

GIT & GIT HUB

*Ciencia de Datos en
Python*

*Orly Elizabeth Trachtenberg Pérez
Carné: 0900102*

¿Qué es GIT?

Este es un software de control de versiones para desarrolladores: es el proceso de guardar diferentes archivos o versiones de los mismos a lo largo de las diferentes etapas de un. Esto permite a los desarrolladores hacer un seguimiento de lo que se ha hecho y volver a una fase anterior si deciden que quieren revertir algunos de los cambios que han hecho.

Beneficios:

- Facilita la resolución de errores
- Facilita la corrección de errores que puedan ocurrir durante el desarrollo
- Permite notar los cambios en cada versión para que cualquier persona sepa que se hizo y que falta por hacer.
- Permite empujar y tirar datos hacia y desde las instalaciones de otros ordenadores. (sistema de control de versiones distribuido)

A diferencia de la mayoría de los otros sistemas de control de versiones, git almacena cada versión guardada como una 'instantánea' en lugar de una lista de los cambios realizados en cada archivo. Puede hacer referencia a antiguas instantáneas siempre que lo necesite y las nuevas instantáneas se crean cuando se modifica el proyecto.

Contras:

- Es mejor para el uso individual.
- No puede visualizar las ediciones que otros desarrolladores estén realizando en tiempo real.



La Diferencia Entre git y GitHub

GitHub, es una plataforma que puede mantener basado en la nube para que varios desarrollador ediciones de cada uno en tiempo real.

Además, también incluye funciones de organización y gestión de proyectos. Puede asignar tareas a individuos o grupos, establecer permisos y roles para los colaboradores y usar la moderación de comentarios para mantener a todos en la tarea

Acciones principales:

- **Bifurcación:** El proceso de copiar el código de otra persona del repositorio para modificarlo.
- **Pull:** Cuando haya terminado de hacer cambios en el código de otra persona, puede compartirlos con el propietario original a través de una “solicitud pull”.
- **Fusión:** Los propietarios pueden añadir nuevos cambios a sus proyectos a través de una fusión, y dar crédito a los contribuyentes que los han sugerido.

Diferencias

git es un software de VCS local que permite a los desarrolladores guardar instantáneas de sus proyectos a lo largo del tiempo. Generalmente es mejor para uso individual.

GitHub es una plataforma basada en la web que incorpora las características de control de versiones de **git** para que puedan ser utilizadas de forma colaborativa. También incluye características de gestión de proyectos y equipos, así como oportunidades para la creación de redes y la codificación social.

Tutorial GIT

Paso 1: Instalar git y Añadir un Repositorio

Descargar git

Ejecutar el instalador

Glosario

- Repositorio: La ubicación del archivo donde se almacena su proyecto.
- Comprometerse: El comando utilizado para guardar los nuevos cambios en su proyecto en el repositorio.
- Escenario: Antes de que puedas confirmar los cambios en Git, necesitas prepararlos -esto le da la oportunidad de preparar su código antes de añadirlo formalmente a su proyecto.
- Rama: La parte de su proyecto que está desarrollando activamente.

Paso 2: Crear una cuenta GitHub

Crear cuenta GitHub <https://github.com/join>

Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.
[Learn more.](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

Después de registrarse ya podas ingresar un nuevo repositorio, tomar en cuenta que se debe de tener instalado Git en el equipo.

Para agregar un nuevo repositorio se debe ingresar a:

objects Packages Stars

Type ▾

Language ▾

Sort ▾

New

Se debe de colocar el nombre del repositorio y una descripción si lo desea.

Se selecciona Publico o privado y se crea.

Create a new repository

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

Owner *

Orlytrach

Repository name *

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

Description (optional)



Public

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



Private

You choose who can see and commit to this repository.

Initialize this repository with:

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

☐ Add a README file

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

☐ Add .gitignore

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

☐ Choose a license

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

Al dar en crear te dará una serie de formas, yo utilice la de command line.

Esto se debe ejecutar en git.

...or create a new repository on the command line

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

Descripción de que es cada uno:

Workin directory: en donde se trabaja con todos tus archivos

Stagin área: en donde se agregan todos los archivos que vas a preparar para el guardado

Repository: Cuando ya se desea guardar el campio

Git init: proyecto nuevo o proyecto que vas a empezar a utilizar en Git

Git add: es para pasar los archivos del working directory al stagin área

Git status: estado en el que están mis archivos

Git commit: para pasar del stagin area al repositorio

Git push: es para subirlo a un repositorio remoto

Git pull: si se trabajan con más desarrolladores te traer los cambios que los otros hayan realizado

Git clone: realiza copia del central a otra computadora para trabajarlo