



Git + GitHub



“

Versiones de una misma historia

Git

Sistema de control de
versiones

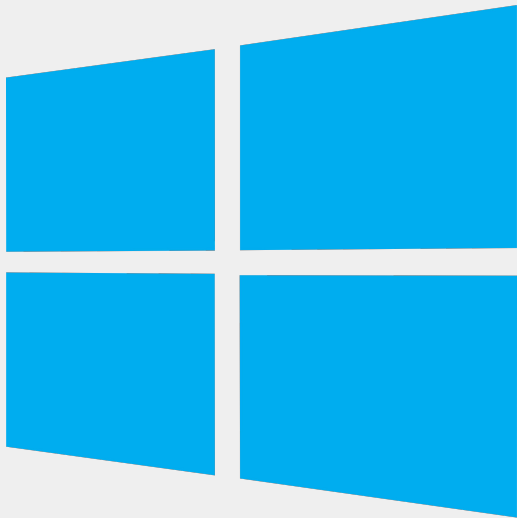
GitHub

Plataforma de desarrollo
colaborativo

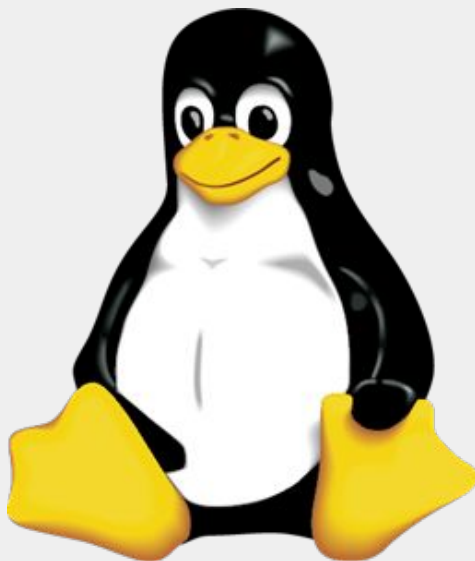


git

Configuración



Windows



Linux



OSX

Configuración [Windows



Configuración [Windows

Git 2.20.1 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change

<https://gitforwindows.org/>

Next >

Cancel

Git 2.20.1 Setup

Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to
install. Click Next when you are ready to continue.

- ☐ Additional icons
 - ☐ On the Desktop
- ☒ Windows Explorer integration
 - ☒ Git Bash Here
 - ☒ Git GUI Here
- ☒ Git LFS (Large File Support)
- ☒ Associate .git* configuration files with the default text editor
- ☒ Associate .sh files to be run with Bash
- ☐ Use a TrueType font in all console windows
- ☐ Check daily for Git for Windows updates

Current selection requires at least 244,9 MB of disk space.

<https://gitforwindows.org/>

< Back

Next >

Cancel

Configuración [Windows

Git 2.20.1 Setup

Choosing the default editor used by Git

Which editor would you like Git to use?

Use Vim (the ubiquitous text editor) as Git's default editor

Use the Nano editor by default

Use Vim (the ubiquitous text editor) as Git's default editor

Use Notepad++ as Git's default editor

Use Visual Studio Code as Git's default editor

Use Visual Studio Code Insiders as Git's default editor

Use Sublime Text as Git's default editor

Use Atom as Git's default editor

Select other editor as Git's default editor

may set it to some other editor of your choice.

Adjusting your PATH environment

How would you like to use Git from the command line?

☐ Use Git from Git Bash only

This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

☒ Git from the command line and also from 3rd-party software

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows I

☐ Use Git and optional Unix tools from the Command Prompt

Both Git and the optional Unix tools will be added to your PATH.

Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

<https://gitforwindows.org/>

<https://gitforwindows.org/>

< Back

Next >

Cancel

< Back

Next >

Cancel

Configuración [Windows]

The image shows the Git 2.20.1 Setup wizard with several overlapping windows. The main window in the background is titled "Git 2.20.1 Setup" and shows the "Choosing HTTPS transport backend" step. It has two radio button options: "Use the OpenSSL library" (selected) and "Use the native Windows Secure Channel". Below these are descriptions of each option. To the right, another window titled "Configuring the line ending conversions" shows three radio button options: "Checkout Windows-style, commit LF" (selected), "Checkout as-is, commit Unix-style", and "Checkout as-is, commit as-is". Below these are descriptions. To the right of that, a third window titled "Configuring the terminal emulator to use with Git Bash" shows two radio button options: "Use MinTTY (the default terminal of MSYS2)" (selected) and "Use Windows' default console window". Below these are descriptions. To the right of that, a fourth window titled "Configuring extra options" shows three checkboxes: "Enable file system caching" (unchecked), "Enable Git Credential Manager" (checked), and "Enable symbolic links" (checked). Below these are descriptions. At the bottom of the main window, there is a progress bar and a "< E" button. At the bottom right, there are three buttons: "< Back", "Install", and "Cancel".

Git 2.20.1 Setup

Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?

☒ **Use the OpenSSL library**
Server certificates will be validated using the OpenSSL library.

☐ **Use the native Windows Secure Channel**
Server certificates will be validated using Windows. This option also allows you to use your company's certificates distributed e.g. via Active Directory Domain Services.

Configuring the line ending conversions
How should Git treat line endings in text files?

☒ **Checkout Windows-style, commit LF**
Git will convert LF to CRLF when checking out text files, CRLF will be converted to LF when committing text files. This is the recommended setting on Windows.

☐ **Checkout as-is, commit Unix-style**
Git will not perform any conversion when checking out text files, CRLF will be converted to LF when committing text files. This is the recommended setting on Unix.

☐ **Checkout as-is, commit as-is**
Git will not perform any conversions when checking out text files. Choosing this option is not recommended for projects ("core.autocrlf" is set to "false").

Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

☐ **Use MinTTY (the default terminal of MSYS2)**
Git Bash will use MinTTY as terminal emulator, non-rectangular selections and a Unicode font (as interactive Python) must be launched via "cmd.exe /c git bash".

☒ **Use Windows' default console window**
Git will use the default console window of Windows. With Win32 console programs such as interact you have very limited default scroll-back, needs to be configured in order to display non-ASCII characters correctly. The default window was not freely resizable and it only allowed a limited number of lines.

Configuring extra options
Which features would you like to enable?

☐ **Enable file system caching**
File system data will be read in bulk and cached in memory for certain operations ("core.fsck" is set to "true"). This provides a significant performance boost.

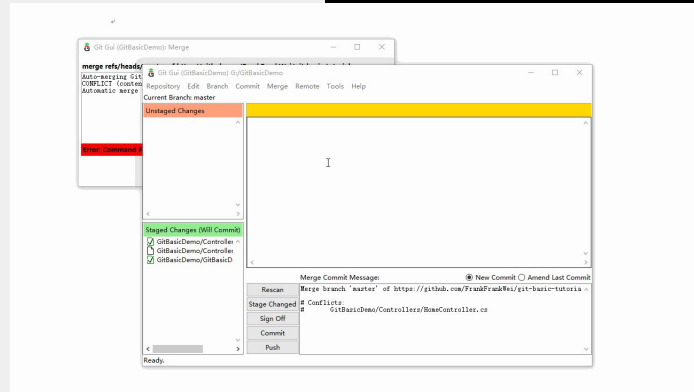
☒ **Enable Git Credential Manager**
The [Git Credential Manager for Windows](#) provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or later).

☒ **Enable symbolic links**
Enable [symbolic links](#) (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.

<https://gitforwindows.org/>

A partir de aquí podemos usar Git por...

- ★ línea de comandos
- ★ interfaz: Git GUI



Conceptos clave



entendiendo Git



Se recomienda
crear **repositorios**
para almacenar
datos de proyectos,
en cada una de sus
versiones.

Se recomienda añadir
mensajes

identificativos por
cada cambio
realizado en una
misma versión.





Se recomienda crear **etiquetas** para cada nueva versión publicada de un software.

Se recomienda
crear **ramas** para
trabajar sobre diferentes
versiones de tu repositorio a la vez.



Configuración inicial

primeros pasos con Git

Configuración base

Primero, comprobamos si en nuestra máquina ya hay configurada una cuenta de usuario con el comando...

```
git config --global -l
```

Configuración base

Algunos comandos para establecer datos de usuario son:

```
git config --global user.name "Name"
```

```
git config --global user.email "Mail"
```



Repositorios

init

Crea un directorio nuevo, ábrelo y ejecuta

```
git init
```

para crear un nuevo repositorio de git.

Tu **repositorio local** está compuesto por tres «árboles» administrados por **git**.

El primero es tu **Directorio de trabajo**, que contiene los archivos; el segundo es el **Index**, que actúa como una **zona intermedia**, y el último es el **HEAD**, que apunta al último **commit** realizado.



status

Para ver los archivos cuyo control de cambios gestiona git, usa el comando

```
git status
```

status

Aparecen en **rojo** aquellos archivos que no están siendo gestionados por git.

Aparecen en **verde** aquellos archivos que están en la **zona intermedia** esperando a ser confirmados.

checkout

Puedes descartar cambios locales usando

```
git checkout -- <filename>
```

Este comando reemplaza los cambios en tu directorio de trabajo con el último contenido de HEAD.

add

Puedes añadir archivos o registrar cambios

```
git add <filename>
```

```
git add .
```

(pasarán de rojo a verde)

reset

Puedes deshacer la acción del add

```
git reset HEAD <filename>
```

```
git reset
```

commit

Para perpetrar estos cambios usa

```
git commit -m "commit message"
```

Ahora el archivo está incluido en el **HEAD** de tu copia local, pero aún no en tu repositorio remoto.

commit's controls

Para cambiar el mensaje del último commit:

```
git commit --amend
```

Para mostrar (y poder eliminar) los últimos *n* commits:

```
git rebase -i HEAD~n
```

diff

Se puede revisar las diferencias entre la **versión de control** de un archivo y la **actual**

```
git diff
```

Para descartar cambios, `git checkout .`

Para guardar, `git add .` + `git commit`

log

También podemos ver un histórico de todos los commits en un repositorio

```
git log
```

Arriba se mostrará el más reciente.

clone

Crea una copia local del repositorio ejecutando

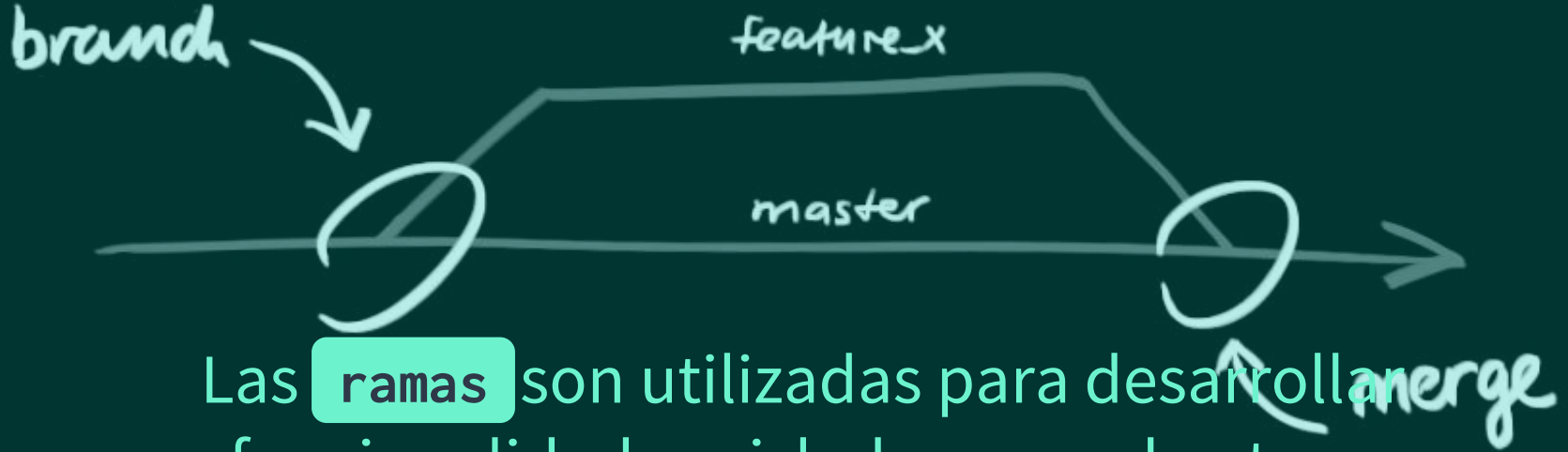
```
git clone /path/to/repository
```

Si utilizas un servidor remoto, ejecuta

```
git clone username@host:/path/to/repository
```


Ramas





Las **ramas** son utilizadas para desarrollar funcionalidades aisladas unas de otras.

La rama master es la rama «por defecto» cuando creas un repositorio. Crea nuevas ramas durante el desarrollo y fusi6nalas a la rama principal cuando termines.

ramas

Crea una nueva rama «feature_x»
y cámbiate a ella usando

```
git checkout -b feature_x
```

vuelve a la rama principal

```
git checkout master
```

y borra la rama

```
git branch -d feature_x
```

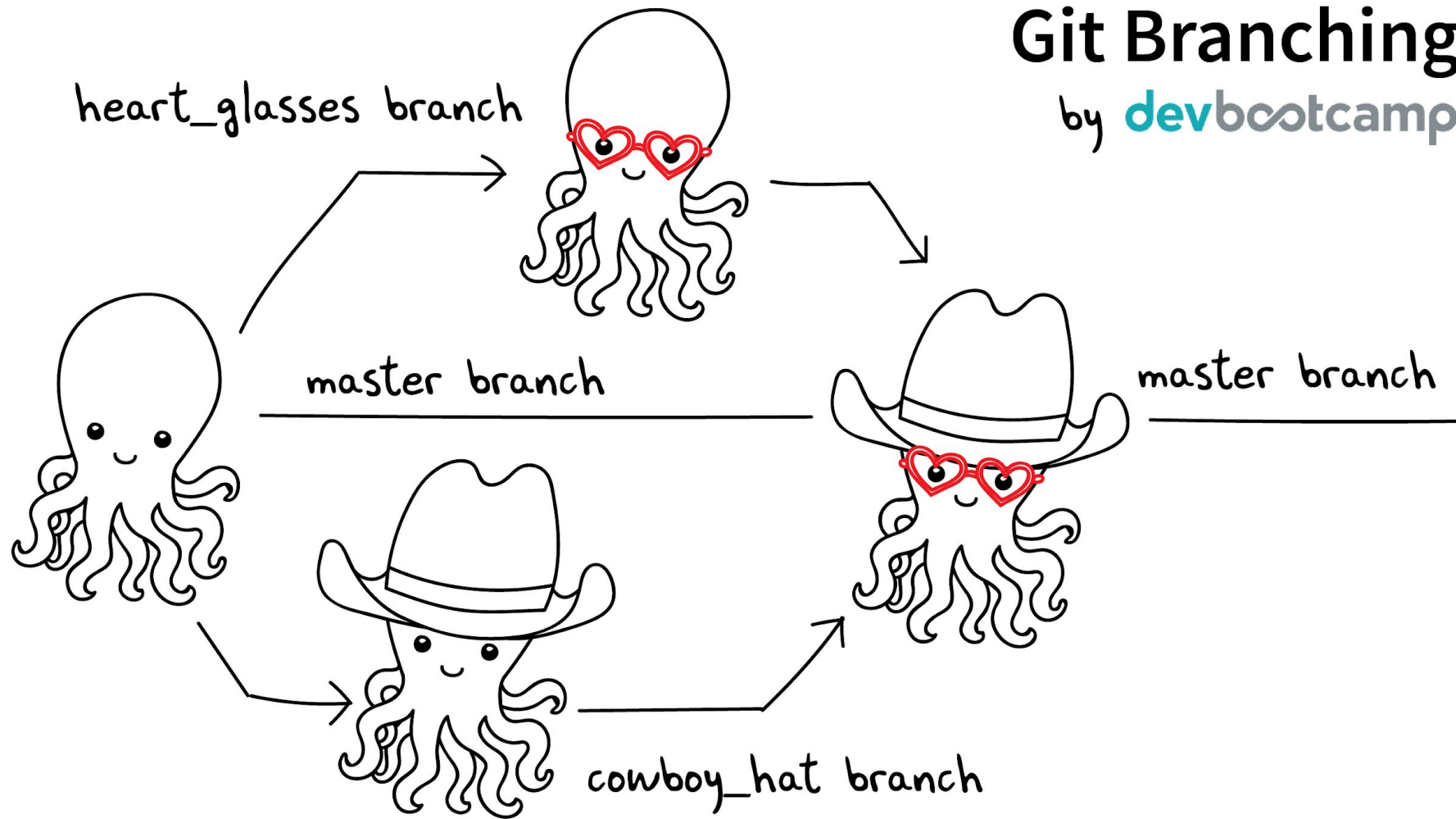
ramas

Para fusionar otra rama a tu rama activa (por ejemplo master), utiliza

```
git merge <branch>
```

Git Branching

by **dev**bootcamp





Etiquetas

etiquetas

Crea una nueva etiqueta llamada 1.0.0
ejecutando

```
git tag 1.0.0 <commitID>
```

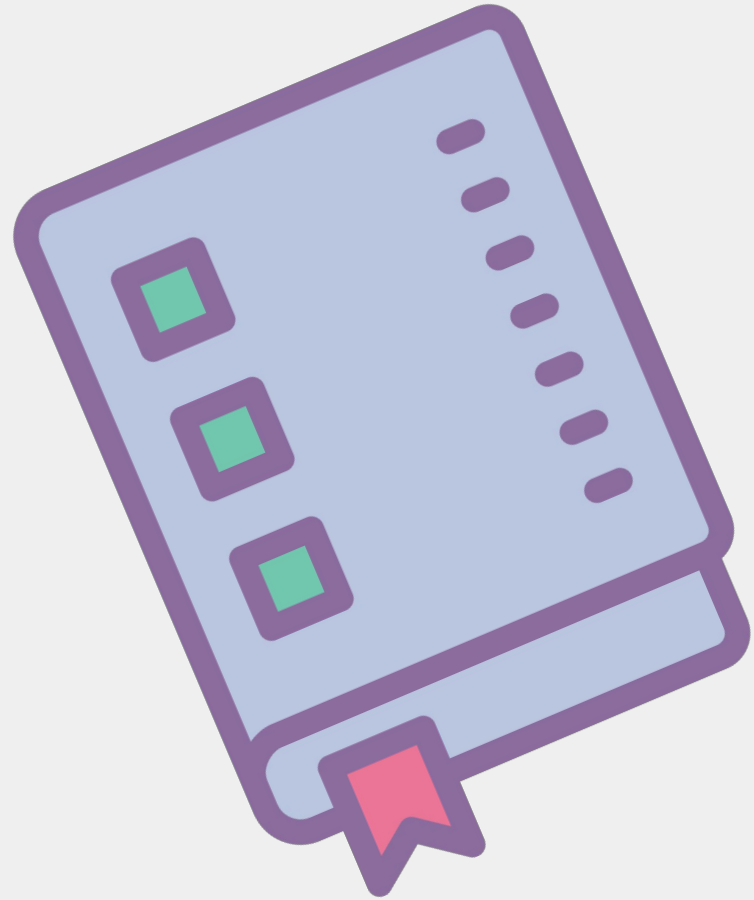
<commitID> son los 10 caracteres del
commit id al cual quieres referirte.

etiquetas

Puedes obtener el commit id con

```
git log
```


Repositorio remoto



push

Para enviar estos cambios a tu repositorio remoto ejecuta

```
git push origin master
```

Reemplaza **master** por la rama a la que quieres enviar tus cambios.

push branch

Una rama nueva no estará disponible para los demás a menos que la subas (push) a tu repositorio remoto

```
git push origin <branch>
```

pull

Para actualizar tu repositorio local al commit más nuevo, ejecuta

```
git pull
```

en tu directorio de trabajo para bajar y fusionar los cambios remotos.

remote add

Si no has clonado un repositorio ya existente y quieres conectar tu repositorio local a un repositorio remoto, usa

```
git remote add origin <server>
```

reemplazar cambios locales

Los cambios que ya han sido agregados al **Index**, así como los nuevos archivos, se mantendrán como estaban.

reemplazar cambios locales

Por otro lado, si quieres deshacer todos los cambios locales y commits, puedes traer la última versión del servidor y apuntar a tu copia local principal para sustituirla

```
git fetch origin
```

```
git reset --hard origin/master
```

Links de interés

Web Git

Cientes Gráficos

ProGit (ES)

Guías GitHub





GitHub

Crea tu cuenta en GitHub



Step 1:
Create personal account



Step 2:
Choose your plan



Step 3:
Tailor your experience

Create your personal account

Username

This will be your username. You can add the name of your organization later.

Email address

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

You'll love GitHub

Unlimited collaborators

Unlimited public repositories

- ✓ Great communication
- ✓ Frictionless development
- ✓ Open source community

Crea tu cuenta gratuita en GitHub

Welcome to GitHub

You're a few steps away from building better software, @Imorafp.



Completed
Set up your account



Step 2:
Choose your plan



Step 3:
Personalize your experience

Choose your plan

With tools developers love and the world's largest open source community, there's no wrong choice.



Free

The basics of GitHub for every developer

\$0

per month

Includes:

- ∞ Unlimited public and private repositories
- ✓ 3 collaborators for private repositories
- ✓ Issues and bug tracking
- ✓ Project management

Are you a student? Get access to the best developer tools for free with the [GitHub Student Developer Pack](#).



Pro

Pro tools for developers with advanced requirements

\$7


per month

[\(view in EUR\)](#)

Includes:



- ∞ Unlimited public and private repositories
- ∞ Unlimited collaborators
- ✓ Issues and bug tracking
- ✓ Project management
- ✓ [Advanced tools and insights](#)


Crea tu repositorio en GitHub

Owner:  Imoraft ▾ / Repository name *: first-repo ✓


Great repository names are short and memorable. Need inspiration? How about **bookish-octo-robot**.

Description (optional):
Mi primer repositorio en GitHub

☒  **Public**  SERÁ LISTADO EN TU PERFIL PÚBLICO
Anyone can see this repository. You choose who can commit.

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

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾ | Add a license: **None** ▾ 

ESTA OPCIÓN ES UNA
AYUDA DE GITHUB SI
TODAVÍA NO HAS
CREADO UNO LOCAL
EN TU ORDENADOR.

.GITIGNORE → LISTA DE ARCHIVOS SOBRE LOS QUE GIT NO HARÁ SEGUIMIENTO
LICENSE → TIPO DE LICENCIA

Trae tu repositorio de GitHub a local

- ★ El nombre de tu repositorio local debe coincidir con el de tu repositorio en **GitHub**.
- ★ Asegúrate de que **HTTPS** está seleccionado.
- ★ Copia la dirección de tu repositorio en GitHub.
- ★ Agrega tu repositorio remoto a tu máquina local:

```
git remote add <NOMBRE-REMOTO> <URL>
```

Si tienes **GitHub Desktop**, se crea automáticamente un remoto llamado **origin** en tu repositorio local. En este caso, sólo necesitarás indicar qué URL asociar con **origin**:

```
git remote set-url origin <URL>
```

Actualiza tu repositorio en GitHub con tu repositorio local

- ★ Empuja (**push**) los cambios que has realizado en local a tu repositorio remoto:

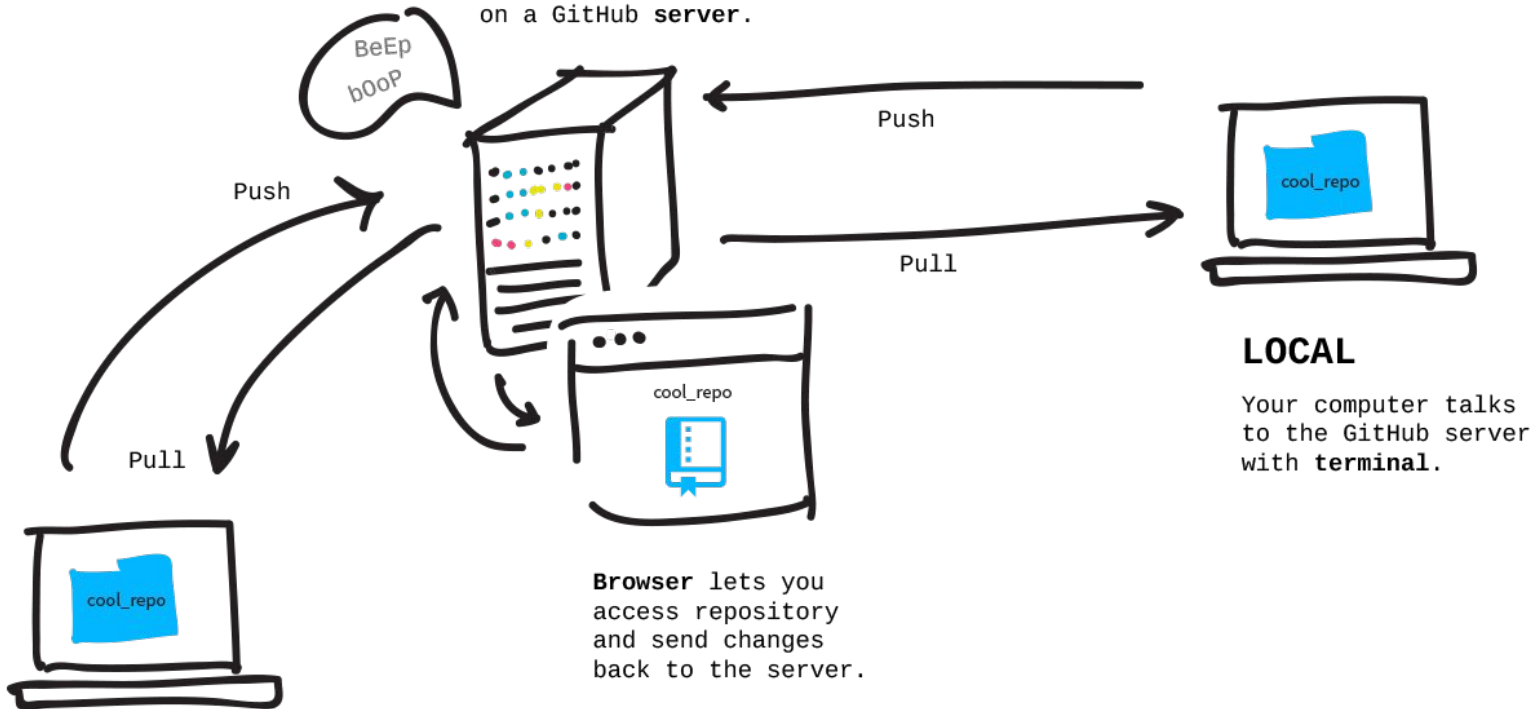
```
git push <NOMBRE-REMOTO> <NOMBRE-RAMA>
```

```
git push origin master
```

- ★ Ve a la página de tu repositorio remoto en GitHub.com y actualízala para verificar que los cambios se han subido.

REMOTE

Repositories live on a GitHub server.



LOCAL

Someone else's computer talks to the GitHub server.

LOCAL

Your computer talks to the GitHub server with **terminal**.

Otros comandos para conexiones remotas

Agregar conexiones remotas

```
git remote add <NOMBRE-REMOTO> <URL>
```

Modificar la URL de un remoto

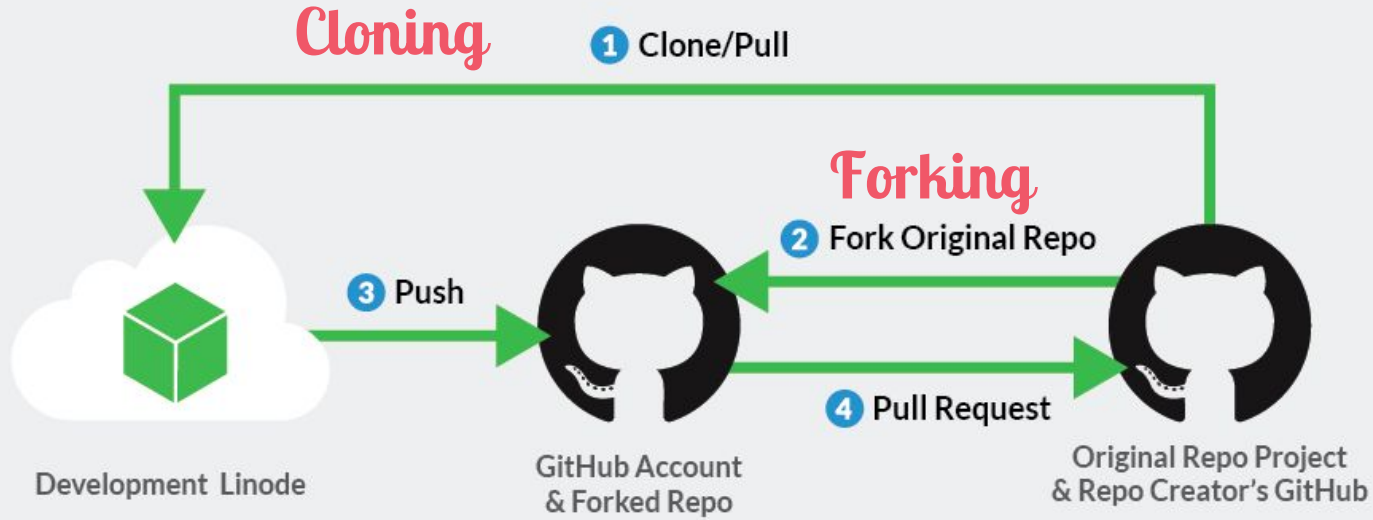
```
git remote set-url <NOMBRE-REMOTO> <URL>
```

Ver las direcciones remotas

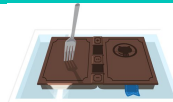
```
git remote -v
```


Participa en un proyecto existente en GitHub

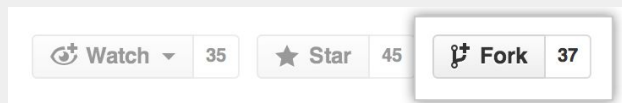
Git and GitHub



Bifurca un repositorio de GitHub



- ★ Busca un repositorio de terceros en **GitHub**.
- ★ Haz clic en el botón Fork:



- ★ Después, podrás clonar tu repositorio bifurcado (**fork**) en tu repositorio local.

Clona un repositorio de GitHub

- ★ El nombre de tu repositorio local debe coincidir con el de tu repositorio en **GitHub**.
- ★ Asegúrate de que **HTTPS** está seleccionado.
- ★ Copia la dirección de tu repositorio en GitHub.
- ★ Agrega tu repositorio remoto a tu máquina local:

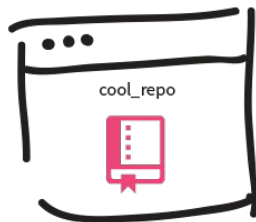
```
git remote add <NOMBRE-REMOTO> <URL>
```

Si tienes **GitHub Desktop**, se crea automáticamente un remoto llamado **origin** en tu repositorio local. En este caso, sólo necesitarás indicar qué URL asociar con **origin**:

```
git remote set-url origin <URL>
```

REMOTE

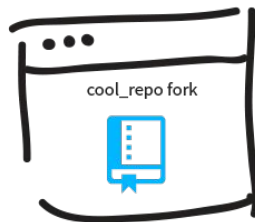
Someone else's repository.



Fork!

REMOTE

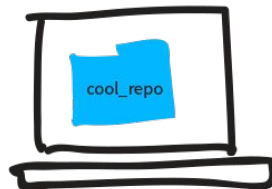
Your fork of the repository.



Clone to
your computer
from GitHub.

Push and Pull to your
fork 'origin'.

Pull from 'upstream'
changes to original.



LOCAL

Use your computer's
terminal to talk to
two repositories via
two remotes to the
GitHub servers.


Crea una rama en GitHub

Just another repository — Edit

 1 commit

 1 branch



 branch: **master** ▼

hello-world / 

Initial commit



hubot authored just now



[README.md](#)

Initial



README.md

Haz cambios en tu rama con GitHub

Crea una petición **pull** con GitHub

Haz un merge con GitHub

Imagenes GitHub

