La arquitectura y la historia de Git: un sistema de control de versiones distribuido

https://medium.com/@willhayjr/the-architecture-and-history-of-git-a-distributed-version-control-system-62b17dd37742

Historia temprana:

Git nacio de la frustación de la comunidad de Linux Kernel con los VCS disponibles , los desarrolladores de Linux no se vieron cubiertos sus necesidades con estas herramientas. Comenzando a utilizar BitKeeper que proporcionaba un control de revisión distribuido y mientas CVS (Concurrent Revisions System) era un sistema de control de versiones cliente-servidor.

En 2005, Larry McVoy , titular de BitKeeper anuncio la revocación de licencias, con lo cual dejo de ser gratuita por culpa de un desarrollador que había violada su licencia. Varios usuarios quedaron sin poder utilizar la herramienta lo cual genero un descontento en la comunidad de desarrolladores y usuarios.

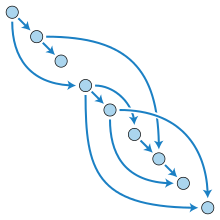
Linus Torvalds, el principal desarrollador del Kernel de Linux, comenzó a trabajar en un nuevo CVS después de no ver otras versiones gratiruitas que satisfaciera sus necesidades.

Desarrollo Inicial:

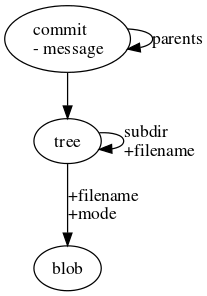
Linus, mantuvo los beneficios de BitKeeper. Se destacaron 3 caracteristicas claves: salvaguardar contra la corrupción del contenido, alto rendimiento y flujo de trabajo de desarrollo distribuido. También se enfatizo que la aplicación de parches de parches no debería tomar mas de 3 segundos.

Descripción de la Arquitectura

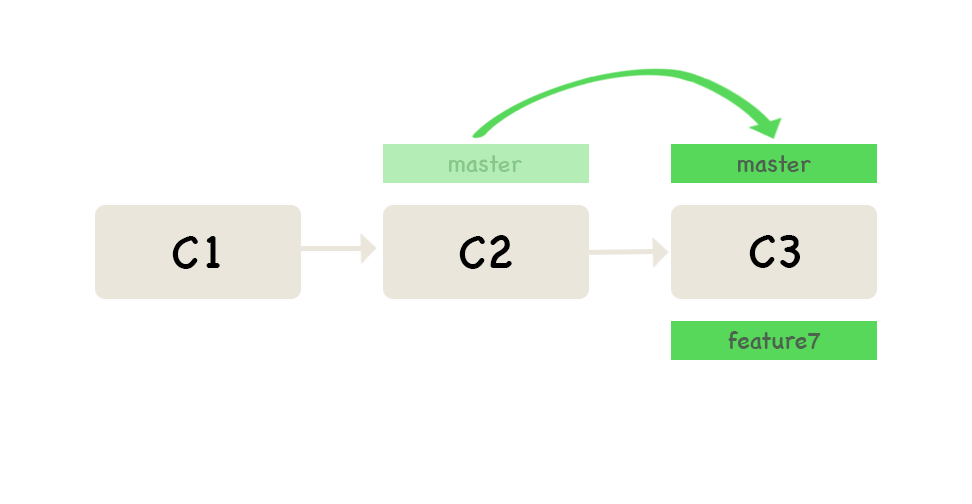
Un Sistema de Control de Versiones generalmente tiene 3 funcionalidades principales: Debe poder almacenar el contenido, realizar un seguimiento de los cambios en dicho contenido y, opcionalmente, distribuir el contenido y confirmar el historial con los colaboradores del proyecto.



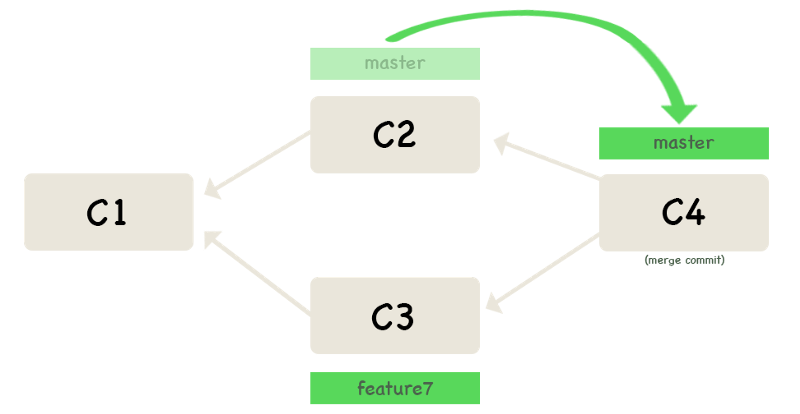
Git utiliza un DAG (grafico aciclico dirigido) , para el almacenamiento de contenido. Ser acíclico significa que no hay manera de pasar del Nodo A al Nodo B y regresar al Nodo A a través de cualquier número de bordes. Un DAG también debe tener ordenamiento topológico.

  
Git también utiliza DAG para el almacenamiento de contenido. Git tiene 3 tipos primitvos principales que utiliza para presentar el contenido de un repositorio: arboles, blobs y confirmaciones. DAG también se utiliza para dar seguimiento al historial de cambios de contenido y fusión que permite mantener una capacidad de bifurcación completa.

Fusionar Estrategias



Al fusionar la rama feauture7 en la rama maestra, Git realiza una combianción de “avance rápido” , que solo es factible cuando el historial de confirmación de la rama “feature7” actual contiene la ultima confirmación (HEAD) de la rama que esta fusionando (maestro).



Git utiliza una estrategia de fusión diferente cuando el compromiso de la rama en la que se encuentra no es un antepasado directo de la rama en la que se está fusionando, lo que significa que su historial de desarrollo divergió. El uso por parte de Git de un sistema de almacenamiento de contenido no lineal y de historial de compromisos le permite fusionar dos ramas de un proyecto a la perfección.

Distribución e Inicialización

Git maneja la distribución de proyectos e historial de proyectos entre los colaboradores que utilizan el modelo distribuido, donde los usuarios pueden trabajar sin conexión y realizar confirmaciones en el repositorio local. Cuando un colaborador esta listo para compartir sus cambios, se puede llevar a un repositorio de acceso publico para que otros colaboradores puedan acceder a ella.