**git - la guía sencilla**

una guía sencilla para comenzar con git. sin complicaciones ;)

Descarga git para windows <https://gitforwindows.org/>

### [Instalando en Windows](https://git-scm.com/book/es/v1/Empezando-Instalando-Git#Instalando-en-Windows)

Instalar Git en Windows es muy fácil. El proyecto msysGit tiene uno de los procesos de instalación más sencillos. Simplemente descarga el archivo exe del instalador desde la página de GitHub, y ejecútalo:

http://msysgit.github.com/

Una vez instalado, tendrás tanto la versión de línea de comandos (incluido un cliente SSH que nos será útil más adelante) como la interfaz gráfica de usuario estándar.

Nota para el uso en Windows: Se debería usar Git con la shell provista por msysGit (estilo Unix), lo cual permite usar las complejas líneas de comandos de este libro. Si por cualquier razón se necesitara usar la shell nativa de Windows, la consola de línea de comandos, se han de usar las comillas dobles en vez de las simples (para parámetros que contengan espacios) y se deben entrecomillar los parámetros terminándolos con el acento circunflejo (^) si están al final de la línea, ya que en Windows es uno de los símbolos de continuación.

Instalando ConEmu

ConEmu es un emulador de terminal de pestañas gratuito y de código abierto para Windows. ConEmu presenta múltiples consolas y aplicaciones de GUI simples como una ventana de GUI personalizable con pestañas y una barra de estado

[https://conemu.github.io/](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwjavLu-w4XiAhVlMn0KHQtgCUAQFjAAegQIBxAC&url=https%3A%2F%2Fconemu.github.io%2F&usg=AOvVaw0Skb9BV7RBXS1npIOfraWQ)

### [Instalando en Linux](https://git-scm.com/book/es/v1/Empezando-Instalando-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

## Configurando Git por primera vez

Ahora que tienes Git en tu sistema, querrás hacer algunas cosas para personalizar tu entorno de Git. Sólo es necesario hacer estas cosas una vez; se mantendrán entre actualizaciones. También puedes cambiarlas en cualquier momento volviendo a ejecutar los comandos correspondientes.

Git trae una herramienta llamada git config que te permite obtener y establecer variables de configuración, que controlan el aspecto y funcionamiento de Git. Estas variables pueden almacenarse en tres sitios distintos:

* Archivo /etc/gitconfig: Contiene valores para todos los usuarios del sistema y todos sus repositorios. Si pasas la opción --system a git config, lee y escribe específicamente en este archivo.
* Archivo ~/.gitconfig file: Específico a tu usuario. Puedes hacer que Git lea y escriba específicamente en este archivo pasando la opción --global.
* Archivo config en el directorio de Git (es decir, .git/config) del repositorio que estés utilizando actualmente: Específico a ese repositorio. Cada nivel sobrescribe los valores del nivel anterior, por lo que los valores de .git/config tienen preferencia sobre los de /etc/gitconfig.

En sistemas Windows, Git busca el archivo .gitconfig en el directorio $HOME (%USERPROFILE% in Windows’ environment), que es C:\Documents and Settings\$USER para la mayoría de usuarios, dependiendo de la versión ($USER es %USERNAME% en el entorno Windows). También busca en el directorio /etc/gitconfig, aunque esta ruta es relativa a la raíz MSys, que es donde quiera que decidieses instalar Git en tu sistema Windows cuando ejecutaste el instalador.

### [Tu identidad](https://git-scm.com/book/es/v1/Empezando-Configurando-Git-por-primera-vez#Tu-identidad)

Lo primero que deberías hacer cuando instalas Git es establecer tu nombre de usuario y dirección de correo electrónico. Esto es importante porque las confirmaciones de cambios (commits) en Git usan esta información, y es introducida de manera inmutable en los commits que envías:

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

De nuevo, sólo necesitas hacer esto una vez si especificas la opción --global, ya que Git siempre usará esta información para todo lo que hagas en ese sistema. Si quieres sobrescribir esta información con otro nombre o dirección de correo para proyectos específicos, puedes ejecutar el comando sin la opción --global cuando estés en ese proyecto.

### [Tu editor](https://git-scm.com/book/es/v1/Empezando-Configurando-Git-por-primera-vez#Tu-editor)

Ahora que tu identidad está configurada, puedes elegir el editor de texto por defecto que se utilizará cuando Git necesite que introduzcas un mensaje. Si no indicas nada, Git usa el editor por defecto de tu sistema, que generalmente es Vi o Vim. Si quieres usar otro editor de texto, como Emacs, puedes hacer lo siguiente:

$ git config --global core.editor emacs

### [Tu herramienta de diferencias](https://git-scm.com/book/es/v1/Empezando-Configurando-Git-por-primera-vez#Tu-herramienta-de-diferencias)

Otra opción útil que puede que quieras configurar es la herramienta de diferencias por defecto, usada para resolver conflictos de unión (merge). Digamos que quieres usar vimdiff:

$ git config --global merge.tool vimdiff

Git acepta kdiff3, tkdiff, meld, xxdiff, emerge, vimdiff, gvimdiff, ecmerge, y opendiff como herramientas válidas. También puedes configurar la herramienta que tú quieras; véase el Capítulo 7 para más información sobre cómo hacerlo.

### [Comprobando tu configuración](https://git-scm.com/book/es/v1/Empezando-Configurando-Git-por-primera-vez#Comprobando-tu-configuración)

Si quieres comprobar tu configuración, puedes usar el comando git config --list para listar todas las propiedades que Git ha configurado:

$ git config --list

user.name=Scott Chacon

user.email=schacon@gmail.com

color.status=auto

color.branch=auto

color.interactive=auto

color.diff=auto

...

Puede que veas claves repetidas, porque Git lee la misma clave de distintos archivos (/etc/gitconfig y ~/.gitconfig, por ejemplo). En ese caso, Git usa el último valor para cada clave única que ve.

También puedes comprobar qué valor cree Git que tiene una clave específica ejecutando git config {clave}:

$ git config user.name

Scott Chacon

## crea un repositorio nuevo

Crea un directorio nuevo, ábrelo y ejecuta git init para crear un nuevo repositorio de git.

## hacer checkout a un repositorio

Crea una copia local del repositorio ejecutando git clone /path/to/repository Si utilizas un servidor remoto, ejecuta git clone username@host:/path/to/repository

## flujo de trabajo

Tu repositorio local esta compuesto por tres "árboles" administrados por git. El primero es tu Directorio de trabajo que contiene los archivos, el segundo es el Index que actua como una zona intermedia, y el último es el HEAD que apunta al último commit realizado.



## add & commit

Puedes registrar cambios (añadirlos al **Index**) usando git add <filename> git add . Este es el primer paso en el flujo de trabajo básico. Para hacer commit a estos cambios usa git commit -m "Commit message" Ahora el archivo esta incluído en el **HEAD**, pero aún no en tu repositorio remoto.

## envío de cambios

Tus cambios están ahora en el **HEAD** de tu copia local. Para enviar estos cambios a tu repositorio remoto ejecuta git push origin master Reemplaza *master* por la rama a la que quieres enviar tus cambios. Si no has clonado un repositorio ya existente y quieres conectar tu repositorio local a un repositorio remoto, usa git remote add origin <server> Ahora podrás subir tus cambios al repositorio remoto seleccionado.

## ramas

Las ramas son utilizadas para desarrollar funcionalidades aisladas unas de otras. La rama *master* es la rama "por defecto" cuando creas un repositorio. Crea nuevas ramas durante el desarrollo y fusiónalas a la rama principal cuando termines.



Crea una nueva rama llamada "feature\_x" y cámbiate a ella usando git checkout -b feature\_x vuelve a la rama principal git checkout master y borra la rama git branch -d feature\_x Una rama nueva *no estará disponible para los demás* a menos que subas (push) la rama a tu repositorio remoto git push origin <branch>

## actualiza & fusiona

Para actualizar tu repositorio local al commit más nuevo, ejecuta git pull en tu directorio de trabajo para *bajar* y *fusionar* los cambios remotos. Para fusionar otra rama a tu rama activa (por ejemplo master), utiliza git merge <branch> en ambos casos git intentará fusionar automáticamente los cambios. Desafortunadamente, no siempre será posible y se podrán producir *conflictos*. Tú eres responsable de fusionar esos *conflictos* manualmente al editar los archivos mostrados por git. Después de modificarlos, necesitas marcarlos como fusionados con git add <filename> Antes de fusionar los cambios, puedes revisarlos usando git diff <source\_branch> <target\_branch>

## etiquetas

Se recomienda crear etiquetas para cada nueva versión publicada de un software. Este concepto no es nuevo, ya que estaba disponible en SVN. Puedes crear una nueva etiqueta llamada *1.0.0* ejecutando git tag 1.0.0 1b2e1d63ff *1b2e1d63ff* se refiere a los 10 caracteres del commit id al cual quieres referirte con tu etiqueta. Puedes obtener el commit id con git log también puedes usar menos caracteres que el commit id, pero debe ser un valor único.

## reemplaza cambios locales

En caso de que hagas algo mal (lo que seguramente nunca suceda ;) puedes reemplazar cambios locales usando el comando git checkout -- <filename> Este comando reemplaza los cambios en tu directorio de trabajo con el último contenido de HEAD. Los cambios que ya han sido agregados al Index, así como también los nuevos archivos, se mantendrán sin cambio.

Por otro lado, si quieres deshacer todos los cambios locales y commits, puedes traer la última versión del servidor y apuntar a tu copia local principal de esta forma git fetch origin git reset --hard origin/master

## datos útiles

Interfaz gráfica por defecto gitk

Colores especiales para la consola git config color.ui true Mostrar sólo una línea por cada commit en la traza git config format.pretty oneline Agregar archivos de forma interactiva git add -i

## enlaces & recursos

### guías

* [Git Community Book](http://book.git-scm.com/)
* [Pro Git](http://progit.org/book/)
* [Think like a git](http://think-like-a-git.net/)
* [GitHub Help](http://help.github.com/)
* [A Visual Git Guide](http://marklodato.github.com/visual-git-guide/index-en.html)