

# MANUAL DE USO: GIT Y GITHUB



<b>Introducción</b>	<b>3</b>
<b>Git --local-branching-on-the-cheap</b>	<b>3</b>
<b>Administración de un repositorio en Git</b>	<b>4</b>
Creación de un repositorio	4
Añadir archivos al repositorio	4
Definición de usuario e email	5
Hacer commit de los archivos locales	5
Información del repositorio y los commit	5
Resolución de conflictos	6
Crear excepciones	6
Crear ramas	7
<b>GitHub: Built for developers</b>	<b>8</b>
Creación de repositorios en GitHub	8
Sincronizar el repositorio local Git con el repositorio compartido de GitHub	9
Archivo README	10
Descargar el repositorio (Pull)	10

# Introducción

Las herramientas como Git o GitHub, son software de control de versiones (CVS en inglés, Concurrent Versions System).

Administran un proyecto durante su desarrollo, permitiendo al equipo de trabajo recuperar el proyecto de un determinado instante del desarrollo, control de versiones, compartir el proyecto, actualizar el proyecto... En definitiva, permite tener un mayor control durante la etapa de desarrollo.

Estas herramientas utilizan un repositorio, el cual es el lugar donde se almacena el proyecto. Este repositorio inicial (trunk) será el que se vaya modificando durante el desarrollo (branch) y una vez se haya modificado hasta cierto punto, se hayan añadido las funciones y objetivos fijados, el branch pasará a ser el trunk.

Este proceso se repetirá hasta que se finalice el desarrollo.

## Git --local-branching-on-the-cheap

Git es uno de los software de control de versiones más importantes, ya que es de gratuito y de código libre, pudiendo administrar tanto proyectos pequeños como grandes.

La web oficial [Git SCM](#) nos permitirá descargarlo, también dispone de un foro y un manual de usuario.

# Administración de un repositorio en Git

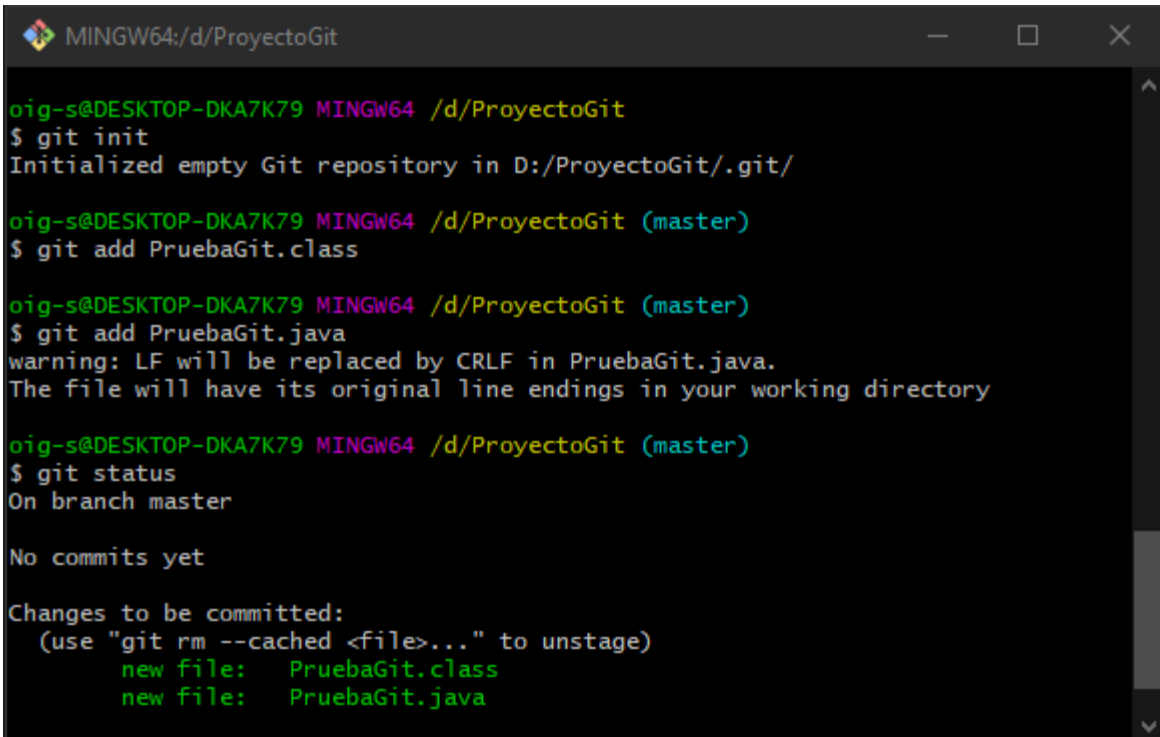
## Creación de un repositorio

Una vez tengamos instalado Git, lo primero que necesitaremos crear será el lugar donde se guarde nuestro proyecto, y si ya lo tenemos creado, necesitamos indicarle a Git donde se encuentra.

Iniciaremos Git Bash, navegaremos al directorio del proyecto mediante comandos Linux (cd). Una vez nos encontremos dentro del proyecto, lanzaremos el comando `git init` para que Git nos cree los archivos necesarios para realizar el control de versiones.

## Añadir archivos al repositorio

También debemos añadir los archivos del proyecto con `git add [archivo]`.



```
MINGW64:/d/ProyectoGit
oig-s@DESKTOP-DKA7K79 MINGW64 /d/ProyectoGit
$ git init
Initialized empty Git repository in D:/ProyectoGit/.git/

oig-s@DESKTOP-DKA7K79 MINGW64 /d/ProyectoGit (master)
$ git add PruebaGit.class

oig-s@DESKTOP-DKA7K79 MINGW64 /d/ProyectoGit (master)
$ git add PruebaGit.java
warning: LF will be replaced by CRLF in PruebaGit.java.
The file will have its original line endings in your working directory

oig-s@DESKTOP-DKA7K79 MINGW64 /d/ProyectoGit (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   PruebaGit.class
        new file:   PruebaGit.java
```

Con `git add .` añadiremos todos los archivos sin añadir.

## Definicion de usuario e email

Con los comandos `git config --global user.name 'nombre'` y `git config --global user.email 'email'` añadiremos un nombre y un email a todos los cambios que hagamos.

```
oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git config --global user.name 'Sergio'

oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git config --global user.email 'sergio.mm116@gmail.com'
```

## Hacer commit de los archivos locales

Una vez hayamos añadido los archivos para que formen parte del repositorio, será necesario realizar un commit para que estos se actualicen y guarden como tal. Para esto lanzaremos el commando `git commit -m 'Comentario'`.

Si hacemos un `git status` podremos ver que ya tendremos un commit hecho y que no hay diferencias entre los archivos del trunk y de nuestro repositorio local.

## Información del repositorio y los commit

Podremos ver la información de los commit, entre ellos el usuario, la fecha, el comentario y una hash identificador mediante `git log`. Además de las diferentes ramas.

```
oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git log
commit ab75e687b4aebccf353ab09289575d6d2a0171ee (HEAD -> master)
Author: sergiomm116 <sergio.mm116@gmail.com>
Date:   Tue Mar 24 21:50:33 2020 +0100

    Primer commit
```

## Resolución de conflictos

Durante el desarrollo será necesario que modifiquemos los archivos, cuando hagamos esto, si lanzamos el comando `git status`, Git nos indicará que hay archivos modificados.

```
oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git status
On branch master
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:   PruebaGit.class
        modified:   PruebaGit.java

no changes added to commit (use "git add" and/or "git commit -a")
```

En estos casos, Git nos ofrece varias opciones, revertir los cambios con `git restore [archivo]`, visualizar los cambios con `git diff` o agregar los cambios al repositorio con `git commit -a -m 'Comentario'`.

## Crear excepciones

Para crear excepciones dentro de nuestro repositorio, crearemos un archivo llamado `.gitignore`. Dentro de este archivo añadiremos los archivos o directorios que queremos que no se añadan a nuestro repositorio. Una vez hecho esto, necesitaremos añadir el archivo `.gitignore` a nuestro repositorio con un commit.

```
oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git add .gitignore

oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git commit -m 'agregado .gitignore'
[master 99ad2d2] agregado .gitignore
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
```

## Crear ramas

Mediante el comando `git branch [nombre_rama]`, podremos crear nuevas ramas (branch) en nuestro proyecto y mediante `git branch` podremos listar las ramas que tenemos hasta el momento.

```
oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git branch branch1

oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git branch
  branch1
* master
```

Para trabajar con otra rama usamos `git checkout [nombre_rama]`. Todo lo que modifiquemos en una rama, no modificará al resto.

# GitHub: Built for developers

GitHub es una herramienta de desarrollo colaborativo que usa el sistema Git para compartir los repositorios entre sus usuarios. Los repositorios se guardan de manera pública, por lo que cualquier usuario podrá visualizarlos, para crear repositorios privados es necesario tener una cuenta de pago.


## Creación de repositorios en GitHub

Tras registrarnos, nos dirigiremos a [github.com/new](https://github.com/new) y rellenaremos los campos. Tras esto clickaremos en “Create repository”.

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)



---

Owner	Repository name *
 sergiomm116 ▾	/ ProyectoGit ✓

Great repository names are short and memorable. Need inspiration? How about **refactored-couscous**?

Description (optional)

Repositorio de prueba para la asignatura de ENDES - DAM1

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

---

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: **None** ▾

Add a license: **None** ▾ ⓘ

---

Create repository

---



## Sincronizar el repositorio local Git con el repositorio compartido de GitHub

Tras crear nuestro repositorio compartido en GitHub, debemos sincronizarlo con nuestros archivos locales Git. Para ello, GitHub nos proporciona el comando:

```
git remote add origin https://github.com/sergiomm116/ProyectoGit.git
```

Pegaremos este comando en la consola de Git y después este: `git push -u origin master`.

Nos pedirá nuestro login de GitHub, ya que es necesario para poder modificar el repositorio.

```
oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git remote add origin https://github.com/sergiomm116/ProyectoGit.git

oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git push -u origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.48 KiB | 755.00 KiB/s, done.
Total 11 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/sergiomm116/ProyectoGit.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

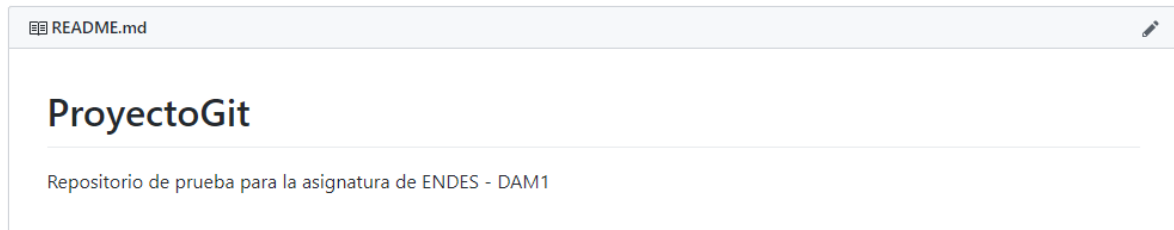
Ahora tendremos nuestro repositorio local en nuestra página de GitHub donde será posible descargarlo con un simple click con el botón “Clone or download”.

The screenshot shows the GitHub repository page for 'sergiomm116 / ProyectoGit'. The repository is described as 'Repositorio de prueba para la asignatura de ENDES - DAM1'. It has 4 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The 'master' branch is selected. A table of recent commits is displayed, showing the creation of a branch, adding a .gitignore file, and adding source files.

Commit	Message	Time
branch1	Creacion de branch	35 seconds ago
.gitignore	agregado .gitignore	35 minutes ago
PruebaGit.class	Añadido sout Adios Mundo	1 hour ago
PruebaGit.java	Añadido sout Adios Mundo	1 hour ago

## Archivo README

Mediante este archivo, daremos información sobre el repositorio a cualquier persona ajena a él. Podremos hacerlo desde el boton “Add a README”.



## Descargar el repositorio (Pull)

Dentro de Git Bash, usaremos el comando `git clone [direccion_GitHub]`. Esto nos clona el repositorio en el directorio donde nos encontremos posicionados en Git Bash.

```
oig-s@DESKTOP-DKA7K79 MINGW64 /d/proyectogit (master)
$ git clone https://github.com/sergiomm116/ProyectoGit.git
Cloning into 'ProyectoGit'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 18 (delta 3), reused 14 (delta 2), pack-reused 0
Receiving objects: 100% (18/18), 2.43 KiB | 1.21 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```