fusionarlos.
    - Issues: Sistema de seguimiento de tareas, errores o nuevas ideas.

3. Gestión de proyectos

- Ofrece tableros similares a Kanban para organizar tareas y flujos de trabajo, útil en metodologías ágiles como Scrum.

4. GitHub Actions

- Herramienta de automatización integrada para crear flujos de trabajo personalizados, como pruebas, despliegues y notificaciones.

5. Alojar páginas web

- Con GitHub Pages, puedes alojar sitios web estáticos directamente desde un repositorio.

6. Open source y contribuciones

- Muchos proyectos de código abierto están alojados en GitHub, lo que permite a los desarrolladores contribuir a grandes proyectos y aprender.

7. Seguridad

- Integración con herramientas para escanear vulnerabilidades, verificar dependencias y proteger el código.
- Soporte para claves SSH y autenticación de dos factores.

Usos principales

1. Desarrollo de software

- GitHub es la plataforma preferida para desarrollar, gestionar y colaborar en proyectos de software.

2. Código abierto

- Miles de proyectos como Linux, React, y TensorFlow están alojados en GitHub, y cualquiera puede contribuir.

3. Proyectos personales

- Ideal para almacenar proyectos personales, como scripts, aplicaciones o blogs.

4. Educación

- Profesores y estudiantes usan GitHub para compartir materiales, colaborar en proyectos y enseñar conceptos de programación.

5. Documentación

- Es excelente para escribir y mantener documentación, gracias al soporte de Markdown.

Impacto

GitHub no solo es una herramienta para almacenar código, sino que también ha creado una cultura de colaboración, aprendizaje y desarrollo comunitario. Es un estándar de facto para los desarrolladores de todo el mundo.

¿Qué alternativas hay para github?

- GitLab
- Bitbucket
- SourceForge
- Codeberg
- AWS CodeCommit

- Gitea
- Phabricator (ahora limitado)
- Azure DevOps (anteriormente Visual Studio Team Services)
- Fossil
- Launchpad