

**Nombre:** Yessica Paola Cárdenas Briñez

## **INVESTIGACION**

### **Guía de Versionamiento**

1. ¿Qué es un repositorio y mención 2 ejemplos de los mismos (proveedores online)?

Un repositorio es un espacio de almacenamiento donde se guardan archivos y datos relacionados con el desarrollo de software, como el código fuente, documentación, configuraciones, entre otros. Estos repositorios permiten a los desarrolladores colaborar y gestionar versiones del código de manera organizada y controlada, lo que facilita el trabajo en equipo y el seguimiento de cambios en el proyecto.

Dos ejemplos de proveedores online de repositorios son:

- **GitHub:** Es uno de los servicios más populares para alojar repositorios Git. Ofrece control de versiones, colaboración en equipo, y funcionalidades como revisión de código, integraciones con CI/CD, y GitHub Actions para automatización de tareas.
- **GitLab:** Similar a GitHub, GitLab también permite alojar repositorios Git, pero con la ventaja de ofrecer opciones de integración continua (CI/CD) y otras funcionalidades avanzadas integradas en la misma plataforma. Se puede utilizar en la nube o como una solución auto-hospedada.

2. ¿Cuáles son los principales componentes de un versionamiento en la herramienta GIT?

Los principales componentes de un sistema de versionamiento en Git son los siguientes:

1. **Repositorio:** Es el almacenamiento que contiene todos los archivos del proyecto y su historial de cambios. Existen dos tipos de repositorios en Git:

- **Repositorio local:** Es el repositorio almacenado en la máquina del usuario donde trabaja el código y realiza las operaciones.
- **Repositorio remoto:** Es una copia del repositorio almacenada en un servidor externo (como GitHub o GitLab), lo que permite la colaboración entre varios usuarios.

2. **Commit:** Es un registro de cambios realizado en el repositorio. Un commit guarda una "instantánea" del estado de los archivos en un momento determinado y permite revertir o revisar esos cambios. Cada commit tiene un mensaje descriptivo que ayuda a entender el propósito de los cambios.

3. **Branch (Rama):** Es una bifurcación del historial de commits. Permite trabajar en diferentes características o correcciones sin afectar la rama principal (normalmente llamada “main” o “master”). Cada rama tiene su propio flujo de commits hasta que se fusiona o elimina.

4. **Merge (Fusión):** Es el proceso de integrar cambios de una rama a otra. Generalmente, se utiliza para unir los cambios de una rama de trabajo a la rama principal, consolidando los desarrollos y nuevas características.

5. **Staging Area (Área de preparación):** Es un espacio intermedio donde se colocan los archivos modificados antes de hacer un commit. Esto permite a los usuarios controlar exactamente qué cambios se incluirán en el próximo commit.

6. **Tag (Etiqueta):** Es una referencia a un commit específico y generalmente se utiliza para marcar versiones importantes del proyecto, como lanzamientos o hitos.

#### 7. Pull y Push:

- Pull: Es la acción de descargar cambios desde un repositorio remoto hacia el repositorio local.

- Push: Es la acción de enviar los cambios locales al repositorio remoto, permitiendo que otros usuarios los accedan y trabajen con ellos.

3. ¿Mencione con sus propias palabras las ventajas que tiene GIT frente a otros proveedores de repositorios?

GIT destaca frente a otros proveedores de repositorios por su flexibilidad, eficiencia en la gestión de ramas y gestiones, su naturaleza distribuida y su integración con una amplia gama de herramientas en el ecosistema de desarrollo moderno. Esto lo ha convertido en el estándar de facto para la mayoría de los equipos de desarrollo

4. Mencione por lo menos 5 ejemplos de los comandos básicos que se usan en GIT.

GIT INIT: Inicializa un nuevo repositorio Git en el directorio actual. Crea un repositorio local vacío que puede comenzar a rastrear cambios

- GIT CLONE <URL>: Clona un repositorio remoto a tu máquina local. Copia todo el historial del repositorio en la URL especificada a un nuevo directorio local.
- GIT ADD: Añade los cambios de un archivo al área de preparación (staging area), preparándolo para el siguiente commit.
- GIT COMMIT -M "MENSAJE": Crea un commit con los cambios que están en el área de preparación. El commit incluye un mensaje que describe los cambios realizados.

- GIT PUSH: Envía los commits locales al repositorio remoto. Es decir, "sube" los cambios realizados en tu máquina local para que estén disponibles en el servidor remoto.
- GIT COMMIT (crear COMMIT)
- GIT BRANCH (crear RAMAS)
- GIT MERGE (combinar RAMAS)

5. ¿Que son y cuáles son las funciones de los branch?

En GIT, es una copia independiente del repositorio que permite trabajar en paralelo con la versión principal (main o master). Las ramas son útiles para:

- Desarrollo de las nuevas características sin afectar la versión principal
- Corrección de errores sin interferir con el desarrollo principal
- Creación de versiones beta o de prueba
- Colaboración en equipo sin afectar el código principal
- Experimentación con cambios radicales sin riesgo

6. ¿Cuál es el Nombre del branch principal?

Master: nombre tradicional y ampliamente utilizado

Main: nombre recomendado por Github y utilizado en algunos proyectos

### **2.3. GESTOR GIT**

Pantallazos de la investigación de 1 de gestor de GIT de forma local y realización del paso a paso de para su instalación y funcionamiento:

### **3. INSTALACION DE GIT**

About

Documentation

Downloads

GUI Clients  
Logos

Community

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

## Download for Windows

[Click here to download](#) the latest (**2.47.0**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **13 days ago**, on 2024-10-08.

### 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.](#)

#### Using winget tool

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.47.0**. If you want the newer version, you can build it from [the source code](#).

### Now What?

Now that you have downloaded Git, it's time to start using it.



#### Read the Book

Dive into the Pro Git book and learn at your own pace.




#### Download a GUI

Several free and commercial GUI tools are available for the Windows platform.



#### Get Involved

A knowledgeable Git community is available to answer your questions.

 MINGW64:/c/Users/LAV3RD3

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~  
$ |
```



```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~  
$ git --version  
git version 2.47.0.windows.1  
  
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~  
$ |
```

## 4. PRIMEROS PASOS

### 4.1 Creación del repositorio local

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~  
$ cd Desktop/git_adsi/  
  
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (main)  
$ git init  
Reinitialized existing Git repository in C:/Users/LAV3RD3/Desktop/git_adsi/.git/
```

### 4.2 Adición de archivos para controlar cambios

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (main)  
$ cat adsi.txt  
Código Original
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (main)  
$ git add .
```

### 4.3. Aplicación de cambios y revisión del estado del proyecto

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (master)  
$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   adsi.txt
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (master)  
$ git commit -m "version original"  
[master (root-commit) eb67392] version original  
1 file changed, 1 insertion(+)  
create mode 100644 adsi.txt
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (master)  
$ git status  
On branch master  
nothing to commit, working tree clean
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adi (master)
$ git log
commit eb673921df23286161720c82c21fb4e4453f6074 (HEAD -> master)
Author: jpaola2 <jpaolacardenasb@gmail.com>
Date: Sun Oct 20 19:37:56 2024 -0500

    version original
```

#### 4.4. Aplicación y seguimiento a nuevos cambios

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adi (master)
$ cat adi.txt
Código Oriniga
Primer cambio
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adi (master)
$ git status
On branch master
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:   adi.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adi (master)
$ git add .
warning: in the working copy of 'adi.txt', LF will be replaced by CRLF the next time Git touches it

LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adi (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   adi.txt
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adi (master)
$ git commit -m "primer cambio"
[master e9f482a] primer cambio
1 file changed, 1 insertion(+)
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adi (master)
$ git log
commit e9f482afb52ae5abcbe2bfdbc06cc4fab7f1b382 (HEAD -> master)
Author: jpaola2 <jpaolacardenasb@gmail.com>
Date: Sun Oct 20 19:45:05 2024 -0500

    primer cambio

commit eb673921df23286161720c82c21fb4e4453f6074
Author: jpaola2 <jpaolacardenasb@gmail.com>
Date: Sun Oct 20 19:37:56 2024 -0500

    version original
```

#### 4.5. Navegación a través del historial de cambios

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (master)
$ git checkout eb67392
Note: switching to 'eb67392'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at eb67392 version original
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi ((eb67392...))
$ cat ads.txt
Código Oriniga
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi ((eb67392...))
$ git checkout master
Previous HEAD position was eb67392 version original
Switched to branch 'master'
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (master)
$ cat ads.txt
Código Oriniga
Primer cambio
```

#### 4.6. Uso de Git con repositorios remotos

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (master)
$ git clone https://github.com/swcraftersclm/curso-de-git.git
Cloning into 'curso-de-git'...
remote: Enumerating objects: 354, done.
remote: Total 354 (delta 0), reused 0 (delta 0), pack-reused 354 (from 1)
Receiving objects: 100% (354/354), 54.31 KiB | 646.00 KiB/s, done.
Resolving deltas: 100% (134/134), done.
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi (master)
$ git remote -v
```

```
LAV3RD3@DESKTOP-LNVLR5P MINGW64 ~/Desktop/git_adsi/curso-de-git (dev)
$ git remote --v
origin https://github.com/swcraftersclm/curso-de-git.git (fetch)
origin https://github.com/swcraftersclm/curso-de-git.git (push)
```