

GIT -- Github

ING. JAIDER OSPINA NAVAS.

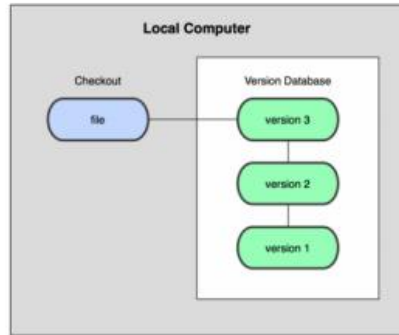


Control de Versiones.

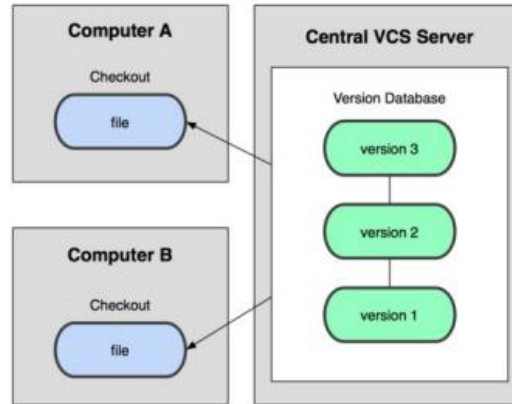
“Gestionar el cambio”

“Creando Inteligencia”

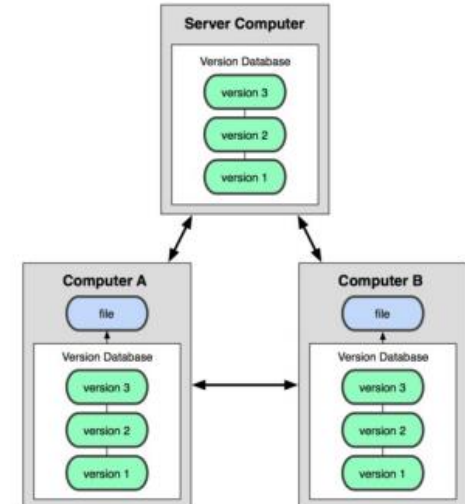
SCV Local (rcs)

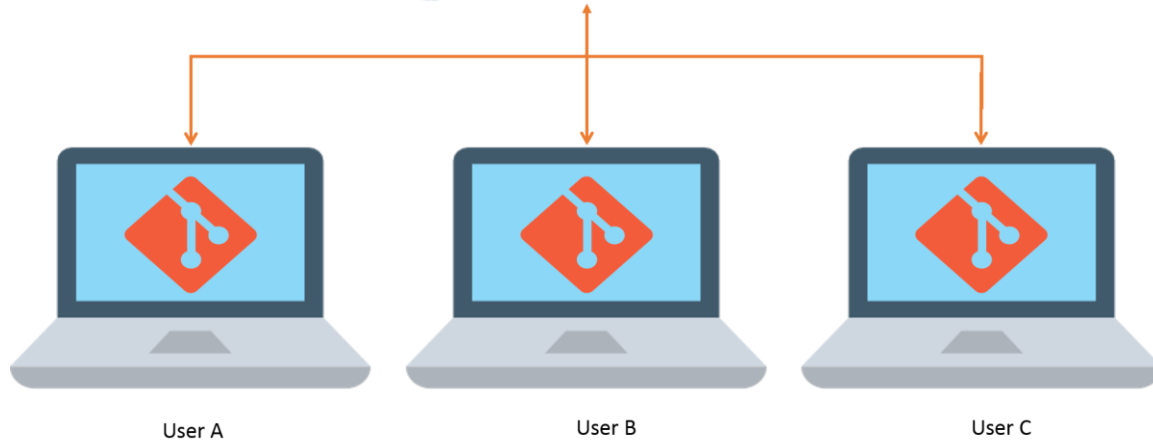


SCV centralizado
(cvs, subversion)



SCV distribuido
(git, mercurial)





Fuente: <https://aprendiendoarduino.wordpress.com/tag/git/>

<https://github.com/>

Git es un **sistema de control de versiones distribuido**.

GitHub es la plataforma de “hosting” de los proyectos. Una comunidad llena de personas que desarrollan y comparten, usando GIT.

GitHub es una página que **ofrece un grupo de servicios que facilitan el uso de Git**, como por ejemplo hosting de proyectos, facilidades de colaboración, reviews de código, perfiles personales, pull requests, issues, etc.



--distributed-even-if-your-workflow-isnt

Search entire site...

About

Documentation

Reference

Book

Videos

External Links

Downloads

Community

This book is available in [English](#).

Full translation available in

[български език](#),

[Deutsch](#),

[Español](#),

[Français](#),

[Ελληνικά](#),

[日本語](#).

Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#). Print versions of the book are available on [Amazon.com](#).



2nd Edition (2014)

1. Inicio - Sobre el Control de Versiones

- 1.1 [Acerca del Control de Versiones](#)
- 1.2 [Una breve historia de Git](#)
- 1.3 [Fundamentos de Git](#)
- 1.4 [La Línea de Comandos](#)
- 1.5 [Instalación de Git](#)
- 1.6 [Configurando Git por primera vez](#)
- 1.7 [¿Cómo obtener ayuda?](#)
- 1.8 [Resumen](#)

Download Ebook



<https://git-scm.com/book/es/v2>

Free

Basics for teams
and developers

- ∞ Unlimited public/private repositories
- ∞ Unlimited collaborators
- ✓ 2,000 Actions minutes/month
Free for public repositories
- ✓ 500MB of GitHub Packages storage
Free for public repositories
- ✓ Community Support

\$0 /month

Create a free organization

Team

Advanced collaboration and support
for teams

- ∞ Unlimited public/private repositories
- ✓ Required reviewers
- ✓ 3,000 Actions minutes/month
Free for public repositories
- ✓ 2GB of GitHub Packages storage
Free for public repositories
- ✓ Code owners

\$4 per user/month

Continue with Team ▾

Enterprise

Security, compliance, and flexible
deployment for enterprises

- ← Everything included in Team
- ✓ SAML single sign-on
- ✓ 50,000 Actions minutes/month
Free for public repositories
- ✓ 50GB of GitHub Packages storage
Free for public repositories
- ✓ Advanced auditing

\$21 per user/month

Contact Sales

GitHub One

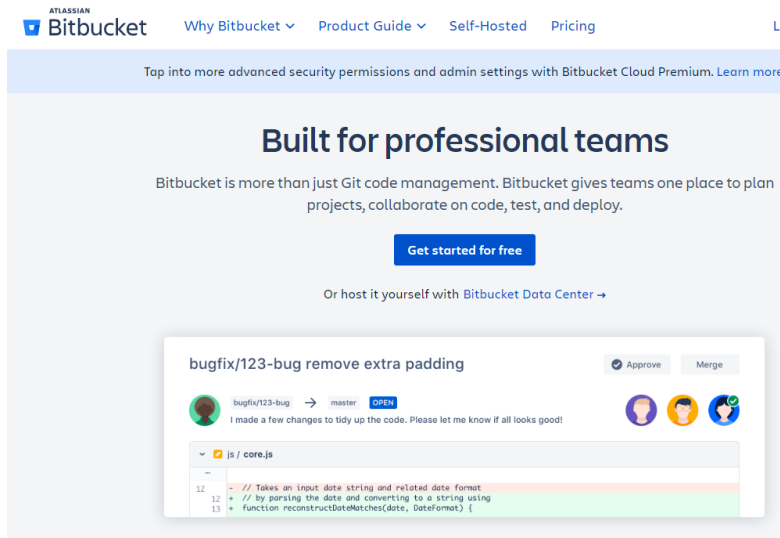
All of our best tools, support,
and services

- ← Everything included in Enterprise
- ✓ Community-powered security
- ✓ Actionable metrics
- ✓ 24/7 support
- ✓ Continuous learning

[Learn more](#)

Contact Sales

Bitbucket /// Gitlab



The screenshot shows the Bitbucket homepage. At the top, the Bitbucket logo is followed by navigation links: 'Why Bitbucket', 'Product Guide', 'Self-Hosted', and 'Pricing'. A blue banner below the navigation bar reads: 'Tap into more advanced security permissions and admin settings with Bitbucket Cloud Premium. Learn more'. The main heading is 'Built for professional teams'. Below this, a paragraph states: 'Bitbucket is more than just Git code management. Bitbucket gives teams one place to plan projects, collaborate on code, test, and deploy.' A blue button labeled 'Get started for free' is centered. Below the button, it says 'Or host it yourself with Bitbucket Data Center'. A code review interface is shown at the bottom, featuring a commit titled 'bugfix/123-bug remove extra padding' with an 'Approve' button and a 'Merge' button. The commit message is 'I made a few changes to tidy up the code. Please let me know if all looks good!'. Below the message is a diff view for 'core.js' showing changes to a function.

ATLASSIAN
Bitbucket

Why Bitbucket Product Guide Self-Hosted Pricing

Tap into more advanced security permissions and admin settings with Bitbucket Cloud Premium. Learn more

Built for professional teams

Bitbucket is more than just Git code management. Bitbucket gives teams one place to plan projects, collaborate on code, test, and deploy.

[Get started for free](#)

Or host it yourself with [Bitbucket Data Center](#)

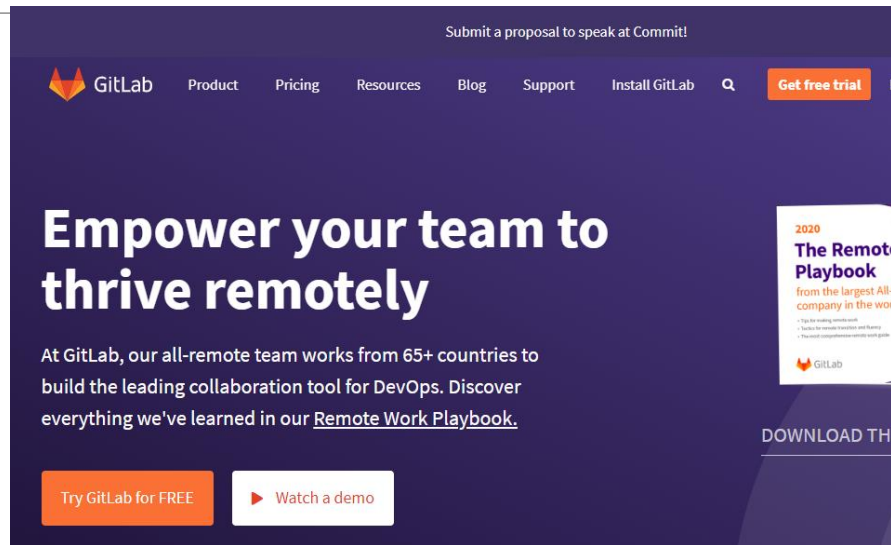
bugfix/123-bug remove extra padding Approve Merge

bugfix/123-bug → master OPEN
I made a few changes to tidy up the code. Please let me know if all looks good!

js / core.js

```
12 // Takes an input date string and related date format  
12 // by parsing the date and converting to a string using  
13 function reconstructDateMatches(date, dateFormat) {
```

<https://bitbucket.org/>



The screenshot shows the GitLab homepage. At the top, the GitLab logo is followed by navigation links: 'Product', 'Pricing', 'Resources', 'Blog', 'Support', 'Install GitLab', and a search icon. A blue button labeled 'Get free trial' is on the right. A top right link says 'Submit a proposal to speak at Commit!'. The main heading is 'Empower your team to thrive remotely'. Below this, a paragraph states: 'At GitLab, our all-remote team works from 65+ countries to build the leading collaboration tool for DevOps. Discover everything we've learned in our [Remote Work Playbook](#).' Two buttons are at the bottom: 'Try GitLab for FREE' and 'Watch a demo'. On the right side, there is a '2020 The Remote Playbook' badge with the text 'from the largest All-remote company in the world' and a list of tips for existing remote work.

Submit a proposal to speak at Commit!

GitLab Product Pricing Resources Blog Support Install GitLab [Get free trial](#)

Empower your team to thrive remotely

At GitLab, our all-remote team works from 65+ countries to build the leading collaboration tool for DevOps. Discover everything we've learned in our [Remote Work Playbook](#).

[Try GitLab for FREE](#) [Watch a demo](#)

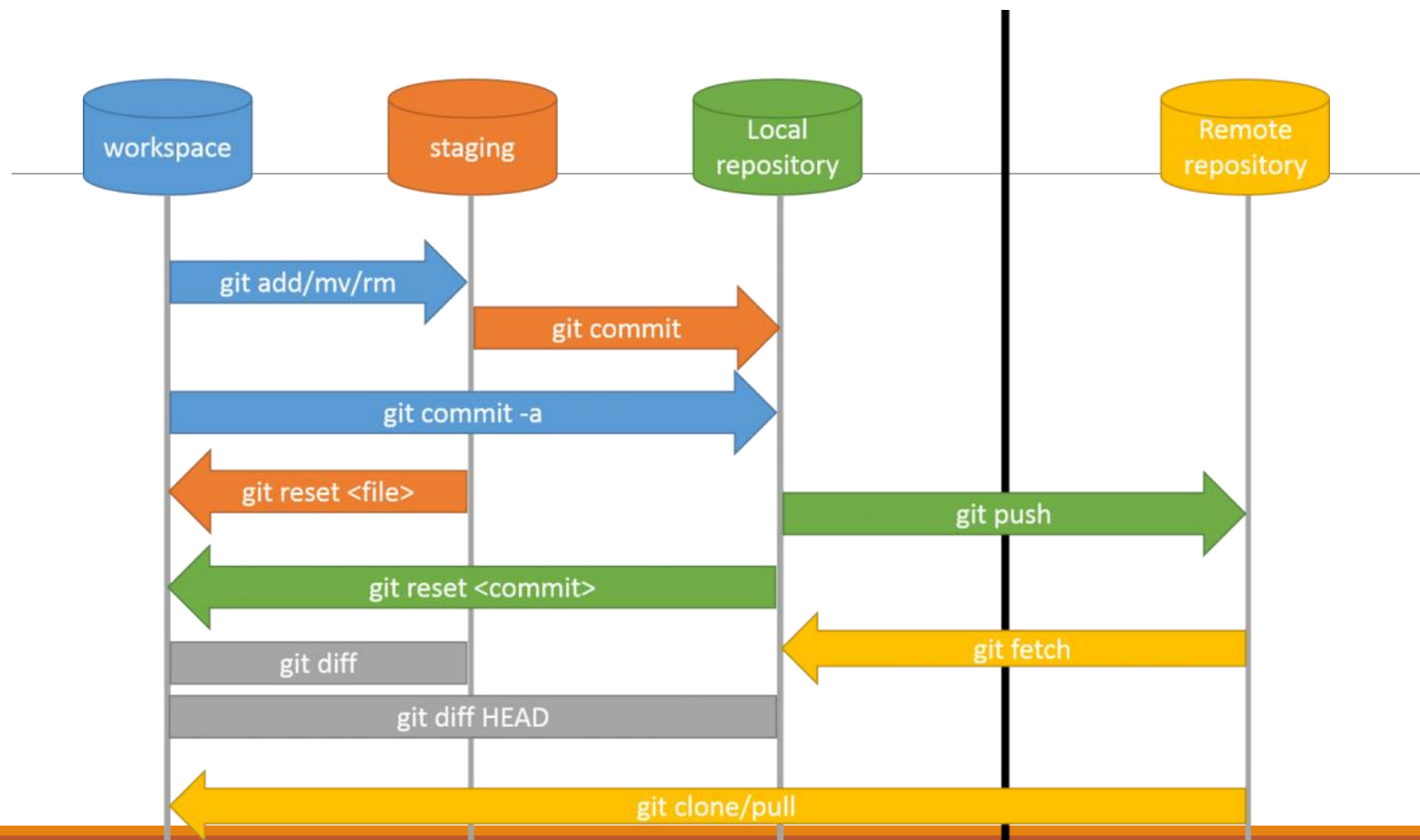
2020
The Remote Playbook
from the largest All-remote company in the world

- Tip for existing remote work
- Tip for new remote workers and teams
- The world's most successful remote work guide

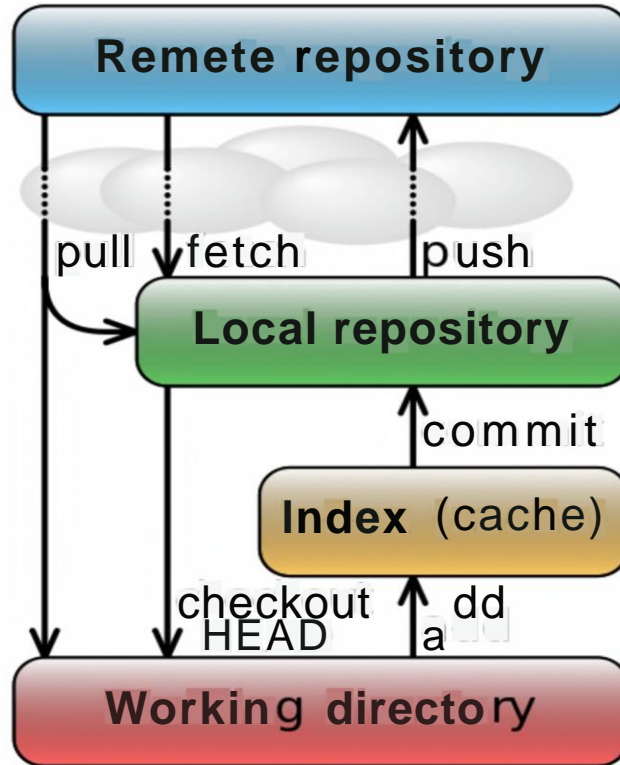
GitLab

DOWNLOAD THE PLAYBOOK

<https://about.gitlab.com/>



Flujo interno de Git



In case of fire



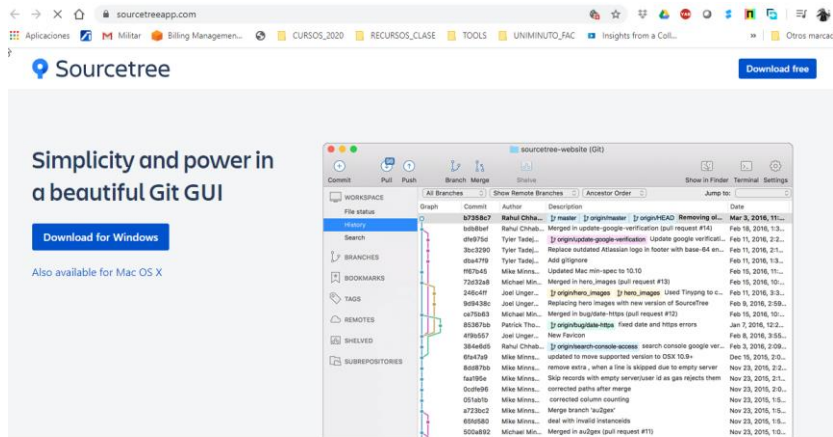
1. git commit



2. git push

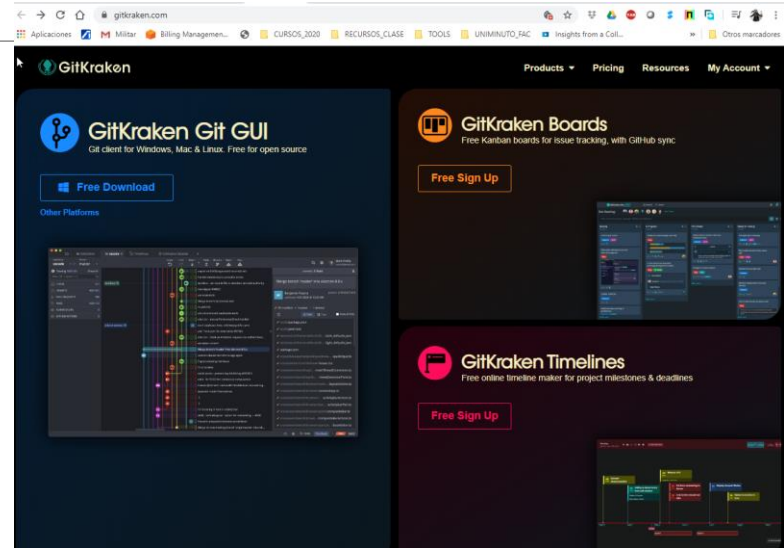


3. leave building



A free Git client for Windows and Mac

<https://www.sourcetreeapp.com/>



<https://www.gitkraken.com/>

INSTALAR GIT

GitHub le ofrece a los clientes de computadoras de escritorio que incluye una interfaz gráfica de usuario para las acciones de repositorio más comunes y una edición de línea de comando de actualización automática de Git para escenarios avanzados.

GitHub para Windows

<https://windows.github.com>

GitHub para Mac

<https://mac.github.com>

Hay distribuciones de Git para sistemas Linux y POSIX en el sitio web oficial Git SCM.

Git para toda plataforma

<http://git-scm.com>

CONFIGURAR HERRAMIENTAS

Configura la información del usuario para todos los repositorios locales

```
$ git config --global user.name "[name]"
```

Establece el nombre que desea esté anexado a sus transacciones de commit

```
$ git config --global user.email "[email address]"
```

Establece el e-mail que desea esté anexado a sus transacciones de commit

```
$ git config --global color.ui auto
```

Habilita la útil colorización del producto de la línea de comando

CREAR REPOSITORIOS

Inicia un nuevo repositorio u obtiene uno de una URL existente

```
$ git init [project-name]
```

Crea un nuevo repositorio local con el nombre especificado

```
$ git clone [url]
```

Descarga un proyecto y toda su historia de versión

EFFECTUAR CAMBIOS

Revisa las ediciones y elabora una transacción de commit

```
$ git status
```

Enumera todos los archivos nuevos o modificados que se deben confirmar

```
$ git diff
```

Muestra las diferencias de archivos que no se han enviado aún al área de espera

```
$ git add [file]
```

Toma una instantánea del archivo para preparar la versión

```
$ git diff --staged
```

Muestra las diferencias del archivo entre el área de espera y la última versión del archivo

```
$ git reset [file]
```

Mueve el archivo del área de espera, pero preserva su contenido

```
$ git commit -m "[descriptive message]"
```

Registra las instantáneas del archivo permanentemente en el historial de versión

CAMBIOS GRUPALES

Nombrar una serie de commits y combina esfuerzos ya culminados

```
$ git branch
```

Enumera todas las ramas en el repositorio actual

```
$ git branch [branch-name]
```

Crea una nueva rama

```
$ git checkout [branch-name]
```

Cambia a la rama especificada y actualiza el directorio activo

```
$ git merge [branch]
```

Combina el historial de la rama especificada con la rama actual

```
$ git branch -d [branch-name]
```

Borra la rama especificada

NOMBRES DEL ARCHIVO DE REFACTORIZACIÓN

Reubica y retira los archivos con versión

```
$ git rm [file]
```

Borra el archivo del directorio activo y pone en el área de espera el archivo borrado

```
$ git rm --cached [file]
```

Retira el archivo del control de versiones, pero preserva el archivo a nivel local

```
$ git mv [file-original] [file-renamed]
```

Cambia el nombre del archivo y lo prepara para commit

SUPRIMIR TRACKING

Excluye los archivos temporales y las rutas

```
*.log  
build/  
temp-*
```

Un archivo de texto llamado `.gitignore` suprime la creación accidental de versiones de archivos y rutas que concuerdan con los patrones especificados

```
$ git ls-files --other --ignored --exclude-standard
```

Enumera todos los archivos ignorados en este proyecto

GUARDAR FRAGMENTOS

Almacena y restaura cambios incompletos

```
$ git stash
```

Almacena temporalmente todos los archivos tracked modificados

```
$ git stash pop
```

Restaura los archivos guardados más recientemente

```
$ git stash list
```

Enumera todos los sets de cambios en guardado rápido

```
$ git stash drop
```

Elimina el set de cambios en guardado rápido más reciente

REPASAR HISTORIAL

Navega e inspecciona la evolución de los archivos de proyecto

```
$ git log
```

Enumera el historial de la versión para la rama actual

```
$ git log --follow [file]
```

Enumera el historial de versión para el archivo, incluidos los cambios de nombre

```
$ git diff [first-branch]...[second-branch]
```

Muestra las diferencias de contenido entre dos ramas

```
$ git show [commit]
```

Produce metadatos y cambios de contenido del commit especificado

REHACER COMMITS

Borra errores y elabora historial de reemplazo

```
$ git reset [commit]
```

Deshace todos los commits después de `[commit]`, preservando los cambios localmente

```
$ git reset --hard [commit]
```

Desecha todo el historial y regresa al commit especificado

SINCRONIZAR CAMBIOS

Registrar un marcador de repositorio e intercambiar historial de versión

```
$ git fetch [bookmark]
```

Descarga todo el historial del marcador del repositorio

```
$ git merge [bookmark]/[branch]
```

Combina la rama del marcador con la rama local actual

```
$ git push [alias] [branch]
```

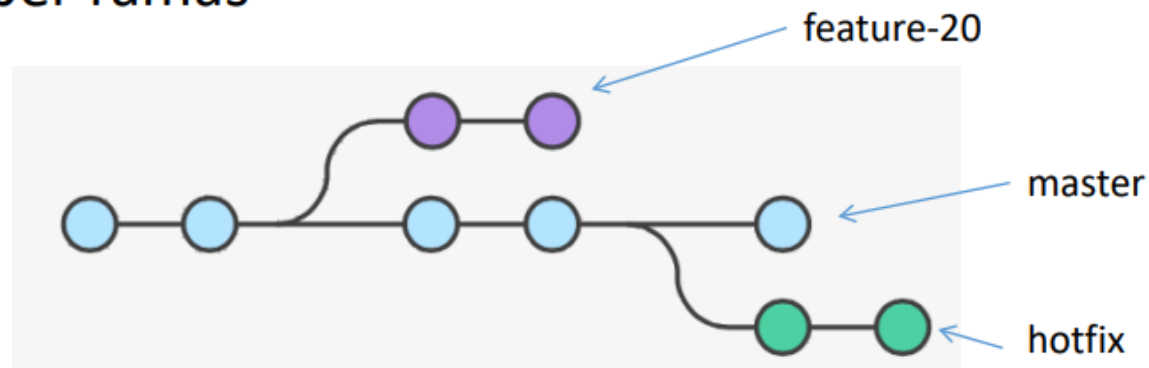
Carga todos los commits de la rama local al GitHub

```
$ git pull
```

Descarga el historial del marcador e incorpora cambios

Ramas

- En Git los commits no tienen porque seguir una secuencia lineal sino que puede haber ramas



- A cada rama se le asigna un nombre para distinguirla de la demás
- Aunque desde un punto de vista técnico no hay ninguna diferencia, se considera que hay una rama por defecto la cual se llama **master**

Ramas

- Al igual que una rama se puede diversificar, también se puede fusionar con otra
- El comando para consultar las ramas es: `git branch`
- La rama activa se muestra con un asterisco

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITARIOS/keepass2 (master)
$ git branch
  atila
* master
```

Ramas

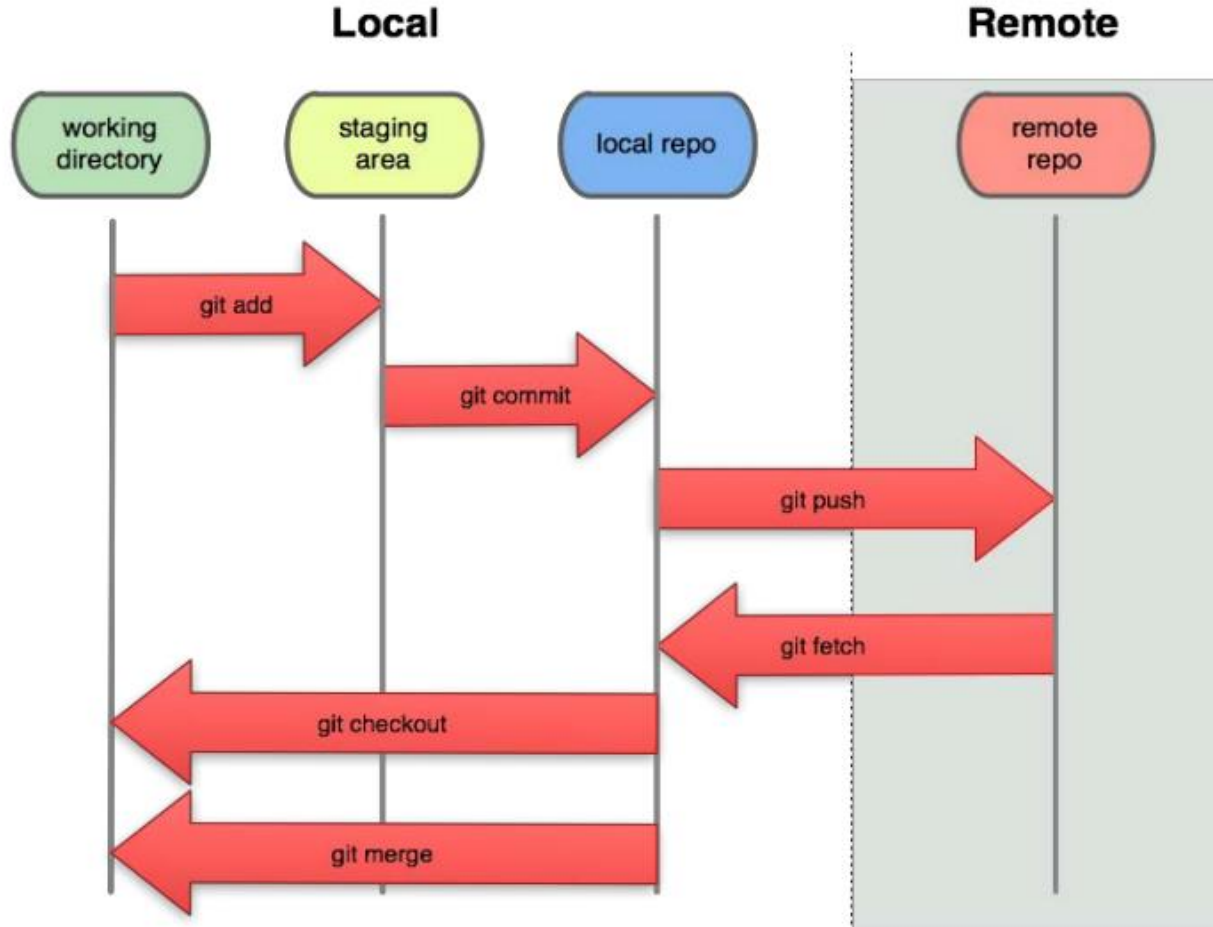
- Creación de una rama: `git branch <nombre rama>`
- Cambiar de una rama a otra: `git checkout <nombre de la rama>`

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITARIOS/keepass2 (master)
$ git branch
  atila
* master

lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITARIOS/keepass2 (master)
$ git checkout atila
Switched to branch 'atila'

lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITARIOS/keepass2 (atila)
$ git branch
* atila
  master
```


Git en modo remoto



Ejercicio



- Crear un repositorio en GitHub

TTG Colombia S.A.S. Zabbix Edit inbound rules [...] COURSES TTG SECURITY DOCTORADO e-REdING. Bibliotec... 27:21

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 ***


 jaiderospina / CEM2023 

Great repository names are short and memorable. Need inspiration? How about [congenial-journey?](#)

Description (optional)

No por saber pescar se es pescador, pero sin ello con dificultad lo seríamos....

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

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

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

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Ejercicio

- Subir el repositorio local al remoto

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



Set up in Desktop

or

HTTPS

SSH

<https://github.com/jaiderospina/CEM2023.git>

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 "# CEM2023" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/jaiderospina/CEM2023.git
git push -u origin main
```

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

```
git remote add origin https://github.com/jaiderospina/CEM2023.git
git branch -M main
git push -u origin main
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (master)
$ echo "# CEM2023" >> README.md
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (master)
$ ls
README.md
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (master)
$ git init
Initialized empty Git repository in G:/UNIMILITAR/2023/REPOSITORIOS/CEM/.git/
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (master)
$ git commit -m "Mi primer commit"
[master (root-commit) 26921dd] Mi primer commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (master)
$ git branch -M main
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (main)
$ git remote add origin https://github.com/jaiderospina/CEM2023.git
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 224 bytes | 224.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jaiderospina/CEM2023.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

```
lenovo@jaider MINGW64 /g/UNIMILITAR/2023/REPOSITORIOS/CEM (main)
$
```

🔗 main ▾

🔗 1 branch

🔖 0 tags

Go to file

Add file ▾

<> Code ▾



jaiderospina Mi primer commit

26921dd 7 minutes ago ⌚ 1 commit



README.md

Mi primer commit

7 minutes ago


README.md



CEM2023

Create a new repository

Owner *

 jaiderospina ▾

Repository name *

/ BLOCKCHAIN 

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

Description (optional)

Laboratorio de clase. |



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

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

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

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



Set up in Desktop

or

HTTPS

SSH

`https://github.com/jaiderospina/BLOCKCHAIN.git`

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 "# BLOCKCHAIN" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/jaiderospina/BLOCKCHAIN.git
git push -u origin main
```

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

```
git remote add origin https://github.com/jaiderospina/BLOCKCHAIN.git
git branch -M main
git push -u origin main
```

...or import code from another repository

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

Import code

y...porqué git????



Instalación de git

<https://git-scm.com/>



The screenshot shows the Git website homepage with a light beige background and a subtle diamond pattern at the top. The main content is organized into four quadrants, each with a circular icon and a title in red. The top-right quadrant features a large monitor graphic displaying the latest source release information. The bottom-left quadrant includes a small book cover for 'Pro Git'. The bottom section is partially visible, showing the heading for 'Companies & Projects Using Git'.



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.



Latest source Release
2.30.2
[Release Notes \(2021-02-12\)](#)
[Download 2.30.2 for Windows](#)



Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

 [Windows GUIs](#)

 [Tarballs](#)

 [Mac Build](#)

 [Source Code](#)

Companies & Projects Using Git

Downloading Git



Your download is starting...

You are downloading the latest (**2.30.2**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **1 day ago**, on 2021-03-09.

[Click here to download manually](#), if your download hasn't started.

Other Git for Windows downloads

Git for Windows Setup

32-bit Git for Windows Setup.

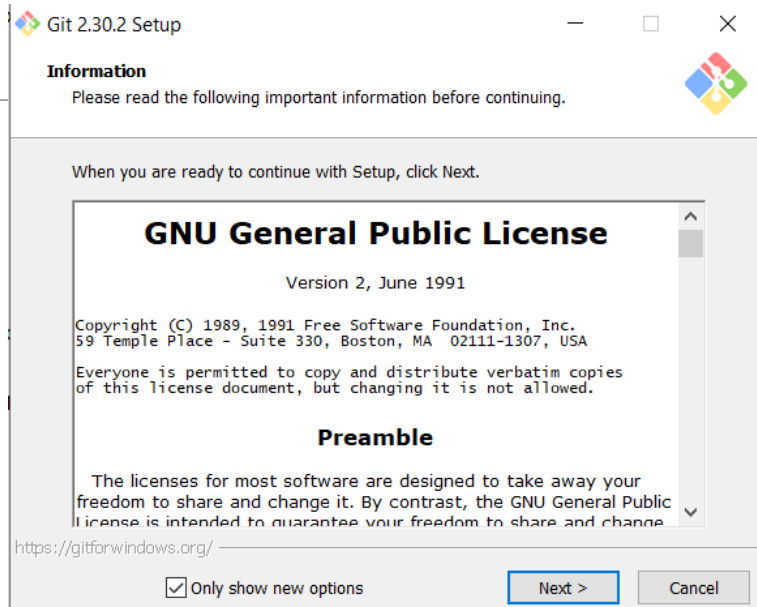
64-bit Git for Windows Setup.

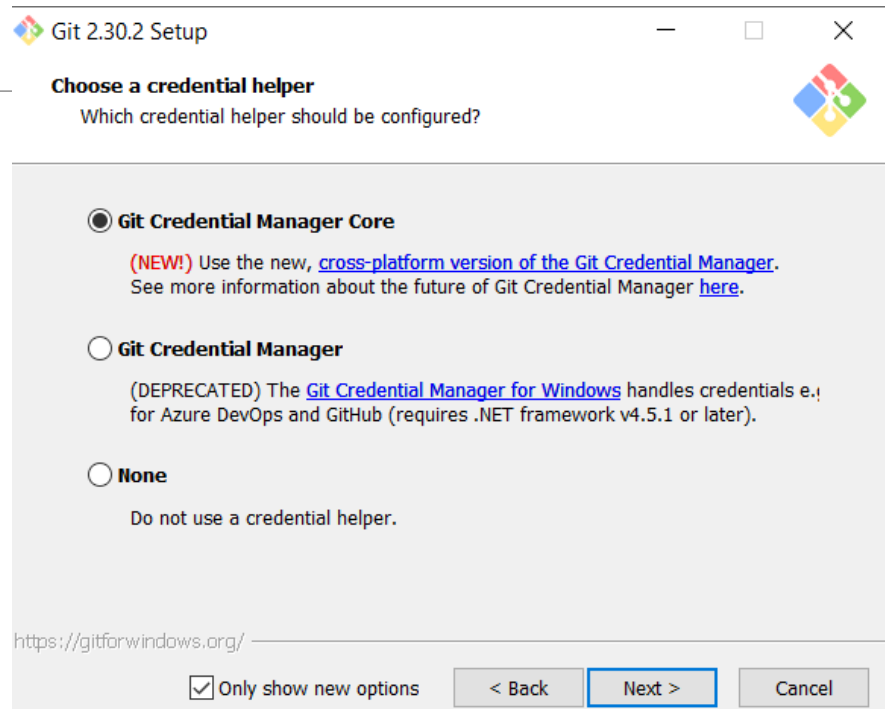
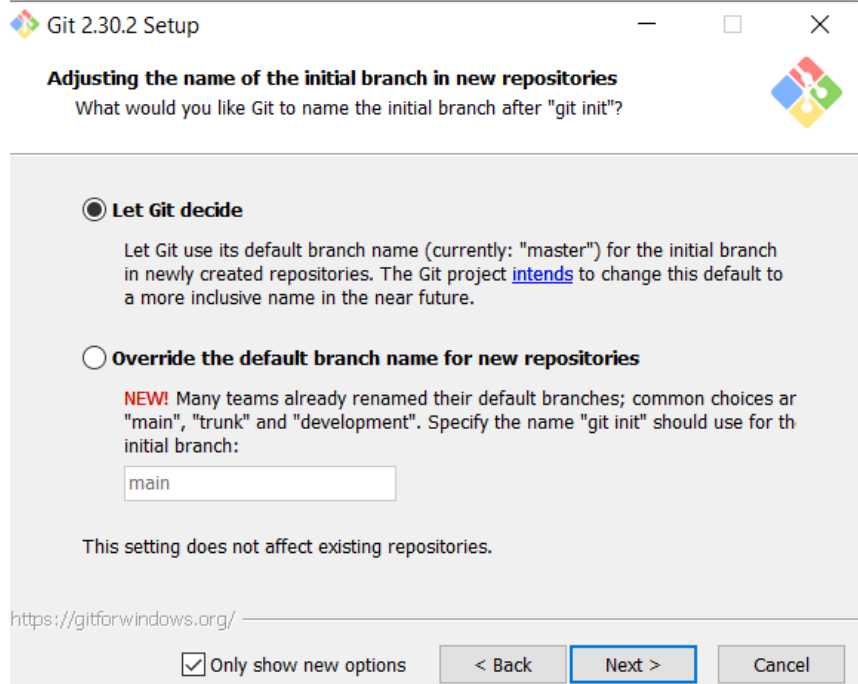
Git for Windows Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

The current source code release is version **2.30.2**. If you want the newer version, you can build it from the [source code](#).





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.

☒ Use Git from the Windows Command Prompt

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 both Git Bash and the Windows Command Prompt.

☐ Use Git and optional Unix tools from the Windows 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.

tps://git-for-windows.github.io/

< Back

Next >

Cancel

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 226,7 MB of disk space.

tps://git-for-windows.github.io/

< Back

Next >

Cancel

Git 2.30.2 Setup



Replacing in-use files

The following applications use files that need to be replaced, please close them.



MSYS2 terminal (PID 22180, closing is required)
bash (PID 1176, closing is required)
Git for Windows (PID 13780, closing is required)

https://gitforwindows.org/

Refresh

☒ Only show new options

< Back

Install

Cancel

Git 2.30.2 Setup



Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

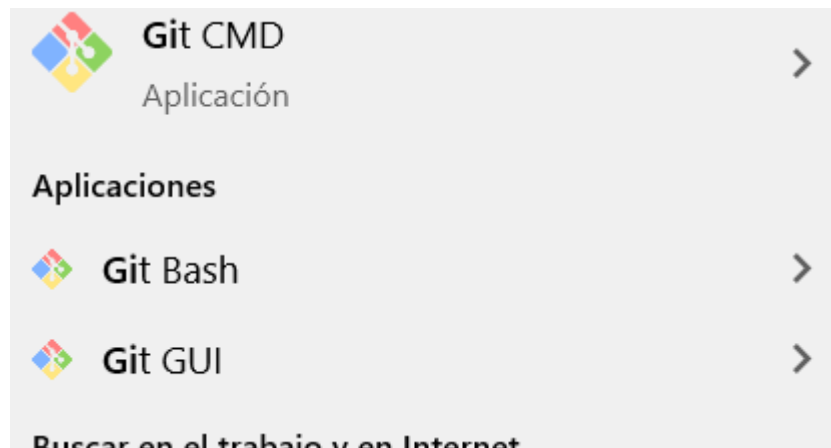
☐ Launch Git Bash

☒ View Release Notes

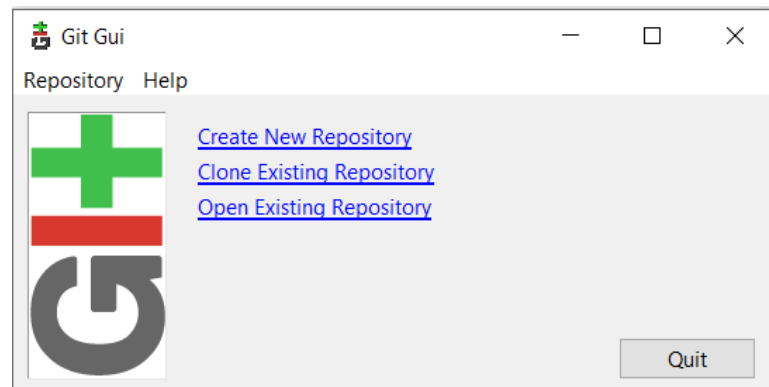


☒ Only show new options

Finish



```
MINGW64:/c/Users/LENOVO
LENOVO@Matrix MINGW64 ~
$ git --version
git version 2.30.2.windows.1
LENOVO@Matrix MINGW64 ~
$
```



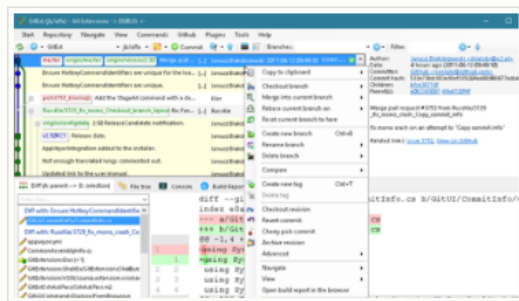
GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just [follow the instructions](#).

[All](#)[Windows](#)[Mac](#)[Linux](#)[Android](#)[iOS](#)

20 Linux GUIs are shown below ↓



Git Extensions

Platforms: Linux, Mac, Windows

Price: Free

License: GNU GPL

GitKraken

Platforms: Linux, Mac, Windows

Price: Free / \$29 / \$49

License: Proprietary

Configuración Básica

- Abrimos la aplicación Git Bash y en ella ejecutamos:
- \$ git config --global user.name "Mi nombre"
- \$ git config --global user.email ["mi@correo.com"](mailto:mi@correo.com)

Git Init

- Luego Solo hay que ubicarnos en la carpeta raíz de nuestro proyecto y ejecutar git init para crear el repositorio:
- \$ cd /proyectos/mi-proyecto
- \$ git init

Git status

- ejecutando `git status` podemos verificar los cambios hechos en el estado actual:
- `$ git status`

Git add

- Para registrar todos los archivos contenidos en nuestro proyecto ejecutamos:
- `$ git add .`
- y verificamos:
- `$ git status`
- Desde luego es posible agregar archivos individualmente con solo especificar la ruta relativa con relación a la raíz del repositorio, por ejemplo:
- `$ git add mi-proyecto/archivo.php`

Git commit

- Finalmente debemos dar por hechos los cambios que llevamos en el repositorio, entonces ejecutamos `git commit` para confirmarlo:
- `$ git commit -m "Primer commit; agregué la estructura de archivos"`

Git branch

- Lo ideal es que siempre trabajemos la versión estable de nuestros proyectos en la rama master.
- para mostrar las ramas existentes localmente ejecutamos:
- `$ git branch`

Git branch

- Es recomendable crear la rama develop para trabajar en el desarrollo:
 - `$ git branch develop`
 - para movernos entre ramas debemos usar checkout:
 - `$ git checkout develop`

Git branch

- a partir de la rama en la que nos encontremos podemos crear ramas inferiores para separar nuestros cambios de acuerdo con la estructura de archivos que usemos en nuestro proyecto. Esto se hace mediante:
 - `$ git checkout -b subrama`

Git branch

- suponiendo que hemos trabajado sobre la subrama creada y no queremos perder los cambios realizados entonces debemos hacer commit:
- `$ git commit -am "Comentario de ejemplo"`
- el atributo `-am` nos permite agregar automáticamente los archivos modificados y establecer un mensaje para el commit.

Git merge

- Para integrar los cambios usamos la instrucción merge:
 - nos ubicamos en la rama de desarrollo:
 - \$ git checkout develop
- y despues integramos los cambios
 - \$ git merge subrama
- Y ahí tenemos, nuestros cambios realizados en la rama auxiliar los hemos integrado a la rama oficial de desarrollo.

Eliminando subramas

- Suponiendo que ya no vamos a usar la subrama creada pues nuestro código se encuentra estable y necesitamos eliminar referencias innecesarias. Hacemos lo siguiente para eliminarla:
- `$ git branch -d subrama`

Clonando un repositorio existente

- Si deseas obtener una copia de un repositorio Git existente el comando que necesitas es git clone
 - `$ git clone git://git.diens.com.co/usuario/proyecto.git`
 - con esto crearemos una copia local de nuestro repositorio.

Trabajando con repositorios Remotos

- Para ver qué repositorios remotos tienes configurados, puedes ejecutar el comando `remote`.
- `$ git remote -v`
- con la opción `-v` podemos ver la URL asociada a cada repositorio remoto.

Añadiendo repositorios remotos

- Para añadir un nuevo repositorio Git remoto, asignándole un nombre con el que referenciarlo fácilmente.
- \$ git remote add [nombre] [url]:
- \$ git remote add us git://git.diens.com.co/usuario/proyecto.git
- Por ejemplo, para recuperar toda la información de 'usuario' que todavía no tienes en tu repositorio, puedes ejecutar:
- \$ git fetch us

Recibiendo de tus repositorios remotos

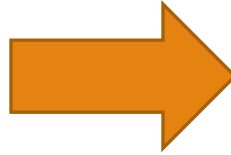
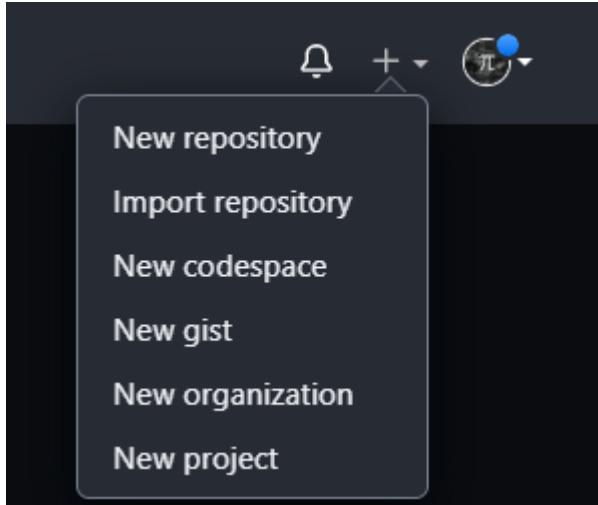
- Como acabas de ver, para recuperar datos de tus repositorios remotos puedes ejecutar:
 - `$ git fetch [remote-name]`
 - Este comando recupera todos los datos del proyecto remoto que no tengas todavía.

Enviando a tus repositorios remotos

- Cuando tu proyecto se encuentra en un estado que quieres compartir el comando que te permite hacer esto es sencillo:
 - `$ git push [nombre-remoto][nombre-rama]`
 - Por ejemplo para enviar a la rama mestra del servidor de origen:
 - `$ git push origin master`

Ejemplo práctico 2

Repositorio desde Github



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 ***

jaiderospina / test33 ✓

Great repository names are short and memorable. Need inspiration? How about [turbo-fiesta?](#)

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

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

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

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None

① You are creating a public repository in your personal account.

[Create repository](#)

Ejemplo práctico 2

Repositorio desde Github

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



Set up in Desktop

or

HTTPS

SSH

`https://github.com/jaiderospina/test33.git`



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 "# test33" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/jaiderospina/test33.git
git push -u origin main
```



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

```
git remote add origin https://github.com/jaiderospina/test33.git
git branch -M main
git push -u origin main
```



...or import code from another repository

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

[Import code](#)

RETO

Terminar el ejercicio, realizando un repo local y enviándolo a github.

Ejemplo práctico 2

Repositorio desde Github

Continuar estos pasos en su repositorio

- Crear una nueva rama
 - *\$ git checkout -b nuevos_cambios*
- Crear archivo7.txt, agregarlo y guardar cambios
 - *\$ git add archivo7.txt*
 - *\$ git commit -m "Agregando archivo 7.txt"*
- Regresar a master y comparar
 - *\$ git checkout master*
 - *\$ git diff master nuevos_cambios*

Ejemplo práctico

- Realizar merge y enviar cambios
 - *\$ git merge nuevos_cambios*
 - *\$ git push origin master*
- Eliminar rama que ya no se empleará
 - *git branch -d nuevos_cambios*
- Ir a newproject y traer los nuevos cambios
 - *\$ git pull/ origin master*

Github / Bitbucket

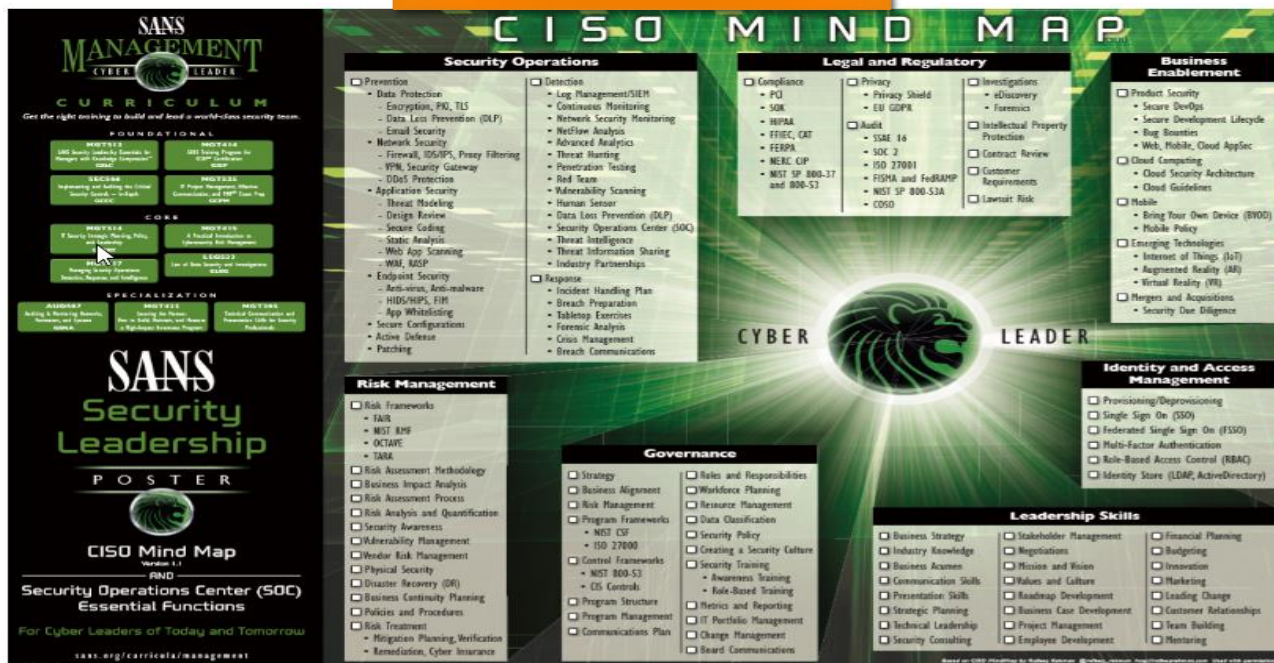
- **Github**

- Trabaja con repositorios netamente de tipo git, tiene una versión de pago para repositorio privados .

- **Bitbucket**

- A diferencia del anterior permite trabajar con repositorios privados y trabaja con mercurial y git.

TALLER



Del poster "CISO MIND MAP" de SANS tomar un grupo específico de tareas y:

1. Clonar repositorio a su máquina local.
2. Crear en su repositorio local una nueva carpeta que lleve como título el grupo elegido.
3. Realizar lista numerada en esta carpeta de todos los elementos de su grupo (traducido). Para ello haga usos de un archivo markdown.
4. Describir los 5 primeros elementos del grupo de trabajo.
5. Realice un (git push -u origin master) a este repositorio. Asegúrese de realizar el comentario pertinente detallando el autor y motivo de commit.

Referencias

<https://aprendiendoarduino.wordpress.com/tag/git/>

<https://bitbucket.org/>

<https://www.sourcetreeapp.com/>
