

GIT

HERRAMIENTAS - GIT

<https://git-scm.com>



Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.



About

The advantages of Git compared to other source control systems.



Documentation

Command reference pages, Pro Git book content, videos and other material.



Downloads

GUI clients and binary releases for all major platforms.



Community

Get involved! Bug reporting, mailing list, chat, development and more.



HERRAMIENTAS - GIT

Comandos de configuración esenciales:

' **git config --global user.name "Nombre"** ' : Configurar el nombre del usuario.

' **git config --global user.email "nobre@gmail.com"** ' : Configurar el email del usuario.

Crear el fichero .gitignore para incluir los ficheros que no queremos subir al repositorio

' **git config --list** ' : Muestra las configuraciones del sistema.

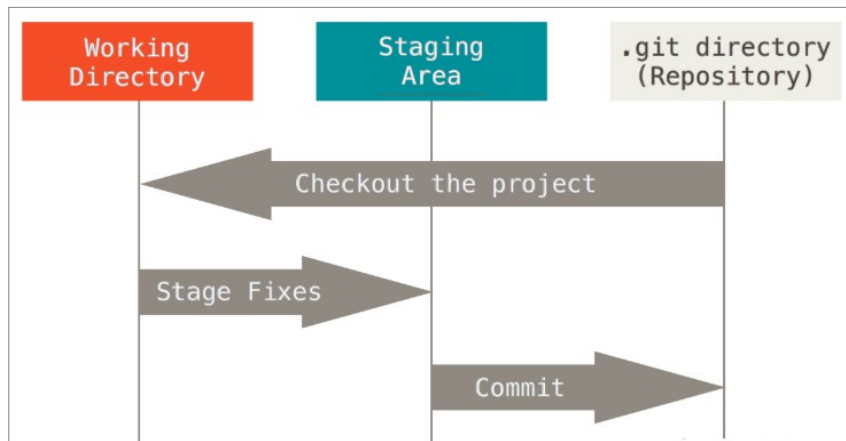
Otros comandos de configuración :

' **git config --global core.editor "code --wait"** ' : Configurar el editor de texto para git.

' **git config --global merge.tool "p4merge"** ' : Configurar la herramienta para comparar diferencias.

' **git config core.autocrlf true** ' : Configurar el salto de línea. (Si da un 'warning: LF will be replaced ...')

HERRAMIENTAS - GIT



Directorio de trabajo: Son los ficheros que hay en nuestro directorio.

Area de Stage: Son los ficheros en los que ha habido cambios y están pendientes de subirse.

Repositorio: Contiene la historia de los ficheros que se han ido subiendo.

Comandos esenciales:

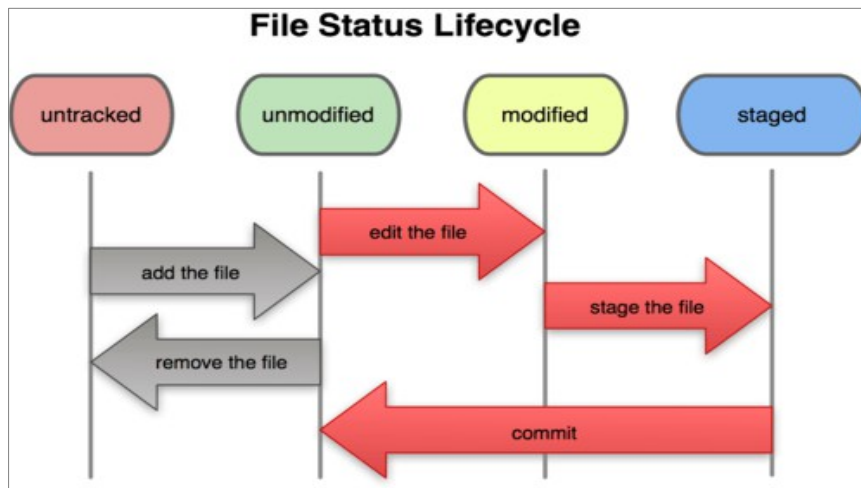
'git init' : Crea un repositorio del directorio activo. Crea el directorio.

'git add .': Añade todos los ficheros al area de stage.

'git status': Muestra los elementos que están en el stage pendientes de subir al repositorio.

'git commit -m "Comentario"': Sube los ficheros del stage al repositorio.

'git checkout': Copia los ficheros del repositorio a la zona de trabajo.



HERRAMIENTAS - GIT



'git log' : Muestra la historia de cambios que hay en el repositorio.

'git log --oneline --decorate --all --graph': Muestra la historia de cambios de manera más esquematizada. Recomendación: Dada la cantidad de elementos se recomienda utilizar un alias (ver abajo). (Solo conoce el pasado).

'git reset --soft 42634534': Vuelve atrás en el tiempo dentro del repositorio. (Ojo: Solo es el repositorio, para verlo en la zona de trabajo habrá que hacer " git checkout -f "). (Viaja en el tiempo a un punto determinado).

'git reset --soft HEAD^': Vuelve atrás en el tiempo (solo 1 paso anterior)

'git reset --hard 42634534': Hace reset --soft y checkout -f

'git checkout -f ': Recupera la versión que hay en ese momento en el repositorio

'git reflog': Muestra la historia del repositorio (incluido puntos realizados en el futuro).

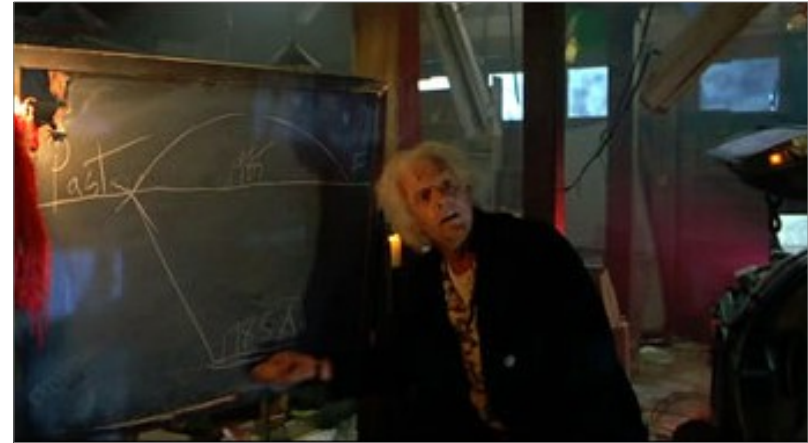
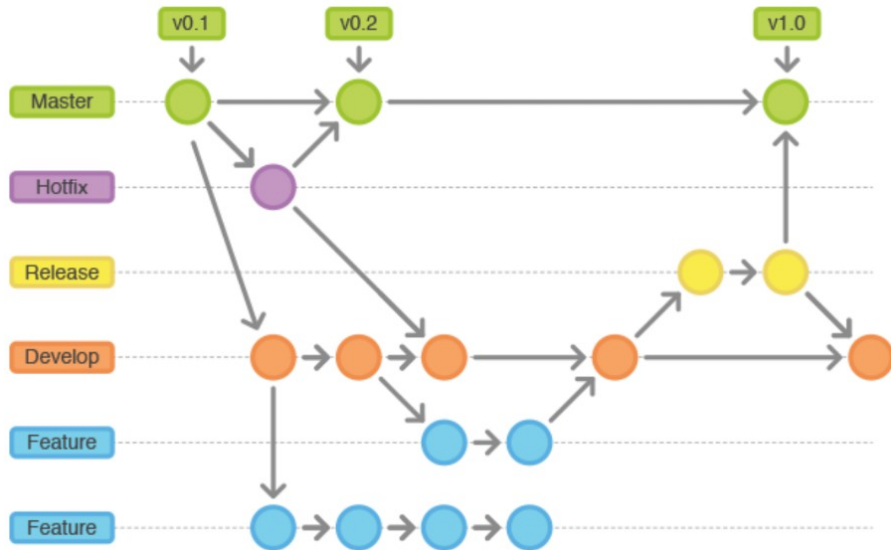
'git reset nombreFichero': Quita del repositorio un fichero (o ficheros)

Crear un Alias:

'git config --global alias.lg "log --oneline --decorate --all --graph ': Sirve para crear un nuevo comando con nombre lg.

'git lg ': Utilización del nuevo alias

HERRAMIENTAS - GIT



'git branch nombreRama' : Crea una nueva rama con el nombre indicado.

'git checkout nombreRama' : Cambiar a otra rama (en el espacio de trabajo y stage).

'git status -s -b': Muestra la rama que estamos editando.

'git diff master nombreRama': Muestra las diferencias entre dos ramas.

'git merge nombreRama': Incorpora los cambios de una rama sobre otra.

'git branch' : Muestra las ramas que tenemos.

'git branch -d nombreRama' : Borra la rama indicada.

HERRAMIENTAS - GIT

```
D:\PruebasGit>git merge rama1
Auto-merging composer.json
CONFLICT (content): Merge conflict in composer.json
Automatic merge failed; fix conflicts and then commit the result.
```

```
<<<<<< HEAD
{222
=====
{111
>>>>>> rama1
```

'**git branch nombreRama**': Crea una nueva rama con el nombre indicado.

'**git checkout nombreRama**': Cambiar a otra rama (en el espacio de trabajo y stage).

'**git status -s -b**': Muestra la rama que estamos editando.

'**git diff master nombreRama**': Muestra las diferencias entre dos ramas.

'**git merge nombreRama**': Incorpora los cambios de una rama sobre otra.

'**git branch**': Muestra las ramas que tenemos.

'**git branch -d nombreRama**': Borra la rama indicada.

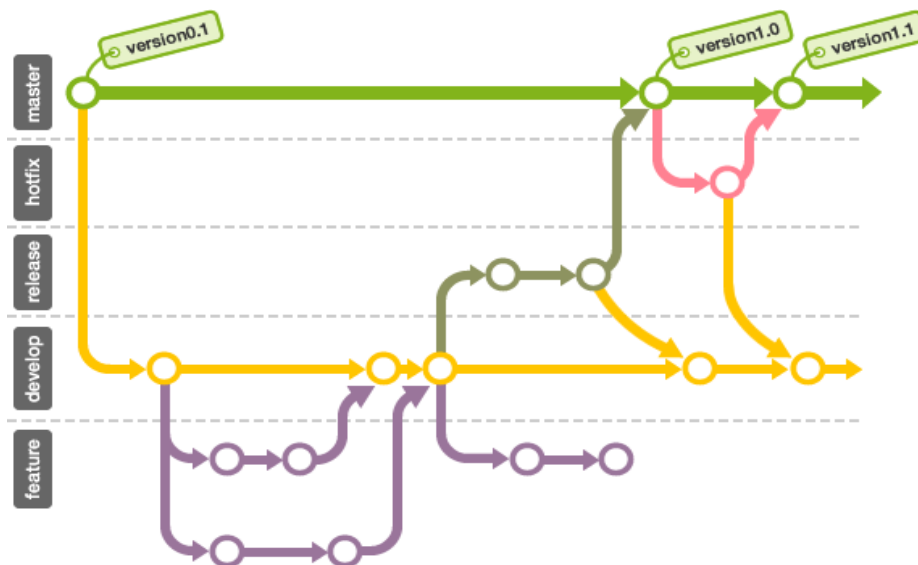
HERRAMIENTAS - GIT

'git tag -a v1.0.0 -m "Version 1.0.0"': Crea una etiqueta sobre el punto actual.

'git tag': Ver la lista de etiquetas creadas.

'git tag -a v0.1.0 4232123 -m "Versión alfa"': Crea una etiqueta sobre un punto anterior.

'git show v0.1.0': Muestra información de una etiqueta concreta.



HERRAMIENTAS - GIT

Taller de GIT - 1

1. Cree una carpeta junto a a los demás ejercicios del proyecto llamada `ejerciciogit`
 2. Dentro de esa carpeta, inicialice el proyecto de git
-
3. Dentro del repositorio cree un archivo llamado `README.md`, ingrese el siguiente contenido:
Información
Fichero para aprender a utilizar GIT
-
4.
 - Agregue el README.md al stage o escenario
 - Realice el primer commit con el mensaje `Creación del readme`
-
5.
 - Cree una carpeta dentro del proyecto llamada `logs`
 - Dentro de la carpeta logs, cree un archivo llamado `hoy.log`
 - Crear otro archivo llamado `historia.log`
-
6. Agregue únicamente el archivo `hoy.log` al stage

HERRAMIENTAS - GIT

7. Realice el segundo commit con el archivo `hoy.log`, y en el mensaje coloque "Creamos el archivo hoy.log"

8.

- Vamos a crear ahora el archivo `.gitignore`
 - Ignoremos completamente el archivo `historia.log`
-

9.

- Agregue al stage el archivo `.gitignore`
 - Realice un commit unicamente con el archivo `.gitignore`
-

10.

- Borre la carpeta LOGS
 - Borre el archivo index.html
 - Borre el archivo .gitignore
 - El único archivo que debe de quedar, es el README.md
-

11. Reconstruir todo lo borrado con un único comando. (El archivo historia.log debe de aparecer)

¿Por qué el archivo `historia.log` no apareció?

Debe de ser capaz de analizar el por qué

HERRAMIENTAS - GIT

Modificaciones de emergencia: Stash

Comandos esenciales:

'git stash' : Hace una copia de seguridad del directorio de trabajo en un área de trabajo temporal para resolver cosas de emergencia. Por lo tanto, almacena los ficheros pero con temporalidad no permanente. En nuestro directorio de trabajo recuperamos la ultima versión subida del repositorio para arreglar la emergencia. Cuando se haya resuelto la emergencia se podrá recuperar el stash del área temporal y lo borraremos.

'git stash save "Poner aquí un comentario"' : Pone un comentario sobre el stash para darnos más información

'git stash list' : Muestra los stash creados.

'git stash pop' : Recupera la copia temporal al directorio de trabajo realizando un merge con los cambios que se hayan arreglado por la emergencia. Una vez recuperada la copia, borra el stash del area temporal.

Otros comandos:

'git stash apply' : Restaura la ultima copia temporal. (Sin borrar el area temporal).

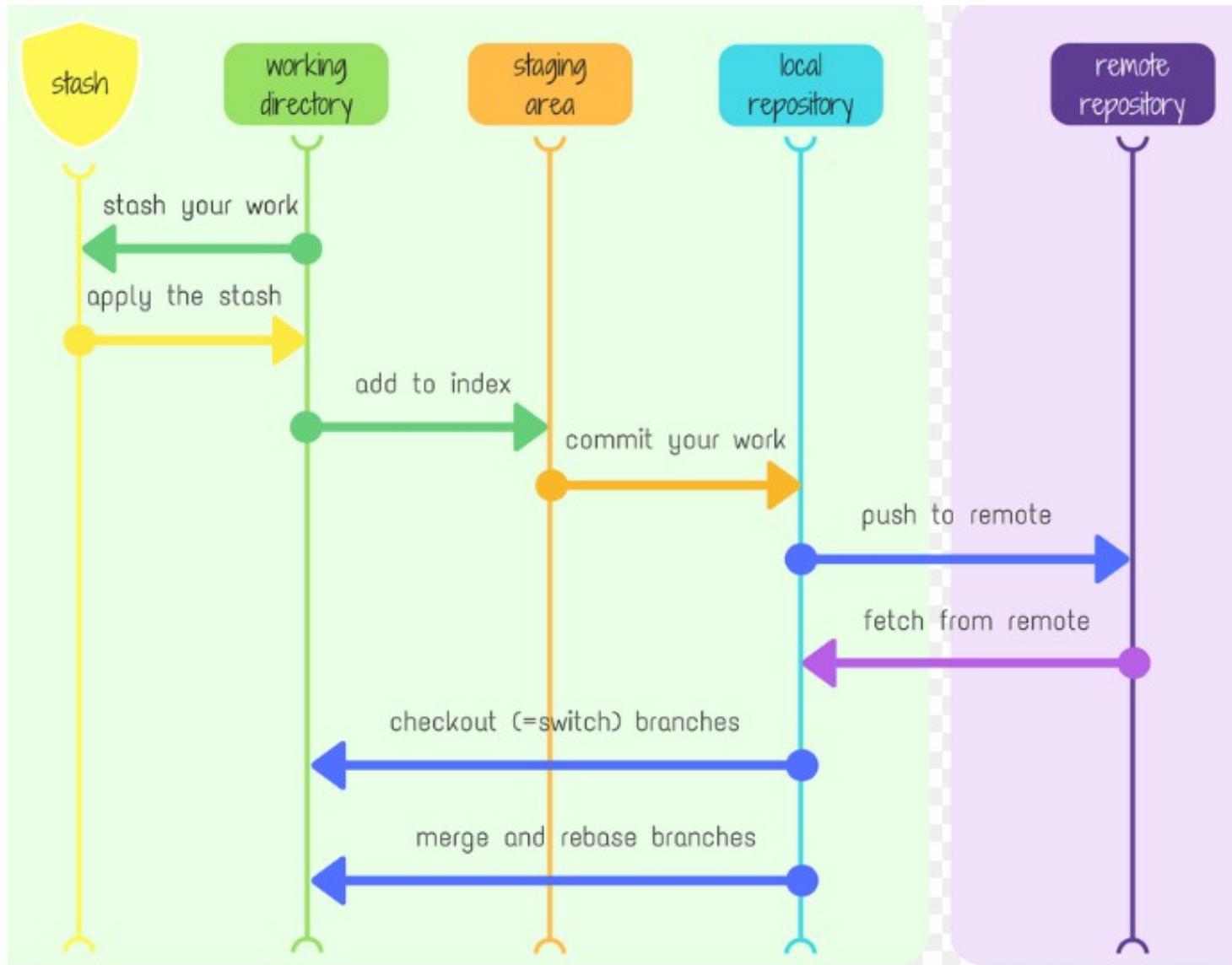
'git stash drop' : Borra la primera entrada stash del área temporal.

'git stash list --stat' : Muestra todos los stash de una manera más detallada

'git show stash' o 'git show stash@{1}' : Muestra información de los cambios de un stash concreto

'git stash clear' : Borra todos los stash.

HERRAMIENTAS - GIT



HERRAMIENTAS - GIT

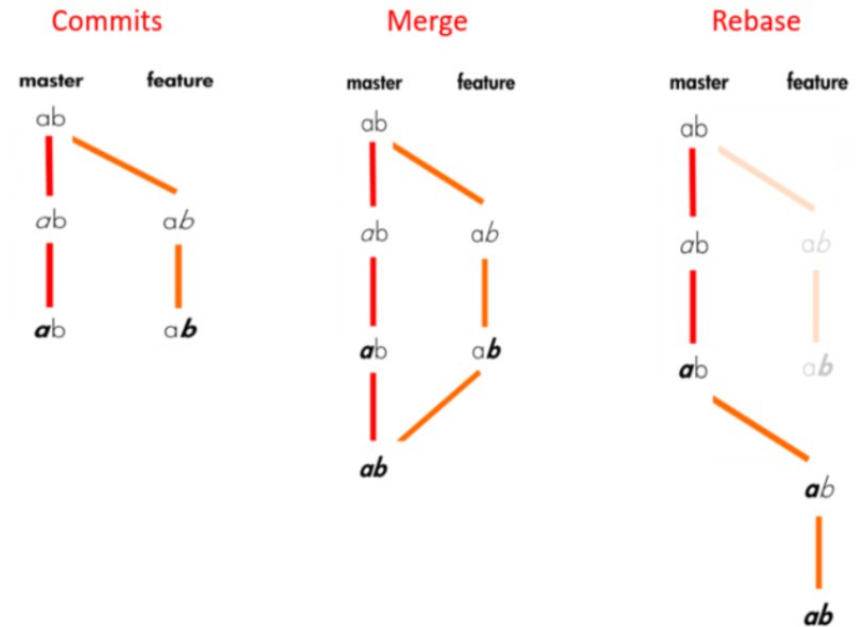
Modificaciones de emergencia: Rebase

Comandos esenciales:

'**git rebase master**': Aplica sobre la rama todos los cambios que están en el master. Ha diferencia del merge, se generan todos los puntos de commit realizados en el master. Es como si la rama se hubiese creado en otro punto.

Otros comandos:

Utilizando "git rebase -i" es posible editar la historia y en un editor y utilizar las palabras "Squash" y "Reword" para unir o separar commits.



HERRAMIENTAS - GIT

1 Guión VS 2 Guiones

`git log --online` Con dos guiones se expresa que a continuación va a ir una palabra que representa un comando

`git commit -am "My commit"` Con un guión se expresa que pueden ir una o varias letras, cada una representa un comando.

HERRAMIENTAS - GIT

Servidor remoto Git: Es posible que además deseemos guardar los cambios en un servidor remoto, bien por compartir el código entre diferentes usuarios o bien para no depender de una sola máquina.

Servidores de GIT externos:

- **BitBucket:** <https://bitbucket.org/>
- **GitLab:** <https://gitlab.com/>

Servidores GIT para instalar:

- **Gitosis**

Operaciones:

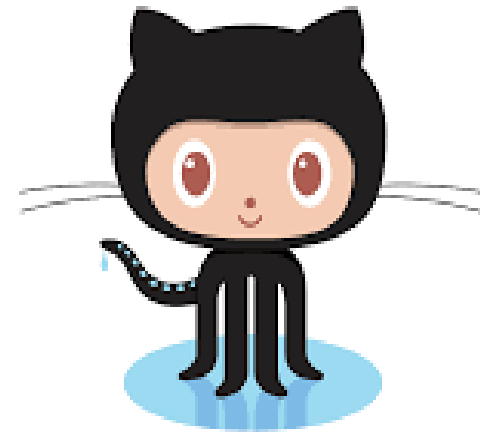
Push: Enviar los cambios de nuestro repositorio a un servidor remoto.

Pull: Recibir los cambios a nuestro repositorio de un servidor remoto.

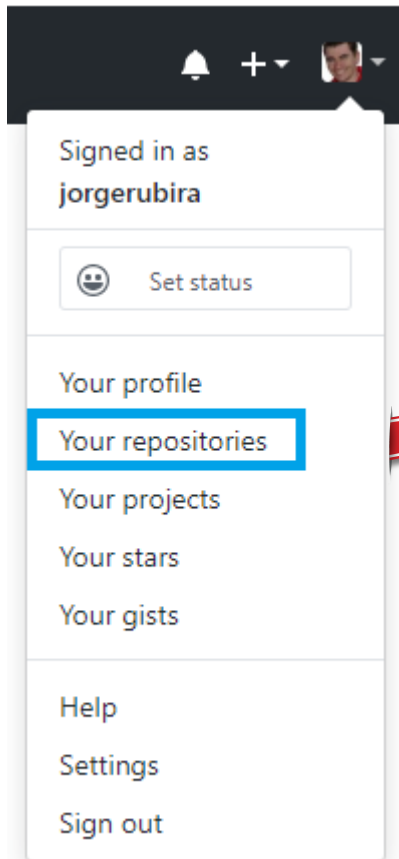
HERRAMIENTAS - GIT

GitHub: Es uno de los repositorios más populares. De manera gratuita ofrece:


- Repositorios ilimitados.
- Subir ficheros ilimitados.
- Push, Pull y Clone ilimitados.
- Wikis y estadísticas.
- Organizaciones ilimitadas.



HERRAMIENTAS - GIT



Signed in as
jorgerubira

 Set status

Your profile

Your repositories

Your projects

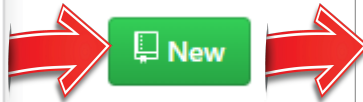
Your stars

Your gists

Help



Settings

Sign out



Create a new repository

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

Owner:  **jorgerubira** / Repository name: 

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


Description (optional)

☒ **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: | Add a license: 

Create repository

HERRAMIENTAS - GIT

jorgerubira / 01pruebainicial Watch 0 Star 0 Fork 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# 01pruebainicial" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/jorgerubira/01pruebainicial.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/jorgerubira/01pruebainicial.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

💡 **ProTip!** Use the URL for this page when adding GitHub as a remote.

HERRAMIENTAS - GIT

Trabajo con servidor remoto

Comandos esenciales:

'git remote add origin URLRemota' : Nos conectamos a un repositorio remoto. Origin es el nombre del repositorio.

'git remote -v' : Comprobar las conexiones remotas que tenemos configuradas.

'git push -u origin master' : Sube una rama indicada (master). El comando -u nos recuerda la rama utilizada para futuros push.

'git push' : Una vez especificada la rama con el comando -u. Se puede utilizar simplemente "git push".

'git push -- tags' : Sube todos los Tags del repositorio local.

'git pull origin master' : Obtiene todos los cambios del repositorio remoto.

'git clone URLRemota' : Crea una copia del proyecto remoto sin estar conectado.

'git clone URLRemota nombreCarpeta' : Igual que el clone pero el nombre de la carpeta que genera se puede definir.

'git fetch' : Es parecido al pull pero sin hacer merge.

HERRAMIENTAS - GIT

Readme.md: Si subimos un fichero llamado README.md lo pondrá como documentación del proyecto.

```
<> Edit file  Preview changes  Spaces 2 Soft wrap

1  # Objetivos de la repositorio
2
3  Este proyecto se encarga de manejar los planes de la liga de la justicia
4
5
6  ## Notas
7  Pueden hacer lo que quieran...
8
9
10 # h1
11 ## h2
12 ### h3
13 #### h4
14 ##### h5
15
16 Un gran poder requiere _una_ gran responsabilidad
17 > Ben Parker
18
19 1. item
20 2. item
21 3. item
22   * subitem
23   * subitem
24   * subitem
25
26 ![Batman](https://s-media-cache-ak0.pinimg.com/736x/e5/a0/69/e5a06942fa42823c88be5f3a834e063d--fantastic-art-bat-family.jpg)
```

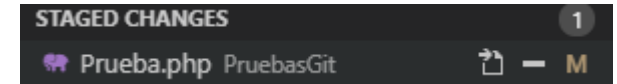
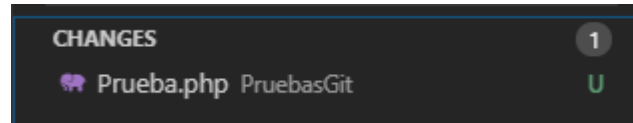
HERRAMIENTAS - GIT

Visual Studio Code: Nos permite trabajar con GIT sin necesidad de extensiones adicionales.
Solo hay que pulsar el icono:



Los códigos por fichero son:

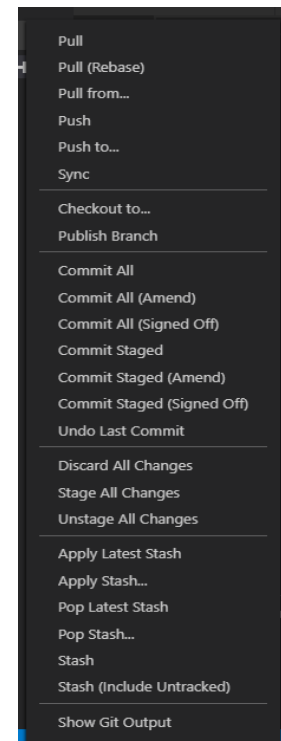
U – Untracked
A – Para añadir
M – Para modificar
D – Para borrar



Para subir fichero pulsar el botón de commit:



Finalmente existe un menú donde se pueden hacer el resto de operaciones.



HERRAMIENTAS - GIT

Los conflictos quedar representados gráficamente.

Accept Current Change → Toma los cambios que están en local.

Accept Incoming Change → Toma los cambios que hay en el repositorio

```
<?php
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
echo 'Hola mundo 2';
=====
echo 'Hola mundo 3';
>>>>>> ba3787c2d499cd43447ef7e2b923e1f735280480 (Incoming Change)
```