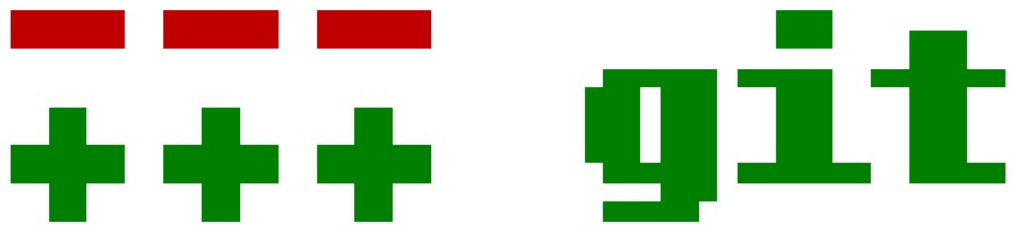



Tutorial de



Autor: Sócrates Díaz S.

 @socrates_xD

Introducción

En este pequeño tutorial explicaré qué es git, y cómo utilizarlo.

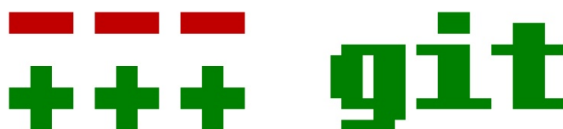
Git es un sistema de control de versiones (RCS). Es software libre y open source.

Un sistema de control de versiones es un software para gestionar el historial de versiones de un proyecto. **Una versión es el estado en el que se encuentra un proyecto en un momento dado.**



Las ventajas de un RCS son copias de seguridad, deshacer cambios, historial de cambios, manejar diferentes versiones del proyecto, entre otros. Un proyecto puede gestionarse de manera individual o en grupo.

Git fue creado por Linus Torvalds en 2002. Después de ciertos problemas con un RCS creado por la compañía BitKeeper. BitKeeper no era lo suficientemente rápido para las necesidades del kernel como proyecto, muchos parches complicados tomaban hasta 30 segundos en aplicarse, sincronizar las versiones entre Linus Torvalds y Andrew Morton (mantenedor de la rama 2.6) demoraba 2 horas; se necesitaba algo veloz y distribuido, entonces Torvalds decidió crear un RCS llamado Git.



¡Comencemos!

Instalación

Lo primero que debemos hacer es instalar git. A continuación pongo las instrucciones para varias distros.

- Ubuntu/Debian/Linux Mint/(cualquier derivado de debian o ubuntu):

```
sudo apt-get install git git-core
```

- Fedora/RedHat (y derivados):

```
sudo yum install git
```

- Archlinux/Chakra:

```
sudo pacman -S git
```

Alguna otra pues puede ser compilada desde el código fuente. Si usas Windows o Mac puedes ir a la página oficial de git, descargarlo e instalarlo (tales pasos son muy fáciles de realizar por lo que no pondré detalles aquí sobre ello).

La página de descargas es <http://git-scm.com/download/> por lo que puedes visitar y obtener información sobre otras distros.

Estaré utilizando Github como página para trabajar con los repositorios: www.github.com

Utilizar una interfaz gráfica no es mala idea para trabajar con git, aunque no es recomendable si estás aprendiendo a usarlo. Las interfaces gráficas las puedes bajar desde aquí <http://git-scm.com/downloads/guis>. Aunque git trae unas interfaces gráficas por defecto: git-gui (para hacer committing) y gitk (para navegar entre los archivos). Si usas Windows o Mac te recomiendo Github for Windows, o Github for Mac, ya que estaremos usando Github como nuestra página para gestionar repositorios.

Github

Es una plataforma para el desarrollo colaborativo de software que te permite almacenar tus proyectos en un repositorio (almacén personal de software), y poder colaborar con otros.

Entre sus características:



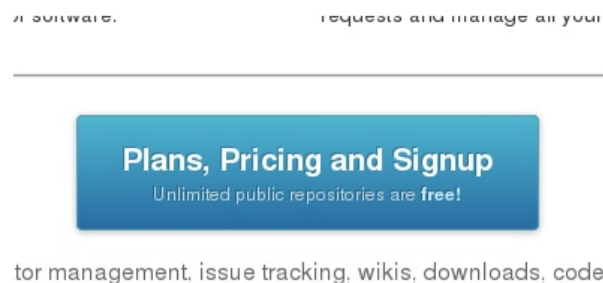
github
SOCIAL CODING

- **Wiki** para cada proyecto.
- **Página web para cada proyecto.**
- **Gráfico** para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto.
- Funcionalidades como si se tratase de una **red social**, como por ejemplo: seguidores.

Creando una cuenta en Github

(Si tienes una cuenta en Github y/o un repositorio no tienes que realizar los siguientes pasos)

1. Entrar a www.github.com
2. Le das al botón azul grande del centro de la página:



3. Entonces, a no ser que puedas pagar por un repositorio privado, haz clic en crear una cuenta gratis (Create a free account):



4. Introduces los datos necesarios y le das al botón "crear una cuenta" (Create an account).

Create your free personal account

Username

Email Address

We promise we won't share your email with anyone.

Password

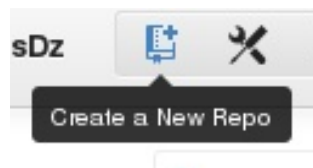
Must contain one lowercase letter, one number, and be at least 7 characters long.

Confirm Password

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

Create an account

5. En la pantalla siguiente busca en la parte superior derecha un botón que dice "Create a New Repo", el cual es de esta forma:



6. Pon un nombre para el repositorio, en este caso, le puse el nombre "myrepo" pero puede ser el que quieras:

Repository name

7. Pon una descripción (este paso es opcional), y lo conviertes en público:

Description (optional)

☒ **Public** Anyone can see this repository. You choose who can contribute to this repository.

☐ **Private** You choose who can see and commit to this repository.

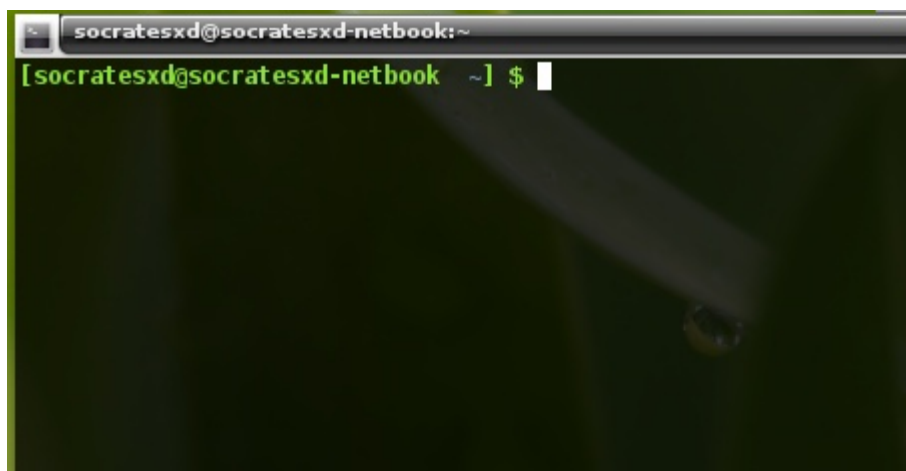
8. Creas el repositorio:

Create repository

Ahora deberías ver los archivos que hay en tu repositorio. Sin embargo no hay nada, porque no hemos subido ningún archivo. En la página hay instrucciones sobre cómo crear tu primer archivo y subirlo. Sigamos esas instrucciones desde nuestra terminal :)

Tengo que decir que estaré usando Linux durante todo este tutorial, usuarios de otros sistemas operativos tendrán que buscar otros métodos (o esperar otra versión de este tutorial). Sin embargo los **comandos de git** son los mismos para cualquier sistema operativo.

1º : Abres una terminal.



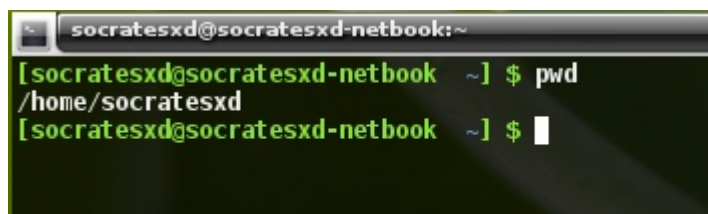
Lo más probable es que tu terminal sea diferente a la mía. Incluso es posible que no sepas qué significa lo que ves actualmente en la terminal, por lo que haré una breve explicación sobre ello. Eso que ves se llama **prompt**, y se compone de la siguiente forma:

```
[usuario@hostname directorio_de_trabajo_actual] $
```

Creo que no queda mucho qué explicar después de eso último, pero lo haré de todas formas.

```
usuario = "socratesxd"  
hostname = "socratesxd-netbook"  
directorio_de_trabajo_actual = "~"
```

El símbolo tilde (~) es un comodín que significa el directorio **/home** del usuario, es posible que lo conozcas como "Carpeta personal", para explicarlo mejor:



pwd es un comando que imprime el directorio de trabajo actual, de ahí su nombre: "print working directory"

Aquí vemos que el directorio de trabajo actual es mi "**Carpeta personal**" o mi directorio **/home**. Y eso es exactamente lo que significa la tilde (~). El símbolo "\$" significa que eres un usuario sin privilegios de administrador, y eso está bien, porque a lo largo de este tutorial no necesitamos dichos privilegios. De hecho, uno nunca debe tener dichos privilegios excepto para tareas administrativas, dichos privilegios deberán ser revocados inmediatamente terminen dichas tareas. Pero como dije, no necesitamos privilegios de administrador para usar git.

2º Clonas o creas la carpeta del repositorio

Estando en la terminal, tienes dos opciones, crear la carpeta o clonarla (de todas formas el resultado es una carpeta). Pero te enseñaré ambas formas:

- Creando una carpeta (y accediendo a ella):

Consiste en ejecutar dos comandos: `mkdir` y `cd`

```
socratesxd@socratesxd-netbook:~
[socratesxd@socratesxd-netbook ~] $ mkdir myrepo
[socratesxd@socratesxd-netbook ~] $
```

En este caso sólo escribí `mkdir myrepo` y presioné la tecla Enter (o Return). `myrepo` es el nombre del repositorio que creamos. `mkdir` significa "make directory"

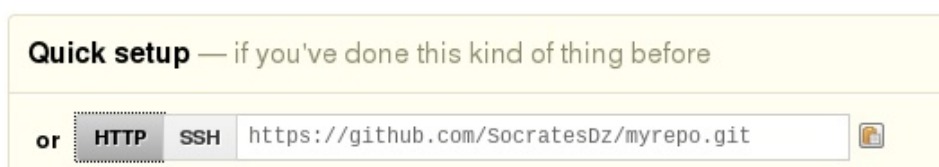
Moviéndome a `myrepo`:

```
socratesxd@socratesxd-netbook:~/myrepo
[socratesxd@socratesxd-netbook ~] $ mkdir myrepo
[socratesxd@socratesxd-netbook ~] $ cd myrepo
[socratesxd@socratesxd-netbook ~/myrepo] $
```

En este caso ejecuté `cd myrepo` para moverme al directorio que había creado llamado `myrepo`. Nota cómo cambia el **prompt** y pone `~/myrepo` en el directorio de trabajo actual, eso significa que estamos dentro de una carpeta llamada `myrepo` que está dentro de mi "**Carpeta personal**" o directorio `/home`. `cd` significa "**change directory**".

- Clonando el repositorio:

Buscamos la dirección del repositorio que hemos creado, la cual suele estar en la barra de direcciones del navegador si estamos en la página de nuestro repositorio. En este caso, en el repositorio que creé para este tutorial:



En este caso el repositorio que usaremos es <https://github.com/SocratesDz/myrepo.git>. Para clonarlo usamos este comando: `git clone https://github.com/SocratesDz/myrepo.git`

```
socratesxd@socratesxd-netbook:~/myrepo
[socratesxd@socratesxd-netbook ~] $ git clone https://github.com/SocratesDz/myrepo.git
Cloning into 'myrepo'...
warning: You appear to have cloned an empty repository.
[socratesxd@socratesxd-netbook ~] $ cd myrepo
[socratesxd@socratesxd-netbook ~/myrepo] $
```

Nos pone un mensaje de advertencia, diciéndonos que hemos clonado un repositorio vacío (obviamente está vacío, pues no tenemos nada en nuestro repositorio). Inmediatamente me muevo a mi nuevo directorio creado llamado `myrepo`. Que conste que también podíamos usar la dirección ssh del repositorio: `git@github.com:SocratesDz/myrepo.git`.

3º Inicializamos git

Primero tenemos que decirle a git que tome nota de los cambios que vayamos haciendo dentro del directorio del repositorio mediante el comando `git init`:



```
socratesxd@socratesxd-netbook: ~/myrepo
[socratesxd@socratesxd-netbook ~] $ git init
Reinitialized existing Git repository in /home/socratesxd/myrepo/.git/
[socratesxd@socratesxd-netbook ~] $
```

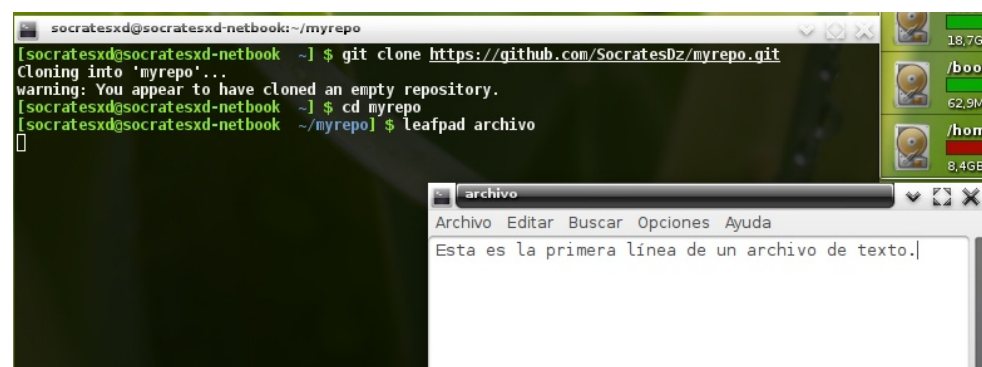
Dependiendo de si creaste el directorio o si lo clonaste el mensaje suele ser ligeramente diferente de este (en este caso lo cloné).

Este comando crea un directorio que se llama `.git` el cual es donde se guarda toda la magia de git :)

4º Creamos un archivo

Vamos a crear un archivo de texto. Para esto simplemente abre un editor, escribe algunas líneas (las que quieras) y guarda el archivo en la carpeta de tu repositorio.

En mi caso usaré un editor llamado leafpad, pero tú puedes usar el que quieras (kate, gedit, nano, vim, emacs).



```
socratesxd@socratesxd-netbook: ~/myrepo
[socratesxd@socratesxd-netbook ~] $ git clone https://github.com/SocratesDz/myrepo.git
Cloning into 'myrepo'...
warning: You appear to have cloned an empty repository.
[socratesxd@socratesxd-netbook ~] $ cd myrepo
[socratesxd@socratesxd-netbook ~] $ leafpad archivo
[]
```

archivo

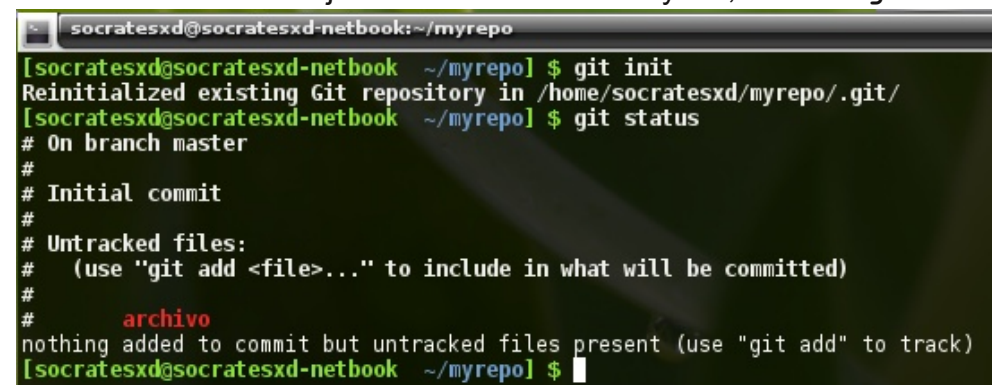
Archivo Editar Buscar Opciones Ayuda

Esta es la primera línea de un archivo de texto.

Abrí leafpad con un nuevo documento nombrado **archivo**

5º Lo agregamos a la lista de cambios

Primero deberíamos ejecutar un comando muy útil, llamado `git status`.



```
socratesxd@socratesxd-netbook: ~/myrepo
[socratesxd@socratesxd-netbook ~] $ git init
Reinitialized existing Git repository in /home/socratesxd/myrepo/.git/
[socratesxd@socratesxd-netbook ~] $ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   archivo
nothing added to commit but untracked files present (use "git add" to track)
[socratesxd@socratesxd-netbook ~] $
```

La salida de este comando es auto explicativa, en el branch master, commit inicial, hay archivos que no han sido agregados (los que están en rojo). E incluso nos sugiere que los agreguemos con `git add`.

Ahora lo agregamos mediante `git add`:

```
[socratesxd@socratesxd-netbook ~/myrepo] $ git add archivo
[socratesxd@socratesxd-netbook ~/myrepo] $
```

Y mediante un `git status` veremos los archivos que agregamos:

```
[socratesxd@socratesxd-netbook ~/myrepo] $ git add archivo
[socratesxd@socratesxd-netbook ~/myrepo] $ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   archivo
#
[socratesxd@socratesxd-netbook ~/myrepo] $
```

Vemos los archivos que agregamos, en este caso nos dice que "archivo" es un nuevo archivo en el repositorio. Y nos sugiere un comando para eliminar ese archivo de la lista de agregados `git rm --cached <file>`

6º Haz un commit

Un commit es un comentario que indica los cambios que hemos hecho a los archivos. Cada vez que vayamos a subir archivos o modificaciones, debemos hacer un commit indicando los cambios que hayamos hecho mediante el comando `git commit`.

```
socratesxd@socratesxd-netbook:~/myrepo
[socratesxd@socratesxd-netbook ~/myrepo] $ git commit -m "Subiendo mi primer archivo"
[master (root-commit) 150cd46] Subiendo mi primer archivo
1 file changed, 1 insertion(+)
create mode 100644 archivo
[socratesxd@socratesxd-netbook ~/myrepo] $
```

En este caso usamos `git commit -m <mensaje>`. Donde el mensaje es "Subiendo mi primer archivo". La opción `-m` se utiliza para que escribir el mensaje en la misma línea, porque si usáramos simplemente `git commit` entonces aparecería ante nosotros un editor en la terminal (llamado vi) permitiéndonos escribir un commit más extenso.

7º Sube tus archivos (push)

Ahora es tiempo de subir el archivo, pero necesitamos un paso más. que a la larga nos hará las cosas más cómodas. `git remote add <nombre> <repositorio>`. Este comando sirve para ponerle un nombre a ese repositorio y para que podamos llamarlo por ese nombre en vez de poner la dirección completa del repositorio. Es una costumbre usar el nombre de origen. **Sin embargo, si clonaron el repositorio (en vez de haber creado la carpeta), éste por defecto ya tiene el nombre de origen, si ejecutamos el comando nos va a da error, pero esto es normal.**

```
socratesxd@socratesxd-netbook:~/myrepo
[socratesxd@socratesxd-netbook ~/myrepo] $ git remote add origin https://github.com/SocratesDz/myrepo.git
[socratesxd@socratesxd-netbook ~/myrepo] $
```

Ya realizado el paso anterior, entonces podemos subir nuestro archivo. Utilizamos el comando `git push <nombre> <branch>`. El nombre es el nombre del repositorio que definimos anteriormente (origin) aunque en vez del nombre puedes poner también la dirección del repositorio. Branch es la rama actual del proyecto, el branch por defecto es **master**.

En un repositorio es posible trabajar con varias versiones del mismo trabajo, dividiéndolo ya sea para hacer pruebas o experimentar con características que no quieres que sean parte del trabajo principal.

A estas divisiones o ramificaciones del proyecto se les llama branch, y todo repositorio tiene al menos un branch, el cual por defecto se llama master.

```
[socratesxd@socratesxd-netbook ~/myrepo] $ git push origin master
Username for 'https://github.com': SocratesDz
Password for 'https://SocratesDz@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 273 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/SocratesDz/myrepo.git
 * [new branch]      master -> master
[socratesxd@socratesxd-netbook ~/myrepo] $
```

Nos va a pedir nuestro nombre de usuario (o email) y nuestra contraseña, cuando escribamos nuestra contraseña no nos va a mostrar nada, y eso es normal, es una medida de seguridad, para que no se sepa siquiera cuántos caracteres tiene la contraseña ;)

Ahora si vamos a nuestro repositorio en la página de Github podremos comprobar que el archivo que subimos está ahí:

branch: master
Files
Commits
Branches 1

Latest commit to the **master** branch

Subiendo mi primer archivo

SocratesDz authored 9 hours ago

myrepo /

name	age	message
archivo	9 hours ago	Subiendo mi primer archivo [SocratesDz]

We recommend adding a README to this repository. Visit [github/markup](https://github.com/markup) for details on what formats