Control de versiones

Sistema de control de versiones

El sistema de control de versiones (VCS) es un software que ayuda a los desarrolladores de software a trabajar juntos y a mantener un historial completo de su trabajo.

A continuación, se enumeran las funciones de un VCS

- Permite a los desarrolladores trabajar simultáneamente.
- No permite sobrescribir los cambios de los demás.
- Mantiene un historial de cada versión.

Los tipos de VCS son los siguientes

- Sistema de control de versiones centralizado (CVCS).
- Sistema de control de versiones distribuido/descentralizado (DVCS).

Sistema de control de versiones distribuido

El sistema de control de versiones centralizado (CVCS) utiliza un servidor central para almacenar todos los archivos y permite la colaboración en equipo. Pero el mayor inconveniente del CVCS es su único punto de fallo, es decir, el fallo del servidor central. Desgraciadamente, si el servidor central se cae durante una hora, durante esa hora nadie puede colaborar en absoluto. E incluso en el peor de los casos, si el disco del servidor central se corrompe y no se hace una copia de seguridad adecuada, se perderá todo el historial del proyecto. Aquí entra en escena el sistema de control de versiones distribuido (DVCS).

Los clientes de DVCS no sólo comprueban la última instantánea del directorio, sino que también reflejan completamente el repositorio. Si el servidor se cae, el repositorio de cualquier cliente puede copiarse al servidor para restaurarlo. Cada comprobación es una copia de seguridad completa del repositorio. Git no depende del servidor central y por eso puedes realizar muchas operaciones cuando estás desconectado. Puedes confirmar cambios, crear ramas, ver registros y realizar otras operaciones cuando estás desconectado. Sólo necesitas conexión a la red para publicar tus cambios y tomar los últimos cambios.

Ventajas de Git

Código abierto y gratuito

Git se publica bajo la licencia de código abierto de GPL. Está disponible libremente en Internet. Puedes utilizar Git para gestionar proyectos inmobiliarios sin pagar un solo céntimo. Al ser un código abierto, puedes descargar su código fuente y también realizar cambios según tus necesidades.

Información tomada de: https://www.tutorialspoint.com/git/index.htm

Rápido y pequeño

Como la mayoría de las operaciones se realizan localmente, supone una gran ventaja en términos de velocidad. Git no depende del servidor central; por eso, no hay necesidad de interactuar con el servidor remoto para cada operación. El núcleo de Git está escrito en C, lo que evita los gastos de ejecución asociados a otros lenguajes de alto nivel. Aunque Git refleja todo el repositorio, el tamaño de los datos en el lado del cliente es pequeño. Esto ilustra la eficiencia de Git a la hora de comprimir y almacenar datos en el lado del cliente.

Copia de seguridad implícita

Las posibilidades de perder datos son muy escasas cuando existen múltiples copias de estos. Los datos presentes en cualquier lado del cliente reflejan el repositorio, por lo que pueden ser utilizados en caso de un fallo o corrupción del disco.

Seguridad

Git utiliza una función hash criptográfica común llamada función hash segura (SHA1), para nombrar e identificar objetos dentro de su base de datos. Cada archivo y confirmación se comprueba y se recupera por su suma de comprobación en el momento de la comprobación. Esto implica que, es imposible cambiar el archivo, la fecha y el mensaje de confirmación y cualquier otro dato de la base de datos de Git sin conocer Git.

No se necesita un hardware potente

En el caso de CVCS, el servidor central debe ser lo suficientemente potente como para atender las peticiones de todo el equipo. Para los equipos más pequeños, no es un problema, pero a medida que el tamaño del equipo crece, las limitaciones de hardware del servidor pueden ser un cuello de botella en el rendimiento. En el caso de DVCS, los desarrolladores no interactúan con el servidor a menos que necesiten introducir o retirar cambios. Todo el trabajo pesado se realiza en el lado del cliente, por lo que el hardware del servidor puede ser muy sencillo.

Facilidad de bifurcación

CVCS utiliza un mecanismo de copia barato, si creamos una nueva rama, copiará todos los códigos a la nueva rama, por lo que consume tiempo y no es eficiente. Además, la eliminación y la fusión de ramas en CVCS es complicada y requiere mucho tiempo. Pero la gestión de ramas con Git es muy sencilla. Sólo se tarda unos segundos en crear, eliminar y fusionar ramas.

Terminologías del DVCS

Local Repository

Cada herramienta VCS proporciona un puesto de trabajo privado como copia de trabajo. Los desarrolladores realizan cambios en su puesto de trabajo privado y, tras su confirmación, estos cambios pasan a formar parte del repositorio. Git va un paso más allá al proporcionarles una

copia privada de todo el repositorio. Los usuarios pueden realizar muchas operaciones con este repositorio, como añadir un archivo, eliminar un archivo, renombrar un archivo, mover un archivo, confirmar los cambios, y muchas más.

Working directory and Staging área or Index

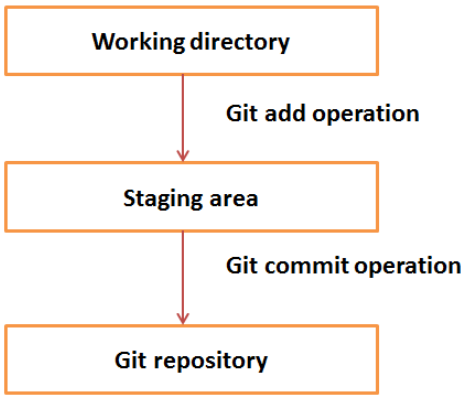
El directorio de trabajo es el lugar en el que se extraen los archivos. En otros CVCS, los desarrolladores suelen hacer modificaciones y confirmar sus cambios directamente en el repositorio. Pero Git utiliza una estrategia diferente. Git no rastrea todos y cada uno de los archivos modificados. Cada vez que se confirma una operación, Git busca los archivos presentes en el área de staging. Sólo los archivos presentes en el área de preparación se consideran para la confirmación y no todos los archivos modificados.

Flujo de trabajo básico de Git.

1. Se modifica un archivo del directorio de trabajo.

2. Se añaden estos archivos al área de preparación.

3. Se realiza la operación de commit que mueve los archivos del área de staging. Después de la operación push, almacena los cambios permanentemente en el repositorio Git.



Blobs

Blob son las siglas de Binary Large Object. Cada versión de un archivo está representada por un blob. Un blob contiene los datos del archivo pero no contiene ningún metadato sobre el mismo. Es un archivo binario, y en la base de datos de Git, se nombra hash SHA1 de ese archivo. En Git, los archivos no están dirigidos por nombres. Todo está dirigido por el contenido.

Trees

Información tomada de: https://www.tutorialspoint.com/git/index.htm

El árbol es un objeto que representa un directorio. Contiene blobs así como otros subdirectorios. Un árbol es un archivo binario que almacena referencias a blobs y árboles que también son nombrados como hash SHA1 del objeto árbol.

Commits

Los commits guardan el estado actual del repositorio. Un commit también se nombra con el hash SHA1. Puedes considerar un objeto commit como un nodo de la lista enlazada. Cada objeto commit tiene un puntero al objeto commit padre. A partir de una confirmación dada, se puede viajar hacia atrás mirando el puntero del padre para ver la historia de la confirmación. Si una confirmación tiene varias confirmaciones padre, entonces esa confirmación en particular ha sido creada mediante la fusión de dos ramas.

Branches

Las ramas se utilizan para crear otra línea de desarrollo. Por defecto, Git tiene una rama maestra, que es lo mismo que el tronco en Subversion. Normalmente, se crea una rama para trabajar en una nueva característica. Una vez que la característica se ha completado, se fusiona de nuevo con la rama maestra y se elimina la rama. Cada rama está referenciada por HEAD, que apunta a la última confirmación de la rama. Cada vez que se hace un commit, HEAD se actualiza con el último commit.

Tags

Las etiquetas asignan un nombre significativo con una versión específica en el repositorio. Las etiquetas son muy similares a las ramas, pero la diferencia es que las etiquetas son inmutables. Esto significa que la etiqueta es una rama que nadie pretende modificar. Una vez que se crea una etiqueta para una confirmación particular, incluso si se crea una nueva confirmación, no se actualizará. Por lo general, los desarrolladores crean etiquetas para los lanzamientos de productos.

Clone

La operación de clonación crea una instancia del repositorio. La operación de clonación no sólo comprueba la copia de trabajo, sino que también refleja el repositorio completo. Los usuarios pueden realizar muchas operaciones con este repositorio local. El único momento en que la red se involucra es cuando las instancias del repositorio se sincronizan.

Pull

La operación pull copia los cambios de una instancia de repositorio remota a una local. La operación pull se utiliza para la sincronización entre dos instancias de repositorio. Es lo mismo que la operación de actualización en Subversion.

Push

Información tomada de: https://www.tutorialspoint.com/git/index.htm

La operación Push copia los cambios de una instancia de repositorio local a una remota. Se utiliza para almacenar los cambios permanentemente en el repositorio Git. Es lo mismo que la operación de confirmación en Subversion.

HEAD

HEAD es un puntero, que siempre apunta a la última confirmación en la rama. Cada vez que se hace una confirmación, HEAD se actualiza con la última confirmación. Las cabezas de las ramas se almacenan en el directorio .git/refs/heads/.

Revision

La revisión representa la versión del código fuente. Las revisiones en Git están representadas por commits. Estos commits se identifican mediante hashes seguros SHA1.

URL

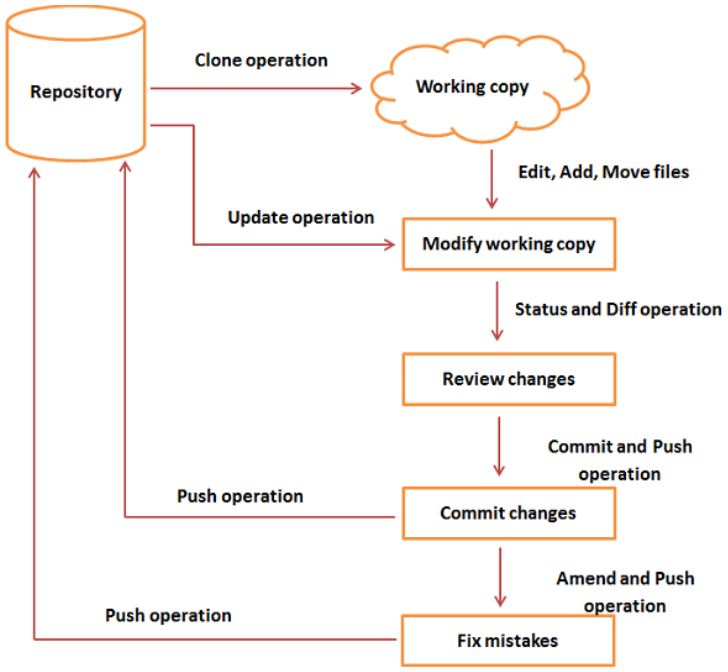
La URL representa la ubicación del repositorio Git. La URL de Git se almacena en el archivo de configuración.

Flujo General

El flujo de trabajo general es el siguiente

1. Se clona el repositorio Git como copia de trabajo.
2. Se modifica la copia de trabajo añadiendo/editando archivos.
3. Si es necesario, también se actualiza la copia de trabajo tomando los cambios de otros desarrolladores.
4. Revisa los cambios antes de confirmarlos.
5. Confirme los cambios. Si todo está bien, entonces empuja los cambios al repositorio.
6. Después de confirmar, si te das cuenta de que algo está mal, entonces corriges la última confirmación y empujas los cambios al repositorio.

A continuación, se muestra la representación pictórica del flujo de trabajo.



Para más detalles sobre las acciones y operaciones con GIT, consulta la fuente indicada en los pie de página o revisar el documento “GIT-cheat-sheet.pdf”

Información tomada de: https://www.tutorialspoint.com/git/index.htm