Introducción

Un sistema de control de versiones (VCS, por sus siglas en inglés) es una herramienta que registra y gestiona los cambios en archivos a lo largo del tiempo. Es en proyectos extensos o colaborativos donde se necesita:

- Registrar quién realizó modificaciones, cuándo y qué se cambió.
- Recuperar versiones anteriores en caso de errores, pérdidas o conflictos (Atlassian, s.f.-a; Chacon & Straub, 2014).

En proyectos con múltiples documentos, carpetas y colaboradores, la falta de control de versiones puede ocasionar pérdida de información, dificultad para organizar los cambios y riesgo de sobrescribir trabajo ajeno.

Git: Sistema de Control de Versiones Distribuido

Git es un sistema de control de versiones distribuido, creado por Linus Torvalds en 2005. Fue desarrollado para gestionar el código fuente del núcleo Linux, y desde entonces se ha convertido en el estándar de facto en la industria del software (Chacon & Straub, 2014).

Características principales

- **Distribuido**: cada colaborador tiene una copia completa del historial del proyecto en su máquina.
- **Ramas livianas y fusiones rápidas**: permite una gestión eficiente del trabajo paralelo.
- **Trabajo sin conexión**: la mayoría de las operaciones pueden realizarse sin conectividad, lo que aporta independencia y rapidez (Loeliger & McCullough, 2012).

Commits

Un *commit* es una instantánea (snapshot) del proyecto en un momento determinado. Incluye los archivos modificados, la identidad del autor, la fecha y un mensaje que describe los cambios realizados. Los archivos atraviesan tres estados principales: **modificado**, **staged** (preparado) y **committed** (confirmado).

Buenas prácticas

- Realizar commits frecuentes y acotados facilita el seguimiento de cambios y la identificación de errores.
- Escribir mensajes descriptivos y concisos mejora la trazabilidad del proyecto y la colaboración (Loeliger & McCullough, 2012).

Ramas (Branches)

Una rama permite desarrollar funcionalidades nuevas, corregir errores o experimentar, sin alterar la línea principal del proyecto. Cada rama es una copia lógica del proyecto que puede evolucionar de forma independiente (GitLab, s.f.-a).

Al fusionar ramas, pueden surgir conflictos si dos ramas modificaron el mismo archivo de manera incompatible. Git intenta resolver estos conflictos automáticamente, pero si no puede, el usuario debe decidir manualmente qué cambios conservar (Atlassian, s.f.-b; GitLab, s.f.-a).

Repositorios remotos y GitHub

Los repositorios remotos permiten sincronizar el trabajo entre diferentes equipos y dispositivos. Plataformas como GitHub, GitLab o Bitbucket ofrecen alojamiento de repositorios para colaboración en línea, historial compartido y herramientas de integración continua.

Comandos principales

- git clone: clona un repositorio remoto en el entorno local.
- git pull: obtiene y fusiona los cambios más recientes desde el remoto.
- git push: envía los cambios locales al repositorio remoto.

Aunque estos comandos pueden ejecutarse manualmente desde la línea de comandos, herramientas gráficas como GitHub Desktop los ejecutan automáticamente en segundo plano.

Forks y Pull Requests

- **Fork**: copia de un repositorio en la cuenta del usuario, útil para realizar cambios sin afectar el proyecto original.
- **Pull Request (PR)**: solicitud formal para integrar los cambios realizados en una rama o fork. Es revisada por los mantenedores del repositorio, quienes deciden si se aceptan (Atlassian, s.f.-a; GitLab, s.f.-a).

Ventajas del control distribuido

- **Trabajo sin conexión**: cada usuario puede desarrollar localmente sin conectarse al servidor.
- **Redundancia**: cada copia local del repositorio actúa como respaldo completo del historial.
- **Colaboración descentralizada**: se facilita el desarrollo en paralelo, incluso sin acceso directo al repositorio principal (Chacon & Straub, 2014; GitLab, s.f.-b).

Referencias

Atlassian. (s.f.-a). *What is version control?* Atlassian Git Tutorials. https://www.atlassian.com/git/tutorials/what-is-version-control

Atlassian. (s.f.-b). *Why Git?* Atlassian Git Tutorials. https://www.atlassian.com/git/tutorials/why-git

Chacon, S., & Straub, B. (2014). *Pro Git* (2.ª ed.). Apress. https://git-scm.com/book/en/v2

GitLab. (s.f.-a). *What is Git version control?* GitLab. https://about.gitlab.com/topics/version-control/what-is-git-version-control

GitLab. (s.f.-b). *Benefits of a distributed version control system*. GitLab. https://about.gitlab.com/topics/version-control/benefits-distributed-version-control-system

Loeliger, J., & McCullough, M. (2012). *Version Control with Git* (2.ª ed.). O'Reilly Media.