

María Prados

# Gestión de Base de Datos

## Primer repositorio

Explicación de las instalaciones, configuración y uso.

1. Instalar **GitHub**, para que el ordenador reconozca los comandos de git.  
Crearnos una cuenta en Git-hub.com

Podemos comprobar primero desde Powershell si nuestro ordenador tuviera instalada alguna versión previa de Git, abriendo la consola powershell, desde inicio o el buscador de Windows, y escribiendo el comando **Git – version**.

Si no lo tenemos instalado nos aparece esto, porque no nos reconoce el comando:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

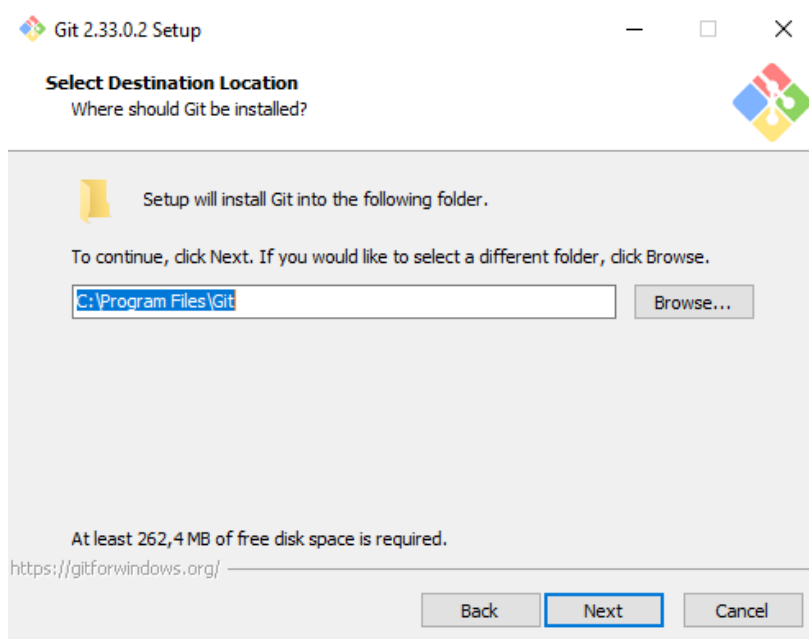
PS C:\Users\darks> git --version
git : El término 'git' no se reconoce como nombre de un cmdlet, función, archivo de script o programa ejecutable.
Compruebe si escribió correctamente el nombre o, si incluyó una ruta de acceso, compruebe que dicha ruta es correcta e
inténtelo de nuevo.
En línea: 1 Carácter: 1
+ git --version
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (git:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

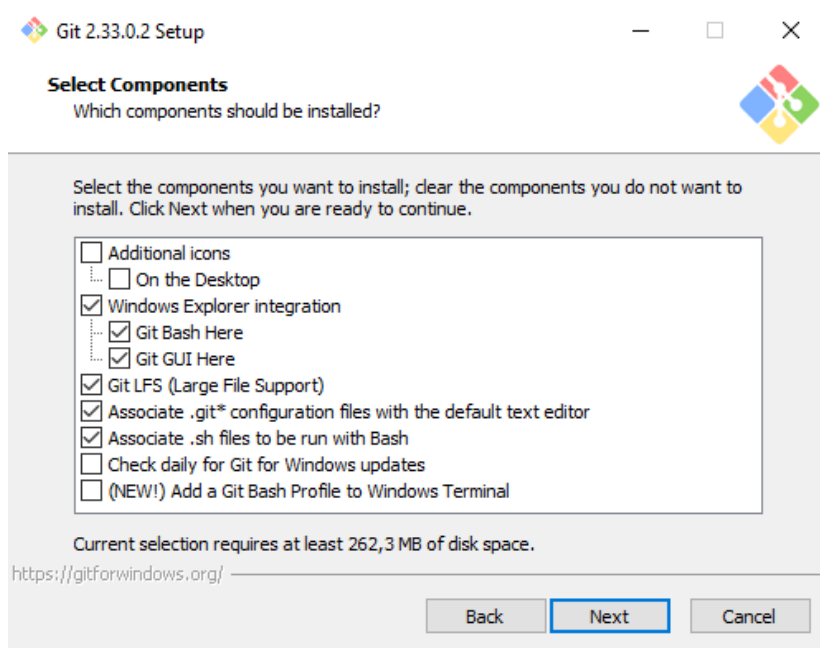
PS C:\Users\darks>
```

Desde la url, [www.git-scm.com](http://www.git-scm.com), nos bajamos la última versión, en este caso la 2.33.

Versión para Windows x64, que serían las características de nuestro ordenador.

Los instalamos, elegimos la ruta, todo por defecto:





Podemos comprobar después en powershell que ya nos reconoce la versión:

```
PS C:\Users\darks> git --version
git version 2.33.0.windows.2
PS C:\Users\darks>
```

## 2. Creación de la cuenta

Nos vamos a la página principal Git-Hub.com y nos creamos una nueva cuenta, donde poder subir nuestros repositorios.

**“Sign up”** para registrarnos, donde elegimos un nuevo usuario y contraseña que luego nos va a valer para todo el uso de nuestras cuentas.

Una vez instalado volvemos a powershell, para comprobar que ya nos reconoce la versión.

Tendremos que verificar nuestra cuenta en nuestro correo electrónico.

## 3. Creación del repositorio.

Repositorio local:

Elegimos una ruta en nuestro ordenador y nos creamos una carpeta para nuestro proyecto (Proyecto\_01).

Repositorio Git-Hub:

En git-hub.com le damos a Create repository, donde pondremos el nombre al nuevo proyecto, en este caso (proyecto\_01).

## 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 \*



PradosCamposMaria ▾



Great repository names are short and memorable. Need inspiration? How about [fictional-octo-succotash?](#)

Description (optional)

Anotación: Ponemos siempre el cero delante para que cuando superen los 10 proyectos nos ordena bien los primeros, e intentamos no usar de momento ni tildes ni espacios en las rutas.

#### 4. Abrimos Powershell, escribimos git (para activar los comandos de git)

```
PS C:\Users\darks> git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone          Clone a repository into a new directory
  init           Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add            Add file contents to the index
  mv            Move or rename a file, a directory, or a symlink
  restore        Restore working tree files
  rm            Remove files from the working tree and from the index
  sparse-checkout Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
  bisect        Use binary search to find the commit that introduced a bug
  diff          Show changes between commits, commit and working tree, etc
  grep          Print lines matching a pattern
  log           Show commit logs
  show          Show various types of objects
  status        Show the working tree status

grow, mark and tweak your common history
  branch        List, create, or delete branches
  commit        Record changes to the repository
  merge         Join two or more development histories together
  rebase        Reapply commits on top of another base tip
  reset         Reset current HEAD to the specified state
  switch        Switch branches
  tag           Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch         Download objects and refs from another repository
  pull          Fetch from and integrate with another repository or a local branch
  push          Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

Nos vamos a nuestra ubicación.

Con `cd` + nombre de la ruta. **(Siempre nos aseguramos de estar en nuestra carpeta de trabajo).**

Introducimos el comando **git init**, el cual nos crea en el directorio una carpeta oculta `.git` que nos servirá para conectar el repositorio local con el repositorio en github.

Manualmente creamos en el explorador de Windows, 2 carpetas, una llamada `doc` y otra llamada `scr`, a las dos les metemos un archivo de prueba.txt para que git nos reconozca las carpetas, SI ESTAN VACIAS NO LAS RECONOCE.

Volvemos a powershell, e introducimos **git status**, siempre que queramos saber, cual es el estado de las cosas de nuestro repositorio local, y luego (cuando vinculemos el repositorio en web), el estado respecto al que está colgado en github.

En color verde, las que, si lo están, en color rojo, las que tienen alguna modificación no actualizada.

**Git add**, añade cosas al repositorio, puedo añadir archivos o carpetas independientes o marcando un punto (`.`) Añado todo lo que contenga.

La primera vez que intento añadir cosas, me va a pedir añadir mis datos, “usuario” e “email”.

Explicación de los comandos

**Git Branch** selecciona la rama de trabajo en la que estás puesto que puedes crear varias, desde donde partir para continuar con el trabajo en este caso, trabajaremos en la principal con **Git Branch -M** (main).

**Git remote add origin** (link) nuestro repositorio de github.

**Git push -u**, sube nuestras tareas a la red.

**Git commit -m** “Añadimos un concepto o comentario para el archivo”

Con `git Status` volvemos a comprobar que los archivos están en verde.

Otros comandos:

**Git pull** (Si la información del repositorio es más actual que la local), para traernos la información al ordenador.

**Git Clone**, Nos clona de nuevo el repositorio en el nuevo pc.

**Git Download**, Descarga directamente los mismos archivos al ordenador.