INVESTIGACIÓN GitHub
Willian Roberto Alonzo Pérez
1390-14-16739
Compiladores

Git

(pronunciado "guit"2) es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. 3 Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. 4 Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.

Órdenes básicas

- `git fetch`:

Descarga los cambios realizados en el repositorio remoto.

- `git merge <nombre_rama>`:

Impacta en la rama en la que te encuentras parado, los cambios realizados en la rama “nombre_rama”.

- `git pull`:

Unifica los comandos *fetch* y *merge* en un único comando.

- `git commit -am "<mensaje>"`:

Confirma los cambios realizados. El “mensaje” generalmente se usa para asociar al *commit* una breve descripción de los cambios realizados.

- `git push origin <nombre_rama>`:

Sube la rama “nombre_rama” al servidor remoto.

- `git status`:

Muestra el estado actual de la rama, como los cambios que hay sin commitear.

- `git add <nombre_archivo>`:

Comienza a trackear el archivo “nombre_archivo”.

- `git checkout -b <nombre_rama_nueva>`:

Crea una rama a partir de la que te encuentres parado con el nombre “nombre_rama_nueva”, y luego salta sobre la rama nueva, por lo que quedas parado en esta última.

- `git checkout -t origin/<nombre_rama>`:

Si existe una rama remota de nombre “nombre_rama”, al ejecutar este comando se crea una rama local con el nombre “nombre_rama” para hacer un seguimiento de la rama remota con el mismo nombre.

- `git branch`:

Lista todas las ramas locales.

- `git branch -a`:

Lista todas las ramas locales y remotas.

- `git branch -d <nombre_rama>`:

Elimina la rama local con el nombre “nombre_rama”.

- `git push origin <nombre_rama>`:

Commitea los cambios desde el branch local origin al branch “nombre_rama”.

- `git remote prune origin`:

Actualiza tu repositorio remoto en caso que algún otro desarrollador haya eliminado alguna rama remota.

- `git reset --hard HEAD`:

Elimina los cambios realizados que aún no se hayan hecho *commit*.

- `git revert <hash_commit>`:

Revierte el *commit* realizado, identificado por el “hash_commit”.

Buenas prácticas

Cada desarrollador o equipo de desarrollo puede hacer uso de Git de la forma que le parezca conveniente. Sin embargo una buena práctica es la siguiente:

Se deben utilizar 4 tipos de ramas: Master, Development, Features, y Hotfix.

- Master:

Es la rama principal. Contiene el repositorio que se encuentra publicado en producción, por lo que debe estar siempre estable.

- Development:

Es una rama sacada de master. Es la rama de integración, todas las nuevas funcionalidades se deben integrar en esta rama. Luego que se realice la integración y se corrijan los errores (en caso de haber alguno), es decir que la rama se encuentre estable, se puede hacer un merge de development sobre la rama master.

- Features:

Cada nueva funcionalidad se debe realizar en una rama nueva, específica para esa funcionalidad. Estas se deben sacar de development. Una vez que la funcionalidad esté desarrollada, se hace un merge de la rama sobre development, donde se integrará con las demás funcionalidades.

- Hotfix:

Son bugs que surgen en producción, por lo que se deben arreglar y publicar de forma urgente. Es por ello, que son ramas sacadas de master. Una vez corregido el error, se debe hacer un merge de la rama sobre master. Al final, para que no quede desactualizada, se debe realizar el merge de master sobre development.

GitHub
Es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

GitHub aloja tu repositorio de código y te brinda **herramientas** muy útiles para el **trabajo en equipo**, dentro de un proyecto.
Además de eso, puedes **contribuir a mejorar el software de los demás**. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un **fork** y solicitar **pulls**.
Realizar un **fork** es simplemente **clonar un repositorio ajeno** (genera una copia en tu cuenta), **para eliminar algún bug o modificar cosas de él**. Una vez realizadas tus modificaciones puedes enviar un **pull** al dueño del proyecto. Éste **podrá analizar los cambios** que has realizado fácilmente, **y** si considera interesante tu contribución, **adjuntarlo con el repositorio original**.

DIFERENCIAS ENTRE GIT Y GITHUB

Que es GIT, diferencias entre GIT y GitHub

GIT es el software que rastrea. El sistema de control de versiones. La herramienta que utilizaremos en la terminal.
GitHub es la plataforma de "hosting" de los proyectos. Una comunidad llena de personas que desarrollan y comparten, usando GIT.

Instalando Git
Vamos a empezar a usar un poco de Git. Lo primero es lo primero: tienes que instalarlo. Puedes obtenerlo de varias maneras; las dos principales son instalarlo desde código fuente, o instalar un paquete existente para tu plataforma.

Instalando desde código fuente

Si puedes, en general es útil instalar Git desde código fuente, porque obtendrás la versión más reciente. Cada versión de Git tiende a incluir útiles mejoras en la interfaz de usuario, por lo que utilizar la última versión es a menudo el camino más adecuado si te sientes cómodo compilando software desde código fuente. También ocurre que muchas distribuciones de Linux contienen paquetes muy antiguos; así que a menos que estés en una distribución muy actualizada o estés usando backports, instalar desde código fuente puede ser la mejor opción.

Para instalar Git, necesitas tener las siguientes librerías de las que Git depende: curl, zlib, openssl, expat y libiconv. Por ejemplo, si estás en un sistema que tiene yum (como Fedora) o apt-get (como un sistema basado en Debian), puedes usar estos comandos para instalar todas las dependencias:

```
$ yum install curl-devel expat-devel gettext-devel \
  openssl-devel zlib-devel

$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \
  libz-dev libssl-dev
```

Cuando tengas todas las dependencias necesarias, puedes descargar la versión más reciente de Git desde su página web:

```
http://git-scm.com/download
```

Luego compila e instala:

```
$ tar -zxf git-1.6.0.5.tar.gz
$ cd git-1.6.0.5
$ make prefix=/usr/local all
$ sudo make prefix=/usr/local install
```

Una vez hecho esto, también puedes obtener Git, a través del propio Git, para futuras actualizaciones:

```
$ git clone git://git.kernel.org/pub/scm/git/git.git
```

Instalando en Linux

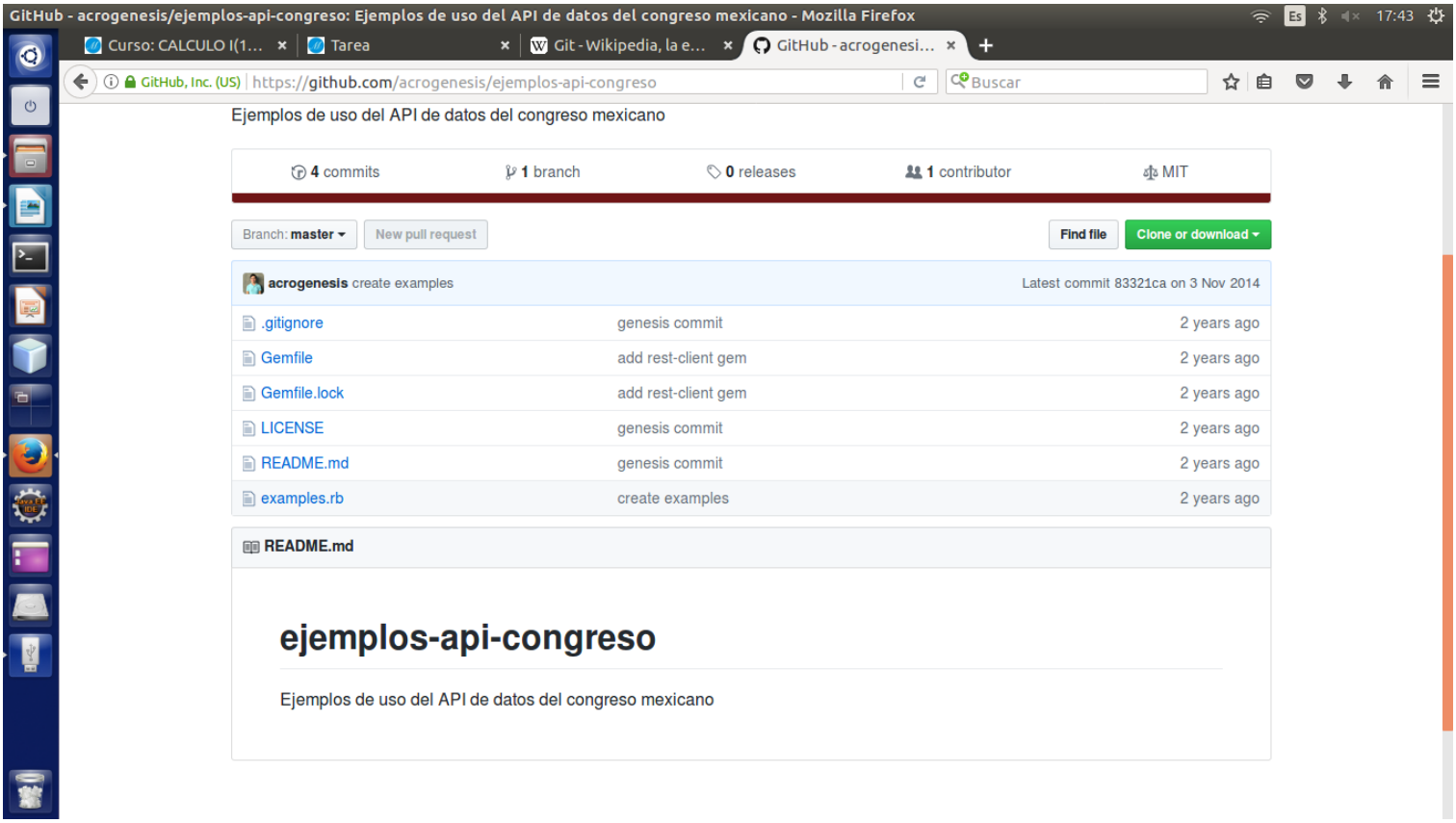
Si quieres instalar Git en Linux a través de un instalador binario, en general puedes hacerlo a través de la herramienta básica de gestión de paquetes que trae tu distribución. Si estás en Fedora, puedes usar yum:

```
$ yum install git-core
```

O si estás en una distribución basada en Debian como Ubuntu, prueba con apt-get:

```
$ apt-get install git
```

Ejemplo


GitHub - acrogenesis/ejemplos-api-congreso: Ejemplos de uso del API de datos del congreso mexicano - Mozilla Firefox
17:43
Curso: CALCULO I(1...
Tarea
Git - Wikipedia, la e...
GitHub - acrogenesi...
GitHub, Inc. (US) | https://github.com/acrogenesis/ejemplos-api-congreso
Buscar
Ejemplos de uso del API de datos del congreso mexicano
4 commits
1 branch
0 releases
1 contributor
MIT
Branch: master
New pull request
Find file
Clone or download
acrogenesis create examples
Latest commit 83321ca on 3 Nov 2014
.gitignore genesis commit 2 years ago
Gemfile add rest-client gem 2 years ago
Gemfile.lock add rest-client gem 2 years ago
LICENSE genesis commit 2 years ago
README.md genesis commit 2 years ago
examples.rb create examples 2 years ago
README.md
ejemplos-api-congreso
Ejemplos de uso del API de datos del congreso mexicano