

GitTutorial

Edgardo Bonzi

March 16, 2022

Con que comprendas y puedas hacer hasta el punto 1.5, es más que suficiente!

1 Git para principiantes

Este notebooks está inspirado en:

<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/tutorial-de-git/>

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

1.1 Git: tutorial básico del Sistema de Control de Versiones

Version Control System (VCS)

No solo las empresas desarrollan proyectos de software de forma colaborativa: también en el sector del código abierto, varios cientos o incluso miles de voluntarios y colaboradores pueden participar en la creación, el mantenimiento, la optimización o la edición de un programa, dependiendo del tamaño del proyecto. Sería prácticamente imposible llevar a cabo estas tareas sin un buen sistema para registrar y controlar los numerosos cambios realizados por todos los desarrolladores.

Una de las soluciones más populares en este sentido es Git, un programa de licencia libre que puedes aprender a manejar rápidamente y utilizar de forma totalmente gratuita. En nuestro tutorial, te enseñamos todos los conceptos básicos para que seas capaz de dar tus primeros pasos con este sistema de control de versiones.

1.2 ¿Qué es Git?

Git es un sistema de control de versiones (VCS) desarrollado en 2005 por Linus Thorvalds, el creador de Linux, y publicado bajo la licencia de software libre GPLv2 de GNU.

La particularidad de esta herramienta es que, aunque guarda un repositorio central para cada proyecto, todos los participantes descargan una copia local del mismo en su propio dispositivo.

Cada una de estas copias constituye una copia completa de todo el contenido del repositorio, por lo que no es necesario estar conectado a la red para trabajar. Además, estos archivos sirven como copia de seguridad en caso de que el repositorio principal falle o resulte dañado. Los cambios en los archivos pueden intercambiarse con todos los demás participantes del proyecto en cualquier momento y, si corresponde, añadirse al repositorio. Consejo

Una de las alternativas más conocidas a **Git** es **Subversion**, una herramienta también de código abierto y más conocida como SVN, que se basa en un sistema de gestión central, contrariamente a Git.

1.3 Cómo instalar Git en el dispositivo

<https://git-scm.com/download/linux>

Si quieres empezar a utilizar Git como programa de control de versiones, en primer lugar, debes familiarizarte con el propio software y su interfaz de usuario.

Git está disponible para Windows, Unix/Linux y macOS, con diferentes versiones que presentan pequeñas diferencias de uso.

Después de llevar a cabo la instalación estándar correspondiente, puedes controlar el programa con el símbolo del sistema o una interfaz gráfica de usuario en cualquiera de los sistemas operativos. Nota

Para poder utilizar los comandos explicados en este tutorial de Git, los usuarios de Windows deben ejecutarlo a través de Git Bash, un shell de estilo Unix que emula la línea de comandos de Git y está incluido en el paquete de instalación. Si lo prefieres, también es posible controlar el software con el símbolo del sistema o la terminal de Windows, aunque debes tener en cuenta que la estructura de los parámetros de los comandos difiere (por ejemplo, se emplean signos de ídem en lugar de comillas).

1.3.1 Descargar para Linux/Ubuntu

Es más fácil instalar Git en Linux utilizando el administrador de paquetes preferido de su distribución de Linux.

Si prefiere compilar desde el código fuente, puede encontrar tarballs en kernel.org. La última versión es la 2.35.1. Debian/Ubuntu

Para obtener la última versión estable para su lanzamiento de Debian/Ubuntu

```
[~] $ sudo apt update
```

```
[~] $ apt-get install git
```

Luego revisa la versión y configuralo

```
$ git version
```

```
$ git config --global user.name "MV Thundergit"
```

```
$ git config --global user.email "mv.thundergit@mail.com"
```

1.4 Git: tutorial para empezar a utilizarlo paso a paso

Una vez instalado Git, ya puedes utilizar el sistema para controlar las versiones de tus proyectos. Como con cualquier otro programa, el primer paso es comprender las funciones y comandos básicos para sacarle el máximo partido a la herramienta. En nuestro completo tutorial de Git, te explicamos los comandos más importantes para configurar y utilizar Git mediante la interfaz de línea de comandos y que puedas crear y administrar fácilmente tu propio repositorio.

1.5 Crear o clonar un repositorio Git

El repositorio Git es el directorio central de un proyecto y, por lo tanto, también el principal punto de contacto para todos los participantes, a través del cual se lleva a cabo el control de todas las versiones.

Por ello, el primer paso consiste en crear este repositorio principal o clonarlo (en forma de copia de trabajo) en caso de que vayas a incorporarte a un proyecto que ya se esté gestionando con Git.

Tanto si quieres configurar el control de versiones para un nuevo proyecto como si acabas de instalar Git para aprender a trabajar con la herramienta, debes crear un nuevo repositorio. Para ello, ejecuta el comando `cd` (change directory) para acceder al directorio de tu dispositivo donde desees ubicarlo:

```
cd ruta al directorio
ejemplo:
mkdir testGit
cd testGit
```

Una vez allí, introduce el siguiente comando para crear un repositorio.git:

```
git init
```

Si ya existe un repositorio Git para el proyecto, solo tienes que introducir el comando `git clone` seguido de la dirección web o de red de ese repositorio para crear una copia de trabajo en el ordenador:

```
por ejemplo, el Git de Fortran de Métodos Numéricos
git clone https://github.com/EdgardoBonzi/Fortran-Examples.git
```

1.6 Comprobar el estado del repositorio y añadir nuevos archivos al control de versiones

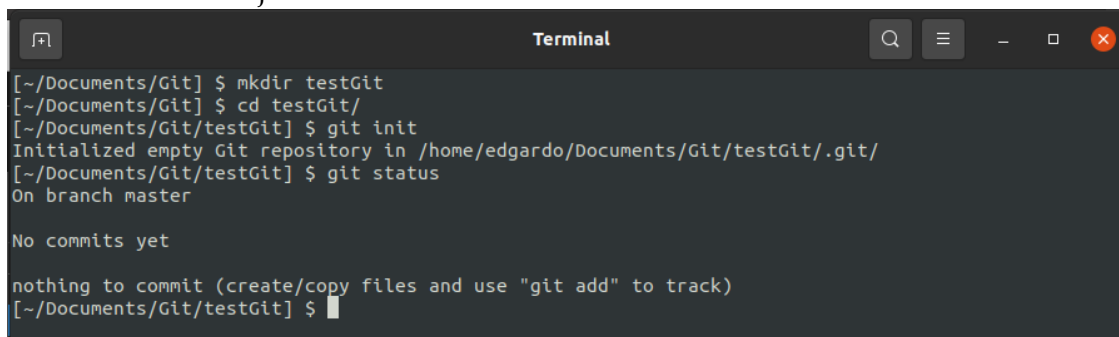
Uno de los conceptos básicos para utilizar Git es organizar adecuadamente el propio directorio de trabajo, lo que permite no solo proponer cambios e innovaciones personales a un proyecto, que luego son aceptados mediante el comando `git commit` (enviar), sino también obtener información sobre las actividades de otros usuarios. Puedes comprobar si tu copia de trabajo está actualizada ejecutando este comando:

```
git status
```

Por lo general, en el caso de los repositorios creados recientemente, o cuando el repositorio principal y la copia de trabajo coinciden por completo, recibirás un aviso de que no se ha modificado el proyecto (*No commits yet*).

Además, Git te informa de que no has compartido tus cambios para el próximo commit (*nothing to commit*).

Así se vería el trabajo desde una terminal de Ubuntu

A screenshot of a terminal window titled "Terminal" with standard Ubuntu window controls. The terminal shows a sequence of commands and their outputs:

```
[~/Documents/Git] $ mkdir testGit
[~/Documents/Git] $ cd testGit/
[~/Documents/Git/testGit] $ git init
Initialized empty Git repository in /home/edgardo/Documents/Git/testGit/.git/
[~/Documents/Git/testGit] $ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
[~/Documents/Git/testGit] $
```

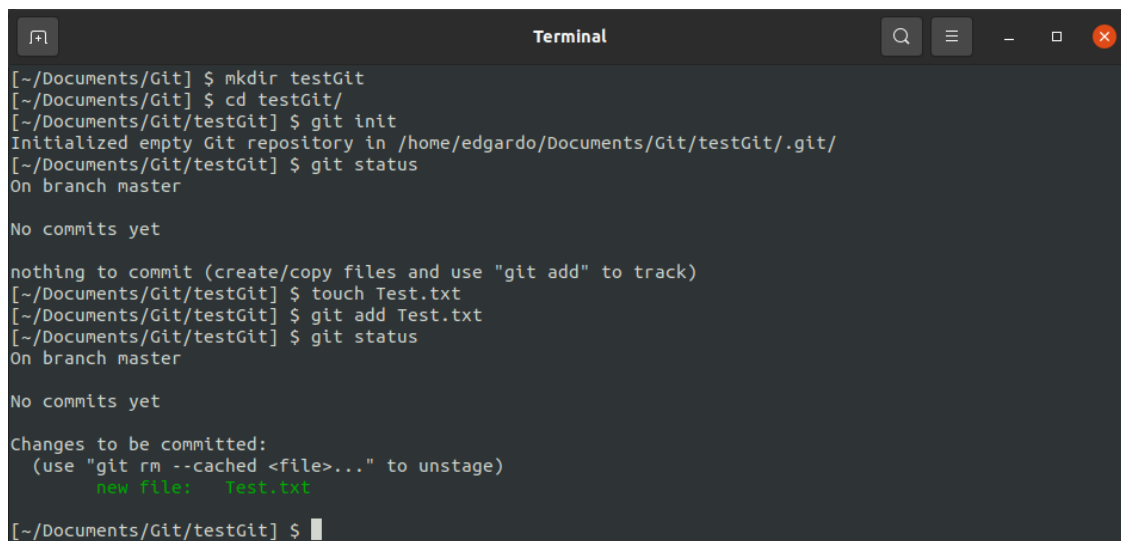
1.7 Agregar un nuevo archivo

Para añadir un nuevo archivo al control de versiones o para registrar algún cambio para el siguiente commit, introduce el comando `git add` y el nombre de este archivo, que debe encontrarse en tu directorio de trabajo. En nuestro tutorial, añadiremos un documento de texto llamado “Test” como ejemplo:

```
touch Test.txt
```

```
git add Test.txt
```

Después, cuando vuelvas a comprobar el estado del repositorio, verás que el documento está a la espera de someterse a la siguiente fase de confirmación de cambios del proyecto, en que estos se aceptarán o no (*Changes to be committed*):



```
Terminal
[~/Documents/Git] $ mkdir testGit
[~/Documents/Git] $ cd testGit/
[~/Documents/Git/testGit] $ git init
Initialized empty Git repository in /home/edgardo/Documents/Git/testGit/.git/
[~/Documents/Git/testGit] $ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
[~/Documents/Git/testGit] $ touch Test.txt
[~/Documents/Git/testGit] $ git add Test.txt
[~/Documents/Git/testGit] $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Test.txt
[~/Documents/Git/testGit] $
```

1.8 Confirmar los cambios mediante commit y añadirlos al HEAD

Cualquier cambio que hayas propuesto incorporar al proyecto, como hemos explicado en el punto anterior, debe confirmarse con `commit` para que se incluya en el HEAD. El HEAD es una especie de índice que apunta al último commit efectuado en el entorno de trabajo Git actual (también llamado “rama”). El comando para hacerlo es el siguiente:

Aquí escribimos una nota, explicando que cambios hicimos, en este caso decimos que ‘Hemos agregado el archivo TestGit.txt’

```
git commit -a -m "Hemos agregado el archivo TestGit.txt"
```

Si no ponemos `-a -m`, luego de ejecutar el comando `git commit`, Git inicia automáticamente el editor que configuraste como predeterminado durante la instalación o que el propio sistema abre por defecto.

En el documento, puedes añadir un comentario personal sobre el commit planificado, en el que las líneas anotadas se separan por punto y coma y, por lo tanto, no se muestran más adelante.

Nota Antes de ejecutar el comando `git commit`, no te olvides de comprobar si has marcado todos los cambios que deseas incluir en el repositorio remoto (con `git add`).

De lo contrario, estos serán ignorados, incluso si se encuentran en la copia de trabajo guardada en el directorio.

Luego Git creará el commit y si además hacemos `git status`, veremos lo siguiente:

```
[~/Documents/Git/testGit] $ git commit -a -m "Hemos agregado el archivo TestGit.txt"
[master (root-commit) bc38770] Hemos agregado el archivo TestGit.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Test.txt
[~/Documents/Git/testGit] $ git status
On branch master
nothing to commit, working tree clean
```

Como se muestra en la captura de pantalla, después de ejecutar `git commit`, obtienes un mensaje que resume el commit:

1. entre corchetes figuran, por un lado, el nombre de la rama del proyecto a la que se transfirieron los cambios (en este caso, *master*, ya que nuestro repositorio de trabajo también es el repositorio principal),
2. por otra parte, la suma de comprobación SHA-1 del commit (en este caso, *bc38770*), y
3. les siguen el comentario que anotó el propio usuario (aquí, "Hemos agregado el archivo TestGit.txt") y algunos datos concretos sobre los cambios:
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Test.txt

1.9 Revisar o deshacer commits ejecutados

Una vez aceptados los cambios mediante el comando `commit`, puedes editar el contenido o eliminarlo por completo en cualquier momento más adelante.

Por ejemplo, un caso típico sería precipitarse al ejecutar `commit` y olvidarse de algún archivo o configuración importante.

En este caso, puedes registrar archivos nuevos o modificados a posteriori mediante el comando `git add` y volver a transferirlos.

Para ello, añade `--amend` al comando estándar:

```
$ git add Test_02.txt
```

```
$ git commit --amend
```

Si quieres deshacer el último commit generado, puedes hacerlo con el siguiente comando de Git:

```
$ git reset --soft HEAD~1
```

Este comando cancela el commit registrado por última vez en el HEAD.

Los archivos que contiene se restablecen como "cambios planificados para el próximo commit" en el estado del proyecto.

Si lo que quieres es eliminar por completo los datos introducidos, ejecuta el siguiente comando en lugar del anterior:

```
$ git reset --hard HEAD~1
```

1.10 Mostrar el historial de commits

Aprender a gestionar proyectos con Git es especialmente útil debido a las características básicas de control de versiones que ofrece el sistema.

Por ejemplo, una gran ventaja de este programa de código abierto es que siempre puedes visualizar los últimos cambios que se han realizado en el repositorio. Para ello, puedes utilizar el siguiente comando de Git:

```
$ git log
```

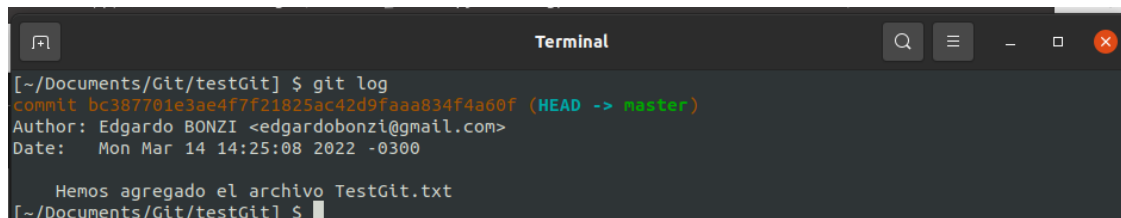
De manera predeterminada, el comando `git log` enumera los *commits* generados en orden cronológico inverso:

1. la suma de comprobación SHA-1, 2. el autor (nombre y dirección de correo electrónico) y 3. la fecha de cada commit.

Además, se muestra un comentario individual que sirve a todos los usuarios como indicador para poder buscar rápidamente las versiones.

En un apartado anterior de este tutorial de Git, generamos un solo commit con el mensaje `“Hemos agregado el archivo TestGit.txt”`.

Al ejecutar el comando, se nos muestra el archivo solicitado:



```
Terminal
[~/Documents/Git/testGit] $ git log
commit bc387701e3ae4f7f21825ac42d9faaa834f4a60f (HEAD -> master)
Author: Edgardo BONZI <edgardobonzi@gmail.com>
Date: Mon Mar 14 14:25:08 2022 -0300

    Hemos agregado el archivo TestGit.txt
[~/Documents/Git/testGit] $
```

El comando `git log` también puede modificarse utilizando varios parámetros.

En la siguiente tabla, te mostramos algunas de las posibilidades:

Parámetros para git log	Descripción
-p	Muestra también los cambios incluidos en un commit
-2	Enumera solo los dos últimos commits ejecutados
-stat	Añade una pequeña estadística a cada registro que muestra qué archivos se han modificado y cuántas líneas se han insertado o eliminado
-pretty	Cambia el formato de salida con diferentes posibilidades;
-abbrev-commit	Muestra solo los primeros caracteres de una suma de comprobación SHA-1
-relative-date	Muestra la fecha de un cambio en formato relativo (por ejemplo, “hace dos semanas”)

1.11 Como configurar la key

```
$ git config --global credential.helper store --file ~/.ssh/git_rsa.pub
```

[]:

To be continue ...