Git & github

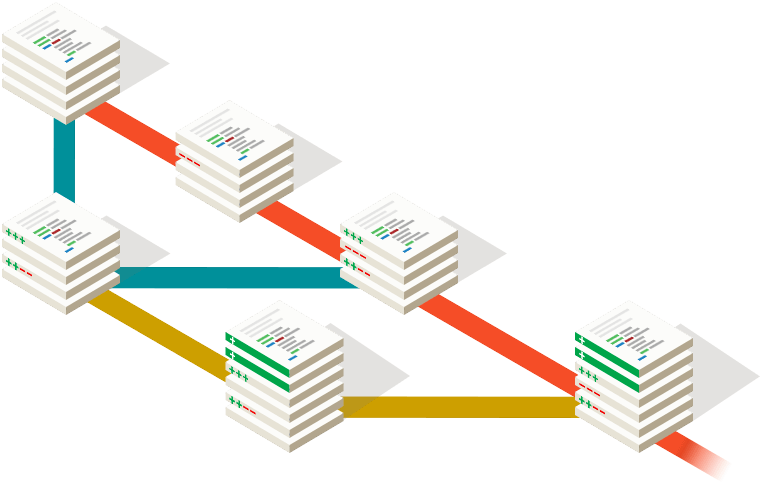
Sistema de control de versiones



Git

Git es [libre y de código abierto](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dgit%26biw%3D652%26bih%3D633&rurl=translate.google.com.mx&sl=en&u=http://git-scm.com/about/free-and-open-source&usg=ALkJrhi3h3jRGnkcSwMnOMTHSvCX3B0G-g) de sistema de control de versiones distribuido diseñado para manejar todo, desde proyectos pequeños hasta muy grandes con velocidad y eficiencia.

Git es [fácil de aprender](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dgit%26biw%3D652%26bih%3D633&rurl=translate.google.com.mx&sl=en&u=http://git-scm.com/documentation&usg=ALkJrhgM5YV6NIBNPYN06VGhZA52zae9LA) y tiene una [pequeña huella con un rendimiento increíblemente rápido](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dgit%26biw%3D652%26bih%3D633&rurl=translate.google.com.mx&sl=en&u=http://git-scm.com/about/small-and-fast&usg=ALkJrhhil529j7ncne6DdXJoPYCXLd2kKQ). Se supera a las herramientas de SCM como Subversión, CVS, Perforce y ClearCase con características como [ramificación barato locales](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dgit%26biw%3D652%26bih%3D633&rurl=translate.google.com.mx&sl=en&u=http://git-scm.com/about/branching-and-merging&usg=ALkJrhgKZGjSBhE395vLatjYYx-PVF-72w) , cómodas [zonas de descanso](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dgit%26biw%3D652%26bih%3D633&rurl=translate.google.com.mx&sl=en&u=http://git-scm.com/about/staging-area&usg=ALkJrhg5jTRVTHViNiZokz6wLFV-ekPVmw) , y [varios flujos de trabajo](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dgit%26biw%3D652%26bih%3D633&rurl=translate.google.com.mx&sl=en&u=http://git-scm.com/about/distributed&usg=ALkJrhjh1W8njhjWQouLOVNnrFnGhIO4Pg).



La última versión estable 1.8.5.3

## Crear ramas y fusionarlas

La función de Git que realmente lo hace destacar, aparte de casi todos los demás SMC por ahí es su modelo de ramificación.

Git permite y anima a tener múltiples ramas locales que pueden ser totalmente independientes entre sí. La creación, fusión y supresión de estas líneas de desarrollo toma segundos.

Esto significa que usted puede hacer cosas como:

* **Cambio de contexto sin fricción**. Crear una rama para probar una idea, cometer un par de veces, cambiar de nuevo a donde usted bifurca desde, aplique un parche, cambiar de nuevo a donde usted está experimentando, y fusionarla
* **Basado en roles Codelines**. Tener una rama que siempre contiene sólo lo que va a la producción, otra que combina el trabajo con las pruebas, y varias más pequeñas para trabajo del día a día.
* **Característica** Based Workflow. Crear nuevas sucursales para cada nueva característica que está trabajando para que la perfección puede cambiar de ida y vuelta entre ellos, a continuación, elimine cada rama cuando esta característica se fusionó con su línea principal.
* **Experimentación desechable**. Crear una rama a experimentar, darse cuenta de que no va a trabajar, y sólo borrarlo - abandonar el trabajo con nadie más verla (incluso si usted ha empujado otras ramas en el íntento).



Cabe destacar que cuando se empuja a un repositorio remoto, usted no tiene que empujar todas sus ramas. Usted puede optar por compartir sólo una de sus ramas, algunos de ellos, o todos ellos. Esto tiende a personas libres de probar nuevas ideas sin preocuparse de tener que planificar cómo y cuándo se van a fusionar en o compartirlo con otros.

Hay maneras de lograr algo de esto con otros sistemas, pero el trabajo en cuestión es mucho más difícil y propenso a errores. Git hace este proceso muy fácil y se cambia la forma en la mayoría de los desarrolladores trabajan cuando se enteran de ello.

## Pequeño y rápido

**Git es rápido**. Con Git, casi todas las operaciones se llevan a cabo a nivel local, lo que supone una gran ventaja de la velocidad en los sistemas centralizados que constantemente tienen que comunicarse con un servidor en alguna parte.

Git fue construido para trabajar en el kernel de Linux, lo que significa que ha tenido que manejar con eficacia grandes repositorios desde el primer día. Git está escrito en C, lo que reduce la sobrecarga de los tiempos de ejecución asociados con lenguajes de alto nivel. La velocidad y el rendimiento ha sido un objetivo de diseño principal de la Git desde el comienzo.

### Puntos de referencia

Vamos a ver cómo las operaciones comunes comparan con Subversion, un sistema de control de versiones centralizado común que es similar a CVS o Perforce. Cuanto más pequeños, más rápido.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio |
| puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio | puntos de referencia de inicio |

Para el ensayo, las grandes instancias de AWS se establecieron en la misma zona de la disponibilidad. Git y SVN se instalaron en las dos máquinas, el repositorio de Ruby fue copiado tanto Git y SVN servidores y operaciones comunes se realizaron en ambos.

En algunos casos, los comandos no coinciden exactamente. En este caso, se intentó que empareja en el mínimo común denominador. Por ejemplo, las pruebas de 'compromisos' incluyen también el momento de empujar para Git, aunque la mayor parte del tiempo que habría en realidad no estar empujando al servidor inmediatamente después de una confirmación donde los dos comandos no se pueden separar en SVN.

Todos estos tiempos son en segundos.

|  |  |  |  |
| --- | --- | --- | --- |
| Operación |  | Git | SVN |
| Comprometerse Archivos (A) | Añadir, cometer y empujar 113 archivos modificados (2164 +, 2259 -) | 0.64 | 2.60 | 4x |
| Commit Imágenes (B) | Añadir, cometer y empujar 1.000 imágenes 1k | 1.53 | 24.70 | 16x |
| Dif. Actual | Dif. 187 archivos cambiados (1664 +, 4859 -) contra la última confirmación | 0.25 | 1.09 | 4x |
| Dif. Recientes | Dif. contra 4 comete espalda (269 changed/3609 +, 6898 -) | 0.25 | 3.99 | 16x |
| Diff Etiquetas | De diferencia de dos etiquetas una contra la otra (v1.9.1.0/v1.9.3.0) | 1.17 | 83.57 | 71x |
| Log (50) | Registro de los últimos 50 commits (19k de salida) | 0.01 | 0.38 | 31x |
| Log (Todos) | Registro de todas las confirmaciones (commits 26056 - 9.4M de la producción) | 0.52 | 169.20 | 325x |
| Iniciar sesión (Archivo) | Log de la historia de un solo archivo (array.c - 483 revoluciones) | 0.60 | 82.84 | 138x |
| Actualización | Tire de la comisión de un escenario (113 archivos cambió, 2164 +, 2259 -) | 0.90 | 2.82 | 3x |
| Culpa | Línea de anotación de un solo archivo (array.c) | 1.91 | 3.04 | 1x |

Tenga en cuenta que este es el mejor de los casos para SVN - un servidor sin carga con una de 80 MB / s de ancho de banda de conexión a la máquina cliente. Casi la totalidad de estos tiempos sería aún peor para SVN si esa conexión era más lento, mientras que muchas de las veces Git no se verían afectados.

Es evidente que en muchas de estas operaciones de control de versiones común, **Git es uno o dos órdenes de magnitud más rápido que SVN**, incluso bajo las condiciones ideales para SVN.

Un lugar donde Git es más lento es en la operación inicial de clones. Aquí, Git es la descarga de toda la historia en lugar de sólo la última versión. Como se observa en las tablas anteriores, no es considerablemente más lento para una operación que sólo se realiza una vez.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operación |  | Git \* | Git | SVN |
| Clon | Clonar y clon sin profundidad (\*) en Git vs pago y envío en el SVN | 21.0 | 107.5 | 14.0 |
| Tamaño (M) | Tamaño de los datos del cliente y el total de los archivos después de la copia / la caja (en M) |  | 181.0 | 132,0 |

Es también interesante observar que el tamaño de los datos en el lado del cliente es muy similar a pesar de que Git también tiene cada versión de cada archivo para toda la historia del proyecto. Esto ilustra qué tan eficiente es en comprimir y almacenar datos en el lado del cliente.

## Repartido

Una de las mejores características de cualquier SCM distribuido, Git incluido, es que se distribuye. Esto significa que en vez de hacer una "comprobación" de la punta actual del código fuente, lo hace un "clon" de todo el repositorio.

### Múltiples copias de seguridad

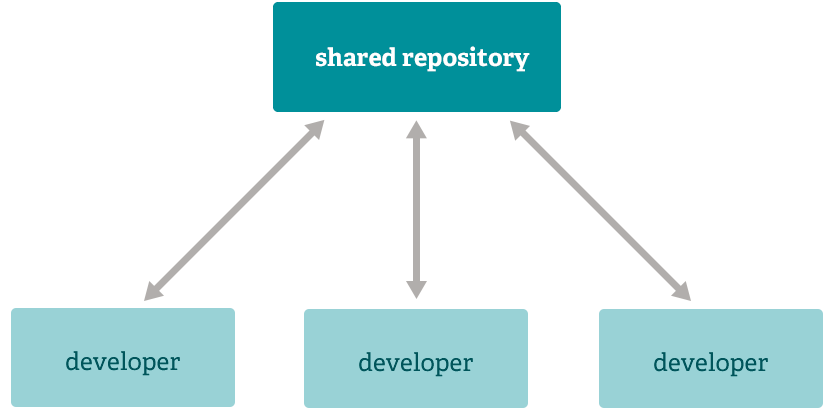
Esto significa que incluso si usted está utilizando un flujo de trabajo centralizado, cada usuario tiene esencialmente una copia de seguridad completa del servidor principal. Cada una de estas copias puede ser empujada hacia arriba para reemplazar el servidor principal en el caso de un accidente o de corrupción. En efecto, no hay ningún punto único de fallo con Git a menos que haya una sola copia del repositorio.

### Cualquier flujo de trabajo

Debido a la naturaleza distribuida y excelente ramificación del sistema de Git, un número casi infinito de los flujos de trabajo se puede implementar con relativa facilidad.

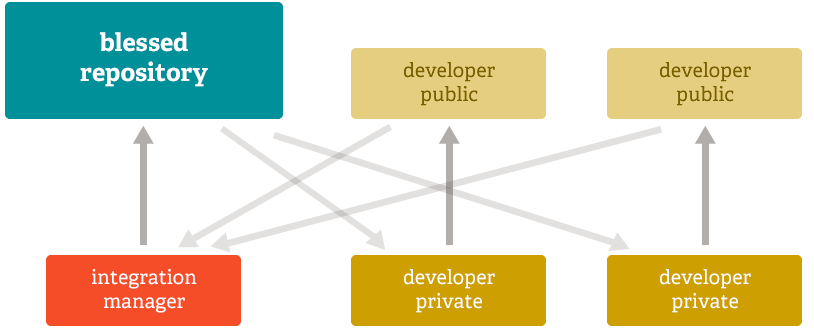
#### Subversión-Estilo de flujo de trabajo

Un flujo de trabajo centralizado es muy común, especialmente de personas en transición de un sistema centralizado. Git no permitirá que usted empuje si alguien ha empujado desde la última vez que se tomó, por lo que un modelo centralizado donde todos los desarrolladores empujan al mismo servidor que funciona muy bien.



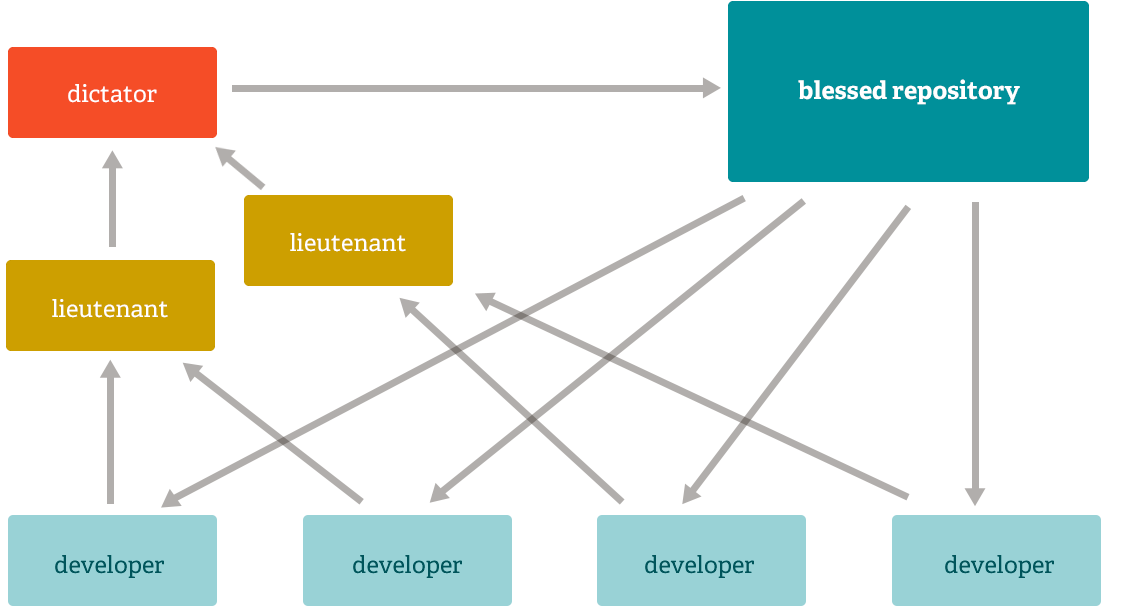
#### Workflow Integration Manager

Otro flujo de trabajo Git común consiste en un gestor de integración - una sola persona que se compromete en el repositorio "bendecida". Un número de desarrolladores a continuación, clonar a partir de ese repositorio, empujar a sus propios repositorios independientes, y pedir al integrador para tirar en sus cambios. Este es el tipo de modelo de desarrollo a menudo visto con el código abierto o repositorios de GitHub.



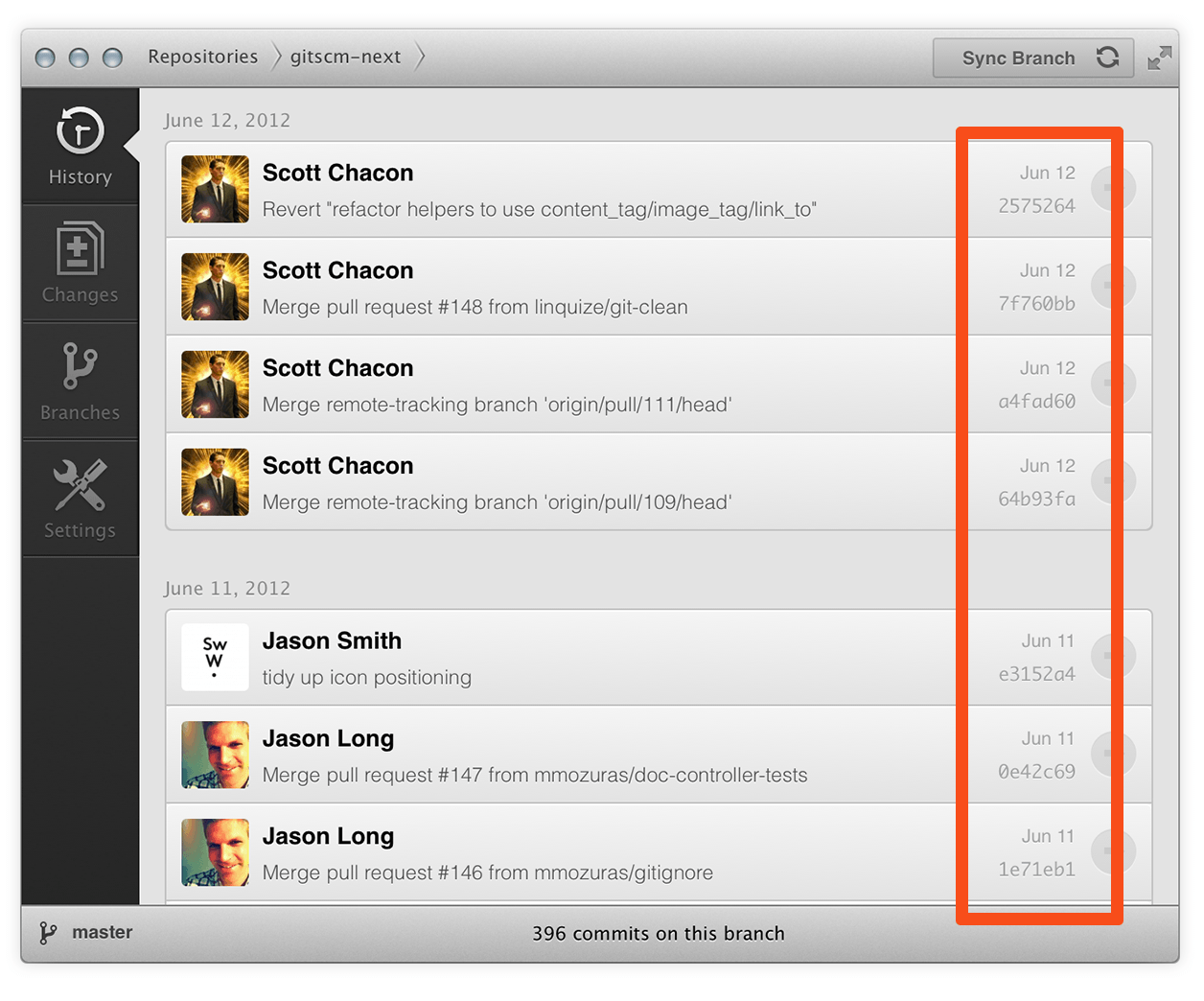
#### Dictador y Tenientes de flujo de trabajo

Para más proyectos masivos, un flujo de trabajo de desarrollo como el del kernel Linux es a menudo eficaz. En este modelo, algunas personas ('tenientes') están a cargo de un subsistema específico del proyecto y se fusionan en todos los cambios relacionados con ese subsistema. Otra integrador (el 'dictador') puede tirar de cambios de sólo sus / sus lugartenientes y luego empujar al repositorio "bendita" que todo el mundo entonces clones de nuevo.



## Datos de Aseguramiento

El modelo de datos que usa Git garantiza la integridad criptográfica de cada pedacito de su proyecto. Cada archivo y comprometerse es una suma de comprobación y se recuperan por su suma de control en caso de control de vuelta. Es imposible sacar nada de Git que no sean los **trozos exactos que usted pone**.



También es imposible cambiar cualquier archivo, fecha, mensaje de entrega o cualquier otro dato en un repositorio Git sin cambiar los identificadores de todo después de ella. Esto significa que si usted tiene un ID de confirmación, puede estar seguro no sólo de que su proyecto es exactamente el mismo que cuando se cometió, pero que nada en su historia fue cambiada.

La mayoría de los sistemas de control de versiones centralizados, no ofrecen esa integridad por defecto.

Área de Ensayo

A diferencia de los otros sistemas, Git tiene algo que se llama la " zona de espera "o" índice. Esta es una zona intermedia donde commits se pueden formatear y revisados ​​antes de completar la cometió.

Una cosa que distingue Git, aparte de otras herramientas es que es posible organizar rápidamente algunos de sus archivos y comprometerlos sin cometer todos los otros archivos modificados en el directorio de trabajo o tener que enumerarlas en la línea de comandos durante la confirmación.



Esto le permite a la etapa de sólo partes de un archivo modificado. Atrás han quedado los días de hacer dos modificaciones lógicamente no relacionadas con un archivo antes de que se dio cuenta de que ha olvidado a cometer uno de ellos. Ahora sólo se puede escenificar el cambio que necesita para cometer la corriente y de la etapa el otro cambio para el próximo commit. Esta característica amplía hasta el mayor número de cambios diferentes en su archivo según sea necesario.

Por supuesto, Git también hace que sea fácil ignorar esta característica si no desea ese tipo de control - sólo tiene que añadir un '-a' a tu comando commit para añadir todos los cambios a todos los archivos en el área de preparación.



Libre y de código abierto

# GitHub

GitHub es un servicio de hosting basado en la web para los proyectos de desarrollo de software que utilizan el Git control de revisión del sistema.

El sitio proporciona las redes sociales funcionalidades tales como alimentaciones, seguidores y la red social gráfica para mostrar cómo los desarrolladores trabajan en sus versiones de un repositorio.

GitHub también opera otros servicios: un pastebin sitio-estilo llamado Gist que proporciona wikis para repositorios individuales y las páginas web que pueden ser editados a través de un repositorio Git, una diapositiva de servicio de alojamiento llama cubierta del altavoz, y una de análisis webde plataforma llamada Medidores.

Limitaciones y restricciones

De acuerdo con los términos del servicio, si el uso de ancho de banda de una cuenta supera considerablemente la media de otros clientes de GitHub, el servicio de alojamiento de archivos de la cuenta puede ser desactivada de inmediato o estrangulado hasta que se reduzca el consumo de ancho de banda. Además, si bien no existe un límite duro, la guía para el tamaño máximo de un repositorio es un gigabyte. Además, no es un cheque de archivos de más de 100 MB en un empujón, si existen este tipo de archivos, el empuje se ser rechazada.

Versiones de software

El 15 de febrero de 2013, GitHub libera Boxen, una fuente abierta Mac herramienta de automatización de medio ambiente. GitHub también tiene su aplicación GUI estándar disponible para su descarga (para de Windows y Mac solamente) directamente desde la página web del servicio, y proporciona una fuente abierta Android app en Google Play