

GIT Course

1. Introduction

1.1. Course Goals and Objectives:



1.2. Course Overview

- Introduction / Key Concepts
- Git Installation and Setup
- Quick Start / Quick Win!
- The Basic Commands
- Comparing
- Branching and Merging
- GitHub Introduction
- Tools Setup (Text Editor, Compare / Merge Tool)

1.3. Why Command Line?

Empezando con la línea de commando es la major manera de aprender GIT, es una forma estándar de comunicar como GIT hace algo:

```
(master) timesheet $ git log --oneline --graph --color --decorate
* b7f67ea (HEAD, origin/master, origin/HEAD, master) remove eclipse project files f
* 58f329e updating to newest dependency versions
* d3e5752 working homepage at hello word stage
* aa8eb4f update to make compile -- still non-functional
* 2eb498a updates for spring mvc
* 076ba8e refactor junit to dependency management and use 4.11
* 2522eb0 initial fileset
* 94782ff first commit
(master) timesheet $ git status
# On branch master
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
```

1.4. Why Source Control?

El Source Control (o control de origen), en un inicio se describe como una copia de seguridad de cada versión de la fuente/fichero que has hecho. De esta forma, se crea una historia o un rastro de los cambios, lo cual permite deshacer los cambios que realizó o restauró desde un estado conocido anterior.

¡Es un control de versiones!

Tipologías de sistemas de Control de versiones:

- Centralized (centralizada) => Requieren de una conexión a un servidor centralizado que es la mejor fuente para la colección de archivos versionados.
 - ✓ Free: Subversion, CVS
 - ✓ Commercial: ClearCase, Perforce, Team Foundation Server (TFS)
 - ✓ Requiere connection to central server for most operations

- Decentralized / Distributed => No requiere de una conexión a un servidor centralizado, ya que permite que la mayoría de las operaciones sean locales.
 - ✓ GIT
 - ✓ Mercurial (Hg)
 - ✓ Most operations are local
 - ✓ Central server not required

1.5. Key GIT Terminology

The GIT Repository

El **repositorio** contiene ficheros, históricos de cambios de versiones y cualquier configuración especial que se mantienen juntas y están manejados por GIT. Por tanto, contiene todos los ficheros relacionados con un proyecto o aplicación.

The States of GIT

A) Local States

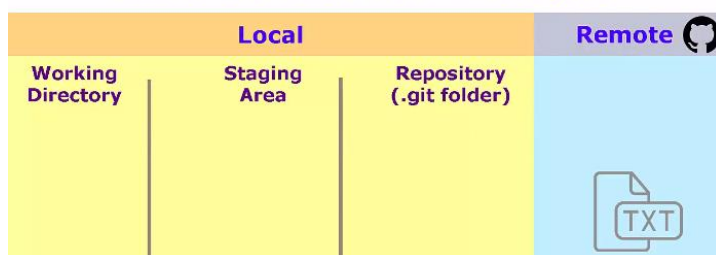
Estados locales relacionados con los archivos administrados por Git:

- **Working directory** => Repositorio de trabajo => El directorio de trabajo es el directorio o carpeta en su computadora que contiene todos los archivos de proyecto o aplicación. Los archivos dentro del directorio de trabajo, pueden o no ser administrados por Git. Dentro del directorio de trabajo hay una carpeta oculta llamada ".git" que contiene el repositorio git real.
- **Staging área / pre-commit holding area** => Área de preparación => Es un área de espera para poner en cola los cambios para la próxima confirmación. Dado que los archivos en el área de preparación aún no se han confirmado, puede moverlos dentro y fuera del área de preparación sin afectar el repositorio de Git y su historial de cambios.
- **Committed – Git Repository (history)** => Repositorio de GIT o el historial de confirmación => Es el área en la que se confirman los archivos editados en el repositorio de trabajo.

B) Remote Repository (GitHub)

El último estado del flujo de trabajo de GIT. Una vez que se finaliza el trabajo en local, se sube la información a remoto.

Basic Git Workflow Life Cycle



Master Branch (Marca maestra)

Son líneas de tiempo que contienen los cambios realizados en las diversas versiones de los ficheros/proyectos:

Branches in Git



En GIT, las ramas contienen commits. Cuando comenzamos, GIT nos proporciona una rama predeterminada denominada **Master**.

2. GIT Installation

Installation for Windows:

- Git for Windows (git-scm.com)
- Google Chrome (install in Bonus section, very optional)

A continuación, se procede con la instalación de GIT para Windows.

Una vez instalado el programa, nos mostrará la versión disponible:

```
MINGW64:/c:/Users/Lorena Mendez Otero

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~
$ git version
git version 2.23.0.windows.1

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~
$ |
```

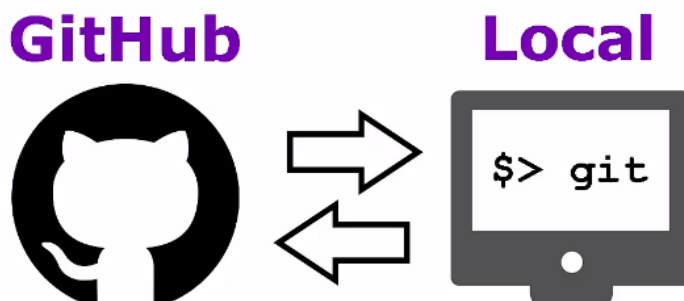
Link para releases notes de GIT:

<file:///C:/Program%20Files/Git/ReleaseNotes.html>

3. GIT Quick Start

3.1. Starting with GitHub and Project Setup

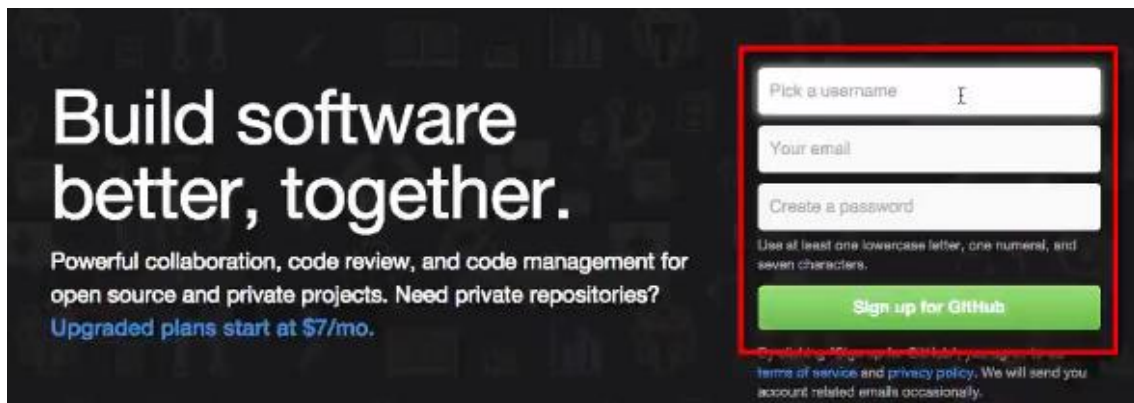
Workflow de GitHub:



Enlace a la página principal de GitHub:

<https://github.com/>

Para crear una cuenta en GIT, hay que crear el siguiente formulario:




Una vez que la cuenta esta creado, vamos a crear un nuevo repositorio:

<https://github.com/new>

Ejemplo de la creación de un repositorio:



- a) Añadimos el owner, el nombre del repositorio y la una descripción breve de lo que guardaremos en el repositorio.

Owner:  Imendezotero ▾ / Repository name: ✓

Great repository names are short and memorable. Need inspiration? How about **fantastic-enigma**?

Description (optional):

- b) Indicaremos que el repositorio sea público:

- ☒  **Public**
Anyone can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

- c) Indicar que el repositorio se inicie con un README (comenzará nuestro repositorio con un simple archivo README)

Skip this step if you're importing an existing repository.

- ☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: | Add a license: ⓘ

Una vez creado el repositorio, necesitamos preparar nuestro sistema local para poder continuar:

Terminal Programs

Windows

Git Bash (*Git for Windows*)

Para arrancar Git Bash, introduzca **pwd**:

```
MINGW64:/c/Users/Lorena Mendez Otero  
  
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~  
$ pwd  
/c/Users/Lorena Mendez Otero  
  
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~  
$ |
```

Crear una carpeta denominada “projects” debajo del directorio de inicio del usuario para administrar todos los proyectos juntos en un mismo directorio. Para ello, introduzca el COMANDO “**MKDIR**”:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT  
$ mkdir "Projects ytugfiug"  
  
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT  
$ rm "Projects ytugfiug"  
rm: cannot remove 'Projects ytugfiug': Is a directory  
  
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT  
$ rm -R "Projects ytugfiug"
```

3.2. Configuration, Clone and Git Basic Workflow

Configurar Git Bash

Para ver la versión de Git que tenemos instalada, introduce “git version”:

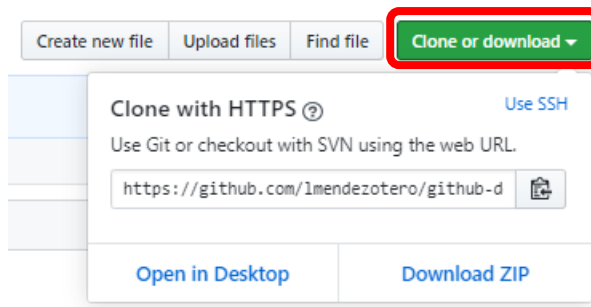
```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~  
$ git version  
git version 2.23.0.windows.1  
  
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~  
$ |
```

Configuración del nombre y correo electrónico del usuario (en este caso Lorena):

```
git version 1.9.3 (Apple Git-50)  
~ $ git config --global user.name "Abe Lincoln"  
~ $ git config --global user.email "mrabe@git.training"  
~ $ git config --global --list  
user.name=Abe Lincoln  
user.email=mrabe@git.training  
~ $
```

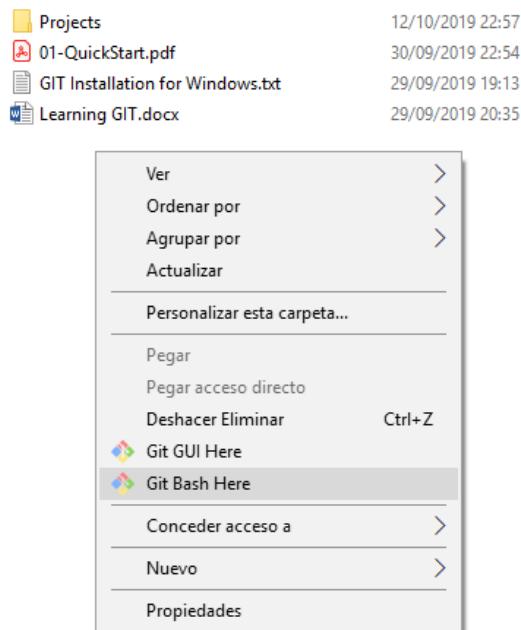
Clonar un repositorio

Para clonar un repositorio, tenemos que volver a Git Hub y seleccionar la opción “Clone with HTTPS”:



Luego copiamos la URL y la pegamos en Git Bash.

Para realizar correctamente el clon, nos vamos a la carpeta de Git en local y :



Una vez abierto Git Bash, chequeamos los directorios disponibles para guardar el clon introduciendo el COMANDO “**LS**”:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT
$ ls
'~$arning GIT.docx'  'GIT Installation for Windows.txt'  Projects/
'01-QuickStart.pdf'  'Learning GIT.docx'
```

El directorio disponible es Projects, por lo que vamos a indicar que vamos a crear aquí nuestro clon. Cambiamos el nombre del directorio introduciendo el COMANDO “**CD**”:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT
$ cd Projects/

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ |
```

Ahora que ya estoy en Projects, inserto el comando “GIT CLONE” + URL copiada para crear el clon:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ git clone https://github.com/lmendezotero/github-demo-begginerscourse.git
Cloning into 'github-demo-begginerscourse'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
```

Resultado de la visualización del clone en Local:

Este equipo > Documentos > 03. CURSOS > 01. PROGRAMACION > 03. GIT > Projects > github-demo-begginerscourse				
Nombre	Fecha de modifica...	Tipo	Tamaño	
.git	12/10/2019 23:05	Carpeta de archivos		
paquita.txt	12/10/2019 23:05	Documento de tex...	1 KB	
README.md	12/10/2019 23:05	Archivo MD	1 KB	

El comando GIT CLONE establece una relación entre el repositorio en GitHub y la referencia origen.

Para ver el status del repositorio clonado, introduzca el nombre del repositorio clonado y luego el COMANDO “GIT STATUS”:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ cd github-demo-begginerscourse

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/github-demo-begginerscourse (master)
$ ls
paquita.txt  README.md

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/github-demo-begginerscourse (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
nothing to commit, working tree clean
```

On branch master es la rama predeterminada para un repositorio Git.

“GIT STATUS” también nos dice que master está actualizado con origin/master, lo que se refiere a la rama maestra en la versión de GitHub del repositorio.

Finalmente, GIT STATUS nos dice si el directorio de trabajo está limpio (clean).

Por tanto, utilizamos GIT STATUS para ver si se ha producido algun cambio entre el directorio de trabajo, el área de preparación, nuestro repositorio local y remoto:



Crear un fichero en el repositorio

Para crear un fichero nuevo en el repositorio, introducimos el COMANDO “ECHO” seguido del nombre del fichero entre “”. Luego, añadimos >> para canalizar el contenido de ECHO en un fichero txt start.txt:


```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/
$ echo "Test Git Quick Start: Paquita es guapa" >> start.txt
```

Nombre del fichero: start.txt

Contenido del fichero: "Test Git Quick Start: Paquita es guapa"

Utilizamos el COMANDO "**CAT**" para mostrar el contenido del fichero (en Python equivale a PRINT):

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/
$ cat start.txt
Test Git Quick Start: Paquita es guapa
```

Git Basic Workflow

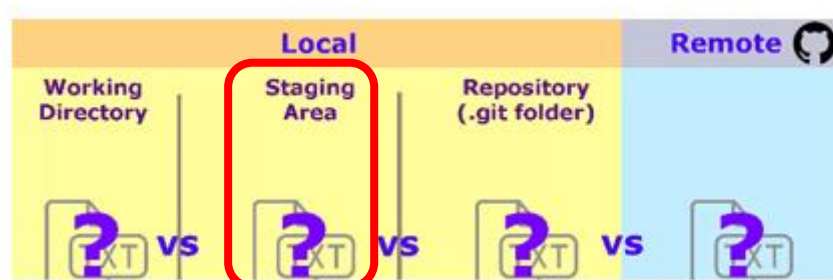
¿Qué significa tener un archivo sin seguimiento? => Un archivo en nuestro directorio de trabajo que aun no se ha añadido a Git.

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  start.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Por tanto, tenemos que añadir el fichero a la Stagin Area.



Para ello, añadimos el COMANDO "GIT ADD" y luego el nombre del fichero:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/
$ git add start.txt
warning: LF will be replaced by CRLF in start.txt.
The file will have its original line endings in your working directory
```

Chequeo del resultado final con GIT STATUS:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
  new file:   start.txt
```

Ahora el fichero está en la Stagin Area, pero pendiente de introducir una confirmación en el commit (parte remarcada en rojo). Para confirmar el nuevo archivo introducimos el COMANDO "**GIT COMMIT -M**" y entre "" el mensaje de confirmación "Agregar archivo de texto de inicio":


```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/
$ git commit -m "Agregar archivo de texto de inicio"
[master 93bd68d] Agregar archivo de texto de inicio
1 file changed, 1 insertion(+)
create mode 100644 start.txt
```

Si introducimos GIT STATUS, nos encontramos en que estamos de vuelta en un estado de directorio de trabajo limpio y que nuestra rama maestra está por delante de “origin/master” por una confirmación:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03.
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Por tanto, el archivo se ha movido hasta el área de Repositorio Local:



Como no hay confirmaciones pendientes, Git marca el fichero como “limpio”.

Para lanzar el fichero al Repositorio en remoto (GitHub), tenemos que hacer un paso final denominado **PUSH** => Impulso. Para ello, introduzca el COMANDO “**GIT PUSH**” seguido de “origin master”:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~
$ git push origin master
```

origin refers to the
GitHub copy of our
repository

master refers to our
default and only branch in
the repository

Después de presionar ENTER, nos vamos al browser en donde tenemos GitHub y refrescamos la página para ver los cambios. Nos pedirá introducir nombre de usuario y contraseña de GitHub. Una vez introducidas las credenciales, vemos los resultados:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PRO
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 372 bytes | 372.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/lmendezotero/github-demo-beggincourse.git
a68b777..93bd68d master -> master
```

Nos vamos al browser de Internet en GitHub y refrescamos la página de nuevo:

Imendezotero Agregar archivo de texto de inicio	
README.md	Initial commit
paquita.txt	Create paquita.txt
start.txt	Agregar archivo de texto de inicio

Comandos de Git analizados en esta sección:

Command Listing, Part 1

```
pwd
mkdir projects
cd projects
pwd
```

Command Listing, Part 2

```
git version
git config --global user.name "Abe Lincoln"
git config --global user.email "mrabe@git.training"
git config --global --list
git clone github-https-url # paste in your GitHub HTTPS clone URL
ls
cd github-demo
ls
git status
echo "Test Git Quick Start demo" >> start.txt
ls
cat start.txt
git status
git add start.txt
git status
git commit -m "Adding start text file"
git status
git push origin master
```

4. Text Editor Installation

4.1. Text Editor Installation Overview

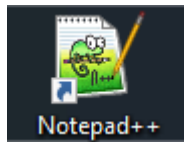
Instalaremos el editor de texto **Notepad++** y veremos cómo configurarlo para que funcione correctamente con Git.

- Notepad++
- Bash alias
- Git configuration

Instalación de Notepad++

<https://notepad-plus-plus.org/downloads/>

Icono del programa **Notepad++** en el escritorio:



Configuración de Notepad++ en Git

Primero, tenemos que encontrar en dónde se ha instalado Notepad++. Para ello, nos vamos al disco Local C y abro la carpeta “Archivos de programa (x86)”:

Este equipo > Disco local (C:) > Archivos de programa (x86)		
Nombre		Fecha de modifica..
Adobe		28/11/2018 17:43
ASUS		25/11/2018 19:15
Battle.net		24/03/2019 13:50
Common Files		02/12/2018 20:13
Google		10/10/2019 21:34
Intel		25/11/2018 19:14
Internet Explorer		13/10/2019 0:30
Microsoft Analysis Services		02/12/2018 20:11
Microsoft Office		02/12/2018 20:12
Microsoft SQL Server		11/12/2018 13:04
Microsoft.NET		02/12/2018 20:12
Mozilla Firefox		02/12/2018 20:16
MSBuild		18/05/2019 13:40
Notepad++		13/10/2019 12:42

Notepad ++ es un software de 32 bits que, que se ejecuta en una versión de Windows de 64 bits, estará en "Archivos de programa (x86)".

Copiamos la dirección de la ruta en al que está instalado Notepad++:

C:\Program Files (x86)\Notepad++

Tenemos que crear un nuevo path en Variables de Entorno, que se encuentra en las Opciones Avanzadas de las Propiedades de Este equipo:

Editar variable de entorno

C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files (x86)\Notepad++

De esta forma, al introducir “Notepad++” en Git Bash se abrirá automáticamente el editor de texto.

4.2. Text Editor for Windows

¿???????

5. Basic GIT Commands

Hay 3 maneras de empezar un proyecto en GIT:

Git Basics Overview

- Starting a Project
 - Fresh (no source yet)
 - Existing source locally
 - GitHub project (Fork and clone)
- Basic Workflow (add, commit, push & pull)
- Working with Files (rename, move & delete)
- History and Aliases
- Ignoring Unwanted Files

5.1. Starting with a Fresh Project

Vamos a iniciar un Proyecto sin ningún código fuente existente. Para ello, utilizaremos información de la siguiente página:

<https://hipsun.co/>

Voy a crear un nuevo repositorio en mi carpeta de Projects, por lo que la abro y arranco Git Bash desde allí.

Introduzco el COMANDO “**GIT INIT**” para crear un nuevo proyecto seguido del nombre del proyecto (fresh-project):

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ git init fresh-project
Initialized empty Git repository in C:/Users/Lorena Mendez Otero/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ ls
fresh-project, github-demo-begginerscourse/
```

» Este equipo » Documentos » 03. CURSOS » 01. PROGRAMACION » 03. GIT » Projects			
Nombre		Fecha de modifica...	Tipo
fresh-project		13/10/2019 13:19	Carpeta de archivos
github-demo-begginerscourse		12/10/2019 23:23	Carpeta de archivos

Si añadimos el comando LS, veremos que la carpeta fresh-proyect no tiene ningun archivo. Sin embargo, si añadimos **LS -AL**, nos encontramos con que el programa le dice al comando list que enumere todos los archivos y carpetas, incluidos los archivos y carpetas de puntos;

mientras que "-l" le dice que haga una lista en el formato de listado. Ahora puedes ver que tenemos un ".git".

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CUR
$ ls -al
total 4
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 13:19 ./
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 13:19 ../
lrwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 13:19 .git/
```

Ahí es donde vive el repositorio de Git; la carpeta en la que estoy ahora es en realidad la carpeta de trabajo. Entonces si entro en el ".git", puedes hacer un "ls", y puedes ver todas las carpetas y archivos que componen el repositorio de Git internamente:

NO ENTIENDO CÓMO LOGRAR ESTO:

```
drwxr-xr-x  3 jason  staff   102B Oct 31 01:30 ./
drwxr-xr-x  3 jason  staff   102B Oct 31 01:30 ../
drwxr-xr-x 10 jason  staff  340B Oct 31 01:30 .git/
(master) fresh-project $ cd .git/
(GIT_DIR!) .git $ ls
HEAD          config      hooks/      objects/
branches/     description info/        refs/
(GIT_DIR!) .git $ cd ..
(master) fresh-project $ pwd
/Users/jason/projects/fresh-project
(master) fresh-project $ git status
On branch master

Initial commit
```

Continuamos con el proyecto y nos vamos a la página de Hipsum para copiar texto de relleno. Copiamos texto, lo guardamos en las notas y creamos un fichero nuevo insertando el COMANDO "MATE" seguido del nombre del archivo hipster.txt:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/fresh-proyect (master)
$ echo "Hola paquita" >> hipster.txt
```

NO LO HE LOGRADO CON MATE, PERO SI CON ECHO!!!!

Actualmente estamos en el directorio de trabajo, por lo que Git aún no está rastreando esto. Para que Git rastree este archivo, simplemente necesitamos agregar/mover ese archivo al índice de Git o al Staging Area de Git. Para hacerlo, simplemente escribimos "git add", espacio y luego el archivo que deseamos agregar.

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01.
$ git add hipster.txt
warning: LF will be replaced by CRLF in hipster.txt.
The file will have its original line endings in your working directory
```

Ahora, si hacemos nuestro "git status", Git nos dice que tenemos "Cambios para confirmar". Eso significa este nuevo archivo, "hipster.txt", ahora está en el área de preparación de Git. Tenemos un archivo que está esperando ser confirmado, pero aún no está confirmado. Para confirmar los

cambios, introducimos “GIT COMMIT” y el mensaje de confirmación “Agregando un nuevo archivo con hipster ipsum”:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ git commit -m "Agregando un nuevo archivo con hipster ipsum"
[master (root-commit) 1c57bea] Agregando un nuevo archivo con hipster ipsum
1 file changed, 1 insertion(+)
create mode 100644 hipster.txt

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ git status
On branch master
nothing to commit, working tree clean
```

¿Y si quiero retroceder hacía atrás?

Inserto el COMANDO “RM-RF” esepficiando la eliminación de la carpeta que quiero hacer:

Por tanto, ya no hay más repositorios de Git.

NO LO HE LOGRADO!

5.2. Adding Git to an exist project (git init)

Para añadir Git a un Proyecto ya inexistente, vamos a utilizar la web:

<http://www.initializr.com/>

Descargamos unos archivos de dicha página web. A continuación, arrancamos Git Hash y descomprimos los ficheros descargados de la web de initializr añadiendo el COMANDO “UNZIP”:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ unzip ~/Downloads/initializr-verekia-4.0.zip
Archive: ~/c:/Users/Lorena Mendez Otero/Downloads/initializr-verekia-4.0.zip
  inflating: initializr/index.html
  inflating: initializr/css/main.css
  inflating: initializr/css/bootstrap-theme.css
  inflating: initializr/css/bootstrap-theme.min.css
  inflating: initializr/css/bootstrap.css
  inflating: initializr/css/bootstrap.min.css
  inflating: initializr/css/bootstrap.css.map
  inflating: initializr/css/bootstrap-theme.css.map
  inflating: initializr/fonts/glyphicons-halflings-regular.eot
  inflating: initializr/fonts/glyphicons-halflings-regular.svg
  inflating: initializr/fonts/glyphicons-halflings-regular.ttf |
```

Chequeo de que existe la carpeta Initializr que contiene todos los documentos descomprimidos en la carpeta de Projects:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ ls
fresh-project/  github-demo-beggincourse/  initializr/
```

Vamos a cambiarle el nombre a la carpeta de Initializr por “web-project” utilizando el COMANDO “MV”:


```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ mv inicializr web-project
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ ls
fresh-project/  github-demo-beggincourse/  web-project/

```

Arrancar la carpeta web-project y ver todos los archivos que hay dentro de ella:

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ cd web-project
pwd
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/web-project
$ pwd
/c/Users/Lorena Mendez Otero/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/web-project
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/web-project
$ ls
404.html  apple-touch-icon.png  browserconfig.xml  crossdomain.xml  css/  favicon.ico  fonts/  humans.txt  img/  index.html  js/  robots.txt  tile.png  tile-wide.png

```

Agregar control de fuente Git al proyecto

Para agregar control de fuente Git al proyecto, utilizamos el COMANDO "GIT INIT":

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/web-project
$ git init
Initialized empty Git repository in C:/Users/Lorena Mendez Otero/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/web-project/.git/

```

Por tanto, con "INIT" Git inicializará un nuevo repositorio, usando la carpeta actual como el directorio de trabajo actual.

Ahora si hago un "ls -al", puedo ver que tengo mi ".git", que es donde realmente vive el repositorio de Git, junto con los otros archivos que forman parte de este proyecto:

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ ls -al
total 95
drwxr-xr-x 1 Lorena Mendez Otero 197121  0 oct. 13 15:20 ./
drwxr-xr-x 1 Lorena Mendez Otero 197121  0 oct. 13 15:12 ../
drwxr-xr-x 1 Lorena Mendez Otero 197121  0 oct. 13 15:20 .git/
-rw-r--r-- 1 Lorena Mendez Otero 197121 39014 oct. 13 13:02 .htaccess
-rw-r--r-- 1 Lorena Mendez Otero 197121  1272 oct. 13 13:02 404.html
-rw-r--r-- 1 Lorena Mendez Otero 197121  3959 oct. 13 13:02 apple-touch-icon.png
-rw-r--r-- 1 Lorena Mendez Otero 197121   416 oct. 13 13:02 browserconfig.xml
-rw-r--r-- 1 Lorena Mendez Otero 197121   603 oct. 13 13:02 crossdomain.xml
drwxr-xr-x 1 Lorena Mendez Otero 197121  0 oct. 13 15:07 css/
-rw-r--r-- 1 Lorena Mendez Otero 197121   766 oct. 13 13:02 favicon.ico
drwxr-xr-x 1 Lorena Mendez Otero 197121  0 oct. 13 15:07 fonts/
-rw-r--r-- 1 Lorena Mendez Otero 197121   191 oct. 13 13:02 humans.txt
drwxr-xr-x 1 Lorena Mendez Otero 197121  0 oct. 13 13:02 img/
-rw-r--r-- 1 Lorena Mendez Otero 197121  5329 oct. 13 13:02 index.html
drwxr-xr-x 1 Lorena Mendez Otero 197121  0 oct. 13 15:07 js/
-rw-r--r-- 1 Lorena Mendez Otero 197121    78 oct. 13 13:02 robots.txt
-rw-r--r-- 1 Lorena Mendez Otero 197121  3482 oct. 13 13:02 tile.png
-rw-r--r-- 1 Lorena Mendez Otero 197121  1854 oct. 13 13:02 tile-wide.png

```

Ahora, si hago un "estado de git", Git me recordará que estoy en la rama maestra, que es la rama predeterminada que obtienes cuando creas un nuevo repositorio de Git.

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git status
On branch master

```

Dado que aún no hemos realizado ninguna confirmación, estamos acumulando nuestra confirmación inicial, y luego Git nota un montón de archivos que aún no está rastreando:


```

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .htaccess
        404.html
        apple-touch-icon.png
        browserconfig.xml
        crossdomain.xml
        css/
        favicon.ico
        fonts/
        humans.txt
        index.html
        js/
        robots.txt
        tile-wide.png
        tile.png

nothing added to commit but untracked files present (use "git add" to track)

```

Entonces, para agregar todos estos archivos a la vez al área de preparación de Git y al índice de Git, vamos a escribir "git add" espacio y un punto:

```

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. G
$ git add .
warning: LF will be replaced by CRLF in .htaccess.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in 404.html.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in browserconfig.xml.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in crossdomain.xml.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in css/bootstrap-theme.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in css/bootstrap-theme.min.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in css/bootstrap.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in css/bootstrap.min.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in css/main.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in fonts/glyphicons-halflings-regular.svg.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in humans.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in index.html.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in js/main.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in js/plugins.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in js/vendor/bootstrap.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in js/vendor/bootstrap.min.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in js/vendor/jquery-1.11.2.min.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in js/vendor/modernizr-2.8.3-respons-1.4.2.min.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in js/vendor/npm.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in robots.txt.
The file will have its original line endings in your working directory

```

Ahora hay muchos más archivos de los que teníamos antes, y eso se debe a que el período agregará recursivamente todos los archivos que forman parte de este proyecto.

Añadiremos el mensaje de comprobación usando el comando “**GIT COMMIT -M**” seguido del mensaje entre comillas “Mi primera confirmación en línea”:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01.
$ git commit -m "Mi primer mensaje en línea"
[master (root-commit) e75d3a2] Mi primer mensaje en línea
29 files changed, 10594 insertions(+)
create mode 100644 .htaccess
create mode 100644 404.html
create mode 100644 apple-touch-icon.png
create mode 100644 browserconfig.xml
create mode 100644 crossdomain.xml
create mode 100644 css/bootstrap-theme.css
create mode 100644 css/bootstrap-theme.css.map
create mode 100644 css/bootstrap-theme.min.css
create mode 100644 css/bootstrap.css
create mode 100644 css/bootstrap.css.map
create mode 100644 css/bootstrap.min.css
create mode 100644 css/main.css
create mode 100644 favicon.ico
create mode 100644 fonts/glyphicons-halflings-regular.eot
create mode 100644 fonts/glyphicons-halflings-regular.svg
create mode 100644 fonts/glyphicons-halflings-regular.ttf
create mode 100644 fonts/glyphicons-halflings-regular.woff
create mode 100644 humans.txt
create mode 100644 index.html
create mode 100644 js/main.js
create mode 100644 js/plugins.js
create mode 100644 js/vendor/bootstrap.js
create mode 100644 js/vendor/bootstrap.min.js
create mode 100644 js/vendor/jquery-1.11.2.min.js
create mode 100644 js/vendor/modernizr-2.8.3-respond-1.4.2.min.js
create mode 100644 js/vendor/npm.js
create mode 100644 robots.txt
create mode 100644 tile-wide.png
create mode 100644 tile.png
```

Utilizamos “Git Status” para chequear que el commit se ha hecho correctamente y el directorio de trabajo está limpio:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~
$ git status
On branch master
nothing to commit, working tree clean
```

Para eliminar una carpeta/archivo, añadimos el comando “**RM -RF**” seguido del nombre de la carpeta/archivo que quiero eliminar.

Luego, haremos una comprobación con el comando “**LS -AL**” y vemos que ya no existe la carpeta .git/:

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. C
$ rm -rf .git/

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. C
$ ls -al
total 91
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 15:32 ./
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 15:12 ../
-rw-r--r-- 1 Lorena Mendez Otero 197121 39014 oct. 13 13:02 .htaccess
-rw-r--r-- 1 Lorena Mendez Otero 197121 1272 oct. 13 13:02 404.html
-rw-r--r-- 1 Lorena Mendez Otero 197121 3959 oct. 13 13:02 apple-touch-icon.png
-rw-r--r-- 1 Lorena Mendez Otero 197121 416 oct. 13 13:02 browserconfig.xml
-rw-r--r-- 1 Lorena Mendez Otero 197121 603 oct. 13 13:02 crossdomain.xml
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 15:07 css/
-rw-r--r-- 1 Lorena Mendez Otero 197121 766 oct. 13 13:02 favicon.ico
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 15:07 fonts/
-rw-r--r-- 1 Lorena Mendez Otero 197121 191 oct. 13 13:02 humans.txt
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 13:02 img/
-rw-r--r-- 1 Lorena Mendez Otero 197121 5329 oct. 13 13:02 index.html
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 15:07 js/
-rw-r--r-- 1 Lorena Mendez Otero 197121 78 oct. 13 13:02 robots.txt
-rw-r--r-- 1 Lorena Mendez Otero 197121 3482 oct. 13 13:02 tile.png
-rw-r--r-- 1 Lorena Mendez Otero 197121 1854 oct. 13 13:02 tile-wide.png

```

Eliminación de la carpeta de proyecto web-project utilizando el comando “RM -RF”:

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents
$ rm -rf web-project

```

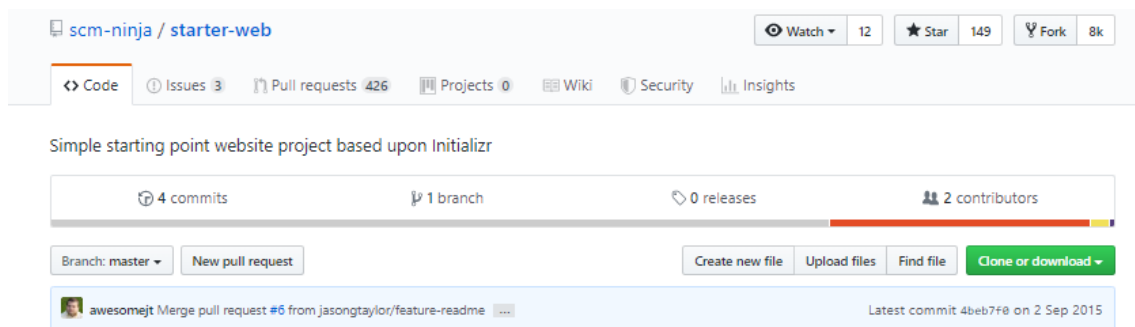
5.3. Starting on GitHub by joining an existing project (git clone)

Vamos a demostrar cómo unirnos a un Proyecto de GitHub ya existente.

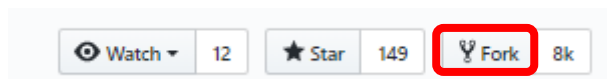
<http://bitly.com/git-start-web>

Copiar el proyecto en GitHub

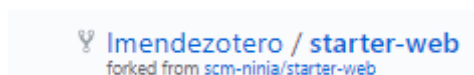
Primero, tenemos que asegurarnos de que nos hemos unido a nuestra cuenta de GitHub y luego entrar en el repositorio de **scm-ninja / starter-web**:



A continuación, clicamos en el botón de Fork (también conocido como "Bifurcación") para crear una copia del repositorio en nuestro espacio personal:



Ahora tenemos nuestra propia copia personal de ese repositorio para trabajar:



Resultado en GitHub:

starter-web

Forked from scm-ninja/starter-web

Simple starting point website project based upon Initializr

ApacheConf 7,967 Updated 6 hours ago

github-demo-begginerscourse

A simple demo repository to show the basic GIT workflow learned in the begginerscourse

Updated 17 hours ago

Clonar el repositorio de GitHub en Local

Luego, necesitamos clonar el repositorio a nuestro sistema local.

Así que vaya a nuestras opciones de clonación y asegúrese de que esté seleccionado "HTTPS" y, si tiene este botón aquí, "Copiar al portapapeles", puede usarlo, o simplemente puede copiar la url que está en este texto campo.

Luego, voy a "proyectos de CD" porque es la ubicación en la que clonaré mi repositorio Git. Para hacerlo, uso el comando "GIT CLONE", espacio, y luego pegue la URL que acabamos de copiar de la página web de GitHub:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ git clone https://github.com/lmendezotero/starter-web.git
Cloning into 'starter-web'...
remote: Enumerating objects: 39, done.
remote: Total 39 (delta 0), reused 0 (delta 0), pack-reused 39
Unpacking objects: 100% (39/39), done.
```

Veamos todos los ficheros dentro del repositorio clonado:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects
$ cd starter-web

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web
$ ls -al
total 84
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 16:43 ./
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 16:43 ../
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 16:43 .git/
-rw-r--r-- 1 Lorena Mendez Otero 197121 30612 oct. 13 16:43 .htaccess
-rw-r--r-- 1 Lorena Mendez Otero 197121 4616 oct. 13 16:43 404.html
-rw-r--r-- 1 Lorena Mendez Otero 197121 1226 oct. 13 16:43 apple-touch-icon-precomposed.png
-rw-r--r-- 1 Lorena Mendez Otero 197121 618 oct. 13 16:43 crossdomain.xml
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 16:43 css/
-rw-r--r-- 1 Lorena Mendez Otero 197121 766 oct. 13 16:43 favicon.ico
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 16:43 fonts/
-rw-r--r-- 1 Lorena Mendez Otero 197121 206 oct. 13 16:43 humans.txt
-rw-r--r-- 1 Lorena Mendez Otero 197121 5260 oct. 13 16:43 index.html
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 16:43 js/
-rw-r--r-- 1 Lorena Mendez Otero 197121 139 oct. 13 16:43 README.md
-rw-r--r-- 1 Lorena Mendez Otero 197121 35 oct. 13 16:43 robots.txt
-rw-r--r-- 1 Lorena Mendez Otero 197121 453 oct. 13 16:43 simple.html
```

En la parte superior, notarás un ". carpeta git "; ese es el repositorio real de Git.

Hacemos un "GIT STATUS" y nos encontramos con que estamos en la rama maestra; master es la rama predeterminada que obtienes con un repositorio de Git. Y, dado que clonamos nuestro repositorio Git desde una ubicación remota, la siguiente línea nos dice que nuestra rama está "actualizada con 'origin / master' ". "

El nombre "origen" es una referencia a nuestro repositorio GitHub, es nuestra referencia remota, y "/" master", es la rama maestra en ese repositorio remoto.

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Document
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Finalmente, la última línea nos dice que no hay nada que comprometer, porque todavía no hemos hecho nada y que nuestro directorio de trabajo está limpio, lo cual, nuevamente, tiene sentido.

5.4. Basic Git Workflow (add, commit, pull & push)

Vamos a demostrar cómo agregar nuevos archivos a un repositorio de Git. Utilizaremos las siguientes URLs:

<http://bitly.com/git-start-web>
<http://meettheipsums.com/>
<http://hipsum.co/>

Las dos últimas URLs son para conseguir texto de relleno.

Vamos a agregar un fichero nuevo utilizando el COMANDO "ECO" llamada hipster.txt.

Al iniciar el comando GIT STATUS, Git nos dice que hay un fichero (hipster.txt) pendiente de seguimiento y dice como incorporarlo para Git pueda rastrearlo:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAM
er-web (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  hipster.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Por lo tanto, utilizaremos el COMANDO "GIT ADD" para lanzar el fichero al Stagin Area y luego añadimos el commit con el COMANDO "GIT COMMIT":

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03.
er-web (master)
$ git add hipster.txt

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03.
er-web (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
  new file:   hipster.txt
```

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. CU
er-web (master)
$ git commit
[master ca7e29c] Por favor, proceda con la subida del fichero.
1 file changed, 1 insertion(+)
create mode 100644 hipster.txt

```

Si hacemos otro GIT STATUS, Git nos dirá que estamos por delante de "origen / maestro" por una confirmación. También nos dice que no tenemos nada que comprometer y que nuestro directorio de trabajo está limpio.

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CU
er-web (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

```












El comando commit es una operación local, ya que no ha impacto en el repositorio en remoto (GitHub). Entonces, para lanzar mi fichero de local a remoto tengo que utilizar el comando GIT PUSH. Por tanto, insertando "git push origin master" logro empujar cualquier confirmación desde el repositorio local al repositorio remoto por nombre de referencia, así como la rama que desea empujar.

```

Lorena Mendez Otero@DESKTOP-50QT6GB
er-web (master)
$ git push origin master
Everything up-to-date

```

Ver resultado en remoto:

This branch is 1 commit ahead of scm-ninja:master.			Pull request	Compare
	Imendezotero	Por favor, proceda con la subida del fichero.	Latest commit ca7e29c 15 minutes ago	
	css	Copying files from initializr project zip file and then creating simp...	5 years ago	
	fonts	Copying files from initializr project zip file and then creating simp...	5 years ago	
	js	Copying files from initializr project zip file and then creating simp...	5 years ago	
	.htaccess	Copying files from initializr project zip file and then creating simp...	5 years ago	
	404.html	Copying files from initializr project zip file and then creating simp...	5 years ago	
	README.md	Adding Purpose section to README	4 years ago	
	apple-touch-icon-precomposed.png	Copying files from initializr project zip file and then creating simp...	5 years ago	
	crossdomain.xml	Copying files from initializr project zip file and then creating simp...	5 years ago	
	favicon.ico	Copying files from initializr project zip file and then creating simp...	5 years ago	
	hipster.txt	Por favor, proceda con la subida del fichero.	15 minutes ago	

Ahora, debería explicar qué significan "origen" y "maestro". Por una parte, "origen" es el nombre del repositorio remoto del que clonamos: de ahí que se llame origen. Por otra parte, "maestro" es nuestra rama maestra, es la única rama que tenemos en este momento.

La operación inversa a GIT PUSH es GIT PULL. Utilizando el comando "**git pull origin master**", esto actualizará nuestro repositorio con cualquier cambio que pueda haber ocurrido en el

repositorio remoto hasta nuestro repositorio local, solo para asegurarnos de que estamos actualizados antes de hacer cualquier envío copia de seguridad en nuestro repositorio remoto.

5.5. Tracked Files

He añadido un cambio en el fichero de hipster.txt añadiendo texto adicional de la web:

<https://hipsum.co/>

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
```

Por ahora, digamos que estoy contento con estos cambios y que solo quiero confirmarlos directamente. Así que realmente puedo atajar el proceso de agregar y confirmar en un solo paso escribiendo "**git commit -am**" (-a para agregar el escenario, y luego "m" para mi mensaje de confirmación. Por tanto, entonces estoy haciendo un mensaje de confirmación en línea, "Agregando más texto ipsum".

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/
$ git commit -am "Agregando más texto ipsum"
[master df7a025] Agregando más texto ipsum
1 file changed, 3 insertions(+), 1 deletion(-)
```

Añadimos GIT STATUS para ver el estado del fichero modificado:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CU
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

"Git commit -am" solo se puede hacer con archivos que se están rastreando. =>> **¿Qué es un archivo rastreado?** Un archivo rastreado es cualquier archivo que Git conoce y rastrea activamente. Ese sería cualquier archivo que ya se haya confirmado en el repositorio de Git, o cualquier archivo que se haya agregado al índice de Git o al área de preparación de Git.

Una forma de averiguar si Git está rastreando su archivo, es usar el comando "**git ls-files**", que me da una lista de todos los archivos que Git está rastreando en el repositorio actual.

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW
$ git ls-files
.htaccess
404.html
README.md
apple-touch-icon-precomposed.png
crossdomain.xml
css/bootstrap-theme.css
css/bootstrap-theme.css.map
css/bootstrap-theme.min.css
css/bootstrap.css
css/bootstrap.css.map
css/bootstrap.min.css
css/main.css
favicon.ico
fonts/glyphicons-halflings-regular.eot
fonts/glyphicons-halflings-regular.svg
fonts/glyphicons-halflings-regular.ttf
fonts/glyphicons-halflings-regular.woff
hipster.txt
```


En caso de que un archivo nuevo "txt " no estuviese en la lista, podemos añadirlo al listado de ficheros rastreados utilizando el comando **"git add newfile". txt "** y a continuación **"git status"** nos encontramos con el nuevo fichero en la lista.

```
(master) starter-web $ git add newfile.txt
(master) starter-web $ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
```

5.6. Editing Files

Cuando estamos haciendo una edición muy simple y directa a un archivo existente, uno de los puntos principales del área de preparación es que desarrollamos nuestros cambios para el compromiso; es decir, solo queremos usar el parámetro **"-a"** en el comando **"commit"** cuando no queremos agregar más cambios a ese commit.

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed
  (use "git checkout -- <file>..." to discard changes in wo
```

Cuando tenemos un cambio que puede abarcar varios archivos, definitivamente queremos usar el área de preparación. Entonces, si hago un "estado de git", nuestro cambio se ha movido al área de preparación, y modifiquemos el archivo una vez más.

Git puede rastrear la diferencia entre los cambios que se deben confirmar, es decir, los que están en escena, y los cambios que aún no se han montado. Así que ahora necesitamos agregar esas modificaciones: **"git add"**.

Entonces, puede que se pregunte por qué estamos usando el comando **"git add"** para un archivo existente. Me gusta pensar en ello como "En realidad estoy agregando los cambios que han ocurrido" de nuevo en el índice de Git, o de nuevo en el área de preparación. Si hago eso, "estado de git", ahora mis dos cambios están ahora en el área de preparación. Entonces **"git commit -m"** Más ipsum para hipsters "", presiona enter. Ahora hagamos un "estado de git"; Bien, volvemos a un directorio de trabajo limpio.

5.7. Recursive Add

En este video, le mostraré cómo agregar archivos de manera recursiva (crear carpetas dentro de otra carpeta) a su repositorio Git.

Quiero crear una estructura de carpetas de anidamiento profundo dentro de mi repositorio Git. Así que ahora voy a usar el COMANDO **"MKDIR" con el parámetro "-p"** para crear esta estructura de carpetas. Si hago un **"ls"**, puede ver que tengo mi directorio de primer nivel, que es **"level1"**.

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMAC
ION/03. GIT/Projects/starter-web (master)
$ mkdir -p level11/level12/level13/level15

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMAC
ION/03. GIT/Projects/starter-web (master)
$ ls
404.html          favicon.ico  index.html  robots.txt
apple-touch-icon-precomposed.png  fonts/      js/         simple.html
crossdomain.xml  hipster.txt level11/
css/             humans.txt  README.md
```

Visualización del resultado en Local:

Este equipo > Documentos > 03. CURSOS > 01. PROGRAMACION > 03. GIT > Projects > starter-web				
Nombre	Fecha de modifica...	Tipo	Tamaño	
.git	13/10/2019 21:25	Carpeta de archivos		
css	13/10/2019 16:43	Carpeta de archivos		
fonts	13/10/2019 16:43	Carpeta de archivos		
js	13/10/2019 16:43	Carpeta de archivos		
level11	13/10/2019 23:05	Carpeta de archivos		
.htaccess	13/10/2019 16:43	Archivo HTACCESS	30 KB	
404.html	13/10/2019 16:43	Archivo HTML	5 KB	
apple-touch-icon-precomposed.png	13/10/2019 16:43	Archivo PNG	2 KB	

Voy a entrar el level11.

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/D
$ cd level11
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/D
$ pwd
/c/Users/Lorena Mendez Otero/Documents/03. CURS
$ ls
level12/
```

Creemos un archivo aquí denominado "leiagame. TXT". Haga un "ls", puede ver ese archivo aquí y el directorio "level12", voy a subir al directorio "level2" y voy a crear el siguiente archivo de nivel.

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01.
$ echo "Recursive Add Test" >> leiagame.txt
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01.
$ ls -al
total 5
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 23:05 ./
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 23:01 ../
-rw-r--r-- 1 Lorena Mendez Otero 197121 19 oct. 13 23:05 leiagame.txt
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 23:01 level12/
```

Visualización del resultado en Local:

Este equipo > Documentos > 03. CURSOS > 01. PROGRAMACION > 03. GIT > Projects > starter-web > level11				
Nombre	Fecha de modifica...	Tipo	Tamaño	
level12	13/10/2019 23:01	Carpeta de archivos		
leiagame.txt	13/10/2019 23:05	Documento de tex...	1 KB	

Creemos otro archivo en el level12 denominado "leigamelevel12.txt":

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACI
$ echo "El iogurt más sano es el natural" >> leiagamelevel12.txt
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACI
$ ls -al
total 1
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 23:12 ./
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 23:05 ../
-rw-r--r-- 1 Lorena Mendez Otero 197121 34 oct. 13 23:12 leiagamelevel12.txt
drwxr-xr-x 1 Lorena Mendez Otero 197121 0 oct. 13 23:01 level13/
```

Retrocedemos a nuestra raíz "web de inicio" del repositorio de Git. Entonces hagamos un "ls"; y n "estado de git". Entonces, el comando "git status" solo le mostrará el directorio de primer nivel; No le mostrará todos los archivos recursivamente hacia abajo.

```

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ pwd
/c/Users/Lorena Mendez Otero/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ ls
404.html          favicon.ico      index.html      robots.txt
apple-touch-icon-precomposed.png  fonts/          js/             simple.html
crossdomain.xml  hipster.txt     level11/
css/              humans.txt      README.md

```

Sin embargo, definitivamente puedo agregar todos los archivos de forma recursiva, simplemente usando un parámetro adicional en el "git add". Entonces escriba "git add", espacio, y si uso un punto, agregará todos los archivos de forma recursiva a partir de ahora.

```

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ git add .
warning: LF will be replaced by CRLF in level11/leia game.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in level11/level12/leia gamelevel12.txt.
The file will have its original line endings in your working directory

```

Ahora si hago un "estado de git", puedes ver ese "nivel1 / nivel1-archivo". txt "," nivel1 / nivel2 / nivel2-archivo. txt ", y así sucesivamente se han agregado al índice Git:

```

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
    new file:   level11/leia game.txt
    new file:   level11/level12/leia gamelevel12.txt

```

Ahora, si quisiera, simplemente puedo cometer esos cambios. Y solo voy a usar mi editor de texto para hacer el mensaje de confirmación "Agregar varios archivos de forma recursiva". Voy a guardar y cerrar, y el comando "commit" regresa, mostrando que cada uno de esos archivos se ha agregado recursivamente al repositorio de Git.

5.8. Backing Out Changes

Mostraremos cómo anular los cambios en su repositorio de Git. Dentro del level11 de mi repositorio starter-web tengo un archivo que lo voy a modificar añadiendo texto.

Guardo y luego cierro. Voy de vuelta en mi terminal hago un "estado de git", tenemos nuestro archivo modificado.

```

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web/level11 (master)
$ pwd
/c/Users/Lorena Mendez Otero/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web/level11

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web/level11 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

```

Avancemos y agreguemos esta versión del archivo al área de preparación de Git: entonces "git add level1-file".txt ", presione enter.

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAM
$ git add leiagame.txt
warning: LF will be replaced by CRLF in level11/leiagame.txt.
The file will have its original line endings in your working directory
```

Entonces, una cosa a tener en cuenta es que puede emitir la mayoría de los comandos en su repositorio Git desde cualquier lugar dentro de su repositorio Git.

Bien, ahora si hacemos un "estado de git", tenemos nuestro archivo modificado. Nuevamente, si abro ese archivo nuevamente. De vuelta en la terminal, digamos que decido que realmente no quiero esos cambios. Lo que puedo hacer es seguir las instrucciones que se encuentran en el comando "git status".

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAM
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   leiagame.txt
        new file:   level12/leiagamelevel12.txt
```

Entonces, lo que quiero hacer es retroceder los cambios desde el área de preparación, o, sin escena. Para hacer eso, voy a usar el comando reset de Git: "git reset HEAD", y luego el nombre del archivo "leiagame". TXT".

```
On branch master
Your branch is ahead of 'origin/master' by 4 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
        modified:   level1-file.txt
```

También podemos utilizar "git restore <file>" para descartar los cambios en el área de trabajo:

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   leiagame.txt
```

NO LOGRO CONSEGUIR LO MISMO QUE EN EL VIDEO!

Entonces, ese comando deshizo el archivo; entonces, si hago un "estado de git", ese archivo ha regresado a los cambios para que no se confirmen, que es solo el estado del directorio de trabajo. Si abro ese archivo, puede ver que este archivo no ha cambiado; todavía está en mi directorio de trabajo como los tres párrafos.

Entonces, digamos que no quiero ninguno de esos cambios para este archivo. Puedo descartar los cambios en mi directorio de trabajo actual usando el comando "**git checkout**". Así que voy

a seguir las instrucciones del comando "git status", así que "git checkout"; y luego dos guiones; y luego el nombre del archivo, entonces "nivel1-archivo. TXT".

Si hago eso, y ahora hago un "estado git", volveremos a un directorio de trabajo limpio. Además, si vuelvo a abrir ese archivo, puede ver que volvemos al estado del archivo que solo tiene dos párrafos; así que este es el mismo estado del archivo que se confirmó por última vez.

```
(master) level1 $ git checkout -- level1-file.txt
(master) level1 $ git status
On branch master
Your branch is ahead of 'origin/master' by 4 commits.
  (use "git push" to publish your local commits)

nothing to commit, working directory clean
```

NO LOGRO CONSEGUIRLO!

5.9. Renaming and Moving Files

Veremos cómo mover y cambiar el nombre de los archivos en Git a través de comandos.

Cambio de nombre de archivos

Para cambiar el nombre de un archivo utilizamos el comando "GIT MV" + espacio + nombre actual del fichero + nuevo nombre del fichero:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web/level11/level12 (master)
$ git mv leiagamelevel12.txt leiabestgame.txt

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web/level11/level12 (master)
$ ls
leiabestgame.txt  level13/
```

Si hago un "GIT Status", Git ha puesto en escena el hecho de que se ha producido el cambio de nombre:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   leiabestgame.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ../leiagame.txt
    ../test.txt
```

Git ha puesto en escena el hecho de que se ha producido el cambio de nombre, pero no lo ha cometido. Eso significa que podría retroceder el cambio de nombre.

Vamos a añadir "git add" + "git commit -m" para aceptar los cambios:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/
$ git add leiabestgame.txt
```

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/
$ git commit -m "Accept the changes"
[master 2046432] Accept the changes
1 file changed, 1 insertion(+)
create mode 100644 level11/level12/leiabestgame.txt

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

```

Recomendación: Cambiar el nombre de sus archivos antes de realizar cambios; esto facilitará a Git rastrear el hecho de que ha ocurrido el cambio de nombre.

¿Cómo retroceder al antiguo nombre de un archivo? => Utilice el comando "git mv". o bien "git mv" + actual nombre del archivo + nuevo nombre de archivo de vuelta al nombre de archivo anterior, presione Enter:

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git mv leiabestgame.txt leiabadgame.txt

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ ls
leiabadgame.txt  level13/

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    leiabestgame.txt -> leiabadgame.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ../leiagame.txt
        ../test.txt

```

Movimientos de archivos

Para mover un archivo de un directorio a otro, utilizaremos el comando "git mv" + "nombre del archivo a mover" + "carpeta de destino" para lograrlo:

```

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git mv leiabadgame.txt level13/

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ ls
level13/

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ cd level13/

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ ls
leiabadgame.txt  level15/

```

¿Cómo movemos los archivos para hacia atrás (a un nivel inferior)? => haciendo la tarea inversa con el comando "git mv".

5.10. Deleting Files

Demostraremos cómo eliminar archivos en Git de diversas formas.

Eliminar archivos NO rastreados por Git

La primera será eliminar un archivo que Git aún no está rastreando, utilizando el comando para eliminar archivos “**git rm**” + nombre del archivo a eliminar:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ ls
404.html          favicon.ico        humans.txt        README.md
apple-touch-icon-precomposed.png  fonts/            index.html       robots.txt
crossdomain.xml   gameover.txt      js/              simple.html
css/              hipster.txt       level11/

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ git rm gameover.txt
```

Git se quejará de que no puede coincidir con el archivo, porque Git aún no está rastreando este archivo:

```
fatal: pathspec 'gameover.txt' did not match any files
```

Entonces, eliminamos el archivo del sistema operativo con el comando “**rm**” + nombre del archivo a eliminar:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ rm gameover.txt

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ ls
404.html          css/              hipster.txt      js/              robots.txt
apple-touch-icon-precomposed.png  favicon.ico       humans.txt       level11/         simple.html
crossdomain.xml   fonts/            index.html       README.md
```

Eliminar archivos SÍ rastreados por Git

Para mostrar todos los archivos que Git está rastreando, introduzco el comando “**git ls-files**”:

```
Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ git ls-files
.htaccess
404.html
README.md
apple-touch-icon-precomposed.png
crossdomain.xml
css/bootstrap-theme.css
css/bootstrap-theme.css.map
css/bootstrap-theme.min.css
css/bootstrap.css
css/bootstrap.css.map
css/bootstrap.min.css
css/main.css
favicon.ico
fonts/glyphicons-halflings-regular.eot
fonts/glyphicons-halflings-regular.svg
fonts/glyphicons-halflings-regular.ttf
fonts/glyphicons-halflings-regular.woff
hipster.txt
humans.txt
index.html
js/main.js
js/plugins.js
js/vendor/bootstrap.js
js/vendor/bootstrap.min.js
js/vendor/jquery-1.11.0.min.js
js/vendor/modernizr-2.6.2-respond-1.1.0.min.js
level11/level12/level13/leibadgame.txt
robots.txt
simple.html
```


¿Cómo hacer que Git rastree un fichero? =>

Por tanto, para eliminar un archivo que git está rastreando es incorporar “git rm” + nombre del archivo a eliminar:

La eliminación aún no es permanente, necesitamos avanzar comprometiendo el cambio que se organiza. Para ello, introducimos un “git commit -m”, para el mensaje en línea "Eliminar nuevo archivo".

Anular eliminaciones por etapas

Vamos a eliminar el fichero hipster.txt => Vamos a añadir “git rm” + nombre del archivo a eliminar:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git rm hipster.txt
rm 'hipster.txt'

Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:   hipster.txt
```

El estado de git muestra que la eliminación está actualmente organizada. Nuevamente, eso significa que el archivo no se ha eliminado permanentemente, al menos en lo que respecta a Git. Por tanto, para eliminarlo definitivamente, incorporamos un “git reset HEAD” + nombre del archivo:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~
$ git reset HEAD hipster.txt
Unstaged changes after reset:
D    hipster.txt
```

Sin embargo, si hacemos un "ls", el "hipster". txt "todavía no está en nuestro directorio de trabajo.

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ ls
404.html  apple-touch-icon-precomposed.png  crossdomain.xml  css/  favicon.ico  fonts/  humans.txt  index.html  js/  level11/  'new file.txt'  README.md  robots.txt  simple.html
```

Entonces, si verificamos nuevamente con Git haciendo un "estado de git", entonces aquí confirmamos que Git está detectando que hay un archivo eliminado, eso es "hipster". txt ", actualmente se está rastreando pero falta en el directorio de trabajo.

Por lo tanto, el comando "reset HEAD" que especifica el archivo, simplemente eliminó la eliminación, no restauró el archivo al sistema de archivos. Por tanto, debemos descartar cualquier cambio temporal que hayamos realizado en el directorio de trabajo utilizando el comando "git checkout":

```
(master) starter-web $ git checkout -- hipster.txt
(master) starter-web $ ls
404.html                humans.txt
apple-touch-icon-precomposed.png  index.html
crossdomain.xml         js
css                      level1
favicon.ico              log
fonts                    robots.txt
hipster.txt              simple.html
```

Vamos a confirmarlo => "ls", ahora podemos ver que el "hipster".txt "está de vuelta en nuestro directorio.

Eliminación de una estructura de carpetas

Para eliminar toda la estructura de carpetas utilizaremos el comando **"rm -rf"**, espacio, y luego el nombre de la carpeta que deseo eliminar. El **"-r"** recorrerá recursivamente la estructura de la carpeta mientras que el **"-f"** forzará la eliminación.

```
(master) starter-web $ rm -rf level1
```

Al hacer un "ls" confirma que la carpeta "level1" y todos sus elementos secundarios ya no están. Si hago un "estado de git", Git observa todos los archivos que hemos eliminado que ha estado rastreando, básicamente todos los archivos de texto desde "nivel1" hacia abajo.

```
(master) starter-web $ git status
On branch master
Your branch is ahead of 'origin/master' by 13 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

       deleted:    level1/level1.txt
       deleted:    level1/level2/level2.txt
       deleted:    level1/level2/level3/level3.txt
```

Entonces, para organizar la eliminación de todos esos archivos a la vez, voy a emitir "git add -A". Posteriormente, si hacemos un "estado de git", la eliminación de todos esos archivos se ha organizado.

```

no changes added to commit (use "git add" and/or "git commit -a")
(master) starter-web $ git add -A
(master) starter-web $ git status
On branch master
Your branch is ahead of 'origin/master' by 13 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    level1/level1.txt
        deleted:    level1/level2/level2.txt
        deleted:    level1/level2/level3/level3.txt

```

Muy bien, sigamos adelante y vamos a confirmar esto: "git commit -m", con el mensaje de confirmación: "eliminando level1 y todos los elementos secundarios".

5.11. History

Mostraremos cómo ver la historia de Git, utilizando el comando "**git log**".

Comando de ayuda de Git: "git help"

Así que sigamos adelante y veamos qué nos da el comando predeterminado "git log": "git log". Entonces, puede ver que tenemos el historial de confirmación en orden cronológico inverso:

```

Lorena Mendez Otero@DESKTOP-5OQT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMAC
ION/03. GIT/Projects/starter-web (master)
$ git log
commit 204643237027084e9514a3fdce5d308fdc17fb1e (HEAD -> master)
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date:   Mon Oct 14 19:53:40 2019 +0200

    Accept the changes

commit df7a02519be9808665c82145274ea146d2822b0d
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date:   Sun Oct 13 21:24:23 2019 +0200

    Agregando más texto ipsum

commit ca7e29cdc57f08d25ba44ad679d11a2f65024192 (origin/master, origin/HEAD)
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date:   Sun Oct 13 20:58:50 2019 +0200

    Por favor, proceda con la subida del fichero.

commit 4beb7f097b54ab7cf08daf17972c3ab14432ae24
Merge: 5c05047 e73f914
Author: Jason Taylor <jason@jasongtaylor.com>
Date:   Wed Sep 2 02:39:19 2015 -0400

...skipping...
commit 204643237027084e9514a3fdce5d308fdc17fb1e (HEAD -> master)
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date:   Mon Oct 14 19:53:40 2019 +0200

    Accept the changes

commit df7a02519be9808665c82145274ea146d2822b0d
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date:   Sun Oct 13 21:24:23 2019 +0200

    Agregando más texto ipsum

```

La parte superior comienza con la última confirmación, y luego trabajamos hacia atrás en el tiempo a medida que bajamos.

El identificador SHA-1 es como Git identifica cada uno de los commits de manera única:

```
commit 204643237027084e9514a3fdce5d308fdc17fb1e (HEAD -> master)
```

Seguido por el nombre y la dirección de correo electrónico del autor, la fecha en que ocurrió la confirmación y luego, sangrado un poco, es el mensaje de confirmación real:

```
commit 204643237027084e9514a3fdce5d308fdc17fb1e (HEAD -> master)
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date: Mon Oct 14 19:53:40 2019 +0200

    Accept the changes
```

Para proporcionar un historial de los commit de forma abreviada inserte el comando "**git log --abbrev-commit**" en Git Bash:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents
$ git log --abbrev-commit
commit 2046432 (HEAD -> master)
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date: Mon Oct 14 19:53:40 2019 +0200

    Accept the changes
```

Para ver el historial de una forma más visual y resumida utilizo el comando "**git log --oneline --graph --decorate**" en Git Bash:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ git log --oneline --graph --decorate
* 2046432 (HEAD -> master) Accept the changes
* df7a025 Agregando más texto ipsum
* ca7e29c (origin/master, origin/HEAD) Por favor, proceda con la subida del fichero.
* 4beb7f0 Merge pull request #6 from jasongtaylor/feature-readme
* e73f914 Adding Purpose section to README
* 34f563b Adding README file
* 5c05047 Copying files from initializr project zip file and then creating simple.html as basis for super simple pages
```

Por tanto, "--oneline" comprimirá nuestras entradas en una línea, "--graph" proporcionará un gráfico ASCII que representa el gráfico de ramificación, "--decorate" agregará cualquier etiqueta o etiqueta o cualquier cosa que anote nuestros compromisos. De esta forma tengo una vista muy diferente de mi historial de confirmaciones. En la parte superior, tenemos la rama maestra que actualmente apunta a la última confirmación.

Es posible hacer búsquedas en git por número de commit introduciendo el comando "**git log + numero del commit**:"

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ git log 34f563b
commit 34f563b40c4c1519026d2e881c75e97728c45502
Author: Jason Taylor <jason@git.training>
Date: Wed Sep 2 02:06:05 2015 -0400

    Adding README file

commit 5c05047e53ecd0913a6f10000014d38829aaecdc
Author: Jason Taylor <jason@screencasts.pro>
Date: Thu Sep 11 22:14:19 2014 -0400

    Copying files from initializr project zip file and then creating simple.html as basis for super simple pages
```

Otra opción que podría usar con el comando "git log" es la búsqueda basada en fechas utilizando el comando "git log --since =" + entre comillas dobles "hace 3 días".

Para ver el historial de archivos renombrados utilizaremos el comando "**git log --follow**", eso seguirá los cambios de nombre; "-"; espacio; y luego la ruta al archivo:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03. GIT/Projects/starter-web (master)
$ git log --follow -- level11/level12
commit 204643237027084e9514a3f0ce5d308f0c17fb1e (HEAD -> master)
Author: Lorena Mendez <mendezotero.lorena@gmail.com>
Date: Mon Oct 14 19:53:40 2019 +0200

Accept the changes
```

Para mostrar información de un específico commit, utilizamos el comando "**git show**" + referencia del commit:

```
Lorena Mendez Otero@DESKTOP-50QT6GB MINGW64 ~/Documents/03. CURSOS/01. PROGRAMACION/03.
$ git show 5c05047e53ecd0913a6f10000014d38829aaecdc
commit 5c05047e53ecd0913a6f10000014d38829aaecdc
Author: Jason Taylor <jason@screencasts.pro>
Date: Thu Sep 11 22:14:19 2014 -0400

    Copying files from initializr project zip file and then creating simple.html as bas

diff --git a/.htaccess b/.htaccess
new file mode 100644
index 0000000..c36e576
--- /dev/null
+++ b/.htaccess
@@ -0,0 +1,665 @@
+# Apache Server Configs v2.2.0 | MIT License
+# https://github.com/h5bp/server-configs-apache
+
+# (!) Using '.htaccess' files slows down Apache, therefore, if you have access
+# to the main server config file (usually called 'httpd.conf'), you should add
+# this logic there: http://httpd.apache.org/docs/current/howto/htaccess.html.
+
+# #####
+# # CROSS-ORIGIN RESOURCE SHARING (CORS)                                     #
+# #####
+
+# -----
+# | Cross-domain AJAX requests                                             |
+# -----
+
+# Allow cross-origins AJAX requests.
+# http://code.google.com/p/html5security/wiki/CrossOriginRequestSecurity
+# http://enable-cors.org/
+
+# <IfModule mod_headers.c>
+#   Header set Access-Control-Allow-Origin "*"
+# </IfModule>
```

5.12. Git Alias

Vamos a demostrar cómo configurar alias dentro de Git (comandos más cortos que nos dan la misma información que un comando largo de Git).

Voy a crear mi propio comando "hist", utilizando el comando "config" de Git. Por tanto, escribo "git config --global", porque queremos que este alias esté disponible independientemente de mi repositorio, de modo que se guardará a nivel de usuario; espacio + "alias". "; y luego el nombre del comando.

```
(master) starter-web $ git config --global alias.hist "git log --all --graph --decorate --oneline"
```

Por tanto, cualquier configuración que comience con " alias. ", nos permitirá crear un alias con el nombre del comando; espacio; así, entre comillas, pondré el comando completo que quiero emitir, dejando fuera la parte " git " del comando.

5.13. Ignoring unwanted files and folders

Demostraremos cómo excluir archivos no deseados de nuestro repositorio de Git.

La facilidad que utiliza Git es un archivo de texto llamado ". **gitignore** " que Git usa para rastrear todos los archivos y carpetas que Git debería ignorar. Debemos crear un archivo nuevo denominado ". gitignore" en nuestro repositorio con el formato una carpeta "*. TXT".

El archivo gitignore es un archivo de texto como cualquier otro archivo, y debe agregarse a Git para el control de versiones, así que hagamos eso ahora: " git add. gitignore ". Hagamos un "estado de git" y luego un "git commit".

5.14. Cleanup and Back to Origin (GitHub)

Cubriremos los comandos básicos de Git sincronizando nuestros cambios con GitHub. Para ello, voy a mi directorio local e introduzco el comando "git pull origin master".

GitHub => Git: GIT PULL ORIGIN MASTER

Git => GitHub: GIT PUSH ORIGIN MASTER

Annex

Basic Linux Commands for Beginners

<https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners>