

FINAL_CORREGIDO_FINALISIMO.V5

Control de versiones de código con Git, GitHub y RStudio

Primer parte: el taller

Ignacio Alcántara, Martín Soñora, Camila Simoes

26 de agosto de 2024

¡Hola!

Ignacio Alcántara (FVet)



Lic. en Ciencias Biológicas/ MSc. en Geociencias/ Estudiante de doctorado en Producción Animal

Bioestadística/Epidemiología Espacial

Camila Simoes (IPMon/Fmed-HC)



Lic. en Ingeniería Biológica/ MSc. en Bioinformática/ Estudiante de doctorado en Ciencias Biológicas

Genómica Humana/Análisis de datos

Martín Soñora (IPMon)



Lic. en Bioquímica/ MSc. en Bioinformática/ Doctor en Ciencias Biológicas

Virología
Física-Computacional/Análisis de datos

Contenidos

01. Introducción

02. Instalación

03. Conceptos y operaciones básicas

04. Creación de usuario y configuración inicial

05. Flujo de trabajo

01. Introducción

¿Qué es Git? ¿Cómo se usa? ¿Por qué usarlo?



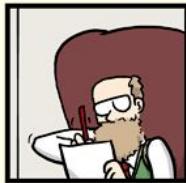
"FINAL".doc



FINAL.doc!



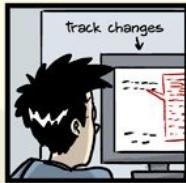
FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc



FINAL_rev.22.comments49.
corrections.10.#@\$%WHYDID
ICOMETOGRAD SCHOOL????.doc



Versiones

- El método de “control de versiones” más utilizado es copiar archivo y guardarlo con otro nombre o en otra carpeta.
- Este enfoque es muy simple, pero también es propenso a errores.
- Es fácil olvidar en qué carpeta estaba y escribir accidentalmente en el archivo equivocado.
- Si trabaja en varias computadoras hay que estar respaldando constantemente.
- Servicios de documentos en la nube (Google Drive, Dropbox, etc) solucionan el almacenamiento centralizado.



¿Qué es un sistema de control de versiones?

El control de versiones es un sistema que registra los cambios en un archivo o conjunto de archivos a lo largo del tiempo para que puedas recuperar versiones específicas más adelante.

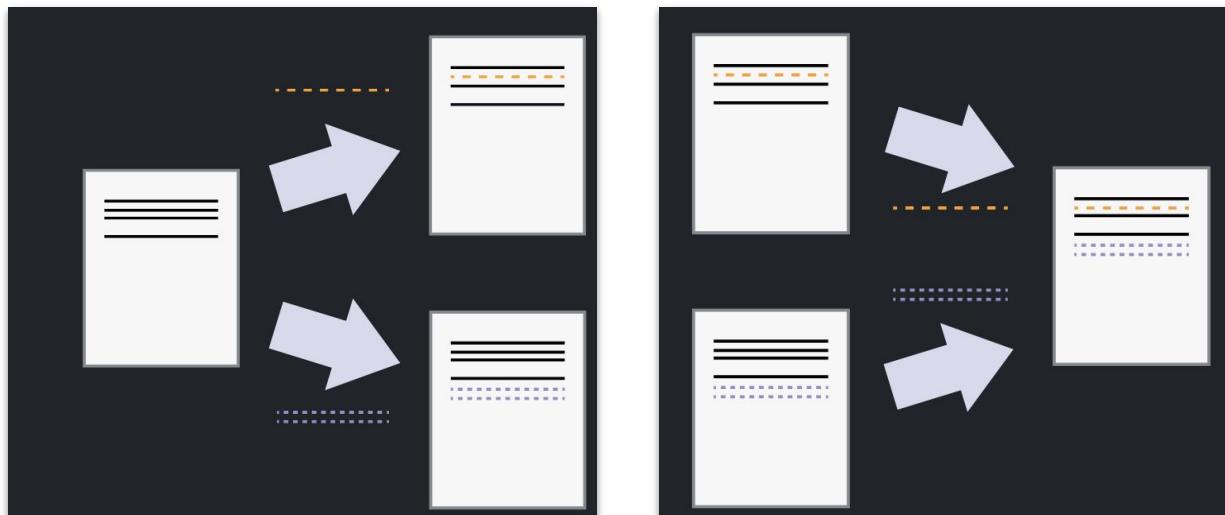


[Software Carpentry](#)

- Los sistemas de control de versiones comienzan con una versión base del documento y luego guardan sólo los cambios que se realizaron en cada paso del proceso.
- Podemos pensar en esto como un registro del progreso del documento solo grabando las modificaciones respecto a la versión anterior.

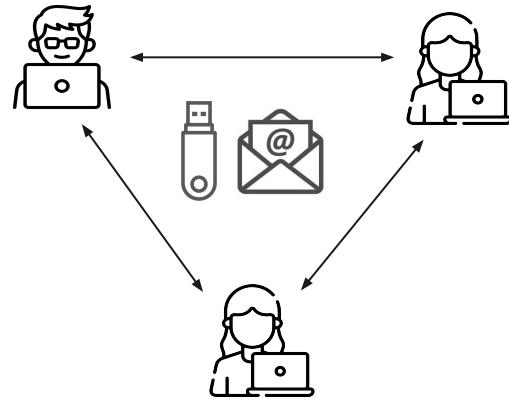
¿Qué es un sistema de control de versiones?

- Los cambios como algo separado del documento en sí.
- Pueden existir historiales de cambio distintos sobre el mismo documento base.
- Esos cambios se incorporan al documento base.



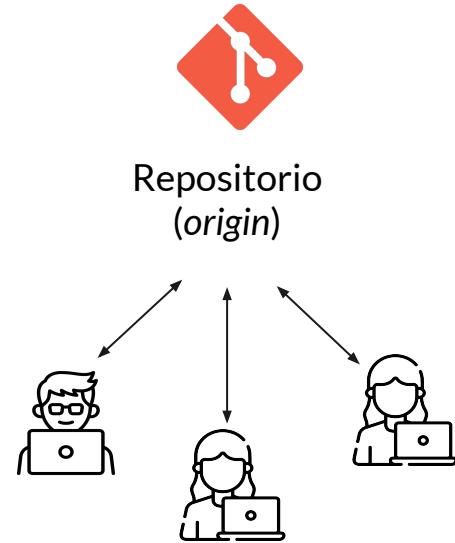
¿Por qué necesitamos control de versiones?

- Nos permite decidir qué cambios se realizarán en la próxima versión
- cada registro de estos cambios se denomina “commits” y guarda metadatos útiles sobre ellos
- El historial completo de “commits” de un proyecto en particular y sus metadatos conforman un repositorio
- Los repositorios se pueden mantener sincronizados en diferentes computadoras, lo que facilita la colaboración entre diferentes personas.



¿Qué es Git?

- Git es un software de control de versiones de código.
- Diseñado para trabajar colaborativamente en grandes equipos de desarrollo de software.
- Git controla la evolución de un conjunto de archivos (repositorio) de manera sensata y muy estructurada.
- Un analista de datos que trabaje solo en una sola computadora puede aplicar control de versiones de sus propios códigos.



¿Qué **no** es Git?



\neq



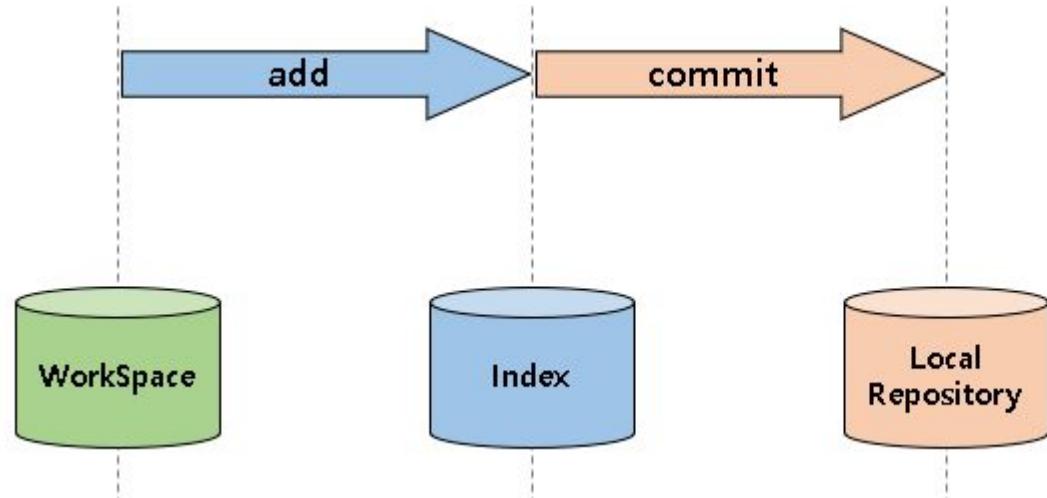
Herramienta

Empresas de hosting

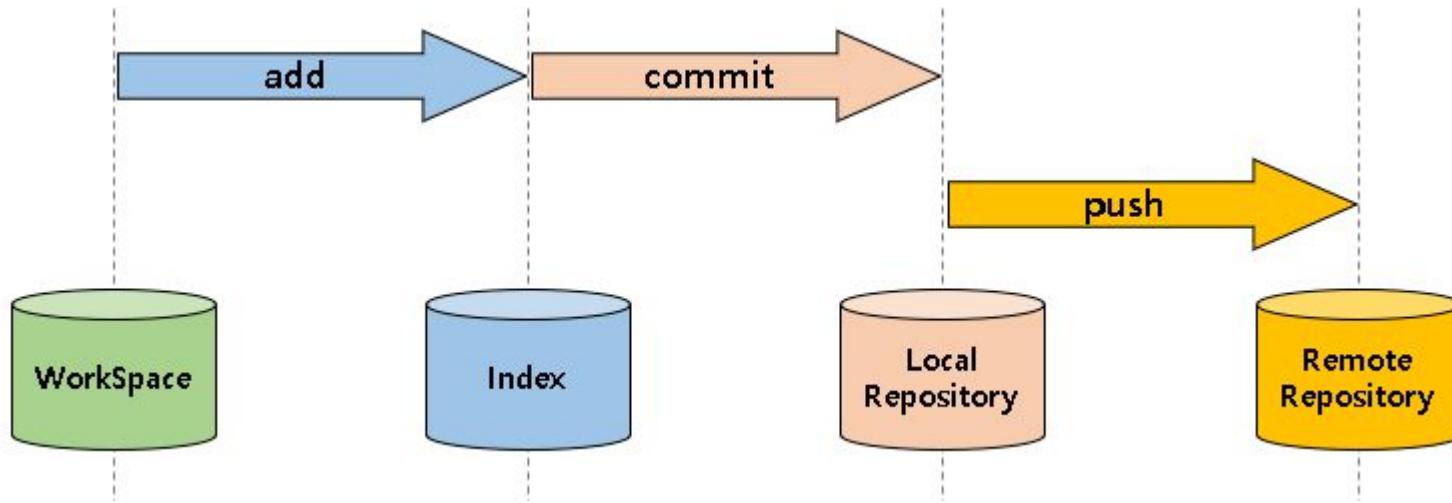


proveen
(y amplían)

¿Cómo funciona Git?



¿Cómo funciona Git + Hosting?



¿Qué podemos hacer con Git?

- Mantener/navegar histórico de cambios
 - Rastrear errores (*debug*)
- Visualizar los cambios entre versiones
 - Diferencial → evolución
- Varias copias de código
 - Desarrollo, testing, estable → flujo de trabajo

¿Qué podemos hacer con Git? (2)

- Mantener/navegar histórico de cambios
 - Rastrear errores (*debug*)
- Visualizar los cambios entre versiones
 - Diferencial → evolución
- Varias copias de código
 - Desarrollo, testing, estable → flujo de trabajo

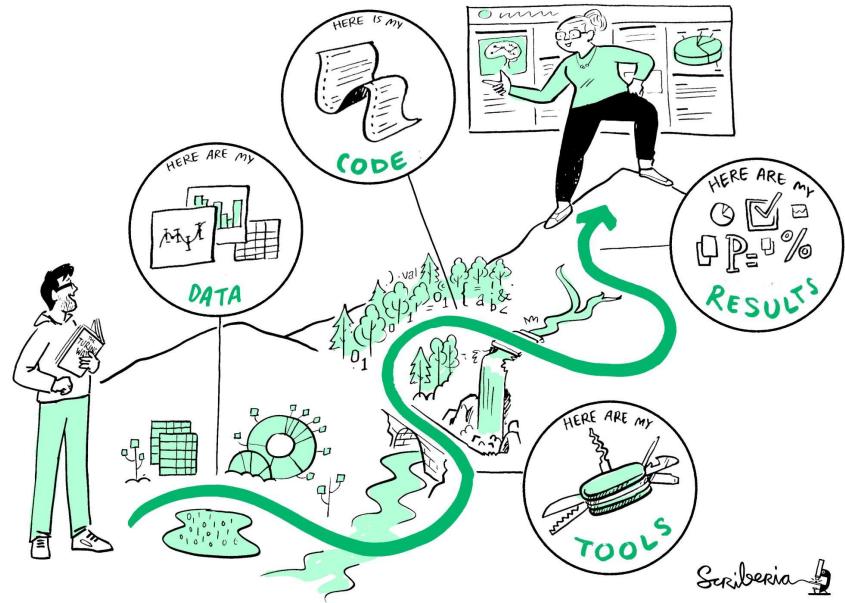
Los archivos de texto plano, como `.txt`, `.md`, `.tex`, `.csv`, `.R`, `.py`, etc., son fácilmente manejables.

Sin embargo, archivos binarios como imágenes (`.jpg`, `.png`), archivos PDF, archivos office (`.doc/.docx`, `.xls/.xlsx`, `.ppt`, etc.) pueden ser trackeados, pero Git no puede mostrar diferencias ("diffs") de forma eficiente en estos casos.

¿Por qué usar git en la academia?

La práctica científica necesita:

- ser colaborativa
- estar documentada
- ser abierta
- ser susceptible de revisión
- incluir código informático verificable



[The Turing Way](#)

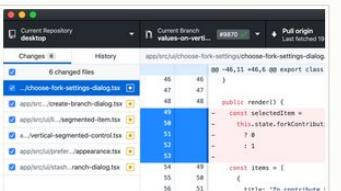
Interfaces

- Terminal / consola

- git en estado puro

- Interfaces gráficas

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

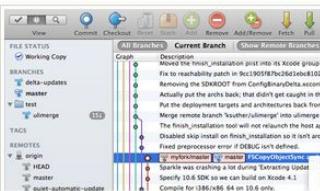


GitHub Desktop

Platforms: Mac, Windows

Price: Free

License: MIT

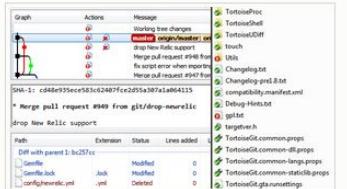


SourceTree

Platforms: Mac, Windows

Price: Free

License: Proprietary

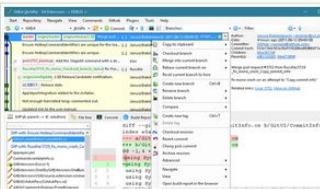


TortoiseGit

Platforms: Windows

Price: Free

License: GNU GPL

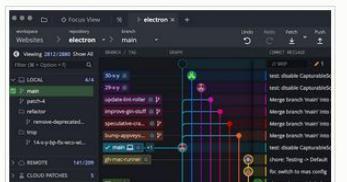


Git Extensions

Platforms: Windows

Price: Free

License: GNU GPL

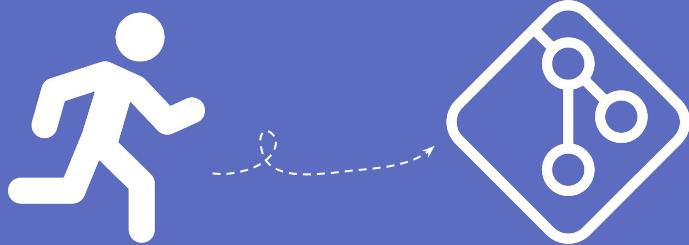


GitKraken Desktop



Magit

La idea es aprender practicando,
¡así que primero consigamos la herramienta!



02.

Instalación

¿Cómo obtengo la herramienta en mi computadora? ¿Cómo se usa?





Instalación de Git

- Windows
- Linux
- Mac



Instalación de Git

Instalación de Git en Windows

1. Descargar el Instalador:

- Página oficial de Git: <https://git-scm.com/>
- <https://git-scm.com/download/win>

The screenshot shows the official Git website (<https://git-scm.com/>). The main header features the Git logo and the tagline "distributed even if your workflow isn't". A search bar is located in the top right corner. The page content includes a brief introduction to Git's benefits and its distributed nature, followed by a diagram illustrating a network of repositories connected by bidirectional arrows. Below this, there are five main navigation links: "About", "Documentation", "Downloads", "Community", and a large "Download for Windows" button. The "Downloads" link is highlighted with a downward arrow icon. The "Community" link is associated with a speech bubble icon. The "Download for Windows" button is prominently displayed on a computer monitor icon.

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

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.

Latest source Release
2.40.0
Release Notes (2023-03-12)

Download for Windows



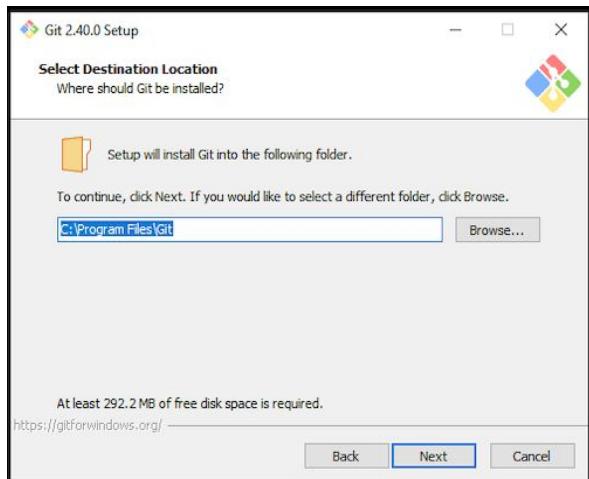
Instalación de Git

Instalación de Git en Windows

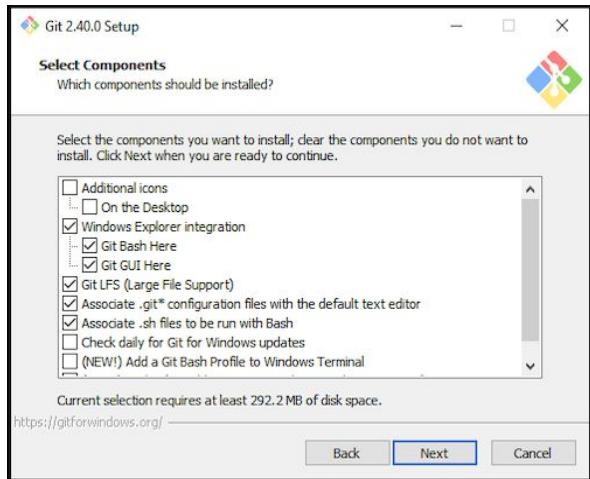
2. Ejecutar el Instalador: archivo `exe`



Acepta la licencia GNU



Selecciona la ubicación de destino para Git en tu computadora.



Selecciona los componentes que quieras instalar con Git.

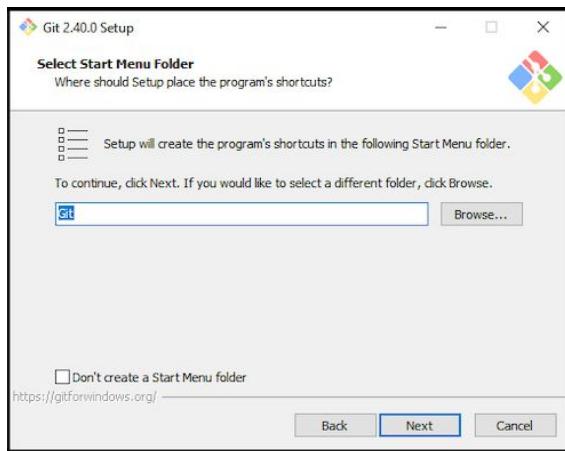


Instalación de Git

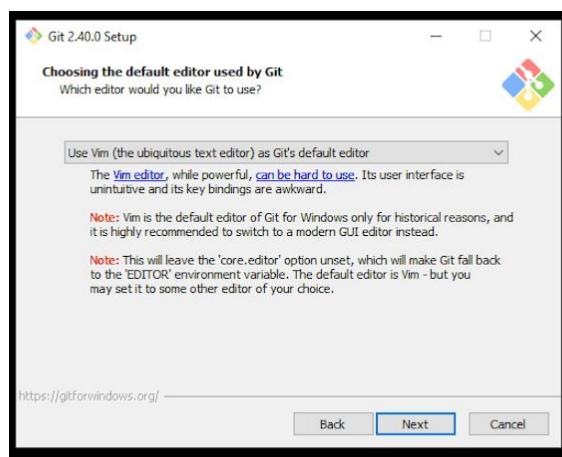
Instalación de Git en Windows

2. Ejecutar el Instalador: archivo `exe`

- Sigue las instrucciones en pantalla, dejando las configuraciones predeterminadas a menos que necesites algo específico.



Si quieres, puedes ajustar el nombre de la carpeta de inicio.



Selecciona el editor por defecto utilizado por Git.



Puedes ajustar el nombre de la rama inicial en los repositorios nuevos.

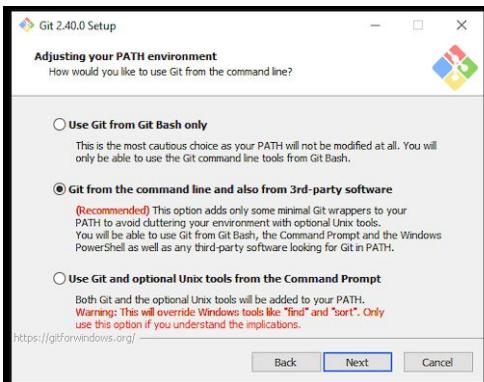


Instalación de Git

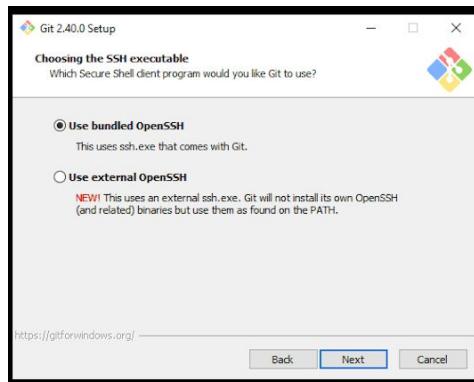
Instalación de Git en Windows

2. Ejecutar el Instalador: archivo ` `.exe`

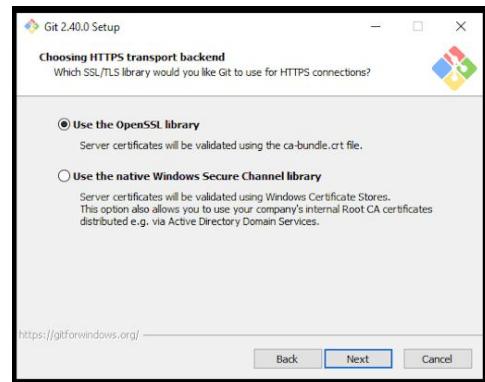
- Sigue las instrucciones en pantalla, dejando las configuraciones predeterminadas a menos que necesites algo específico.



Seleccionar el entorno de la ruta



Elegiendo el ejecutable SSH en el instalador de Git para Windows.



La mayoría debería seleccionar Usar la biblioteca OpenSSL

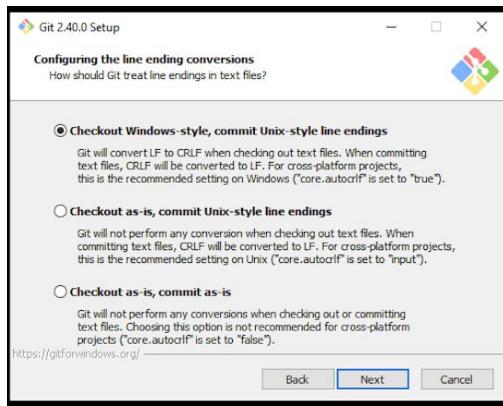


Instalación de Git

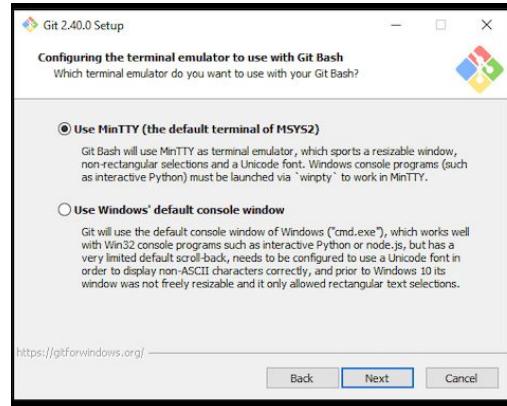
Instalación de Git en Windows

2. Ejecutar el Instalador: archivo ` `.exe`

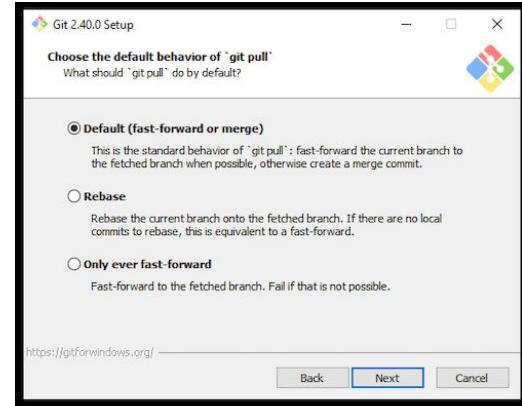
- Sigue las instrucciones en pantalla, dejando las configuraciones predeterminadas a menos que necesites algo específico.



Configurar las conversiones de final de línea.



Configurar el emulador de terminal para utilizarlo con Git Bash.



Selecciona el comportamiento por defecto del comando 'git pull'.

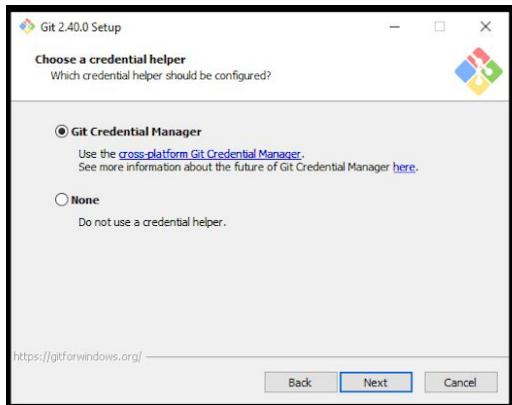


Instalación de Git

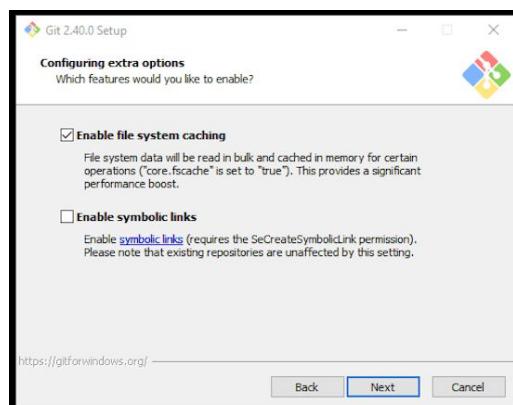
Instalación de Git en Windows

2. Ejecutar el Instalador: archivo `exe`

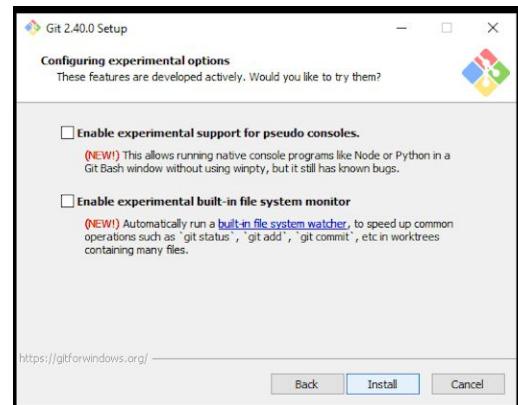
- Sigue las instrucciones en pantalla, dejando las configuraciones predeterminadas a menos que necesites algo específico.



Elegir un ayudante de credenciales.



Selecciona opciones adicionales.



Configurar las opciones experimentales



Instalación de Git

Instalación de Git en Windows

3. Verificar la Instalación:

- Abre la terminal de Git (Git Bash) desde el menú de inicio.

```
git --version
```

git version 2.30.1.windows.1



Instalación de Git

Instalación de Git en Linux

1. Instalar Git Usando el Gestor de Paquetes:

git --fast-version-control

Search entire site...

[About](#)

[Documentation](#)

[Downloads](#)

GUI Clients

Logos

[Community](#)

Download for Linux and Unix

It is easiest to install Git on Linux using the preferred package manager of your Linux distribution. If you prefer to build from source, you can find tarballs on [kernel.org](#). The latest version is [2.40.0](#).

Debian/Ubuntu

For the latest stable version for your release of Debian/Ubuntu

```
# apt-get install git
```

For Ubuntu, this PPA provides the latest stable upstream Git version

```
# add-apt-repository ppa:git-core/ppa # apt update; apt install git
```

Fedor

```
# yum install git (up to Fedora 21)
# dnf install git (Fedora 22 and later)
```



Instalación de Git

Instalación de Git en Linux

1. Instalar Git Usando el Gestor de Paquetes:

```
sudo apt-get install git
```

```
sudo yum install git
```

```
sudo dnf install git
```



Instalación de Git

Instalación de Git en Linux

2. Verificar la Instalación:

```
git --version
```

git version 2.30.0



Instalación de Git

Instalación de Git en macOS

The screenshot shows the official Git website (git-scm.com) with the following content:

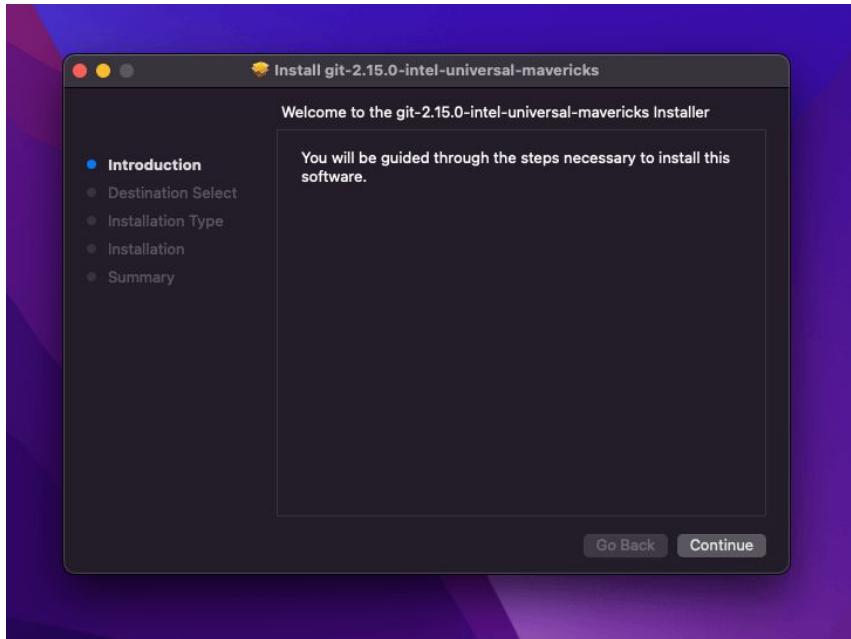
- Header:** The Git logo (a red diamond with a white 'g') and the text "git --fast-version-control". A search bar is located in the top right corner.
- Text:** "Git is a [free and open source](#) distributed version control system designed to handle everything from small to very large projects with speed and efficiency." Below this, another paragraph highlights Git's [easy to learn](#) nature, [tiny footprint with lightning fast performance](#), and its ability to outclass other SCM tools like Subversion, CVS, Perforce, and ClearCase with features like [cheap local branching](#), [convenient staging areas](#), and [multiple workflows](#).
- Diagram:** An illustration showing a network of six computer stacks connected by colored lines (red, blue, yellow) on a grid background, representing the distributed nature of Git.
- Footer:** A section with several links:
 - About**: "The advantages of Git compared to other source control systems." (represented by a gear icon)
 - Documentation**: "Command reference pages, Pro Git book content, videos and other material." (represented by an open book icon)
 - Downloads**: "GUI clients and binary releases for all major platforms." (represented by a download arrow icon)
 - Community**: "Get involved! Bug reporting, mailing list, chat, development and more." (represented by a speech bubble icon)
- Release Information:** A graphic of a Mac desktop computer screen displaying the message "Latest source Release **2.40.0** Release Notes (2023-03-12)" and a "Download for Mac" button.

Descargar Git para macOS.



Instalación de Git

Instalación de Git en macOS

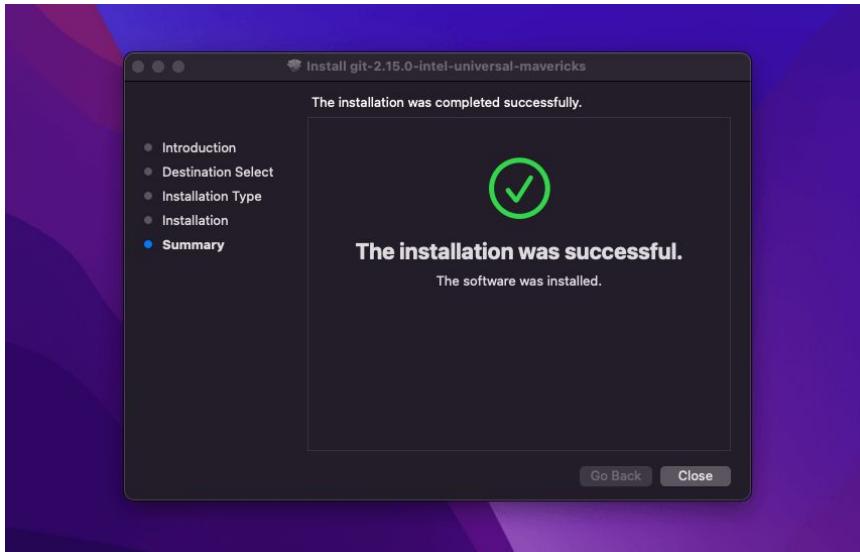


Instalador Git para macOS.



Instalación de Git

Instalación de Git en macOS



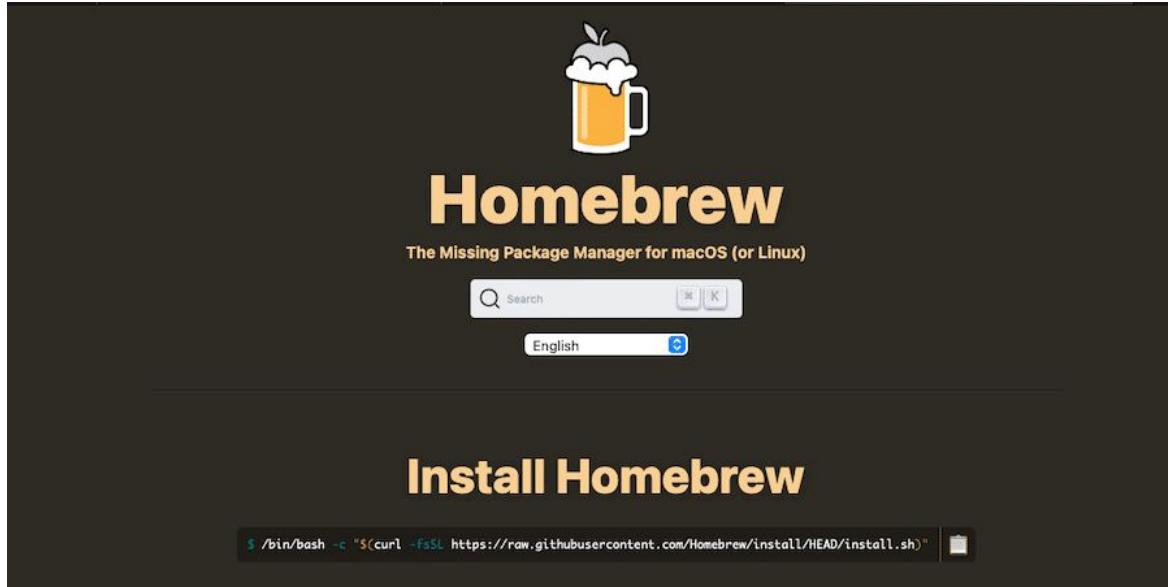
Un mensaje de confirmación de que la instalación de Git se ha realizado correctamente.



Instalación de Git

Instalación de Git en macOS

1. Usar Homebrew (Recomendado):





Instalación de Git

Instalación de Git en macOS

1. Usar Homebrew (Recomendado):

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

```
brew install git
```



Instalación de Git

Instalación de Git en macOS

1. Usar Homebrew (Recomendado):

```
brew --version
```

```
brew install git
```

```
git --version
```



Instalación de Git

Instalación de Git en macOS

2. Instalar Git desde Xcode Command Line Tools:

- Abre la terminal y ejecuta:

```
xcode-select --install
```

- Sigue las instrucciones para instalar las herramientas de línea de comandos, que incluyen Git.



Instalación de Git

Instalación de Git en macOS

3. Verificar la Instalación:

```
git --version
```

Deberías ver la versión de Git instalada.

```
git version 2.31.1
```

03.

Conceptos y operaciones básicas

El ABC del funcionamiento de Git





Comandos Básicos de Git

1. git init

- El comando `git init` se utiliza para crear un nuevo repositorio de Git en tu proyecto actual. Este comando inicializa un repositorio vacío en el directorio en el que te encuentras.

```
mkdir mi_proyecto
```

```
cd mi_proyecto
```

```
git init
```

Initialized empty Git repository in /path/to/mi_proyecto/.git/

Este comando crea un nuevo repositorio de Git en el directorio `mi_proyecto`.

Los commits

- Los commits en Git son un concepto fundamental del control de versiones. Un commit representa una "instantánea" del estado actual de los archivos en un proyecto en un momento determinado. Cada vez que haces un commit, estás guardando un registro de los cambios realizados en el código o en los archivos del proyecto desde el último commit. Este registro incluye qué archivos han sido modificados, agregados o eliminados, junto con un mensaje descriptivo que explica los cambios realizados.

Los commits

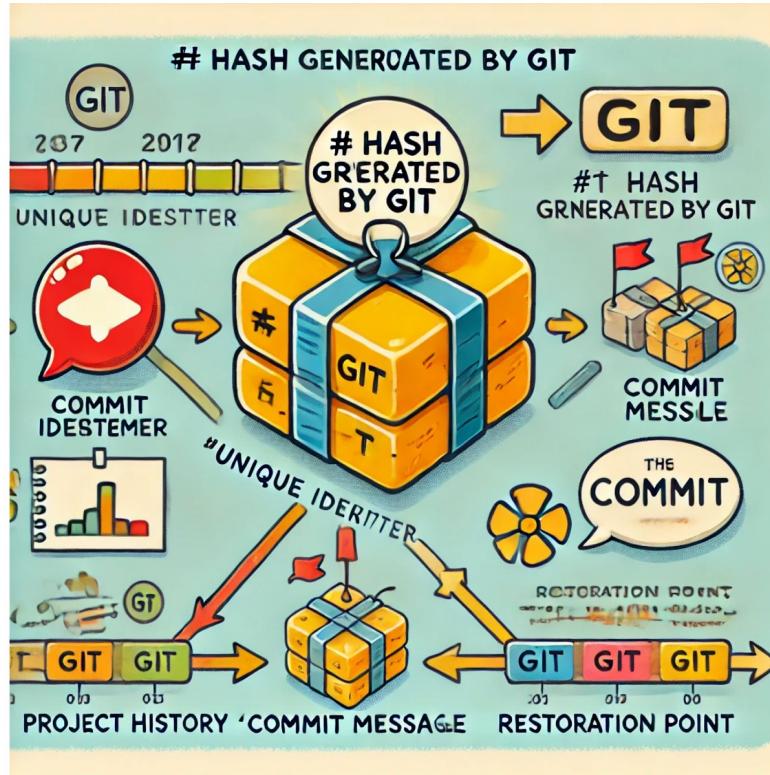
Características principales de un commit:

Identificación Única

Mensaje de Commit

Historia del Proyecto

Punto de Restauración



Los commits

Características principales de un commit:

Si estás trabajando en un proyecto y has modificado algunos archivos, para guardar esos cambios en el repositorio, harías lo siguiente:

- 1. Agregar archivos al área de staging:

```
bash git add archivo.txt
```

- 2. Realizar un commit:

```
bash git commit -m "Agregué una nueva función a  
archivo.txt"
```

Este commit ahora es parte del historial del proyecto, y puedes referenciarlo o volver a él en el futuro si es necesario.

Trabajo con Repositorios en Git

1. git add

- El comando `git add` se utiliza para añadir cambios en el área de staging (o preparación) antes de confirmarlos. Básicamente, prepara los archivos para ser incluidos en el próximo commit.

- Uso:

```
git add <nombre_del_archivo>
```

o

```
git add .
```

Trabajo con Repositorios en Git

1. git add

- El comando `git add` se utiliza para añadir cambios en el área de staging (o preparación) antes de confirmarlos. Básicamente, prepara los archivos para ser incluidos en el próximo commit.

Ejemplo:

```
git add index.html
```

- Añade el archivo `index.html` al área de staging.

```
git add .
```

- Añade todos los archivos modificados y nuevos en el directorio actual al área de staging.

Trabajo con Repositorios en Git

2. git commit

- El comando `git commit` se utiliza para guardar los cambios preparados en el historial del repositorio. Un commit es una "instantánea" de los archivos en el área de staging en un momento específico.

- Uso:

```
git commit -m "Mensaje descriptivo del commit"
```

Ejemplo:

```
git commit -m "Añadido el archivo index.html con la estructura básica"
```

- Crea un commit con el mensaje ``"Añadido el archivo index.html con la estructura básica``.

Trabajo con Repositorios en Git

3. git status

- El comando `git status` muestra el estado de los archivos en el directorio de trabajo y el área de staging. Es útil para ver qué archivos han sido modificados, cuáles están en el área de staging, y cuáles no están siendo rastreados por Git.
- Uso:

```
git status
```

```
git status
```

Trabajo con Repositorios en Git

3. git status

- El comando `git status` muestra el estado de los archivos en el directorio de trabajo y el área de staging. Es útil para ver qué archivos han sido modificados, cuáles están en el área de staging, y cuáles no están siendo rastreados por Git.

- Ejemplo:

```
git status
```

On branch main

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: index.html

- Muestra que `index.html` está en el área de staging listo para ser confirmado.

Trabajo con Repositorios en Git

3. git status

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)

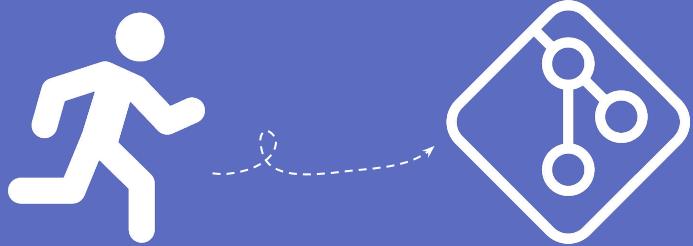
modified: styles.css

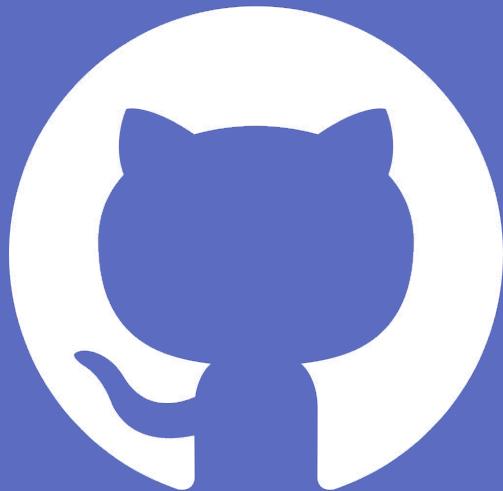
- Muestra que `styles.css` ha sido modificado pero no está en el área de staging.

Pausa 10 minutos



El trabajo local es divertido, ahora empecemos
a ver cómo trabajar con repositorios remotos





GitHub

04.

Creación de usuario y configuración inicial

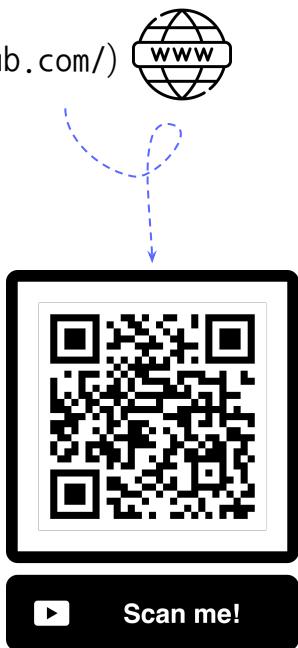
¿Cómo ponemos en marcha esto para trabajar en remoto?





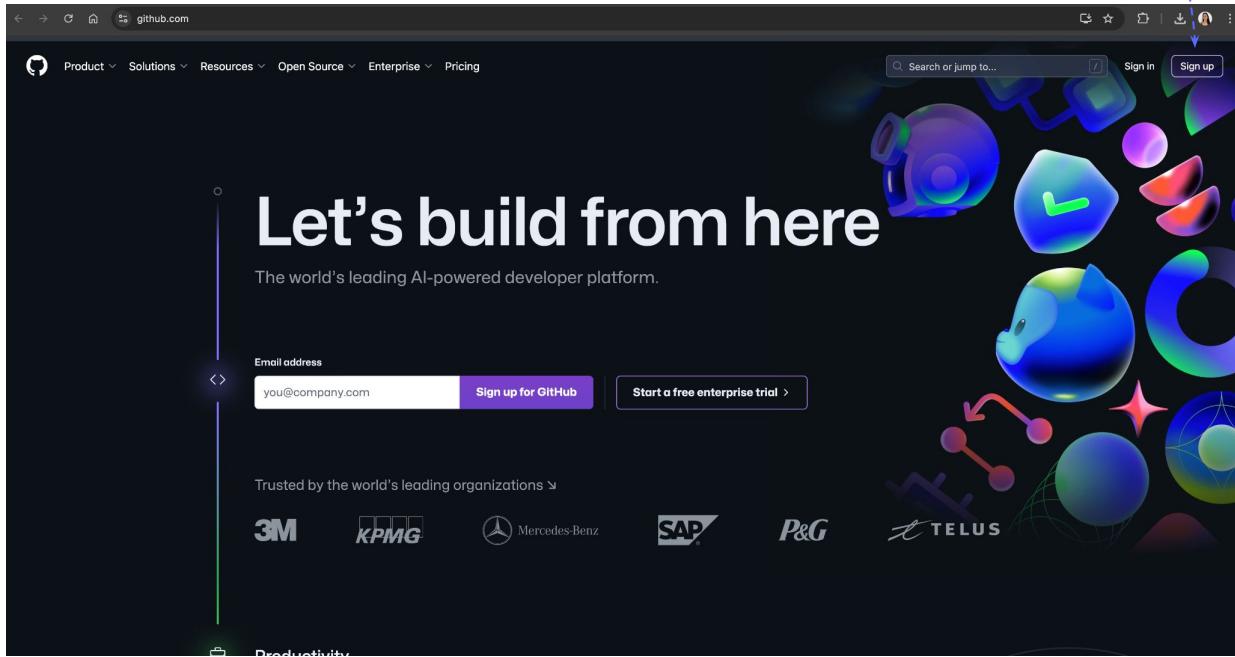
Creación de cuenta (en GitHub)

1. Ingresar al sitio web de GitHub (<https://github.com/>)



Creación de cuenta (en GitHub)

1. Ingresar al sitio web de GitHub
2. Ingresar al registro



Creación de cuenta (en GitHub)

1. Ingresar al sitio web de GitHub
2. Ingresar al registro
3. Completar el formulario de registro

The screenshot shows the GitHub registration process. At the top right, there is a link to "Sign in". The main form area has a dark background with white text. It starts with "Welcome to GitHub! Let's begin the adventure". The first field is "Enter your email*" with the value "csimoes@pasteur.edu.uy" followed by a green checkmark. The second field is "Create a password*" with a partially obscured password followed by a green checkmark. The third field is "Enter a username*" with the value "csimoesa" followed by a green checkmark. Below these fields is a section for "Email preferences" with a checkbox labeled "Receive occasional product updates and announcements." The checkbox is unchecked. To the right of this section is a "Continue" button. At the bottom of the form, there is a small note: "By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails."

Welcome to GitHub!
Let's begin the adventure

Enter your email*

✓ csimoes@pasteur.edu.uy

Create a password*

✓*

Enter a username*

✓ csimoesa

Email preferences

Receive occasional product updates and announcements.

Continue

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

Creación de cuenta (en GitHub)

1. Ingresar al sitio web de GitHub
2. Ingresar al registro
3. Completar el formulario de registro



Comprobar (CAPTCHA)

Verificar (mail)

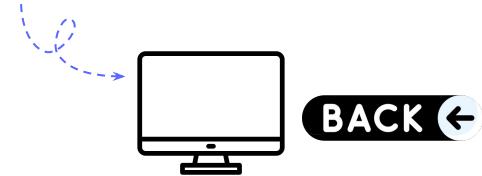
Mini formulario extra (preferencias, etc.)

Elegir plan gratuito o pago 😊

The screenshot shows the GitHub registration interface. At the top right, there is a link to "Sign in". The main form area has a dark background with white text. It starts with "Welcome to GitHub! Let's begin the adventure". Below that, there are fields for "Enter your email*" with a green checkmark and the value "csimoes@pasteur.edu.uy". There is also a "Create a password*" field with a green checkmark and several dots representing the password. The next step is "Enter a username*" with a green checkmark and the value "csimoesa". Under "Email preferences", there is a checkbox labeled "Receive occasional product updates and announcements." which is unchecked. A "Continue" button is located at the bottom right of the form. At the very bottom of the page, there is a small note: "By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails."

Creación de cuenta (en GitHub)

1. Ingresar al sitio web de GitHub
2. Ingresar al registro
3. Completar el formulario de registro
4. Configuración inicial de Git (local)



Creación de cuenta (en GitHub)

1. Ingresar al sitio web de GitHub
2. Ingresar al registro
3. Completar el formulario de registro
4. Configuración inicial de Git (local)



```
git config --global user.name "Tu Nombre"  
git config --global user.email "tu.email@ejemplo.com"
```

Verificar la configuración con:

```
git config --list
```

Creación de cuenta (en GitHub)

- 1.** Ingresar al sitio web de GitHub
- 2.** Ingresar al registro
- 3.** Completar el formulario de registro
- 4.** Configuración inicial de Git (local)



Autenticación segura en GitHub

Acceso a recursos en GitHub:



Web



GitHub Desktop/
aplicaciones



Línea de
comandos



API

Cada forma de acceso soporta distintos modos de autenticación:

- Nombre de usuario y contraseña (2FA recomendada)
- Token de acceso personal (PAT)
- Clave SSH
- Passkeys y SAML

Autenticación segura en GitHub

Acceso a recursos en GitHub:



Web



GitHub Desktop/
aplicaciones



Línea de
comandos



API

Cada forma de acceso soporta distintos modos de autenticación:

- Nombre de usuario y contraseña (2FA recomendada)
- Token de acceso personal (PAT)
- Clave SSH
- Passkeys y SAML

La que vamos a usar

- + básica
- segura

+ conveniente y segura a largo plazo

Configuración: claves SSH

Comprobar claves SSH existentes



1. Abrir la terminal (o Git Bash)
2. `$ ls -al ~/.ssh`
3. Comprobar lista de directorio para ver clave pública.
 - `id_rsa.pub`
 - `id_ecdsa.pub`
 - `id_ed25519.pub`

Configuración: claves SSH

Generación de una nueva clave SSH



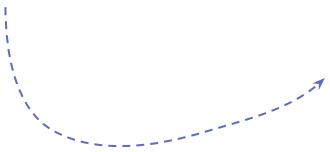
1. Abrir la terminal (o Git Bash)
2. `$ ssh-keygen -t ed25519 -C "your_email@example.com" o
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`
3. (opcional) ingresar frase de contraseña
4. Iniciar el agente SSH
`$ eval "$(ssh-agent -s)"`
5. Añadir la clave privada al agente SSH
`ssh-add ~/ssh/id_ed25519`

Configuración: claves SSH

Añadir clave SSH a cuenta de GitHub

```
$ cat ~/.ssh/id_ed25519.pub
```

(seleccionar y copiar el contenido)



Ir a GitHub

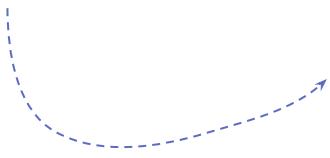
The screenshot shows a GitHub user profile for 'csimoesa'. The sidebar on the right is open, displaying various options: 'Your profile', 'Your repositories', 'Your Copilot', 'Your projects', 'Your stars', 'Your gists', 'Your organizations', 'Your enterprises', 'Your sponsors', 'Upgrade', 'Feature preview', and 'Settings'. The 'Settings' option is highlighted with a white border. The main content area shows a 'Join GitHub Education!' modal with information about GitHub Education services like Copilot, Heroku, and Microsoft Azure. Below the modal is the 'Home' section, which includes fields for starting a new repository and creating a profile README.

Configuración: claves SSH

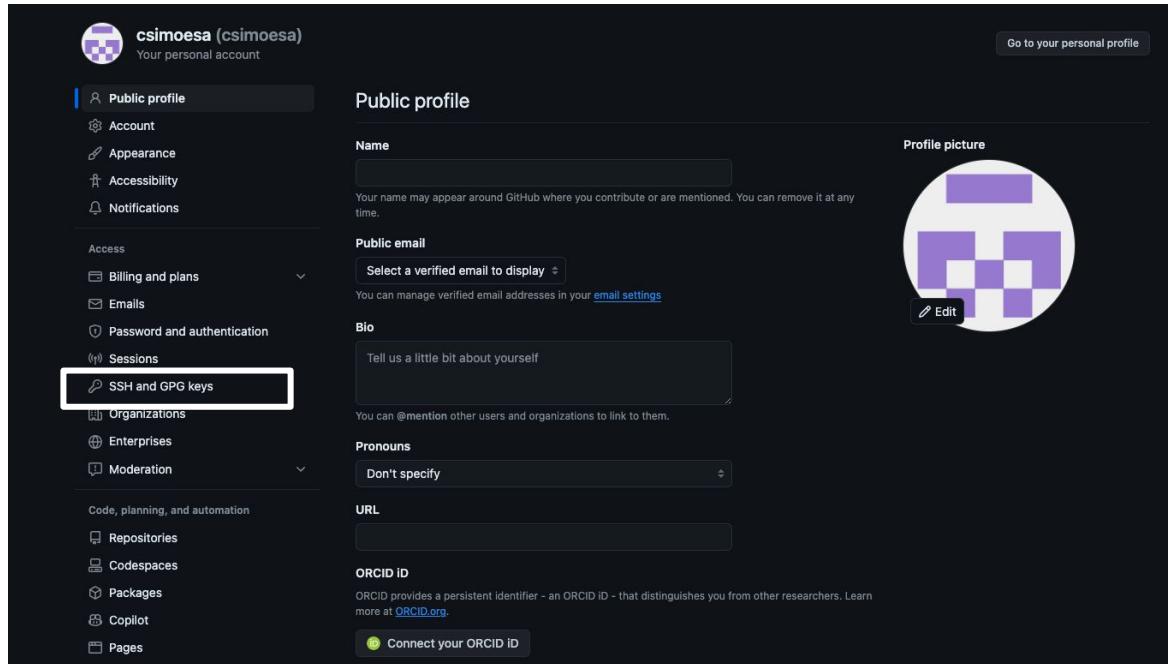
Añadir clave SSH a cuenta de GitHub

```
$ cat ~/.ssh/id_ed25519.pub
```

(seleccionar y copiar el contenido)



Ir a GitHub



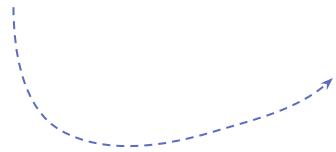
The screenshot shows a GitHub user profile for 'csimoesa'. The sidebar on the left has a dark theme and includes sections like 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access', 'Billing and plans', 'Emails', 'Password and authentication', 'Sessions', 'SSH and GPG keys' (which is highlighted with a white box), 'Organizations', 'Enterprises', 'Moderation', 'Code, planning, and automation', 'Repositories', 'Codespaces', 'Packages', 'Copilot', and 'Pages'. The main content area is titled 'Public profile' and contains fields for 'Name', 'Public email', 'Bio', 'Pronouns', 'URL', and 'ORCID ID'. A circular 'Profile picture' is shown on the right, with an 'Edit' button below it.

Configuración: claves SSH

Añadir clave SSH a cuenta de GitHub

```
$ cat ~/.ssh/id_ed25519.pub
```

(seleccionar y copiar el contenido)



Ir a GitHub

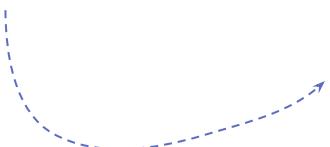
The screenshot shows the GitHub user profile for 'csimoesa'. The left sidebar has a 'SSH and GPG keys' section highlighted. The main content area displays the 'SSH keys' section, which states: 'There are no SSH keys associated with your account.' It includes links to 'connecting to GitHub using SSH keys' and 'troubleshoot common SSH problems'. Below this is the 'GPG keys' section, which also states: 'There are no GPG keys associated with your account.' It includes a link to 'generate a GPG key and add it to your account'. At the bottom is the 'Vigilant mode' section, which contains a checkbox for 'Flag unsigned commits as unverified'.

Configuración: claves SSH

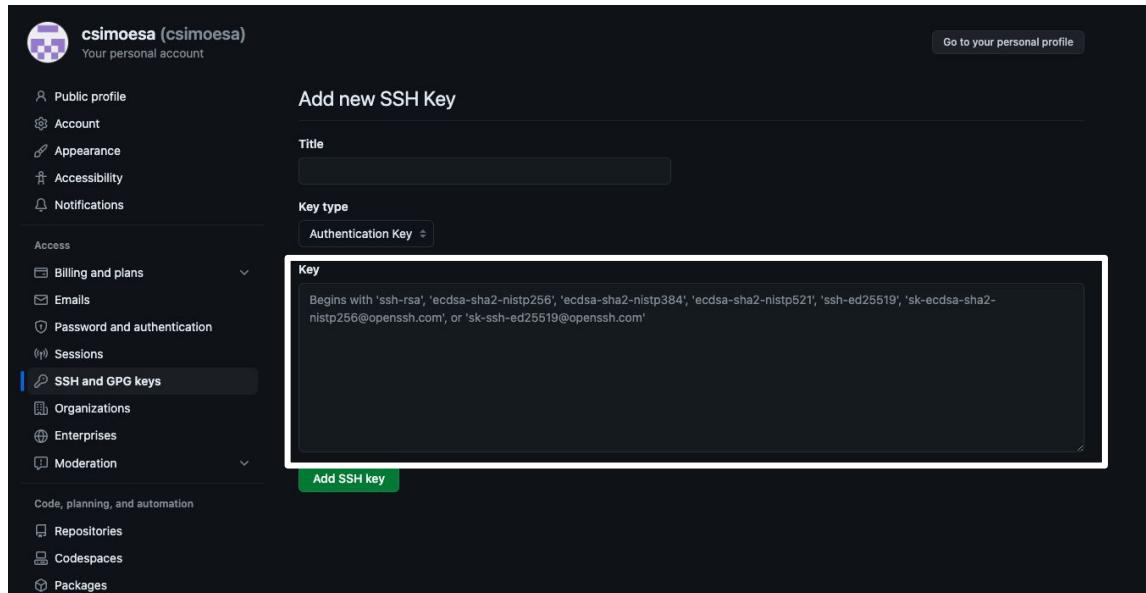
Añadir clave SSH a cuenta de GitHub

```
$ cat ~/.ssh/id_ed25519.pub
```

(seleccionar y copiar el contenido)



Ir a GitHub



The screenshot shows the GitHub user profile page for 'csimoesa'. The left sidebar has a dark theme with white text. The 'SSH and GPG keys' option is highlighted with a blue bar. The main area is titled 'Add new SSH Key'. It has fields for 'Title' (empty) and 'Key type' (set to 'Authentication Key'). The 'Key' field contains the copied SSH public key:
`Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'`
At the bottom right of this field is a green button labeled 'Add SSH key'.

Creemos nuestro primer repositorio en GitHub 😊

Ingresar al sitio web de GitHub (<https://github.com/>)
(y autenticarse)



Creemos nuestro primer repositorio en GitHub

The screenshot shows the GitHub 'Create a new repository' interface. A 'Coming Soon' sign with a hand holding a yellow key is pointing to the 'Add .gitignore' section. Blue dashed arrows point from the text labels to specific fields:

- nombre del repositorio: points to the 'Repository name' field containing 'mi_primer_repo'.
- Descripción (opcional): points to the 'Description (optional)' text area containing 'Este es mi primer repositorio de git.'
- Visibilidad: points to the 'Visibility' section where 'Private' is selected.
- Añadir README inicial: points to the 'Initialize this repository with:' section, specifically the 'Add a README file' checkbox.

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner * csimoesa / **Repository name** * mi_primer_repo [mi_primer_repo is available.](#)

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

Description (optional)
Este es mi primer repositorio de git.

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:

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

Add .gitignore

.gitignore template: None

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

Choose a license

License: None

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

This will set [main](#) as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a private repository in your personal account.

Create repository

Creemos nuestro primer repositorio en GitHub

The screenshot shows a GitHub repository page for a private repository named "mi_primer_repo". The repository has 1 branch and 0 tags. It contains one commit from user "csimoesa" titled "Initial commit" made 4 minutes ago. The README file also has an "Initial commit" 4 minutes ago. The repository description in the README is "Este es mi primer repositorio de git.". The "About" section includes links to Readme, Activity, and Stats (0 stars, 1 watching, 0 forks). The "Releases" section indicates no releases have been published, with a link to "Create a new release". The "Packages" section indicates no packages have been published, with a link to "Publish your first package".

mi_primer_repo · Private

Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

csimoesa Initial commit cad8e25 · 4 minutes ago 1 Commit

README.md Initial commit 4 minutes ago

README

mi_primer_repo

Este es mi primer repositorio de git.

About

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

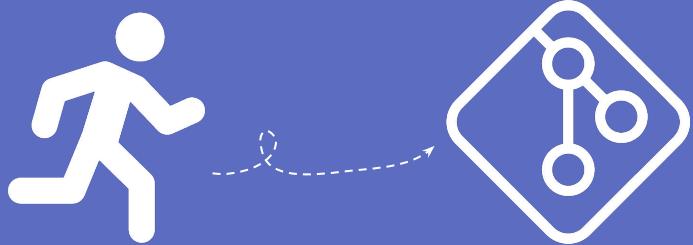
Packages

No packages published Publish your first package



¿Y AHORA QUÉ?

Ahora que tenemos cuenta, y repo,
¿jugamos un poco en remoto?



05.

Flujo de trabajo

A lo que vinimos... (o no)

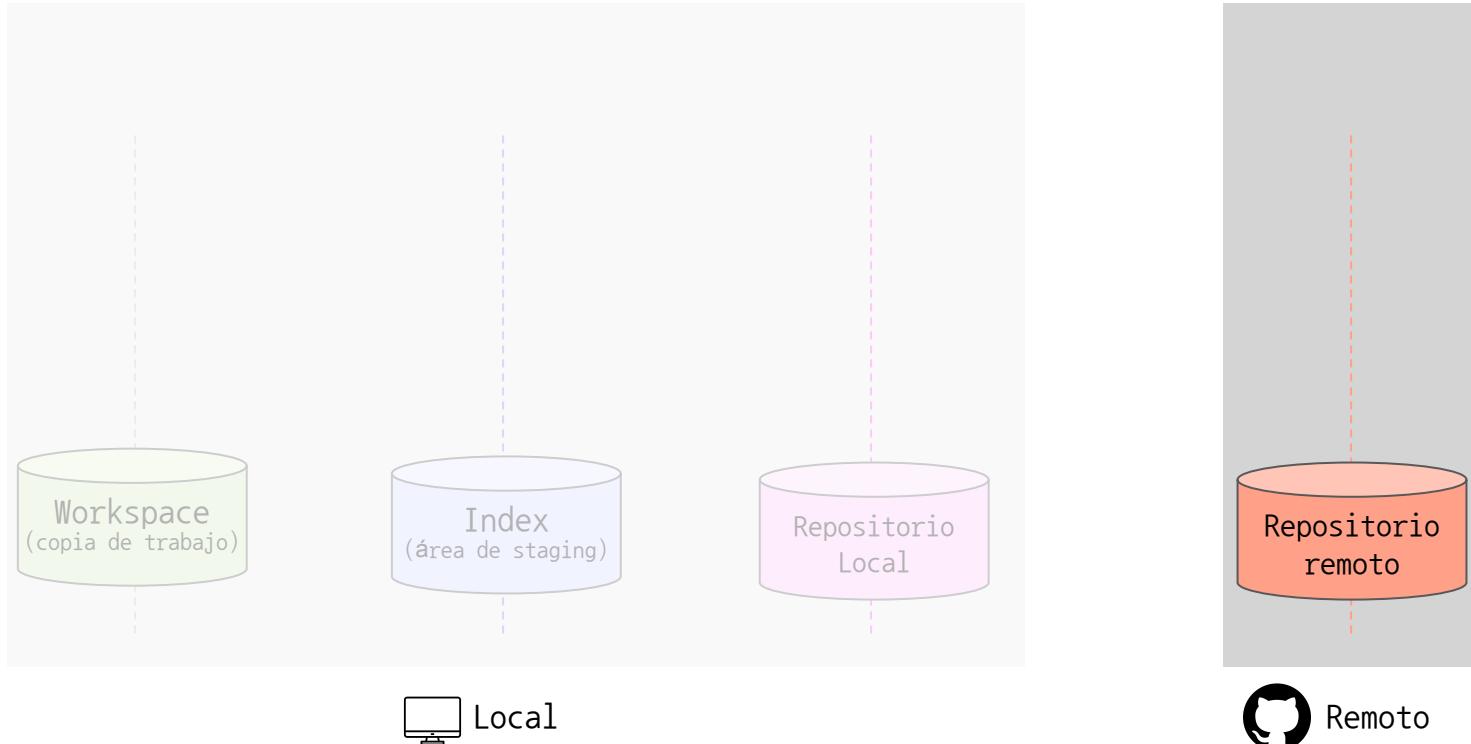


Flujo de trabajo básico: un (mini) ejemplo

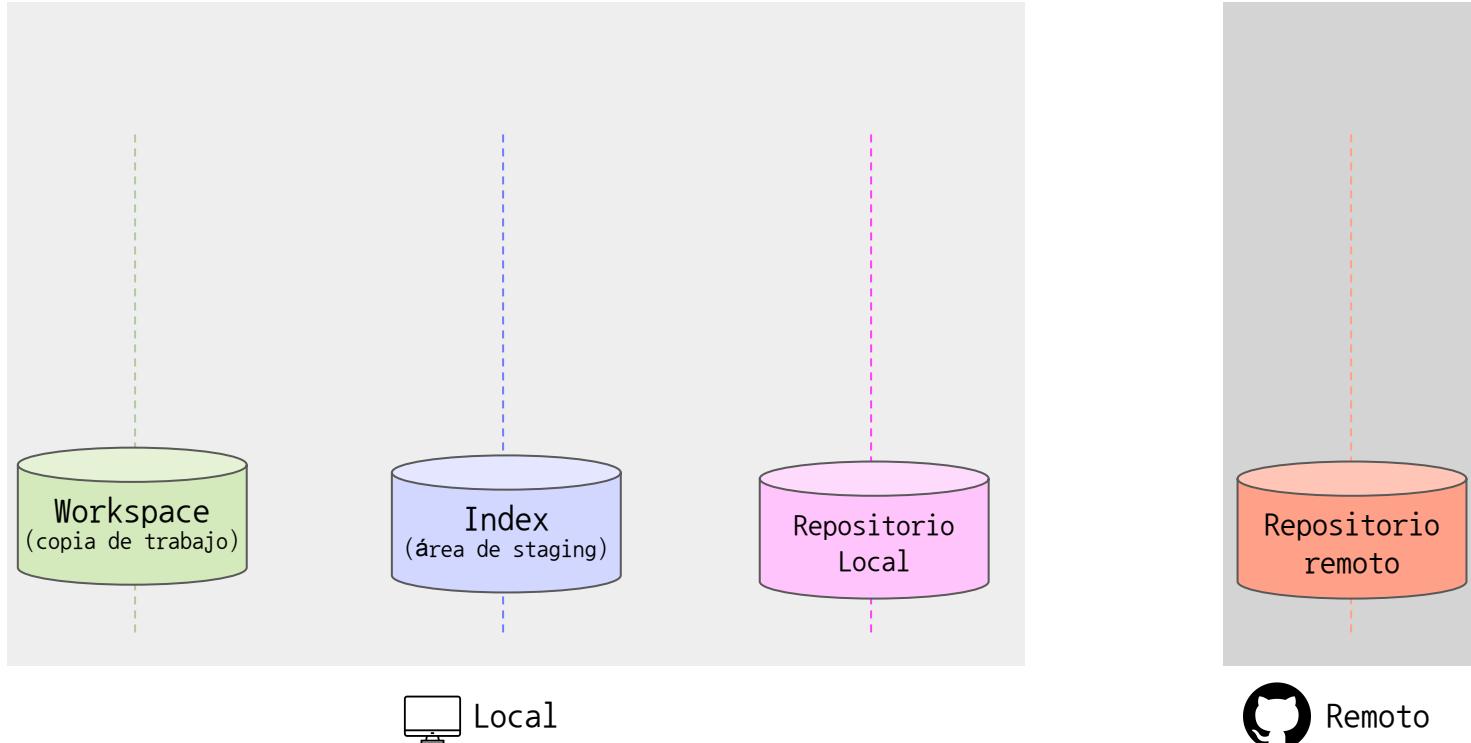
1.  **Clonar** un repositorio existente/descargar cambios
2.  **Modificar** archivos
3.  **Subir** los cambios



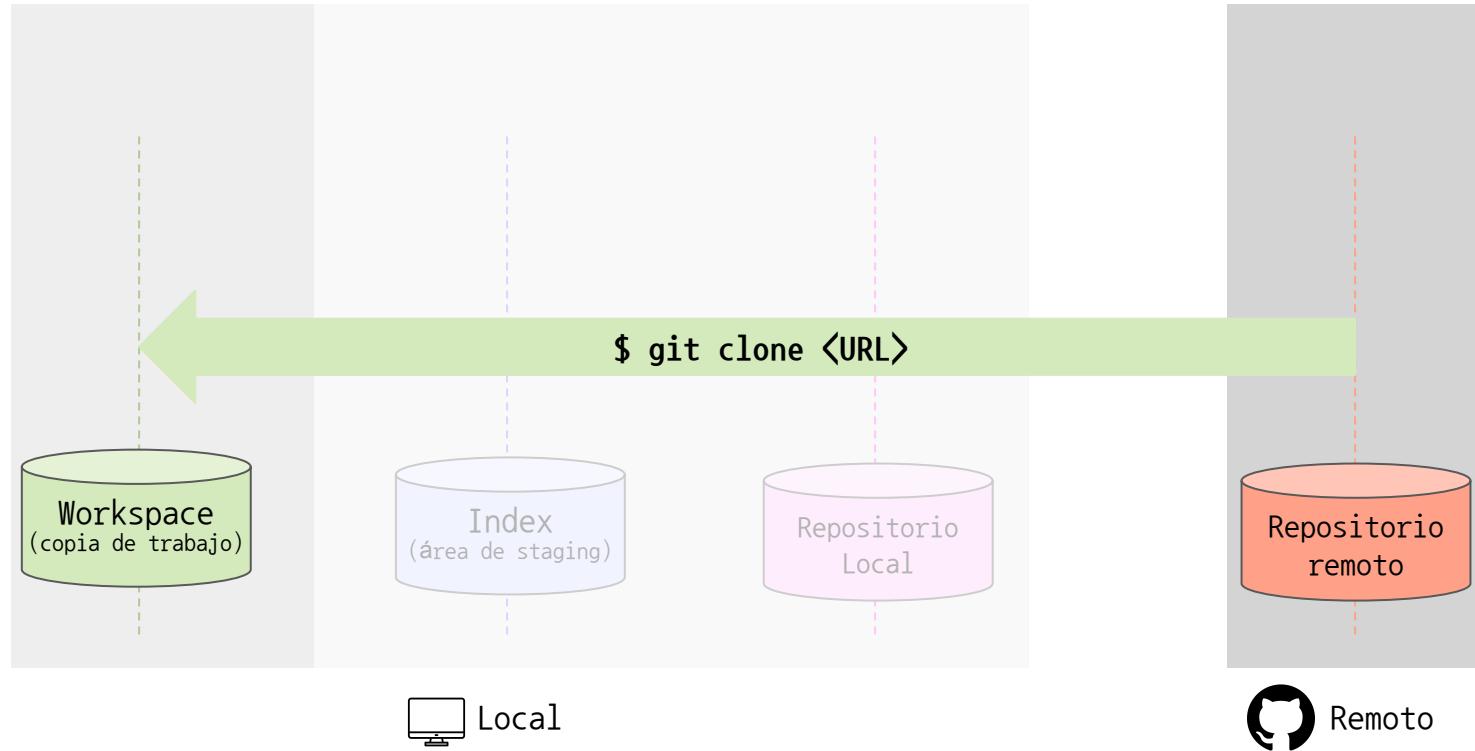
↓ Clonar repositorio desde GitHub



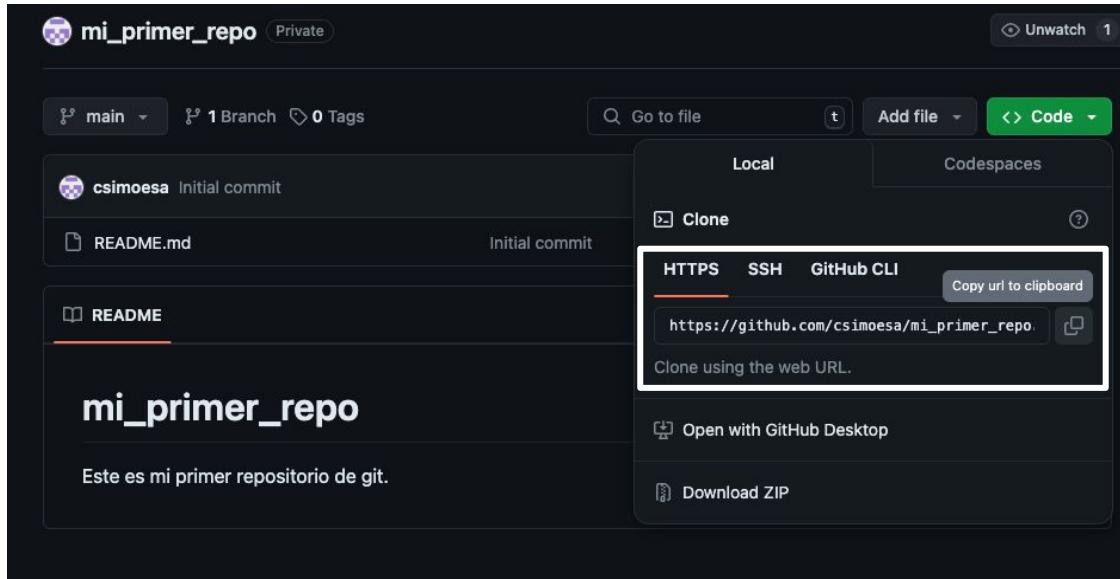
↓ Clonar repositorio desde GitHub



↓ Clonar repositorio desde GitHub



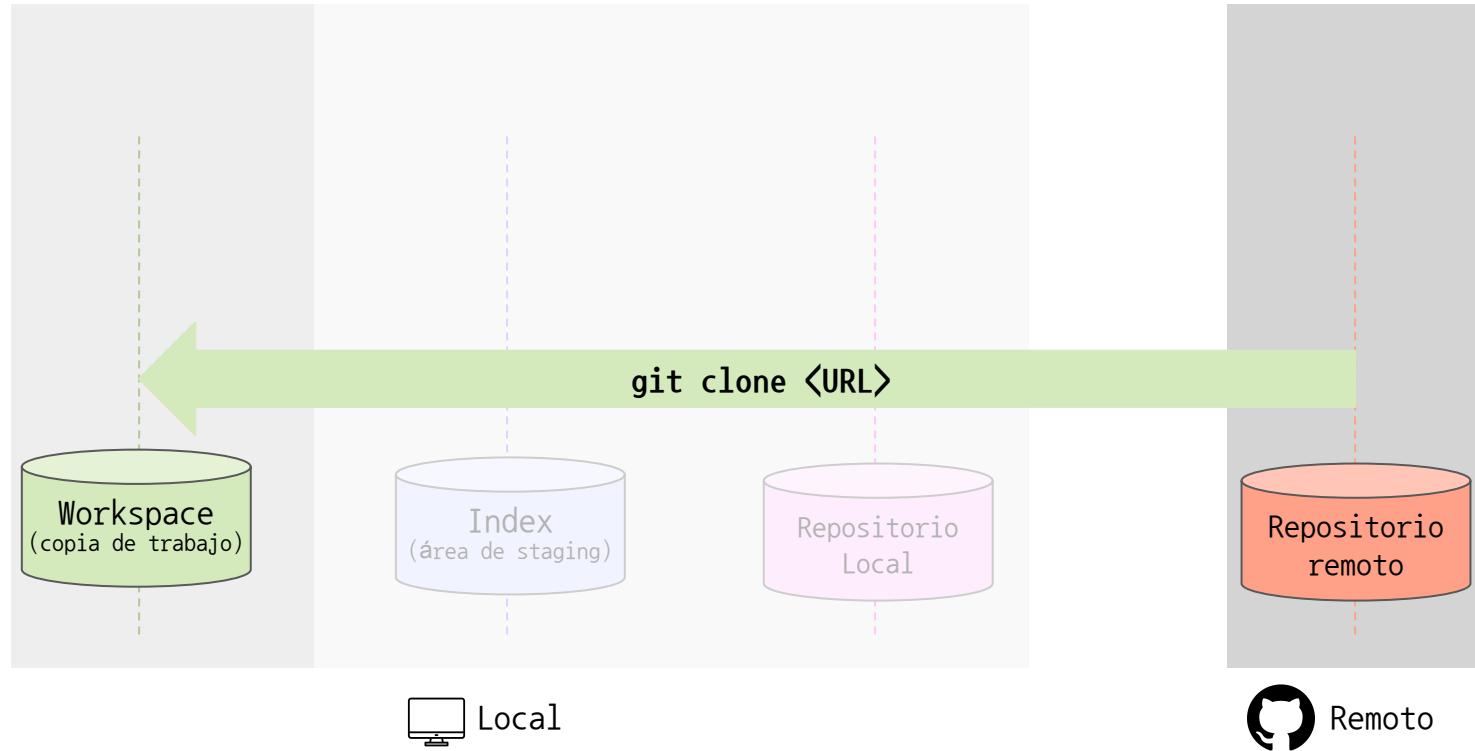
↓ Clonar repositorio desde GitHub



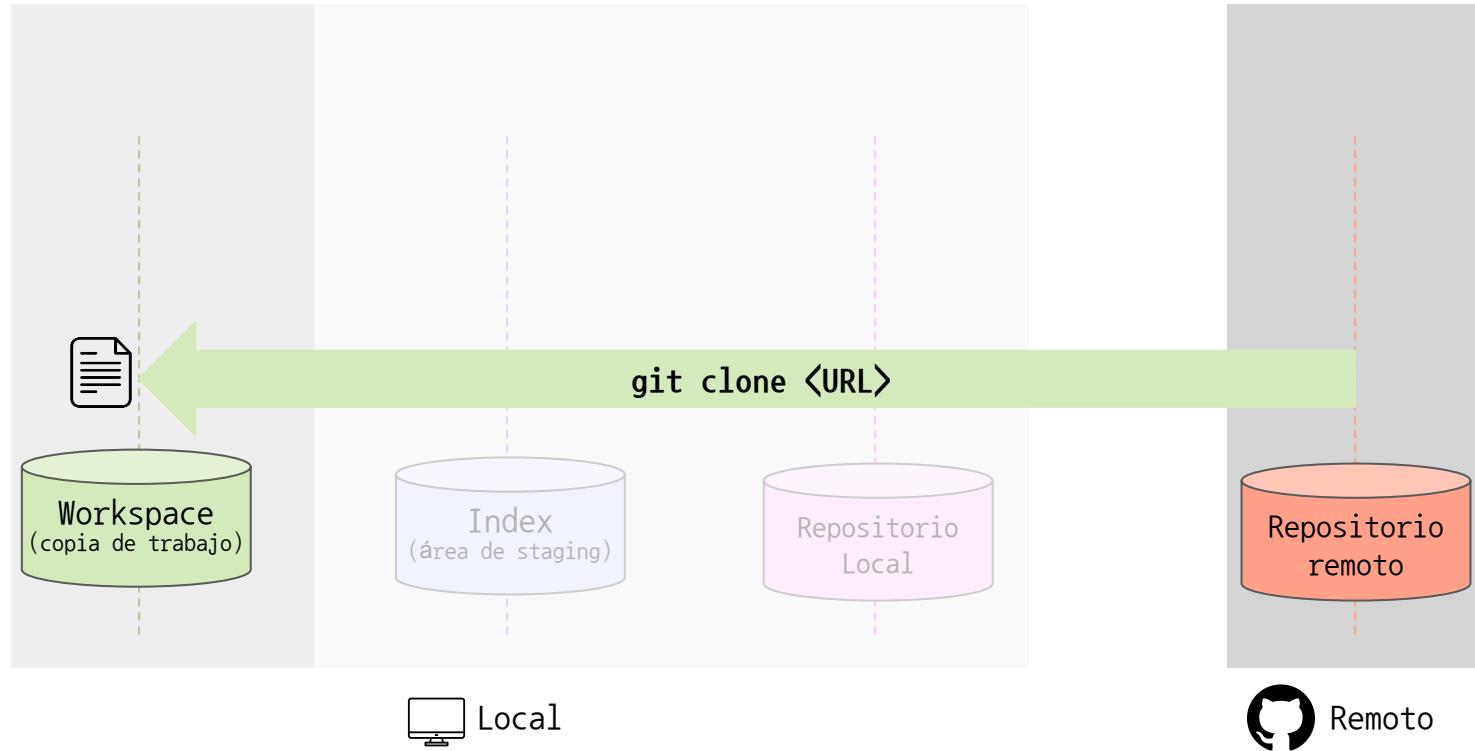
<URL>

- SSH: Clave pública/privada, forma segura, sin usuario/contraseña
- HTTPS: usuario y contraseña

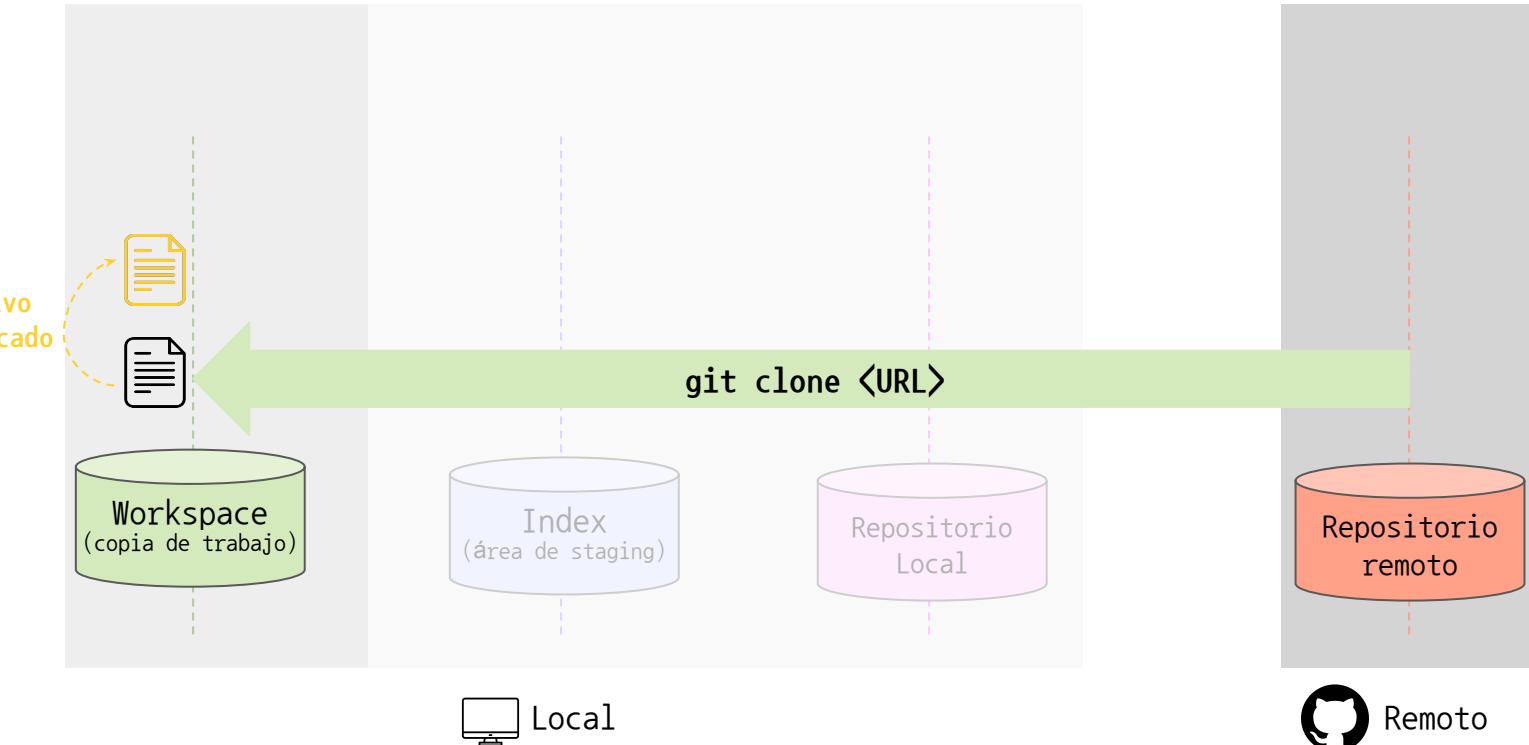
↓ Clonar repositorio desde GitHub



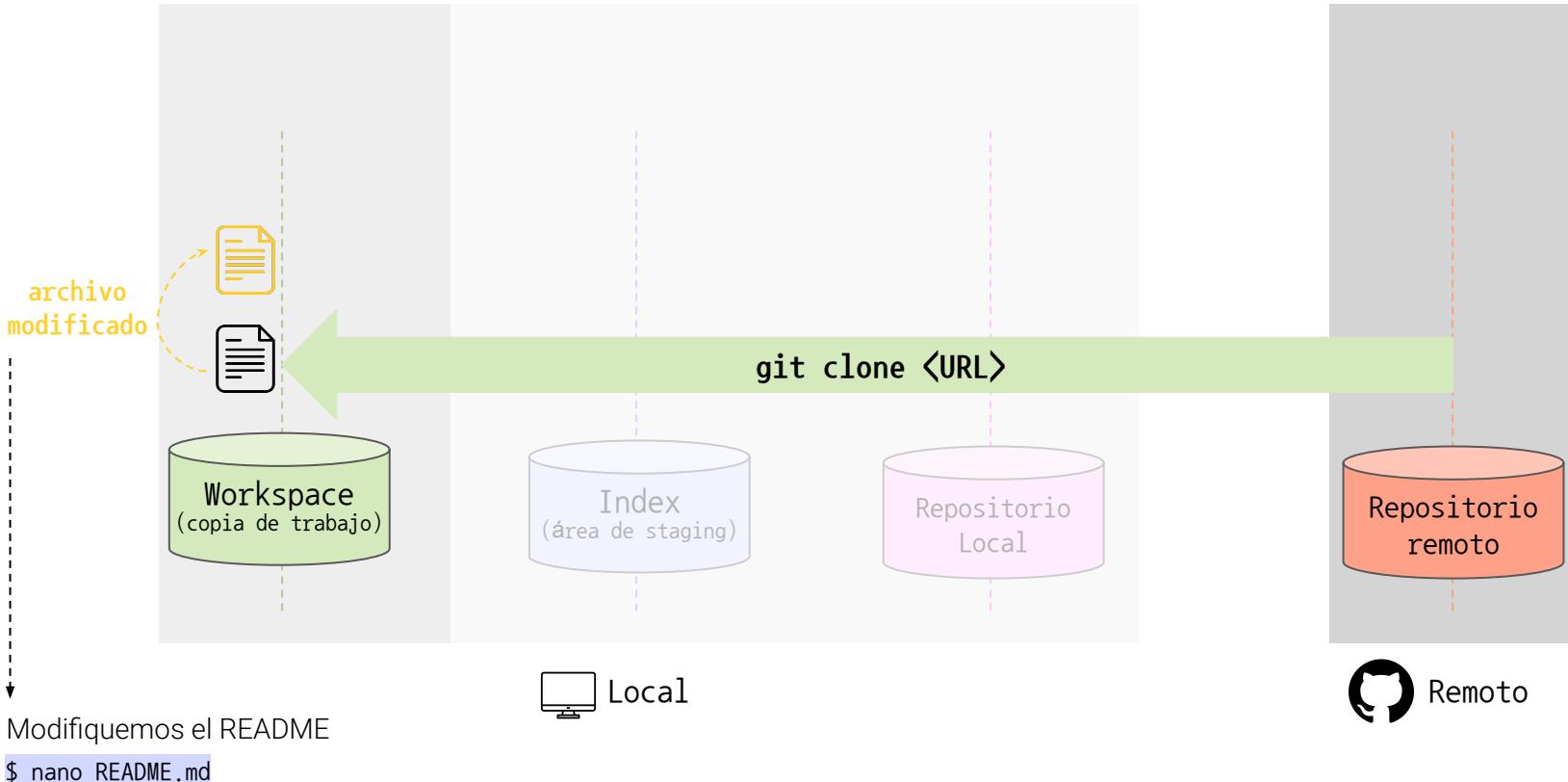
↓ Clonar repositorio desde GitHub



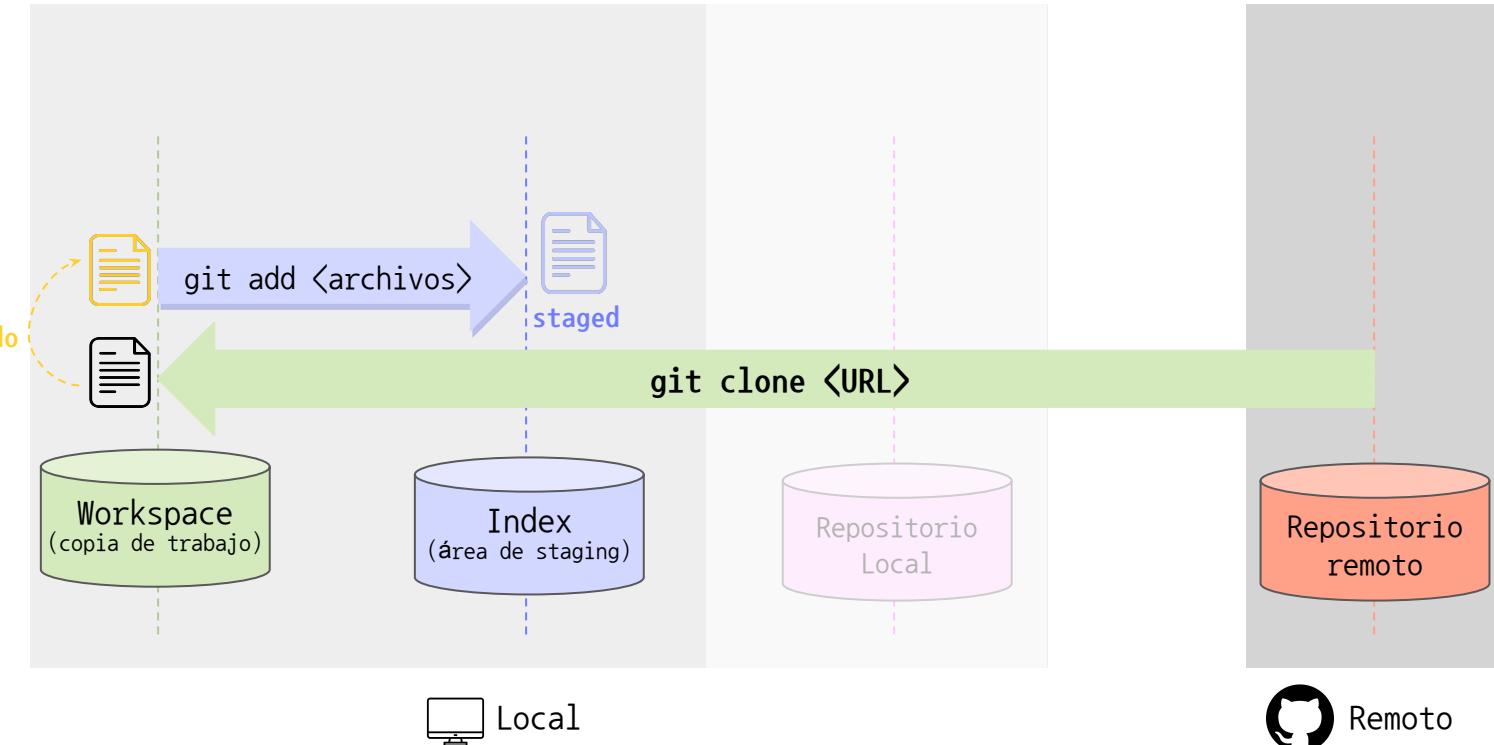
Modificar archivos



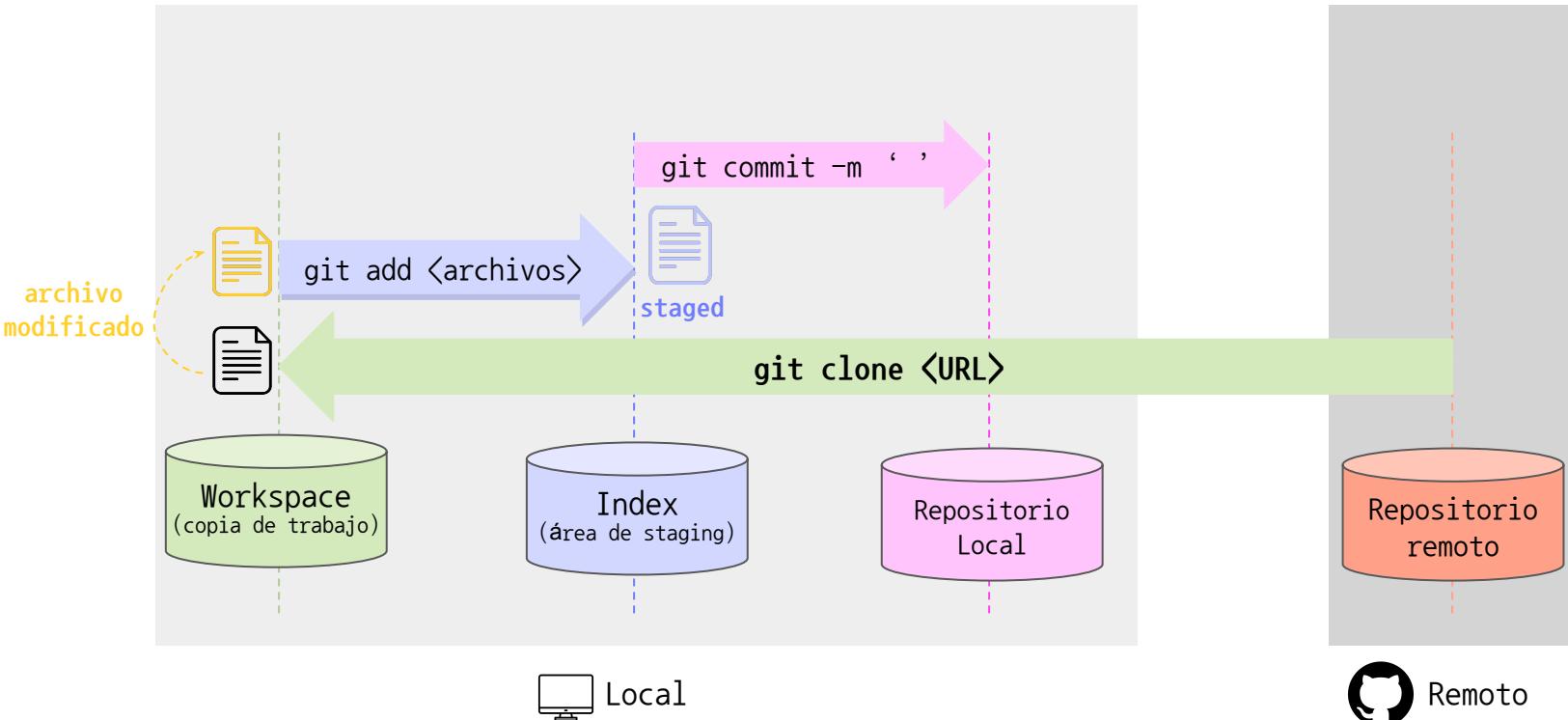
Modificar archivos



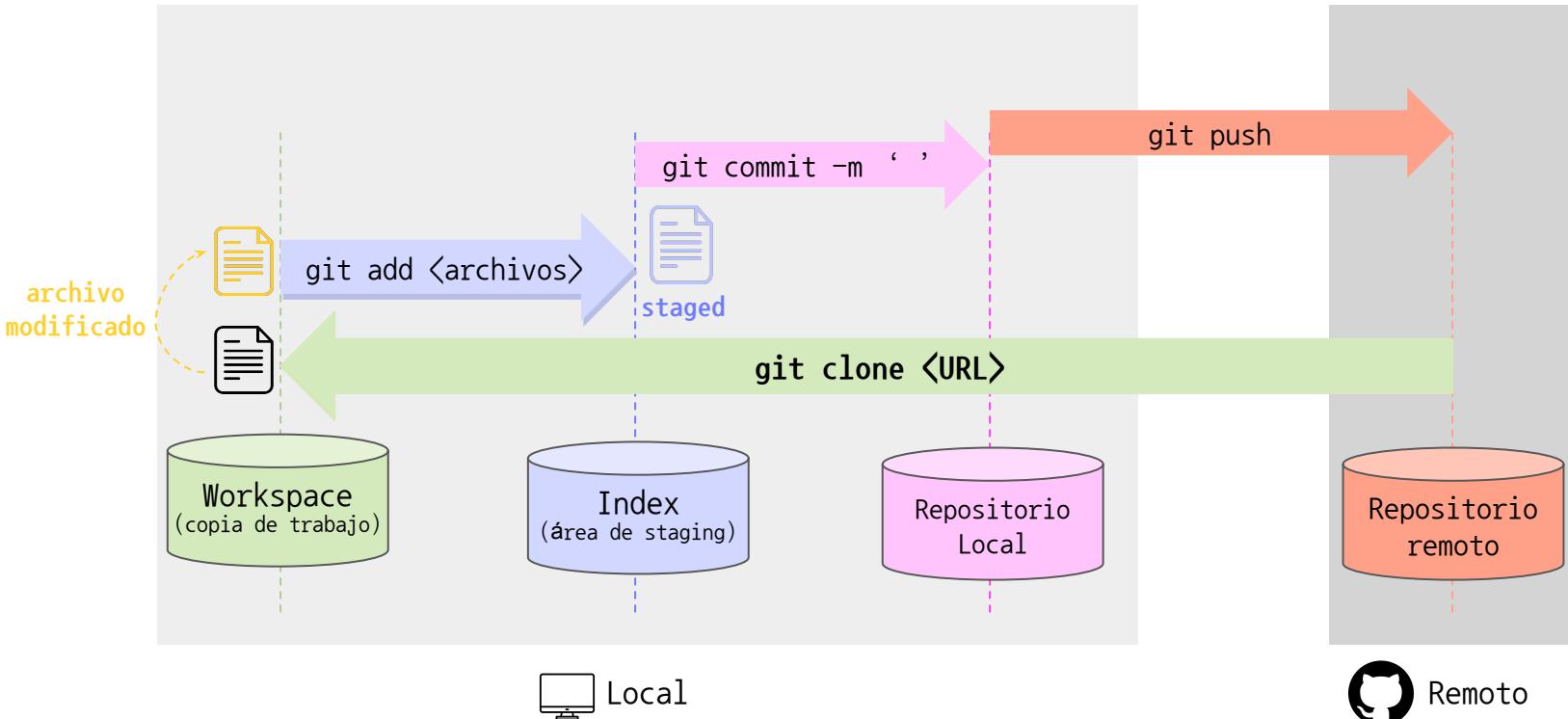
Modificar archivos



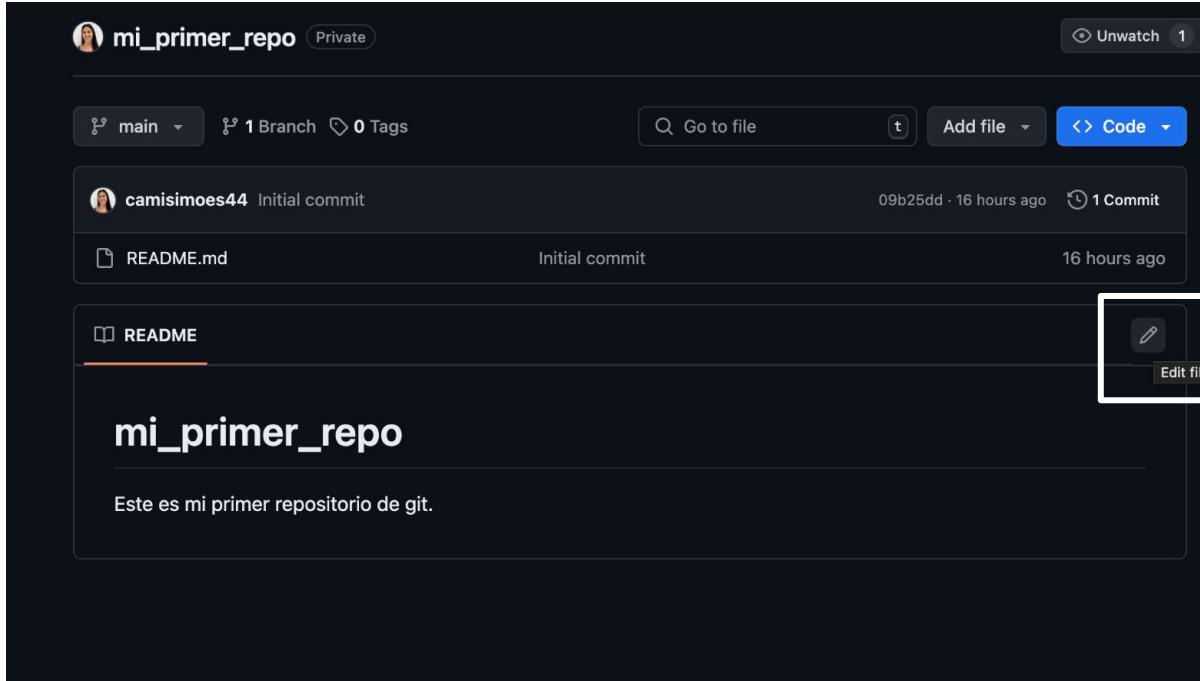
↑ Subir los cambios



↑ Subir los cambios

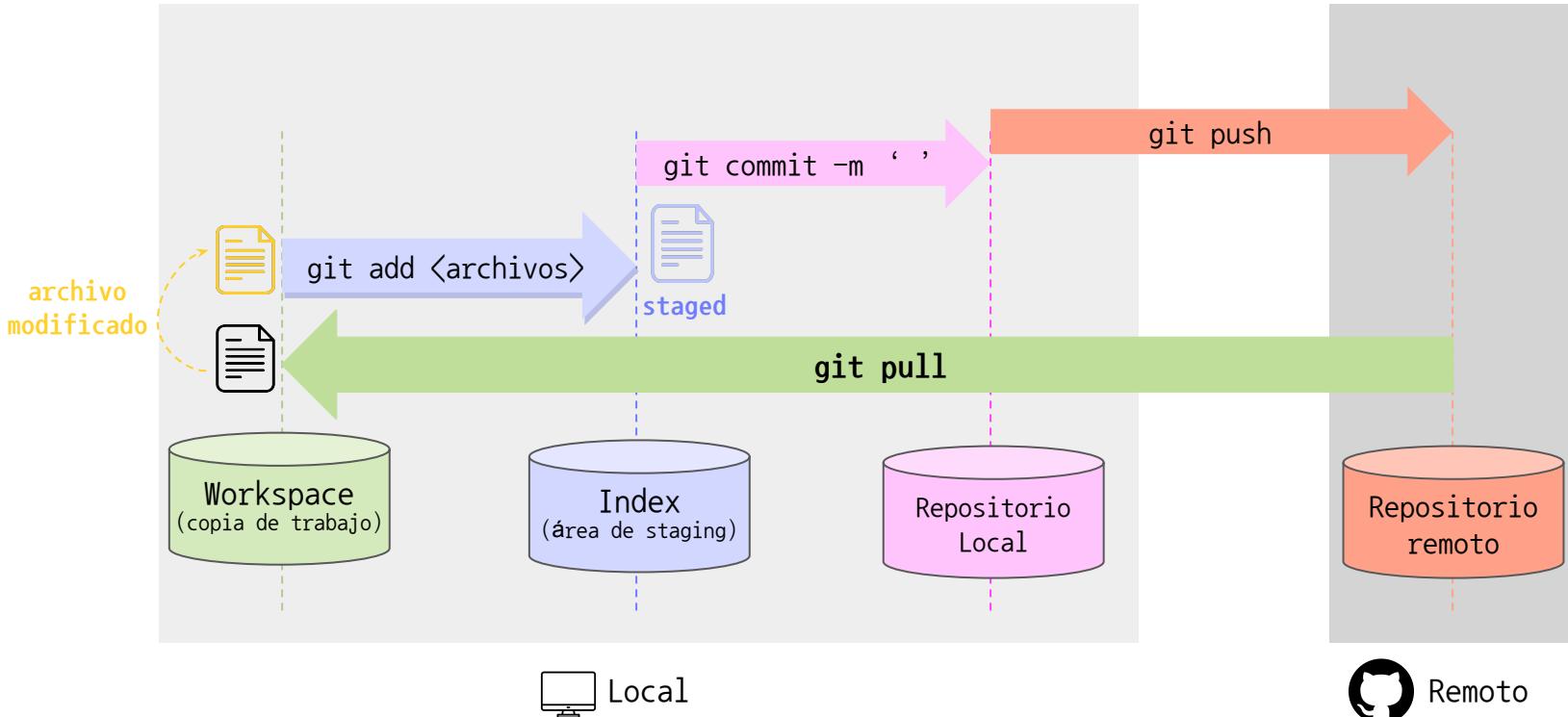


Modificar en remoto





Descargar sólo las diferencias: *git pull*



Pausa 10 minutos



.gitignore

¿Qué es?

- Archivo de texto que le dice a Git qué archivos o directorios deben ser ignorados en un repositorio (no se rastrearán ni serán agregados al historial del repositorio).
- Objetivo:
 - Evitar que archivos no deseados o sensibles se incluyan en los commits (archivos de configuración locales, credenciales, archivos binarios grandes, resultados intermedios, etc.)
 - Mantener el repositorio limpio y enfocado en el código relevante.

.gitignore

¿Qué es?

- Archivo de texto que le dice a Git qué archivos o directorios deben ser ignorados en un repositorio (no se rastrearán ni serán agregados al historial del repositorio).
- Objetivo:
 - Evitar que archivos no deseados o sensibles se incluyan en los commits (archivos de configuración locales, credenciales, archivos binarios grandes, resultados intermedios, etc.)
 - Mantener el repositorio limpio y enfocado en el código relevante.

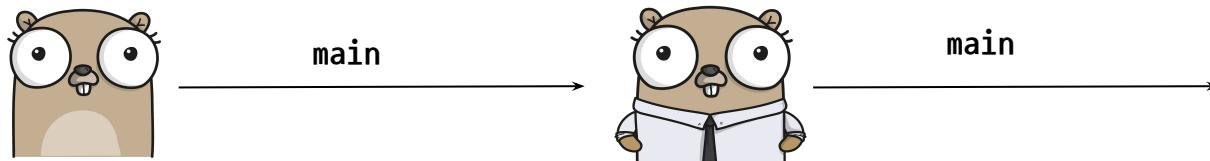
¿Cómo se crea? Hagamos un ejemplo

1. Abrir la consola.
2. Crear un ejemplo de archivo a ignorar:
`$ touch ignore.txt
$ nano ignore.txt`
3. Guardar el .pdf de la presentación en el directorio del repositorio.
4. Crear el .gitignore:
`$ touch .gitignore
$ nano .gitignore`
5. Escribir en .gitignore el archivo a ignorar:
`*.pdf
ignore.txt`
6. `$ git add .gitignore
$ git commit -m "Añadir archivo .gitignore"`

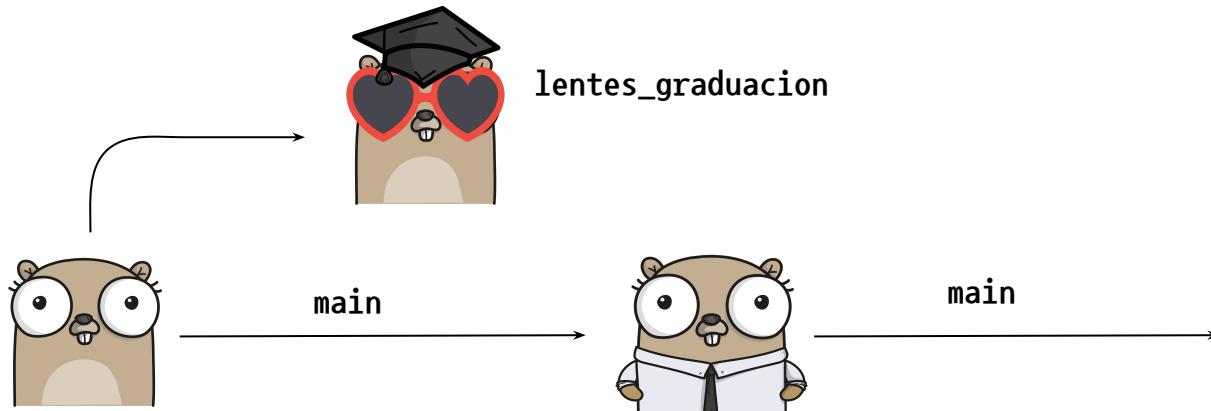
Flujo de trabajo avanzado: trabajo con ramas (branches)

- Versión paralela del proyecto que diverge del proyecto principal.
- Permite desarrollar funcionalidades, corregir errores, o experimentar con nuevas ideas sin afectar la rama principal ([main](#) o [master](#)).
- Objetivo:
 - Facilita el desarrollo aislado de nuevas funcionalidades o correcciones.
 - Permite trabajar en varias tareas en paralelo sin interferir con la estabilidad del proyecto principal.
 - Mejora la colaboración en equipos, ya que cada miembro puede trabajar en su propia rama y luego integrar sus cambios.

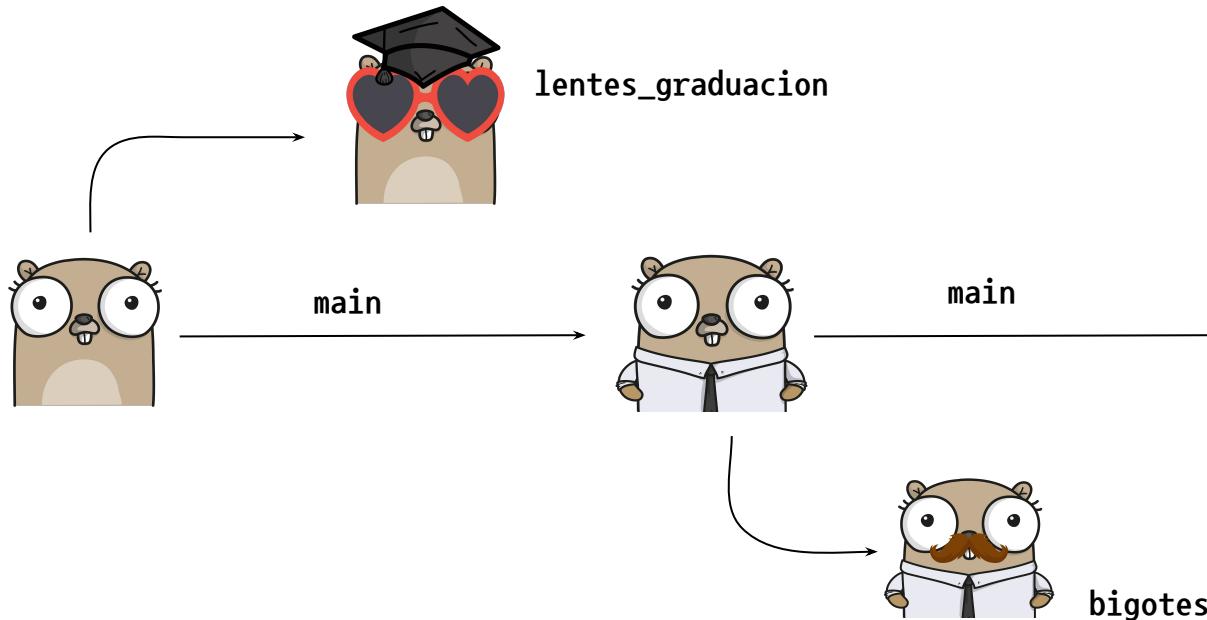
Ramas (branches)



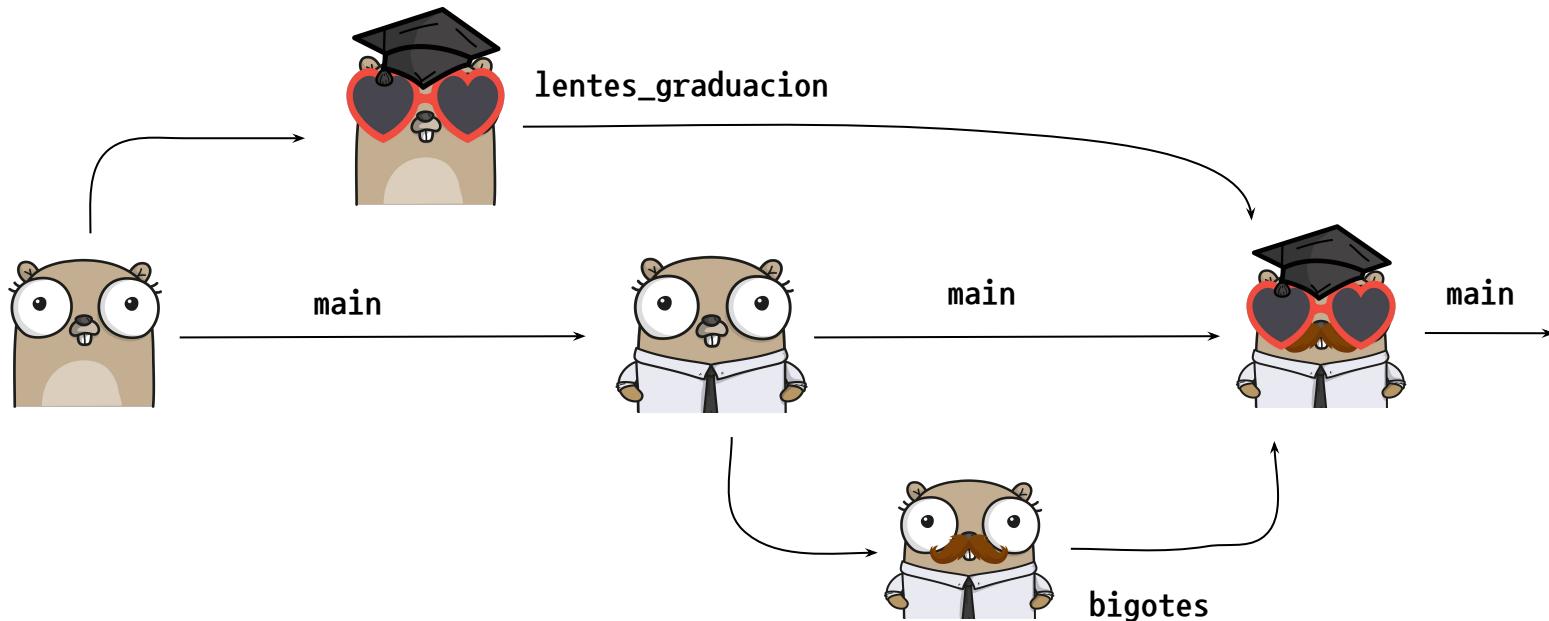
Ramas (branches)



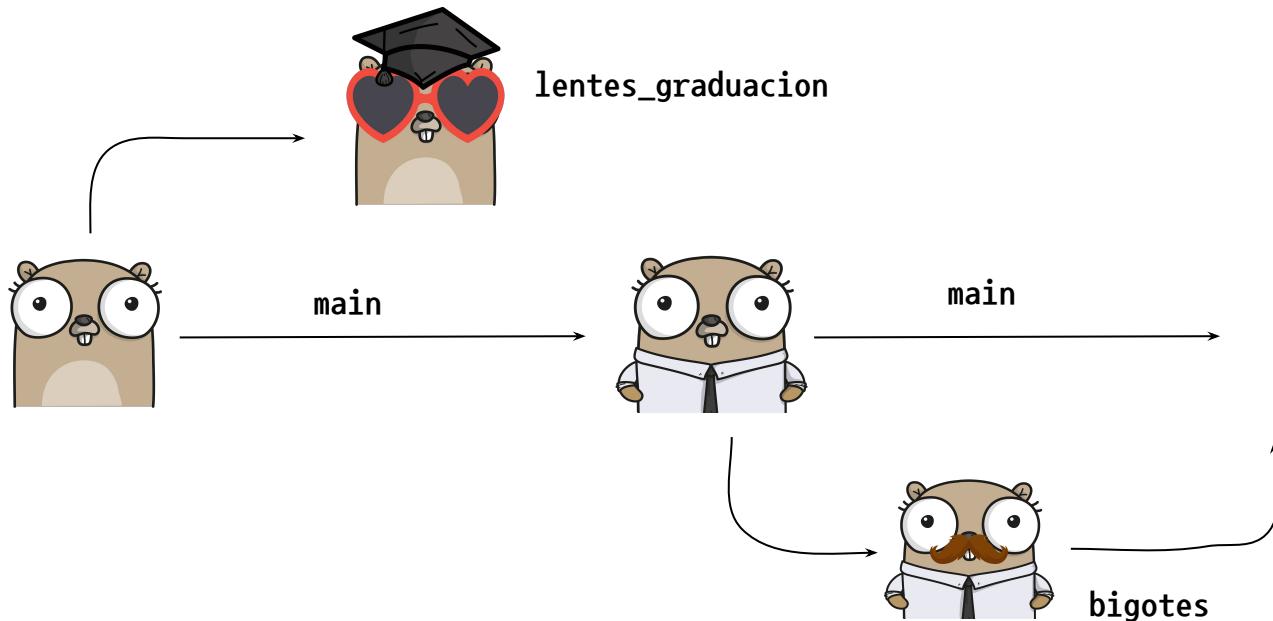
Ramas (branches)



Ramas (branches)

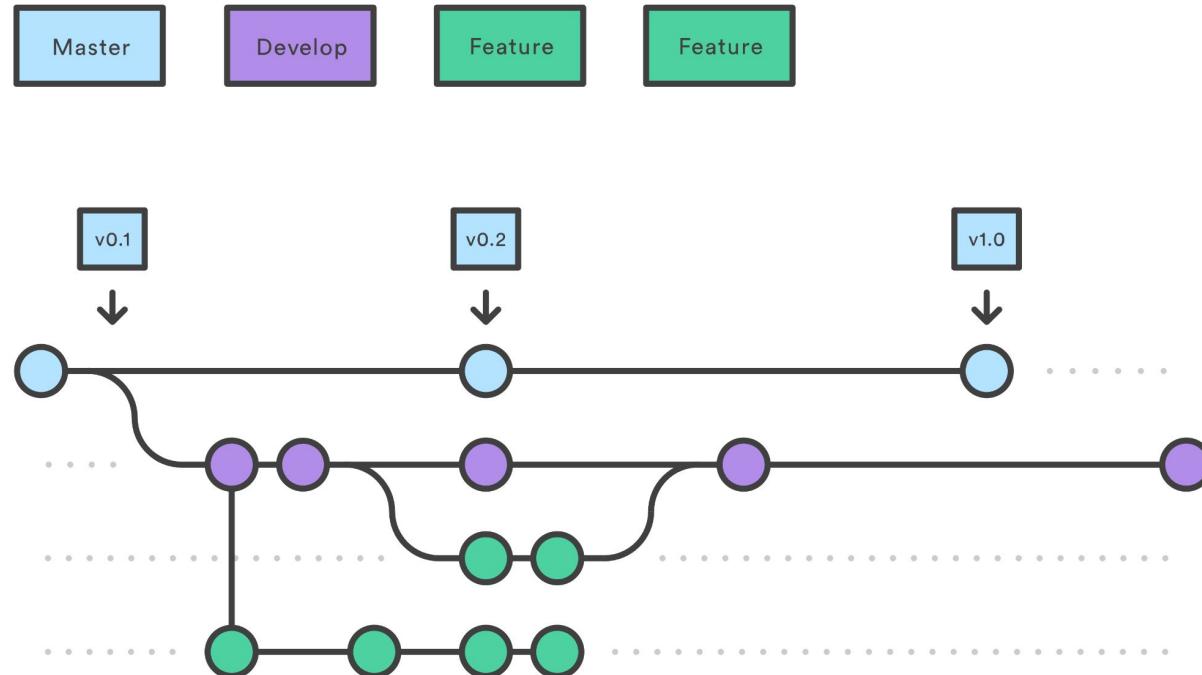


Ramas (branches)



Cada rama es independiente, lo que permite trabajar en características nuevas sin riesgo de dañar el código base.

Flujo de trabajo avanzado: trabajo con ramas (*branches*)



Flujo de trabajo avanzado: trabajo con ramas (branches)

Crear una nueva rama

```
git branch <nombre_de_la_rama>
```

Cambiar a una rama

```
git checkout <nombre_de_la_rama>
```

Listar las ramas

```
git branch
```

Eliminar una rama

```
git -d <nombre_de_la_rama>
```

Hagamos un ejemplo

```
git checkout -b update_readme  
git branch
```

Edita el archivo README.md para agregar más detalles.

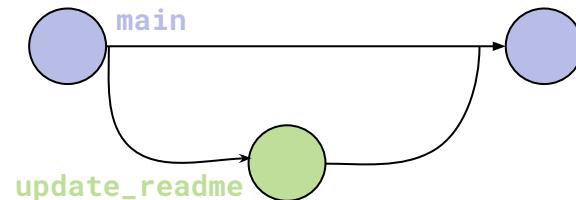
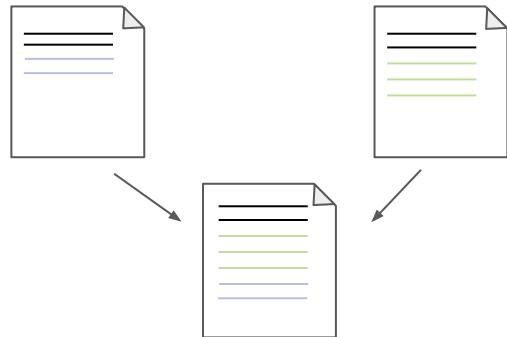
Crea o modifica otros archivos según sea necesario.

```
git add README.md  
git commit -m "Actualizar README"
```



¿Y AHORA QUÉ?

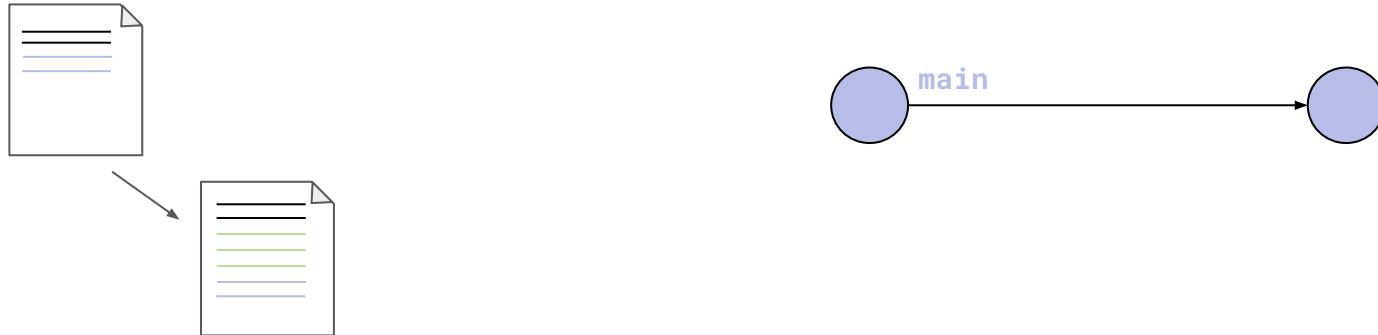
Flujo de trabajo avanzado: fusionar ramas (merge)



Una vez que hemos terminado y probado los cambios, podemos fusionar la rama `update-readme` con la rama `main`.

```
git checkout main
git merge update_readme
```

Flujo de trabajo avanzado: fusionar ramas (merge)



Una vez que hemos terminado y probado los cambios, podemos fusionar la rama `update-readme` con la rama `main`.

```
git checkout main
git merge update_readme
git branch -d update-readme
```

Resolución de Conflictos en Git

Un conflicto surge cuando Git intenta fusionar cambios y encuentra modificaciones en las mismas líneas o en áreas que no pueden reconciliarse automáticamente.





Resolución de Conflictos en Git

Situaciones Comunes:

- Al realizar un `git merge` entre dos ramas con cambios divergentes.
- Durante un `git pull` si los cambios en el repositorio remoto entran en conflicto con los cambios locales.

Cómo Saber que Hay un Conflicto:

- Git te notificará automáticamente durante un merge, pull si encuentra un conflicto.
- Los archivos con conflictos serán marcados en el estado del repositorio, que puedes ver con `git status`.



Resolución de Conflictos en Git

Vamos a clonar el siguiente repositorio: <https://github.com/RSG-Uy/Seminario-RSG-Git>

The screenshot shows the GitHub repository page for 'Seminario-RSG-Git'. The repository is public and was forked from 'NAican/Seminario-RSG-Git'. The main branch has 1 branch and 0 tags. A message indicates it is 3 commits ahead of and 3 commits behind the 'NAican/Seminario-RSG-Git:main' branch. The commit history shows the following entries:

Commit	Message	Time
68a1306 · 44 minutes ago	resuelvo conflicto quedandome con el comentario de camila	6 Commits
repo iniciado	.gitignore	18 hours ago
repo iniciado	Seminario-RSG-Git.Rproj	18 hours ago
resuelvo conflicto quedandome con el comentario de ca...	the_best_plot.R	44 minutes ago

Generemos un conflicto:

1. Modificar en el remoto el archivo: `the_best_plot.R`
2. En la copia local modificamos el mismo archivo con otro cambio
3. Committeamos local, push y ...

Resolución de Conflictos en Git

Identificación de Conflictos

- Marcas de Conflicto en los Archivos:
 - Git inserta marcas en los archivos en conflicto para ayudarte a identificar los cambios:

<<<<< HEAD

Código local en conflicto

=====

Código del remoto en conflicto

>>>>> nombre_de_rama_remota

```
colors.txt ×
src > colors.txt
1 red
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
2 <<<<< HEAD (Current Change)
3 green
4 =====
5 white
6 >>>>> his-branch (Incoming Change)
7 blue

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
react-app-demo ➜ my-branch ➜ git merge his-branch
Auto-merging src/colors.txt
CONFLICT (content): Merge conflict in src/colors.txt
Automatic merge failed; fix conflicts and then commit the result.
✖ react-app-demo ➜ my-branch ➜ >M<
```

Resolución de Conflictos en Git

Resolviendo Conflictos Manualmente

- Paso 1: Editar los Archivos en Conflicto
 - Abre los archivos que están en conflicto y revisa las diferencias entre las versiones.
 - Decide si mantienes la versión local, la versión remota, o una combinación de ambas.
 - Elimina las marcas de conflicto `<<<<<` , `=====`, `>>>>>` una vez que hayas resuelto el conflicto.

Resolución de Conflictos en Git

Resolviendo Conflictos Manualmente

- Paso 2: Marcar los Archivos como Resueltos
- Una vez que hayas resuelto todos los conflictos, marca los archivos como resueltos usando:

```
git add <archivo_resuelto>
```

- Paso 3: Completar el Proceso
- Si estabas haciendo un merge, completa el proceso con:

git commit

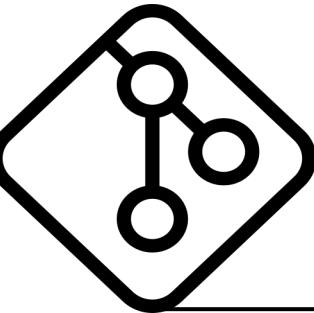
Resolución de Conflictos en Git

Estrategias para Evitar Conflictos

- **Hacer Pull Regularmente:** Mantén tu rama local actualizada con los últimos cambios del repositorio remoto.
- **Pequeños y Frecuentes Commits:** Realiza commits pequeños y frecuentes para reducir el área de posible conflicto.
- **Revisión de Código y Comunicación:** Coordina con tu equipo, revisa los pull requests, y comunica cambios grandes antes de hacer merge.

Pausa 10 minutos





FINAL_CORREGIDO_FINALISIMO.V5

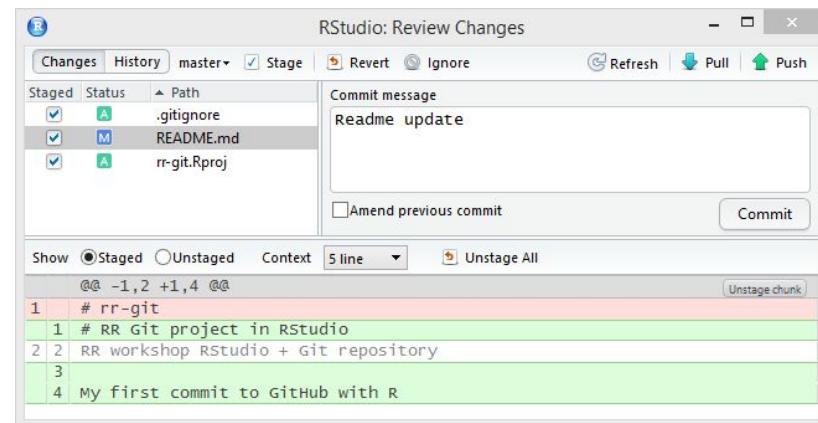
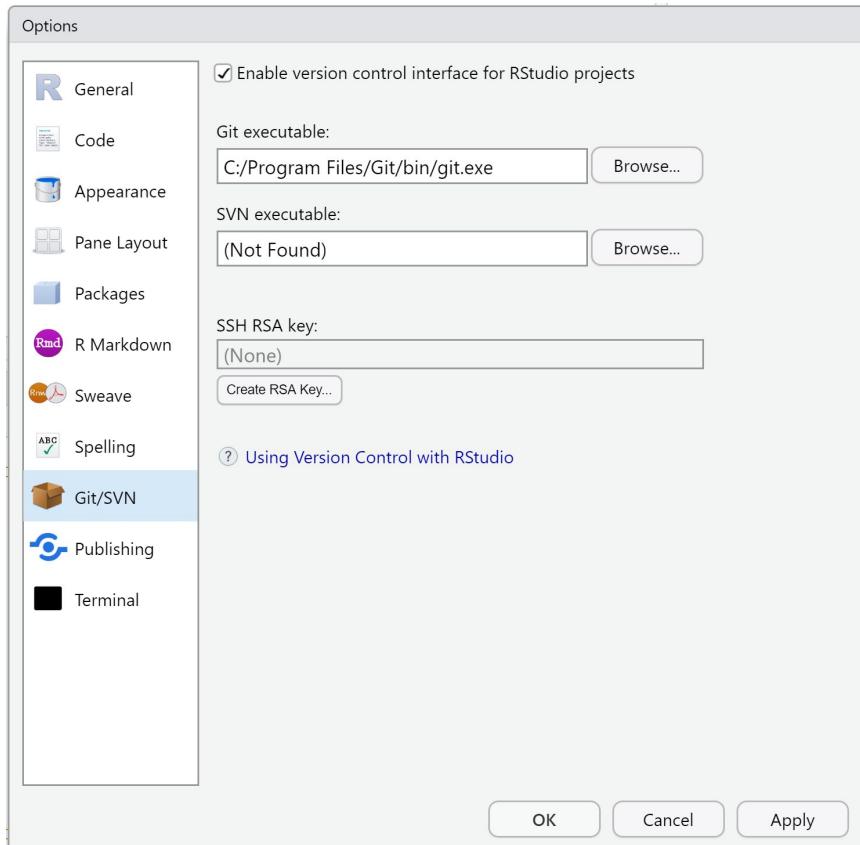
Control de versiones de código con Git, GitHub y RStudio

Segunda parte: la aplicación

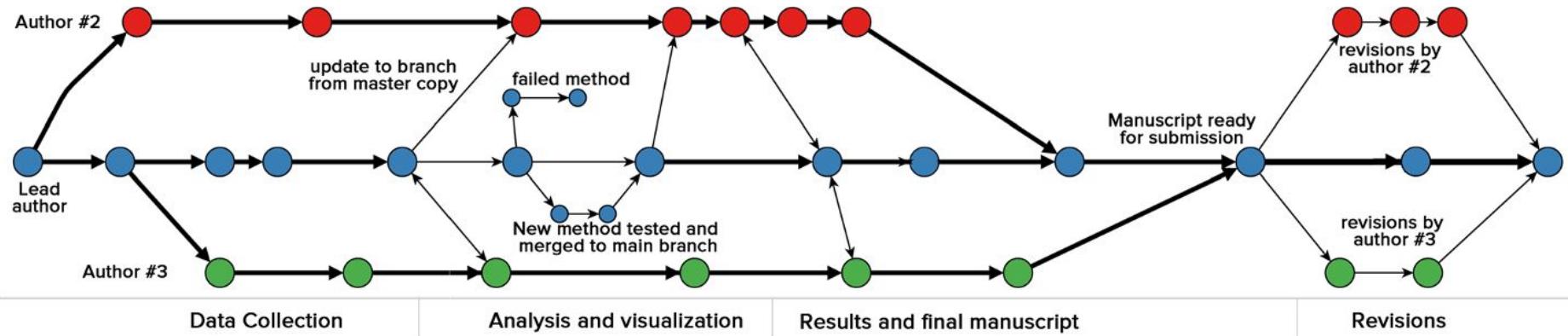
Ignacio Alcántara, Martín Soñora, Camila Simoes

26 de agosto de 2024

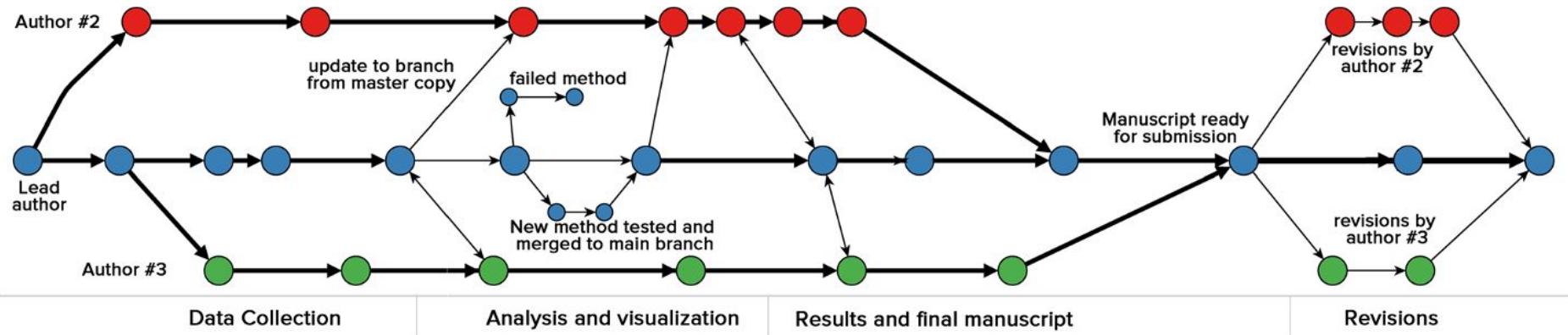
Aplicación 1: Escribiendo y revisando la tesis en Git + RStudio



Aplicación 1: Escribiendo y revisando la tesis en Git + RStudio



Aplicación 1: Escribiendo y revisando la tesis en Git + RStudio



Caso 1: tesis repo privado 😞

Caso 2: reproducibilidad artículo científico 🤔:
https://github.com/NAIcan/Reply_BC2021

Aplicación 2: Repositorio del lab de Simulaciones Biomoleculares

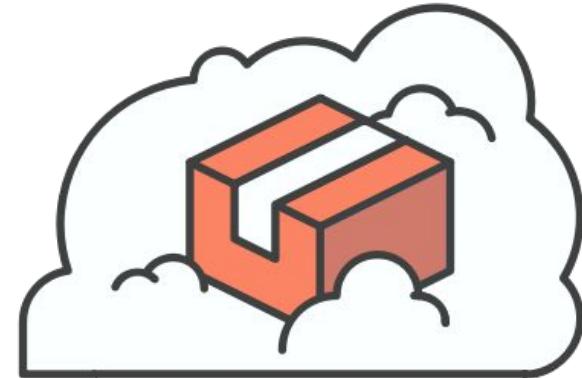


<https://github.com/SIRAHFF>

Aplicación 3: Gestionando archivos (grandes) con Git

Git LFS (Large File Storage)

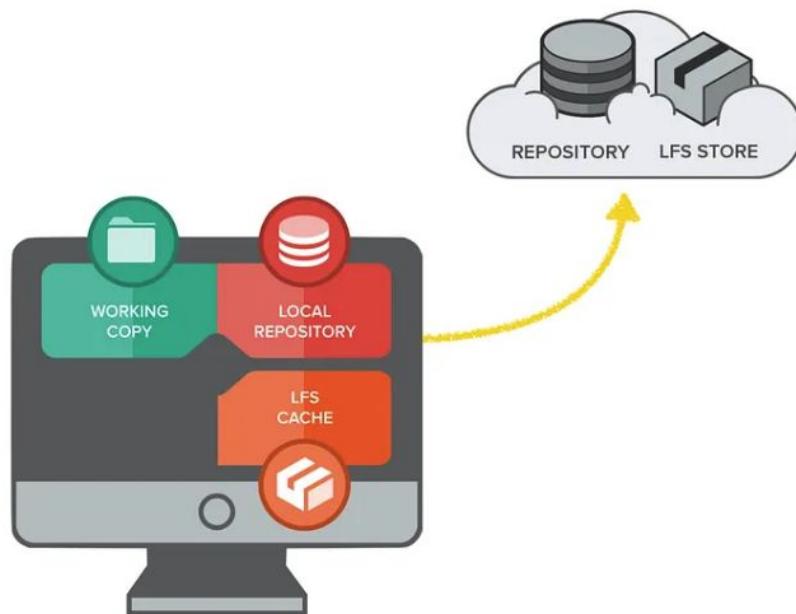
- Extensión de Git.
- permite gestionar grandes archivos en un repositorio sin afectar el rendimiento del mismo.
- Manejar archivos que cambian frecuentemente y son grandes en tamaño, como imágenes, videos, archivos de diseño, y datos.
- Evita que el tamaño del repositorio crezca innecesariamente debido a archivos binarios que no se comprimen bien con el sistema de almacenamiento de Git.



Aplicación 3: Gestionando archivos (grandes) con Git

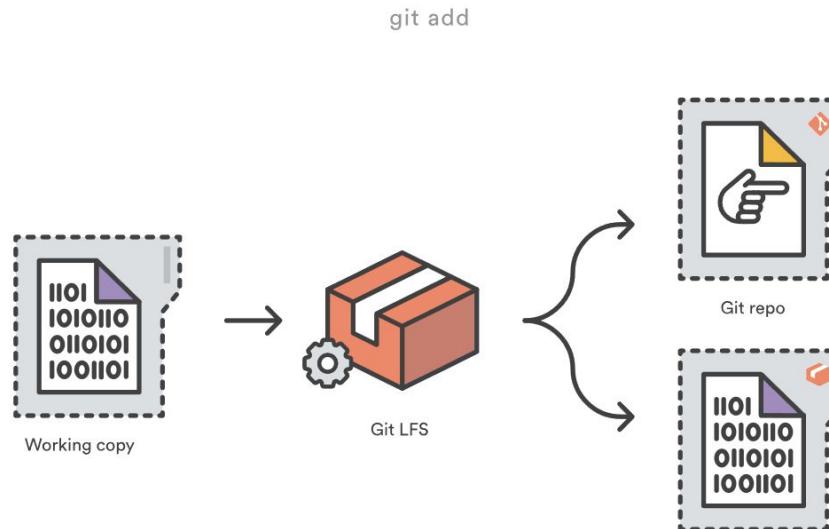
Git LFS (Large File Storage)

- **Reemplazo de archivos grandes con punteros:** indican dónde se almacena la versión completa del archivo en un servidor de Git LFS.
- **Almacenamiento externo en el servidor Git LFS:** solo una referencia pequeña se guarda en el historial de commits del repositorio.
- **Descarga bajo demanda:** reduce el uso de ancho de banda y espacio en disco.



Aplicación 3: Gestionando archivos (grandes) con Git

Git LFS (Large File Storage)

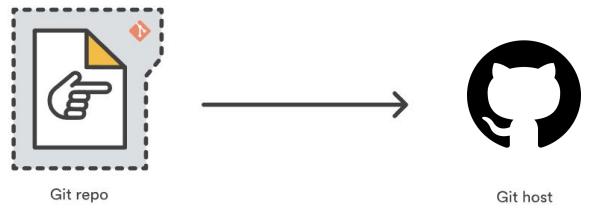


```
git lfs track "*.sql"  
git add .gitattributes  
git commit -m "Rastrear archivos sql con Git LFS"  
git push
```

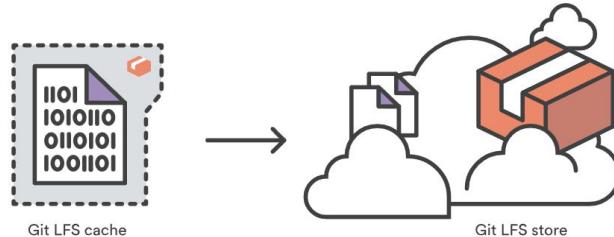
Aplicación 3: Gestiónando archivos (grandes) con Git

Git LFS (Large File Storage)

git push

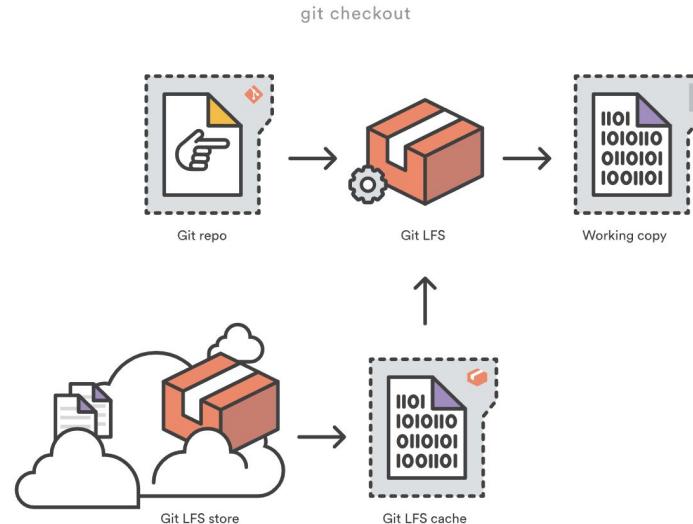


```
git lfs track "*.sql"
git add .gitattributes
git commit -m "Rastrear archivos sql con Git LFS"
git push
```



Aplicación 3: Gestiónando archivos (grandes) con Git

Git LFS (Large File Storage)



```
git lfs track "*.sql"
git add .gitattributes
git commit -m "Rastrear archivos sql con Git LFS"
git push
```

¡ Gracias!



Y recuerden...

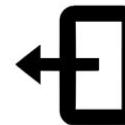
In case of fire



1. git commit



2. git push



3. leave building

Recursos

- Tutorial de GIT <https://swcarpentry.github.io/git-novice/reference.html#version-control>
- Usar Git + RStudio <https://happygitwithr.com/>
- Cheet Sheet de comandos <https://ndpsoftware.com/git-cheatsheet.html#loc=index>
- Introducción a sistemas de control de versiones, GIT y GitHub
<https://techcommunity.microsoft.com/t5/educator-developer-blog/introduction-to-git-github-and-version-control-workshop-o-matic/ba-p/3951511>