Engineer

Git

control de versiones que guarda el historial de este sitio

Qué es

Git es un sistema de control de versiones distribuido: un programa que registra el estado de un conjunto de archivos como una cadena de instantáneas, de modo que cualquier estado anterior se pueda consultar, comparar con el presente o recuperar. El proyecto escribe el nombre como Git y la orden como `git`; ambos designan la misma herramienta.

Cada instantánea es un commit. Un commit guarda cómo era cada archivo bajo seguimiento en ese momento, quién hizo el cambio y cuándo, un mensaje que explica por qué y un puntero al commit o los commits anteriores. Git solo guarda una vez un archivo que ya ha visto: si no cambia entre dos commits, la instantánea nueva apunta a la copia que ya está en disco en vez de escribirla otra vez, por lo que un historial largo ocupa mucho menos que una copia completa por commit.

Todo lo que contiene se nombra por su contenido. Antes de guardar un archivo, el listado de un directorio o un commit, Git calcula una suma de comprobación de sus bytes: en el formato clásico, un hash SHA-1 escrito con 40 caracteres hexadecimales. Esa suma pasa a ser el nombre del objeto. Cambiar un solo byte cambia también el nombre, y con él el de cada commit que se apoya en ese objeto. Así detecta Git la corrupción de datos y por eso un identificador de commit sigue apuntando exactamente al mismo código años después.

Distribuido significa que cada copia es completa. Al clonar un repositorio se descarga todo el historial, no solo los archivos más recientes, de modo que consultar el registro, comparar dos versiones o crear un commit ocurre en el disco local, sin servidor. Compartir trabajo es un paso distinto y deliberado: se envían los commits a otra copia con push, o se recuperan de una con fetch y se

fusionan. Cualquier copia puede actuar como servidor; en la mayoría de los equipos, una copia acordada, a menudo en un servicio alojado, cumple ese papel por convención y no por diseño.

El trabajo diario pasa por tres lugares. El árbol de trabajo contiene los archivos que se editan. El área de preparación, también llamada índice, guarda lo que incluirá el próximo commit, lo que permite confirmar una parte de los cambios y dejar el resto para después. El directorio de Git, `.git` en la raíz del proyecto, guarda la base de datos de objetos y todas las referencias. Una rama no es más que un nombre móvil que apunta a un commit, por eso crearla es instantáneo.

Linus Torvalds y la comunidad del kernel de Linux escribieron Git en 2005, después de que se retirara la gratuidad de BitKeeper, la herramienta propietaria que utilizaban. Sus objetivos eran velocidad, un diseño sencillo, buen soporte para miles de ramas paralelas, distribución completa y capacidad para manejar un proyecto del tamaño del kernel.

Utilice el concepto en una decisión

Aplique «¿Qué es Git?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Cambios probados, publicados y reversibles, para que lleguen a sus clientes un martes por la tarde en vez de un viernes por la noche.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Despliegue GIS SaaS con GitHub Actions — despliegue, datos e informes automatizados

Despliegue GIS SaaS con GitHub Actions Póngase en contacto →

Por qué importa

El control de versiones responde a las preguntas que surgen en cuanto más de una persona, o una sola durante más de una semana, toca los mismos archivos: qué cambió, cuándo, quién y por qué. Si se lleva a mano, las respuestas viven en la memoria, en nombres como `final-v3` y en hilos de correo. Con Git viven junto al código, vinculadas a cada cambio y al alcance de cualquiera que las busque.

El diseño distribuido cambia el trabajo diario de un equipo. Cada clon contiene todo el historial, así que nadie espera a un servidor central para consultar el registro o crear un commit, y todos pueden seguir trabajando aunque el servidor se caiga. Las ramas cuestan poco: cada funcionalidad, corrección o experimento puede tener una propia, revisarse por separado e integrarse cuando esté listo, mientras la línea principal sigue apta para entregarse.

Ese mismo diseño hace que Git sea conservador con los datos. Casi todas las operaciones solo añaden a la base de datos de objetos, así que cuesta perder trabajo ya confirmado: normalmente se puede recuperar una rama eliminada por error o deshacer un rebase que salió mal o un reset al commit equivocado. La red de seguridad es el reflog, un registro local de adónde apuntaba cada rama y cuándo, conservado de forma predeterminada durante 90 días para commits aún alcanzables y 30 días para los que ya no lo son.

Git es además la base común de la mayoría de las otras herramientas de una cadena de entrega. Forjas, herramientas de revisión, sistemas de compilación, scripts de despliegue y generadores de sitios estáticos leen de él. Eso convierte el historial en una fuente de hechos más allá del código: el día en que cambió una página por última vez, el commit desde el que se construyó una versión o el cambio que introdujo un fallo.

Cómo se aplica en la práctica

El código propio de esta empresa muestra tres aspectos de ello.

El código fuente estaba antes en un servicio de alojamiento ajeno. Ahora está en un servidor Git que la empresa opera en su propio hardware: un único binario supervisado por el sistema operativo, elegido por priorizar SSH y carecer de interfaz web administrativa. Los clones y pushes viajan por SSH al puerto 23231, que solo escucha en la interfaz de bucle local, por lo que un cliente remoto externo al host llega mediante un host de salto; se rechazan tanto las sesiones anónimas como las que no tienen clave, y todos los repositorios declarados están marcados como privados. La página dice claramente cuál es el precio: operar una forja propia también exige hacerse cargo de autenticación, control de acceso, almacenamiento y copias de seguridad.

Es fácil confundir copias completas del historial en varias máquinas con una copia de seguridad. La implementación de copias de seguridad y pruebas de restauración de la empresa se diseñó para ampliar la recuperación más allá de una instantánea del proveedor. Sus scripts de restic y unidades de systemd admiten un volcado de base de datos donde esté configurado, instantáneas cifradas y deduplicadas hacia almacenamiento de otro proveedor por SFTP, eliminación según la política de retención y comprobaciones de integridad. Un flujo mensual separado restaura un archivo conocido en una ubicación temporal y compara sus bytes. Esa prueba debe ejecutarse de verdad antes de demostrar una capacidad de recuperación: el registro de implementación fechado el 2026-

10-27 dice que los temporizadores estaban desactivados al no haber un repositorio configurado y no registra ninguna restauración real completada. Ninguna cantidad de clones en portátiles aporta esa evidencia que falta.

El historial también alimenta el propio sitio. La página de créditos registra que el W3C Feed Validator advierte de que la mayoría de las entradas de los feeds comparten fecha de modificación, y que tiene razón: las fechas salen del historial de Git y la mayoría de esas páginas se escribieron realmente en un puñado de commits. El sitio prefiere mostrar la advertencia a inventarse una fecha. El historial cuenta lo que ocurrió, incluido que buena parte ocurrió a la vez.

Una orden permite recuperar uno de esos hechos. Esta imprime el identificador corto, la fecha del commit en formato ISO 8601 y el asunto del último commit que modificó un archivo:

```
git log -1 --format='%h %cI %s' -- README.md
```

Dónde surgen los problemas

- **Un secreto confirmado una vez permanece en el historial.** Eliminar el archivo en el siguiente commit lo quita de la instantánea más reciente, pero lo deja en la anterior, que cada clon ya lleva. Trate la credencial como expuesta y cámbiela; reescribir el historial después limita la propagación, pero las copias hechas entre tanto conservan los commits antiguos.
- **Un push forzado reemplaza trabajo ajeno.** `git push --force` desactiva la comprobación de que la rama remota sea antecesora de lo que se envía, por lo que los commits que un colega haya enviado mientras tanto desaparecen de la rama. `--force-with-lease` solo actualiza el remoto si aún apunta a donde se vio por última vez.
- **Los clones se tratan como copia de seguridad.** La copia de un portátil va por detrás, solo contiene las ramas que su propietario llegó a recuperar y nunca se ha restaurado deliberadamente desde ella. Incidencias, reglas de acceso y demás datos propios de una forja quedan por completo fuera del repositorio.
- **Un archivo binario grande engorda todos los clones para siempre.** Git guarda cada versión de cada archivo y todo clon las descarga todas, así que un vídeo confirmado por accidente y eliminado al día siguiente sigue viajando a cada máquina que clone el proyecto después.

- **El trabajo guardado en un solo lugar desaparece con él.** El reflog que rescata un reset incorrecto vive en una sola máquina y caduca. Los commits que nunca se enviaron no existen en otro sitio, y una rama abandonada durante meses se aleja tanto que volver a integrarla se convierte en un proyecto propio.

Lecturas adicionales

- Documentación de Git — El manual de referencia, el libro y los vídeos, publicados por el propio proyecto.
- Pro Git — Todo el libro en línea, desde el primer commit hasta la base de datos de objetos que hay debajo.
- ¿Qué es Git? — Instantáneas, operaciones locales, sumas de comprobación y los tres estados en un capítulo.
- Breve historia de Git — Por qué la comunidad del kernel escribió su propia herramienta en 2005 y qué se propuso conseguir.
- git-reflog — Cómo encontrar adónde apuntaba una rama antes del error y cuánto dura ese registro.
- git-push — Las opciones de envío forzado y qué protege cada una.



Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Git?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/git/