

# Guía de Trabajo: Control de versiones

---

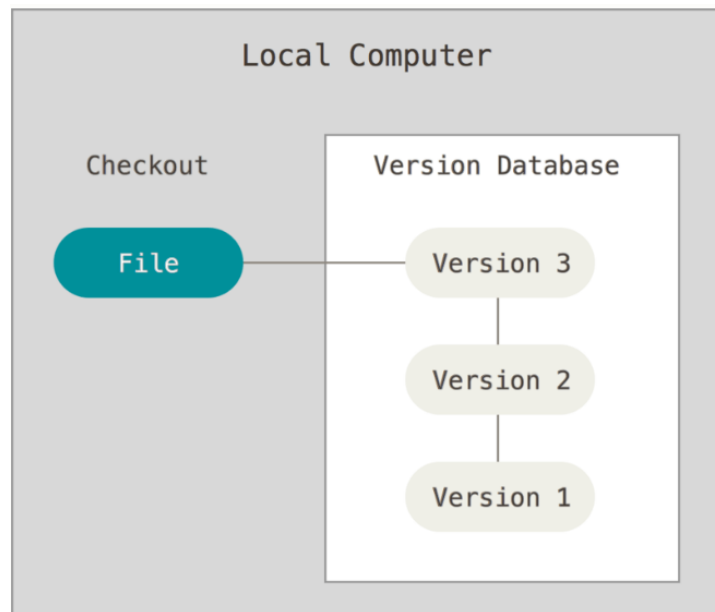
## introducción Control de versiones

Un sistema de control de versiones (VCS, por sus siglas en inglés) es una herramienta que permite a los desarrolladores gestionar los cambios realizados en el código fuente de un proyecto a lo largo del tiempo.

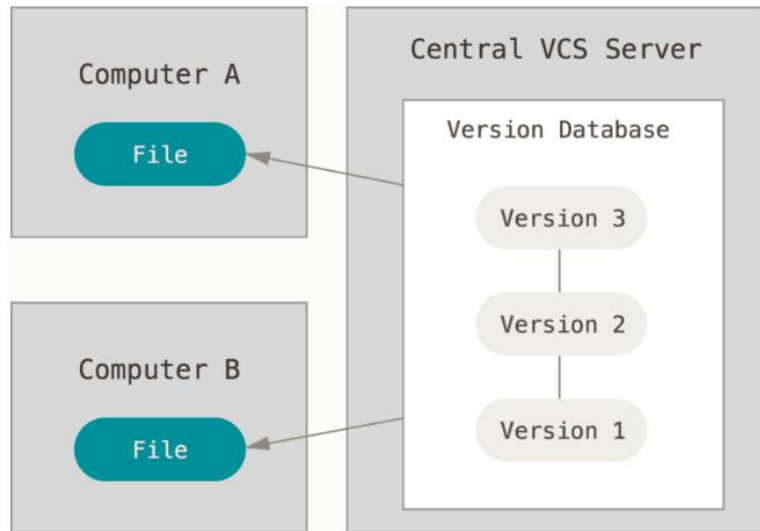
---

### 1. Sistemas de control de versiones

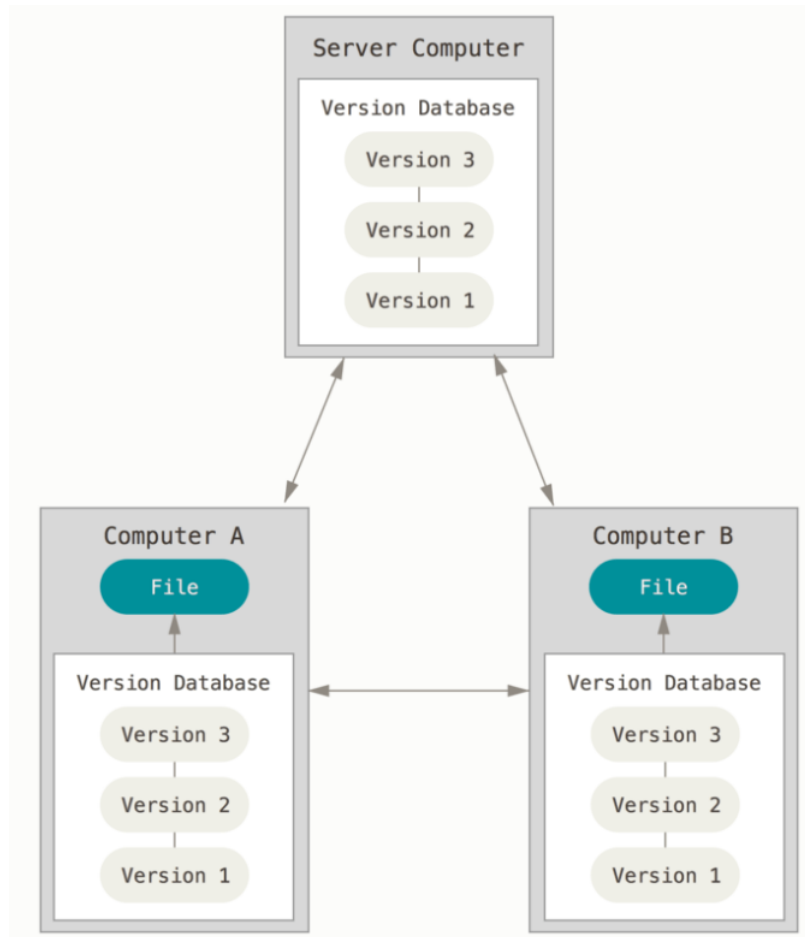
- **Locales:** Un método de control de versiones, usado por muchas personas, es copiar los archivos a otro directorio (quizás indicando la fecha y hora en que lo hicieron).



- **Centrales:** Permiten gestionar y rastrear los cambios realizados en archivos de un proyecto, utilizando un servidor central para almacenar el historial completo de las versiones. Estos sistemas son ideales para proyectos en los que varios usuarios colaboran y requieren sincronización constante.



- **Distribuidos:** Son herramientas diseñadas para gestionar el historial de cambios en proyectos, permitiendo que cada usuario tenga una copia completa del repositorio en su máquina local. Estos sistemas son altamente populares en el desarrollo moderno debido a su flexibilidad, velocidad y capacidad para trabajar sin conexión. De esta manera, si un servidor deja de funcionar y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios disponibles en los clientes puede ser copiado al servidor con el fin de restaurarlo.



## 2. Qué es Git

Es un sistema de control de versiones distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio, incluyendo todo el historial de cambios.



## 2.1. Ventajas:

- **Rendimiento:** Las operaciones locales (commit, log) son rápidas, ya que no requieren comunicación con un servidor.
- **Flexibilidad:** Soporta múltiples flujos de trabajo, desde el desarrollo en solitario hasta colaboraciones complejas.
- **Integración y popularidad:** Amplia adopción en la industria, gran ecosistema de herramientas (GitHub, GitLab, Bitbucket).

## 3. Instalación de Git

**Paso 1:** Escriba en el navegador la siguiente dirección <https://git-scm.com/>

**Paso 2:** Una vez ingresado por defecto aparece en el PC descargar para Windows, clic en esa opción, pero si utilizas otro sistema operativo dar clic en descargas para que los muestre.



### Paso 3: Descargar el archivo

## Download for Windows

 [Click here to download](#) the latest (2.47.1) 64-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **43 days ago**, on 2024-11-25.

### Other Git for Windows downloads

**Standalone Installer**

- [32-bit Git for Windows Setup.](#)
- [64-bit Git for Windows Setup.](#)

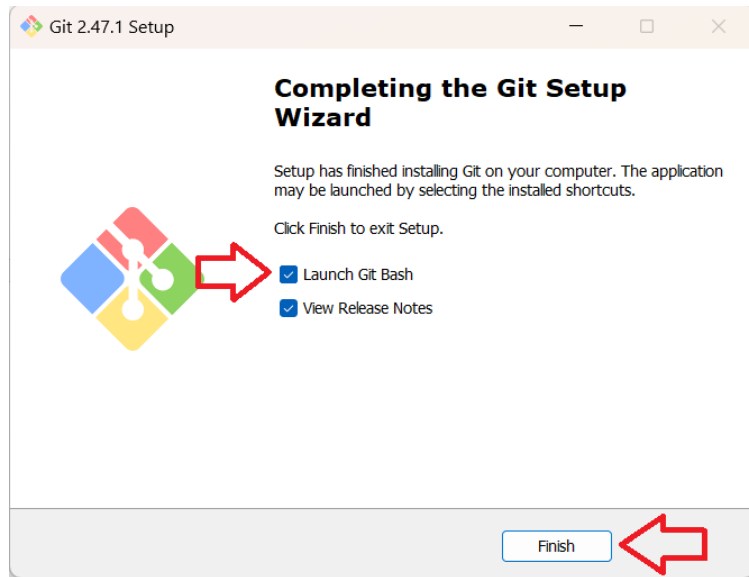
**Portable ("thumbdrive edition")**

- [32-bit Git for Windows Portable.](#)
- [64-bit Git for Windows Portable.](#)

### Paso 4: Se abre el archivo descargado y se da clic en instalar



## Paso 5: Finalice la instalación

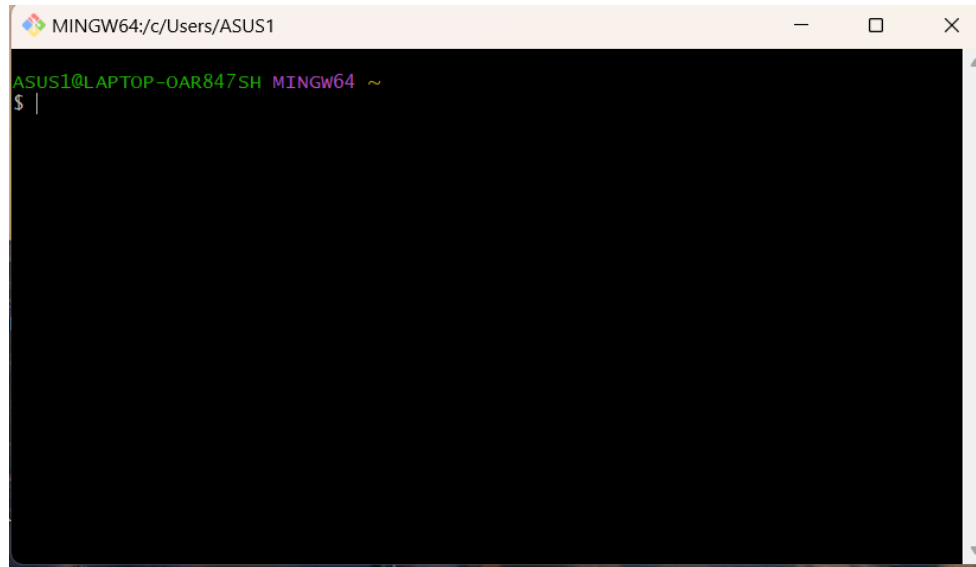


## Paso 6

En el escritorio debe aparecer el icono del Git



y al dar doble clic se muestra su ventana.

A screenshot of a MINGW64 terminal window. The title bar at the top reads "MINGW64:/c/Users/ASUS1" and includes standard window controls (minimize, maximize, close). The terminal area has a black background. The prompt "ASUS1@LAPTOP-OAR847SH MINGW64 ~" is displayed in green and purple text. Below the prompt, a white dollar sign "\$" and a vertical cursor bar are visible, indicating the terminal is ready for input.

```
MINGW64:/c/Users/ASUS1
ASUS1@LAPTOP-OAR847SH MINGW64 ~
$ |
```

#### 4. Comandos básicos de Git

- **Cd <directorio>:** Permite ingresar a un directorio.
- **Cd . . :** Permite retroceder un nivel
- **Mkdir <nombre de la carpeta o directorio>:** Crea un directorio
- **Ls:** Muestra todo el contenido de un directorio
- **Clear:** Borra la pantalla

#### 5. Comandos de configuración de usuario y correo

- **git config --global user.name "usuario"**

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~  
$ git config --global user.name "Devrier15"  
  
ASUS1@LAPTOP-OAR847SH MINGW64 ~  
$ git config user.name  
Devrier15
```

- **Git config --global user.email** “correo”

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~  
$ git config --global user.email "devrier1583@gmail.com"  
  
ASUS1@LAPTOP-OAR847SH MINGW64 ~  
$ git config user.email
```

## 6. Crear repositorios en git y uso de comandos

- **Git init:** Inicializa un repositorio de git

```
Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso  
$ git init  
Initialized empty Git repository in C:/Users/Billy/Documents/git-curso/.git/
```

- **git branch -m master main** **O** **git config --global init.defaultBranch main:** Cambia la rama de master a main para los nuevos repositorios creados, en esta caso se debe eliminar el actual y crear uno nuevo para que muestre la rama main.



```
Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso (master)
$ git branch -m master main

Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso (main)
```

- **Git config --global --get init.defaultBranch:** Verifica en que rama principal esta, si en main o master.

```
Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso (main)
$ git config --global --get init.defaultBranch
main
```

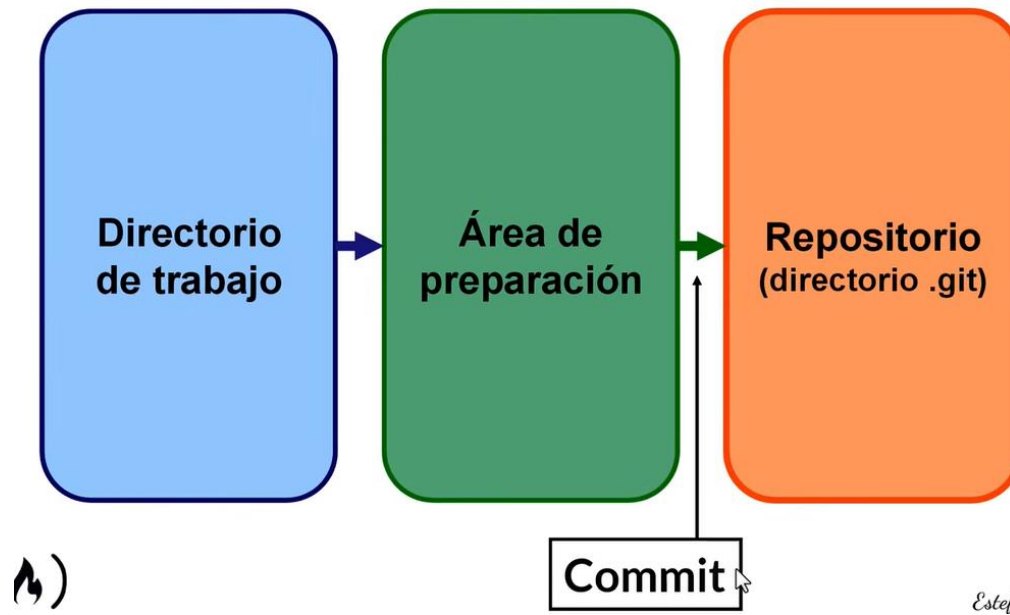
- **git status:** Nuestra el estado de un repositorio.

```
Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso (main)
$ git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

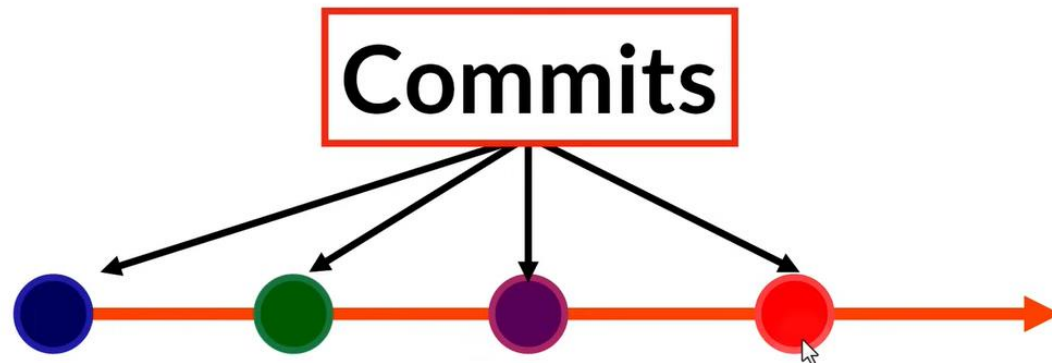
## 7. Tres áreas de Git



- **Directorio de trabajo:** Es la carpeta del proyecto que contiene los archivos y el directorio oculto .git del repositorio.
- **Área de preparación:** Conjunto de archivos y cambios que serán incluidos en el próximo commit.
- **Directorio. Git:** Es la parte más importante de Git, porque contiene todas las versiones y ramas del proyecto.

## 8. Commits

Es un registro o foto del estado de un proyecto en un momento determinado. Registra que líneas se agregaron en cada archivo, si se eliminó un archivo, carpeta etc.



### 8.1. Crear un commits desde línea de comando

- Se crea un archivo en el directorio, en este caso (mi-archivo.txt).
- Se ejecuta el comando **git status** para validar si existen cambios o archivos por rastrear o si ya están en el área de preparación.

```
Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  mi-archivo.txt

nothing added to commit but untracked files present (use "git add" to track)
```

- `git add <nombre del archivo>` o `git add .` : Agrega archivos al área de preparación.

```
Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso (main)
$ git add mi-archivo.txt

Billy@DESKTOP-GSLAQSN MINGW64 ~/Documents/git-curso (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   mi-archivo.txt
```

- `git rm --cached <file>`: Reversa los cambios, es decir, lo quita del área de preparación.
- `Git commit -m "Agregar archivo de texto"`: Crea un mensaje nuevo.

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git commit -m "Agregar archivo de texto"
[main (root-commit) 539387d] Agregar archivo de texto
1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt
```

8.2. **Git log:** Muestra el historial de commits que se han realizado.

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git log
commit 539387df3ea4c92f4c80dc1631fcfe7baeef43a4 (HEAD -> main)
Author: Devrier15 <devrier1583@gmail.com>
Date: Tue Jan 7 23:08:50 2025 -0500

    Agregar archivo de texto

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ |
```

8.3. **Commit con un editor de texto**

- Git commit

```
$ git commit
hint: Waiting for your editor to close the file...
```

- Editar un commit

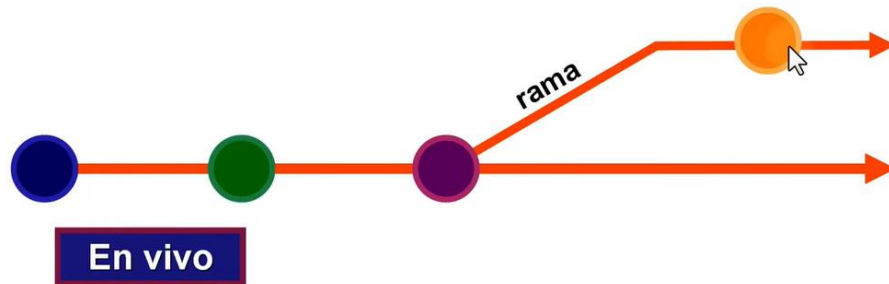
```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git commit --amend
hint: Waiting for your editor to close the file... |
```

- Eliminar un commit

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)  
$ git reset --soft HEAD~1
```

## 9. Ramas o Branch

Un **branch** o **rama** es una copia separada del código base dentro de un repositorio, que permite trabajar en cambios de forma aislada sin afectar la rama principal u otras ramas.



### 9.1. Crear ramas

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)  
$ git branch vesion-javascript
```

## 9.2. Visualizar ramas

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git branch
* main
  vesion-javascript
```

## 9.3. Cambiar de rama

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git checkout vesion-javascript
Switched to branch 'vesion-javascript'

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (vesion-javasc
ript)
$
```

## 9.4. Crear rama y cambiar (combinado)

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git checkout -b version-python
Switched to a new branch 'version-python'

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (version-python)
$
```

## 9.5. Cambiar nombre de una rama

- Primera forma

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (vesion-javascript)
$ git branch -m version-js

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (version-js)
$ |
```

- Segunda forma

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git branch -m version-python version-py

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git branch
* main
  version-js
  version-py
```

- Eliminar rama

Solo aplica para ramas locales



```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git branch -d version-py
Deleted branch version-py (was 95c49d3).

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git branch
* main
  version-js
```

#### 9.6. Observar todos los commit de diferentes ramas

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (version-js)
$ git log --oneline
a5ffd62 (HEAD -> version-js) nueva rama js
95c49d3 (main) Se agrego mas texto
539387d Agregar archivo de texto
```

#### 9.7. Fusionar ramas en Git (merge)

Proceso que permite combinar varias líneas independientes del desarrollo en una solo rama. Se debe realizar el proceso en la rama que va a recibir los cambios.

Comando: git merge <archivo>

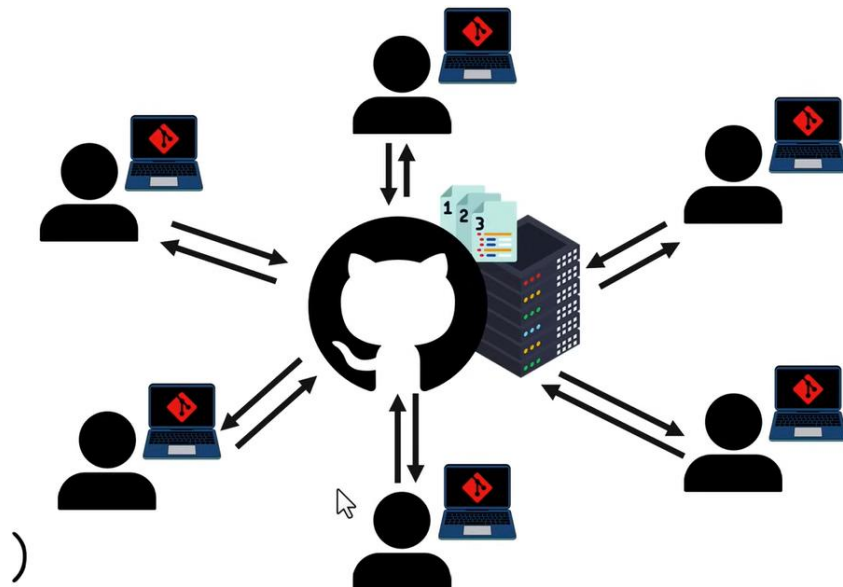
```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git merge texto-expandido
Updating c2f205d..35f6953
Fast-forward
 mi-archivo.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Visualización

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-git (main)
$ git log
commit 35f695307ed9fc807d2138bf144e8060d4888290 (HEAD -> main, texto-expandido)
Author: Devrier15 <devrier1583@gmail.com>
Date:   Fri Jan 10 21:14:29 2025 -0500
```

## 10. GitHub

Servicio de hosting que permite almacenar proyectos de desarrollo de software y control de versiones.



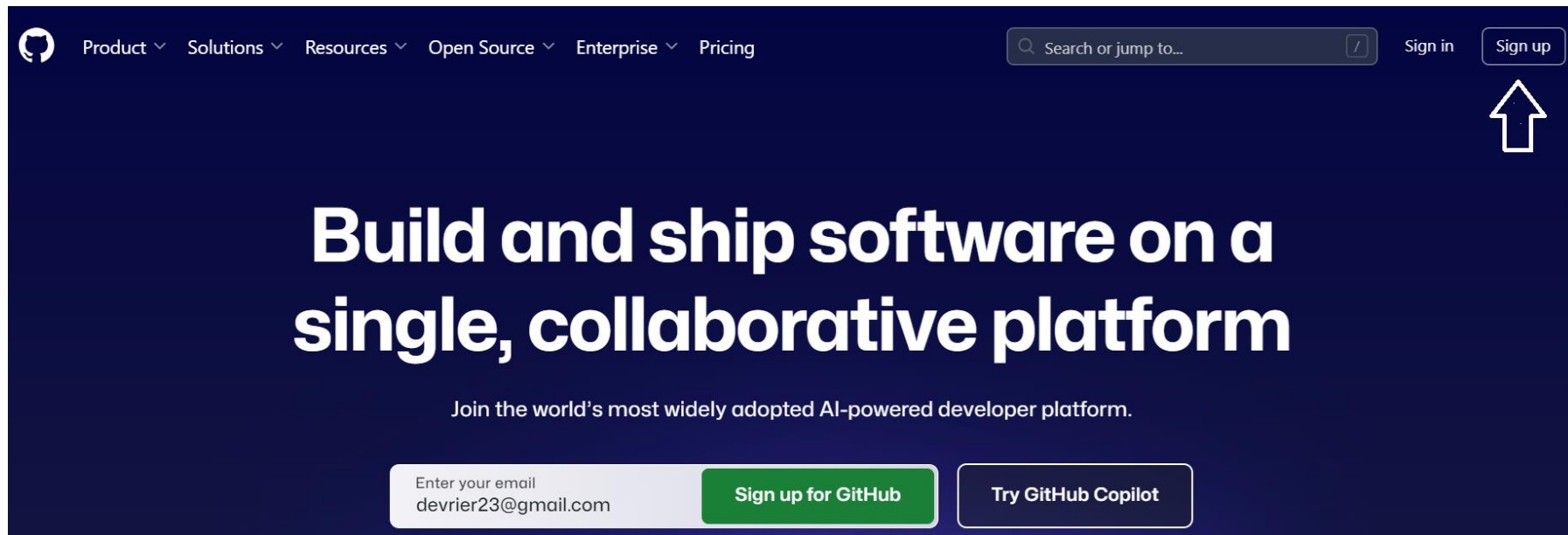
### 10.1. Ventajas de Github

Ventaja	Descripción
<b>Control de versiones con Git</b>	Permite realizar un seguimiento más en detalle de los cambios que se realizan, restauración de las versiones anteriores, además de mantener los proyectos mucho más organizados.
<b>Colaboración y trabajo en equipo</b>	Permite a los desarrolladores enviar «pull requests», revisar el código y debatir sobre los cambios mediante comentarios en el mismo código.
	Con el archivo README.md (para una descripción general), los desarrolladores

<b>Documentación centralizada</b>	pueden tener toda la documentación de sus proyectos en un solo lugar.
<b>Seguridad y control de accesos</b>	Permite tener control de los permisos en cada repositorio, a través de un acceso completo o limitado a los colaboradores según sus necesidades.

## 10.2. Paso 1: Crea una cuenta en GitHub

Primero, accede a [github.com](https://github.com), haz clic en «Sign Up» y rellena el formulario de registro con tu correo electrónico, nombre de usuario y contraseña y sigue todos los pasos.



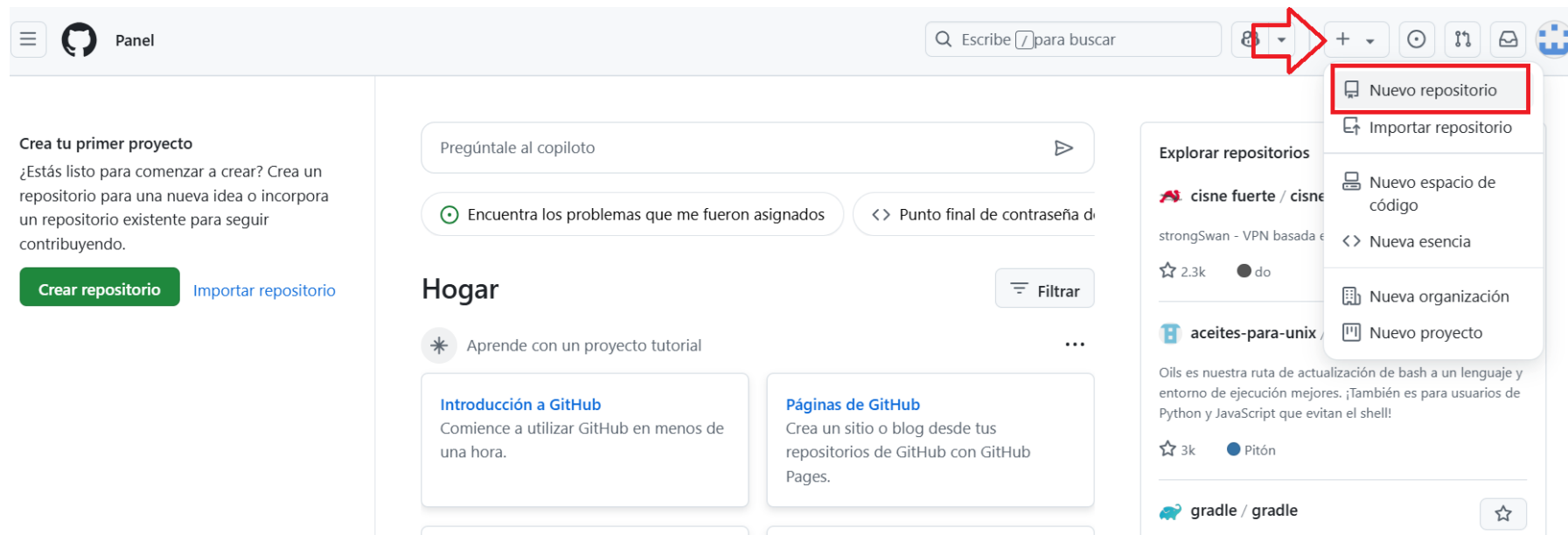
## 10.3. Paso 2: Configura Git en tu ordenador

Si tienes instalado Git digita los siguientes comandos para configurar el usuario y el correo electrónico.

```
1. git config --global user.name "TuNombreDeUsuario"
2.
3. git config --global user.email "TuEmail@example.com"
```

#### 10.4. Paso 3: Crea un nuevo repositorio en GitHub

Ahora, inicia sesión en GitHub y haz clic en el botón «+» ubicado en la esquina superior derecha y selecciona la opción “nuevo repositorio”.



Aquí tienes que asignar un nombre a tu repositorio y elige si deseas que sea público o privado. Puedes añadir una descripción opcional si lo consideras necesario.

## Crear un nuevo repositorio

Un repositorio contiene todos los archivos del proyecto, incluido el historial de revisiones. ¿Ya tienes un repositorio de proyectos en otro lugar? [Importa un repositorio](#).

Los campos obligatorios están marcados con un asterisco (\*).

Dueño \*

 Devrier1583 ▾

Nombre del repositorio \*

/ primer-repositorio

✓ primer-repositorio está disponible.

Los buenos nombres de repositorios son breves y fáciles de recordar. ¿Necesitas inspiración? ¿Qué tal? [gofre agradable](#) ?

Descripción (opcional)

Mi primer repositorio en GitHub



Público

Cualquier persona en Internet puede ver este repositorio. Tú eliges quién puede contribuir.



Privado

Tú eliges quién puede ver y comprometerse con este repositorio.

Inicialice este repositorio con:



Agregar un archivo README

Aquí puedes escribir una descripción detallada de tu proyecto. [Obtén más información sobre los archivos README](#).

Agregar .gitignore

Plantilla .gitignore : Ninguna ▾

Seleccione los archivos que no desea rastrear de una lista de plantillas. [Obtenga más información sobre cómo ignorar archivos](#).

Elija una licencia

Licencia : Ninguna ▾

Una licencia indica a los demás lo que pueden y no pueden hacer con su código. [Obtenga más información sobre las licencias](#).

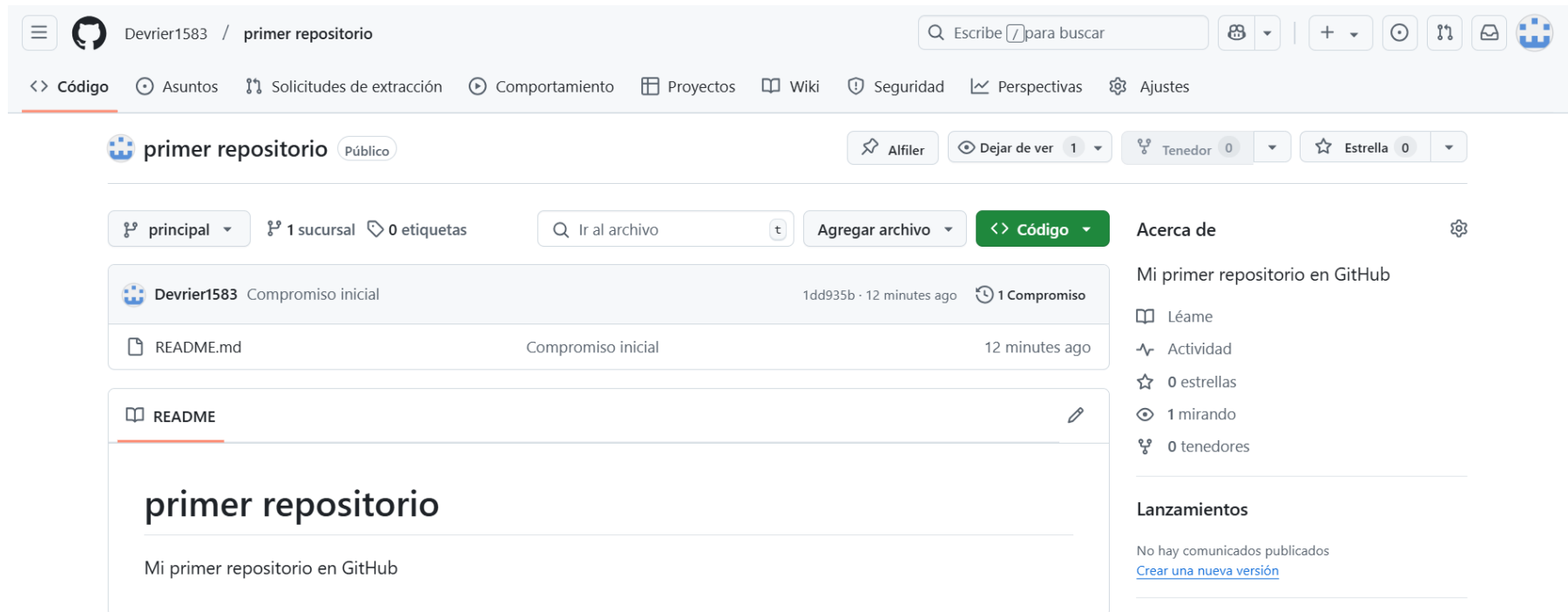
Esto establecerá `principal` como rama predeterminada. Cambie el nombre predeterminado en su [configuración](#).

① Estás creando un repositorio público en tu cuenta personal.

4

Crear repositorio

Cuando se crea el repositorio aparece la siguiente ventana



## 10.5. Agregar un archivo a un repositorio

En la parte superior derecha dar clic en agregar archivo y crear nuevo archivo.

primer repositorio Público

Alfiler Dejar de ver 1 Tenedor 0 Estrella 0

principal 1 sucursal 0 etiquetas Ir al archivo

Agregar archivo

- + Crear nuevo archivo
- Subir archivos

Devrier1583 Compromiso inicial

README.md Compromiso inicial 49 minutes ago

README

# primer repositorio

Mi primer repositorio en GitHub

Acerca de

Mi primer repositorio en GitHub

- Léame
- Actividad
- 0 estrellas
- 1 mirando
- 0 tenedores

Lanzamientos

No hay comunicados publicados

[Crear una nueva versión](#)

Paquetes

Siga todas las instrucciones incluyendo el nombre del archivo y el mensaje commit. Luego aparece en la pantalla principal el archivo creado.



The screenshot shows a GitHub repository page for 'primer repositorio' (Public). The repository is owned by 'Devrier1583'. The commit history shows two commits: 'Compromiso inicial' (1 hour ago) and 'Se creo prueba.txt' (2 minutes ago). The file 'prueba.txt' is highlighted with a red box. The README file is also visible. On the right side, there are sections for 'Acerca de' (About), 'Lanzamientos' (Releases), and 'Paquetes' (Packages).

primer repositorio Público

Alfiler Dejar de ver 1 Tenedor 0 Estrella 0

principal 1 sucursal 0 etiquetas Ir al archivo Agregar archivo Código

Devrier1583 Se creo prueba.txt 4a61685 · 2 minutos ago 2 confirmaciones

Archivo	Descripción	Fecha
README.md	Compromiso inicial	1 hour ago
prueba.txt	Se creo prueba.txt	2 minutos ago

README

## primer repositorio

Mi primer repositorio en GitHub

Acerca de

Mi primer repositorio en GitHub

- Leame
- Actividad
- 0 estrellas
- 1 mirando
- 0 tenedores

Lanzamientos

No hay comunicados publicados

[Crear una nueva versión](#)

Paquetes

## 10.6. Clonar un repositorio

Crea una copia local de tu repositorio remoto en tu maquina local incluyendo sus versiones e historial de commit.

Comando: **git clone "url"**

### Paso 1

En github diríjase a la opción "**código**" y copie la dirección o url HTTPS

The screenshot shows a GitHub repository named "primer repositorio" by user "Devrier1583". The repository is public and contains two files: "README.md" and "prueba.txt". The "Código" button in the top right is highlighted with a red arrow pointing down. The dropdown menu is open, showing options for cloning the repository. The "HTTPS" option is selected, and the URL "https://github.com/Devrier1583/primer-repositorio" is displayed. A red arrow points up to the URL field. Other options in the menu include "SSH", "Interfaz de línea de comandos de GitHub", "Abrir con GitHub Desktop", and "Descargar ZIP".

primer repositorio Público

Alfiler Dejar de ver 1 Tenedor 0 Estrella 0

principal 1 sucursal 0 etiquetas Ir al archivo Agregar archivo <> Código

Devrier1583 Se creo prueba.txt

README.md Compromiso inicial

prueba.txt Se creo prueba.txt

README

primer repositorio

Mi primer repositorio en GitHub

Acerca de

Mi primer repositorio en GitHub

Leame

Actividad

0 estrellas

1 mirando

0 tenedores

Lanzamientos

No hay comunicados publicados

[Crear una nueva versión](#)

## Paso 2

Abra git bash, digite el siguiente comando en una carpeta que desee.  
**git clone <url>**

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github
$ git clone https://github.com/Devrier1583/primer-repositorio.git
Cloning into 'primer-repositorio'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

Validar la clonación

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github
$ ls
primer-repositorio/

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github
$ cd primer-repositorio/

ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github/primer-repositorio (main)
$ ls
README.md  prueba.txt
```

## 10.7. Crear commit local y sincronizarlo con github

**git remote:** muestra el nombre que se le asignó al repositorio remoto.

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github/primer-repositorio (main)
$ git remote
origin
```

Se utiliza **git remote -v** para verificar si existe un repositorio remoto.

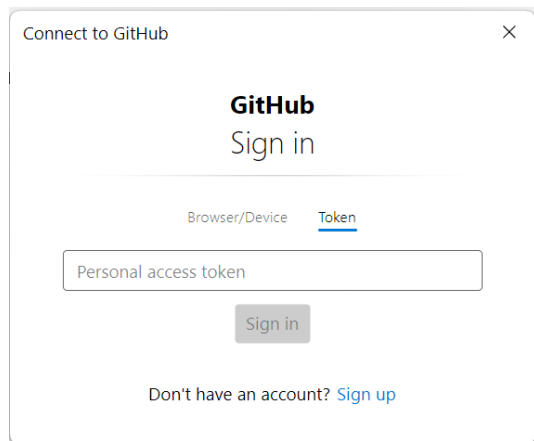
```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github/primer-repositorio (main)
$ git remote -v
origin https://github.com/Devrier1583/primer-repositorio.git (fetch)
origin https://github.com/Devrier1583/primer-repositorio.git (push)
```

**Fetch:** Busca los cambios realizados en el repositorio remoto.

**Puch:** Se envía al repositorio remoto los cambios realizados localmente para que ambos tengan la misma información.

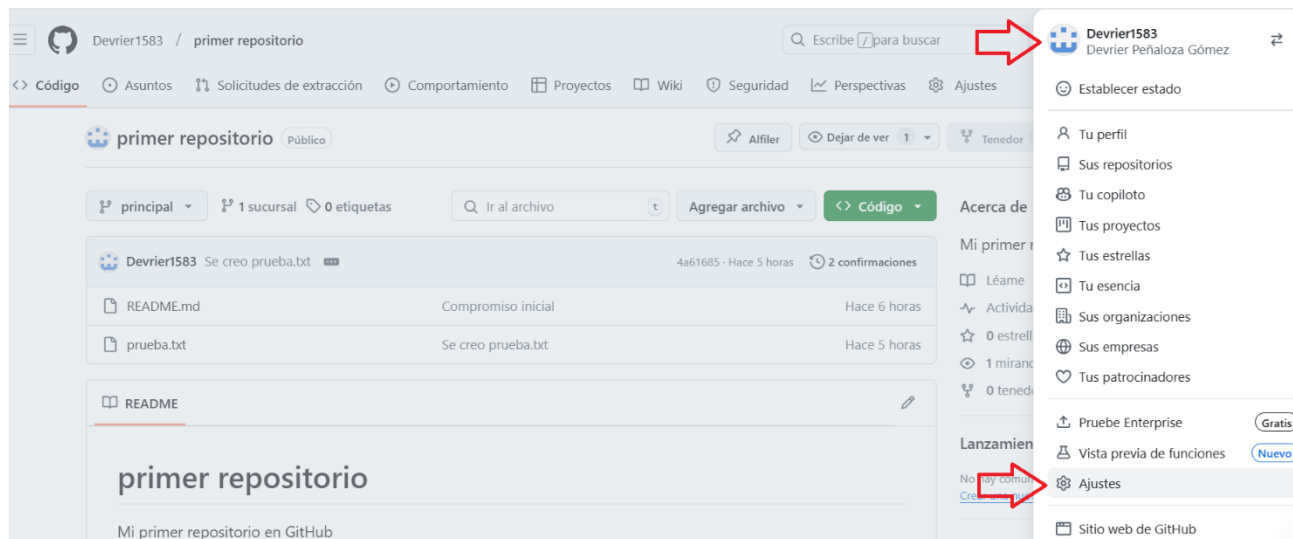
#### 9.7.1. Enviar cambios con push al repositorio remoto

**Git push origin main:** aparece el siguiente mensaje para conectarse a github a través de token o correo.



- **Obtener token en github**

**Paso 1:** Diríjase al repositorio de github, perfil de usuario y ajustes



## Paso 2: Luego de clic en configuración del desarrollador

Copiloto

Páginas

Respuestas guardadas

Seguridad

Seguridad del código

Integraciones

Aplicaciones

Recordatorios programados

Archivo

Registro de seguridad

Registro de patrocinio

Configuración del desarrollador

ORCID proporciona un identificador permanente (ORCID iD) que lo distingue de otros investigadores. Obtenga más información en [ORCID.org](https://orcid.org).

Conecte su ORCID iD

### Cuentas sociales

Enlace al perfil social

Enlace al perfil social

Enlace al perfil social

Enlace al perfil social

### Compañía

P&G

Puedes @mencionar la organización de GitHub de tu empresa para vincularla.

### Ubicación

## Paso 3: Seleccione el tipo de token

Aplicaciones de GitHub

Aplicaciones OAuth

Tokens de acceso personal

Fichas de grano fino

Fichas (clásicas)

Avance

Sin aplicaciones de GitHub

¿Quieres crear algo que se integre con GitHub y lo extienda? Registra una nueva aplicación de GitHub para comenzar a desarrollar en la API de GitHub.

Nueva aplicación de GitHub

[Ver documentación](#)

#### Paso 4: Generar token nuevo



#### No personal access token created

Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.

Generate new token ▾

Generate new token Beta  
Fine-grained, repo-scoped

Generate new token (classic) ←  
For general use

#### Paso 5: Poner el nombre del token y seleccionar permisos.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

#### Note

mi primer token puch

What's this token for?

#### Expiration \*

30 days ▾

The token will expire on Mon, Feb 10 2025

#### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

- ☒ **repo** Full control of private repositories
    - ☒ repo:status Access commit status
    - ☒ repo\_deployment Access deployment status
    - ☒ public\_repo Access public repositories
    - ☒ repo:invite Access repository invitations
    - ☒ security\_events Read and write security events
  - ☐ **workflow** Update GitHub Action workflows
  - ☐ **write:packages** Upload packages to GitHub Package Registry
- Generate token** **Cancel**

## Paso 6: Copiar token y guardarlo

### Personal access tokens (classic)

Generate new token ▾

Tokens you have generated that can be used to access the [GitHub API](#).

🔔 Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_oxddfINhTOYrWdWyspPyL1kavRB95M4DowL4



Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).





### Paso 7: Ingresar a github con token generado

Connect to GitHub ×

**GitHub**  
Sign in

Browser/Device Token





Don't have an account? [Sign up](#)

### Paso 8: Resultado exitoso

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github/primer-r
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 611 bytes | 611.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Devrier1583/primer-repositorio.git
   4a61685..223fed2  main -> main
```

### 9.7.2. Actualizar repositorio local con pull

- **Git pull:** Se usa para descargar el contenido de un repositorio remoto e inmediatamente actualizar el repositorio local para que ambos tengan la misma información.

Realice alguna modificación en algún archivo del repositorio remoto, en este caso manipularemos el archivo prueba agregando la línea 2.

- **Repositorio remoto**



 prueba.txt

 **1 file changed** +1 -1 lines changed

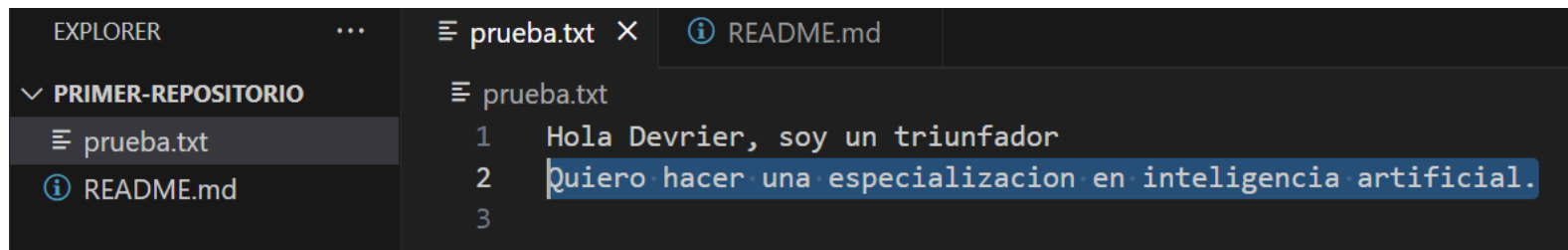
▼ prueba.txt 

...	@@ -1,2 +1,2 @@
1	1    Hola Devrier, soy un triunfador
2	-
2	+ Quiero hacer una especializacion en inteligencia artificial.

- **Comando git pull origin main**

```
ASUS1@LAPTOP-OAR847SH MINGW64 ~/Documents/curso-github/primer-repositorio (main)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1.04 KiB | 50.00 KiB/s, done.
From https://github.com/Devrier1583/primer-repositorio
 * branch                main          -> FETCH_HEAD
    f92c45e..7be6097      main          -> origin/main
Updating f92c45e..7be6097
Fast-forward
 prueba.txt | 2 +
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- Resultado en visual studio



The screenshot shows the Visual Studio Code interface. On the left, the Explorer view displays the file structure of the 'PRIMER-REPOSITORIO' project, with 'prueba.txt' and 'README.md' listed. The main Editor view shows the 'prueba.txt' file open, containing three lines of text. The second line, 'Quiero hacer una especializacion en inteligencia artificial.', is highlighted in blue.

```
EXPLORER  ...  prueba.txt x  README.md
v PRIMER-REPOSITORIO
  prueba.txt
  README.md

prueba.txt
1  Hola Devrier, soy un triunfador
2  Quiero hacer una especializacion en inteligencia artificial.
3
```

## Anexo

Tabla de comandos más utilizados de Git.

Categoría	Comando	Descripción
Configuración	git config --global user.name "TuNombre"	Configura el nombre del usuario para los commits.
	git config --global user.email "TuEmail"	Configura el correo electrónico del usuario para los commits.
	git config --list	Muestra la configuración actual de Git.
Inicialización	git init	Crea un nuevo repositorio Git en el directorio actual.
Clonar	git clone <URL>	Clona un repositorio remoto en tu máquina local.
Estado	git status	Muestra el estado actual del repositorio (archivos modificados, no rastreados, etc.).
Seguimiento	git add <archivo>	Añade un archivo específico al área de preparación (staging).
	git add .	Añade todos los archivos nuevos y modificados al área de preparación.
Commit	git commit -m "Mensaje"	Guarda los cambios en el historial con un mensaje descriptivo.
	git commit -am "Mensaje"	Añade y guarda los cambios rastreados (sin usar git add).
Ramas	git branch	Muestra las ramas disponibles en el repositorio.
	git branch <nombre>	Crea una nueva rama.
	git checkout <rama>	Cambia a una rama específica.
	git checkout -b <rama>	Crea una nueva rama y cambia a ella directamente.
	git branch -d <rama>	Elimina una rama.
Fusionar ramas	git merge <rama>	Fusiona la rama especificada con la actual.
Sincronización remota	git remote add origin <URL>	Conecta el repositorio local a un remoto.
	git pull origin <rama>	Actualiza el repositorio local con los cambios del remoto.
	git push origin <rama>	Sube los cambios de la rama actual al repositorio remoto.
Historial	git log	Muestra el historial de commits.
	git log --oneline	Muestra el historial de commits en formato resumido.
Deshacer cambios	git reset <archivo>	Deshace los cambios en el área de preparación.

## Ejercicio practico

### Requerimientos del Ejercicio

#### 1. Preparación del Repositorio:

- Crea un repositorio público o privado en GitHub con el nombre proyecto-colaborativo.
  - Agrega un archivo README.md con una breve descripción del proyecto.
2. Sube todos los ejercicios realizados hasta el momento.
  3. Clona el repositorio en tu maquina local
  4. Realizar commit, push y pull.
  5. Enviar una invitación a sus docentes.

### Material de apoyo

<https://www.youtube.com/watch?v=mBYSUUnMt9M>

---