



Repositorio Remoto

Contenido

1. Introducción.	2
2. Iniciando el repositorio remoto.	3
3. Trabajando con repositorio remoto.	5



1. Introducción.

En el mundo del desarrollo de software, la colaboración eficiente y la gestión de versiones son esenciales para construir proyectos exitosos. En este documento, exploraremos la función vital que desempeñan los repositorios remotos en la facilitación de la colaboración entre equipos y cómo Git, se convierte en la herramienta fundamental para lograrlo.

Un repositorio remoto en Git es una versión de tu proyecto que se encuentra alojada en un servidor o en otro lugar fuera de tu máquina local. Este repositorio remoto puede ser accesible y compartido entre diferentes colaboradores de un proyecto. Los repositorios remotos permiten la colaboración y la sincronización de cambios entre diferentes desarrolladores.

Algunas características clave de un repositorio remoto son:

- **Colaboración:**

Varios desarrolladores pueden trabajar en un proyecto compartido al acceder y clonar el mismo repositorio remoto.

- **Sincronización de cambios:**

Los desarrolladores pueden enviar (push) sus cambios al repositorio remoto para que otros puedan verlos y descargarlos (pull). Esto facilita la colaboración y la integración de cambios.

- **Copia de seguridad:**

El repositorio remoto sirve como una copia de seguridad centralizada del proyecto. Si tu máquina local falla, puedes recuperar el código desde el repositorio remoto.

- **Facilita la distribución:**

Permite distribuir el código a través de equipos geográficamente dispersos. Los desarrolladores pueden trabajar desde diferentes ubicaciones y contribuir al mismo proyecto.

- **Seguimiento de versiones:**

Los repositorios remotos mantienen un historial de versiones del proyecto, lo que facilita la colaboración y la gestión de cambios a lo largo del tiempo.

Los repositorios remotos suelen estar alojados en plataformas como GitHub, GitLab, Bitbucket u otros servicios de alojamiento de Git. Estas plataformas ofrecen funciones adicionales, como seguimiento de problemas, solicitudes de extracción y otras herramientas de colaboración.



2. Iniciando el repositorio remoto.

En esta sección, exploraremos los pasos necesarios para crear una cuenta en **GitLab** y realizar la configuración inicial para comenzar a trabajar con repositorios remotos.

En primer lugar, debemos crear una cuenta en el [sitio web de GitLab](#). Una vez hecho esto, pasamos a configurar en nuestro equipo el nombre de usuario y la dirección de correo electrónico con la que nos vamos a identificar en el repositorio. Para ello abrimos el terminal de git utilizamos los siguientes comandos:

i El comando **git config** puede ser utilizado principalmente de dos formas:

1. `'git config --global user.name "Tu Nombre"'`
2. `'git config --global user.email "tu@email.com"'`

El parámetro `--global` sirve para fijar esa información de forma global para todos los proyectos. En el caso de que queramos modificar esta configuración para algún proyecto en particular, simplemente tenemos que ejecutar los comandos sin ese parámetro el parámetro `--global`.

El siguiente paso será crear un proyecto dentro de nuestra sesión de GitLab y obtener su URL para clonarlo mediante https:

Your work / Projects

Welcome to GitLab

Faster releases. Better code. Less pain.



Create a project

Projects are where you store your code, access issues, wiki and other features of GitLab.



Create a group

Groups are the best way to manage projects and members.

Create new project



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.



Create from template

Create a project pre-populated with the necessary files to get you started quickly.



Create blank project
Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name
Prueba_Git

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL
https://gitlab.com/blank-project-2023-1/

Project slug
prueba_git

Want to organize several dependent projects under the same namespace? [Create a group](#).

Project deployment target (optional)
Select the deployment target

Visibility Level
☒ Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.
☐ Public
The project can be accessed without any authentication.

Project Configuration
☒ Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
☐ Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more](#).

[Create project](#) [Cancel](#)

P Prueba_Git

main ▾ prueba_git / + ▾

Initial commit
Initial commit authored just now

Name	Last commit
README.md	Initial commit

README.md

Prueba_Git

Getting started

To make it easy for you to get started with GitLab, here's a list of recommended next steps:

Already a pro? Just edit this README.md and make it your own. Want to make it even better?

Clone with SSH
git@gitlab.com:prueba_git/prueba_git/p

Clone with HTTPS
https://gitlab.com/prueba_git/prueba_git/p

Open in your IDE
 Visual Studio Code (SSH)
 Visual Studio Code (HTTPS)
 IntelliJ IDEA (SSH)
 IntelliJ IDEA (HTTPS)

Download Source Code
 zip
 tar.gz
 tar.bz2
 tar

Project information
 1 Commit
 1 Branch
 0 Tags
 3 KiB Project Storage

README
 + Add LICENSE
 + Add CHANGELOG
 + Add CONTRIBUTING
 + Enable Auto DevOps
 + Add Kubernetes cluster
 + Set up CI/CD
 + Add Wiki
 + Configure Integrations

i Disponemos de dos métodos de clonado y la principal diferencia es que HTTPS utiliza autenticación mediante nombre de usuario y contraseña y es fácil de configurar, mientras que SSH utiliza autenticación basada en claves, que ofrece mayor seguridad, pero requiere configuración adicional. Ambos son seguros para interactuar con repositorios de GitLab y la elección depende de las preferencias, requisitos de seguridad y entorno corporativo.

Una vez hecho esto, tenemos que clonar el repositorio remoto en nuestro equipo usando el comando **'git clone'**.



```
MINGW64/c/Users/robertolopez/
robertolopez@L2307001 MINGW64 ~
$ git clone https://gitlab.com/robertolopez/prueba_git.git
Cloning into 'prueba_git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```



A la hora de clonar repositorios tenemos que prestar atención al usuario que tenemos configurado, ya que si el repositorio no es público no podríamos clonarlo.

3. Trabajando con repositorio remoto.

- Subiendo cambios al repositorio remoto:

Para comenzar a trabajar con el repositorio remoto empezaremos añadiendo al directorio local nuestro archivo 'home.html' y hacemos nuestro primer commit:

```
MINGW64/c/Users/robertolopez/prueba_git
robertolopez@L2307001 MINGW64 ~/prueba_git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  home.html

nothing added to commit but untracked files present (use "git add" to track)
robertolopez@L2307001 MINGW64 ~/prueba_git (main)
$ git add .
robertolopez@L2307001 MINGW64 ~/prueba_git (main)
$ git commit -m 'Inicio del repositorio remoto'
[main fc53f0c] Inicio del repositorio remoto
1 file changed, 17 insertions(+)
create mode 100644 home.html
robertolopez@L2307001 MINGW64 ~/prueba_git (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```



Vemos que git nos informa de que nuestro repositorio local está a un commit del repositorio remoto. Para actualizar el repositorio remoto debemos usar el comando **'git push'**:

```
MINGW64/c/Users/fernando/prueba_git
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 552 bytes | 552.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://gitlab.com/fernando/prueba_git.git
355b55e..fc53f0c main -> main
```

Este comando nos sirve para subir los cambios que tenemos en nuestro repositorio local hacia el repositorio remoto.

Una de las ventajas de GitLab es que tenemos la opción de ver toda la información sobre los commits realizados de una forma muy intuitiva:

Commit fc53f0c7 authored 23 hours ago by fernando

[Browse files](#) [Options](#)

Inicio del repositorio remoto

parent 355b55e9

Branches: main

No related merge requests found

Changes

Showing 1 changed file with 17 additions and 0 deletions

[Hide whitespace changes](#) [Inline](#) [Side-by-side](#)

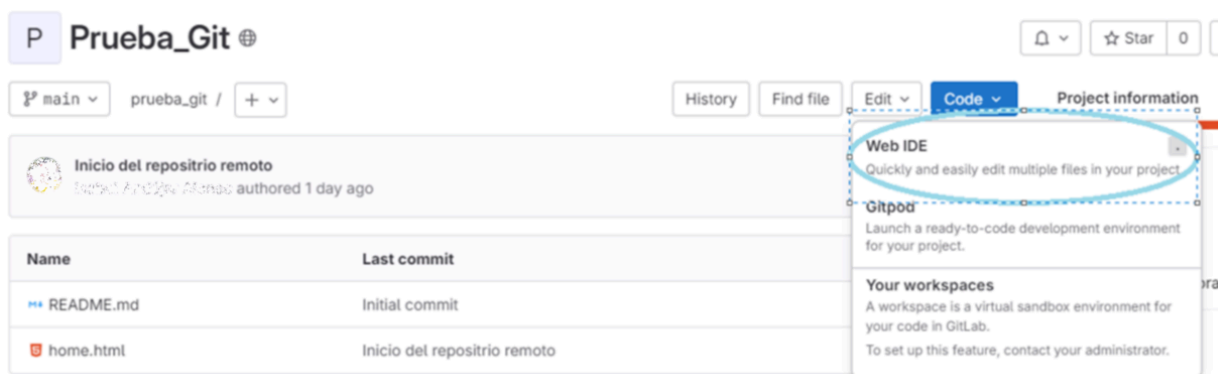
home.html 0 → 100644

```
1 + <!DOCTYPE html>
2 + <html lang="en">
3 + <head>
4 +   <meta charset="UTF-8">
5 +   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 +   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 +   <title>Nter</title>
8 + </head>
9 + <body>
10 + <div>
11 +   <h1>Bienvenidos a Nter 2.0</h1>
12 +   <div>
13 +     <h1>Bienvenidos a Nter</h1>
14 +     <div>parent of 50f7d35 (Estamos trabajando en el proyecto)</div>
15 +   </div>
16 + </div>
17 +
```



• Actualizando el repositorio local:

Por último, nos quedaría ver como actualizar nuestro repositorio local en relación con el repositorio remoto. Para comenzar necesitamos hacer cambios en el repositorio remoto y para ello utilizaremos 'Web IDE', una utilidad de GitLab que nos permite editar los archivos de nuestro proyecto de una forma rápida y sencilla.



Para actualizar los cambios en nuestro repositorio local usamos el comando '**git pull**':

```
MINGW64/c/Users/indianita/prueba_git
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 320 bytes | 45.00 KiB/s, done.
From https://gitlab.com/indianita/prueba_git
   fc53f0c..3bd235d main    -> origin/main
Updating fc53f0c..3bd235d
Fast-forward
 home.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Este comando actualiza la rama del repositorio local en la que se está trabajando y además nos da un resumen de las modificaciones que se han realizado.

¡Listo! Ya estás preparado para trabajar con repositorios remotos.