

FIP

**Carrera
Programador
full-stack**

Sistemas de versionado

¿Qué es?

Gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.





Time



Un Sistema de Versionado de Código (SVC) es lo que nos permite **compartir el código** fuente de nuestros desarrollos y a la vez **mantener un registro de los cambios** por los que va pasando.



Your Project



VCS

Add headline
to index page

Create "about" page



Change page layout

index.html

modified

```
<h1>Headline</h1>
...
```

about.html

created

```
<html>
<head>
...
```

photo.png

created

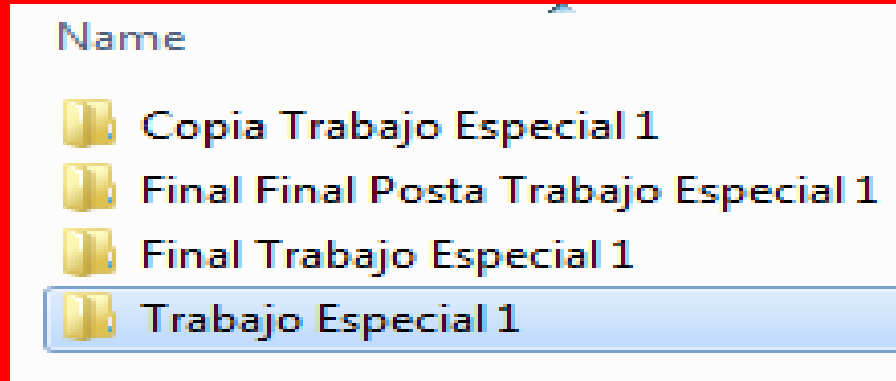
about.html

modified

```
<div>new content</div>
...
```



Copiar y Pegar Archivos



NO

Es control de versiones

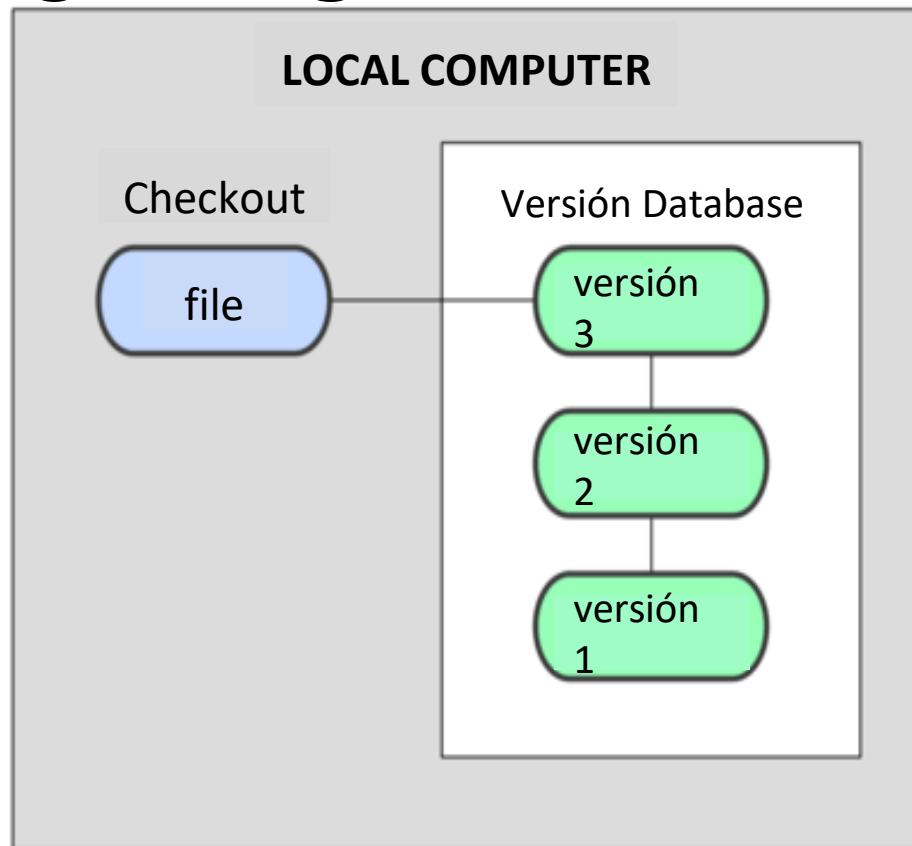
Software de Control de Versiones



SCV LOCALES:

Funciona guardando las diferencias entre archivos de una versión a otra en el disco. Otra manera de hacer esto es copiar los archivos a otro directorio.

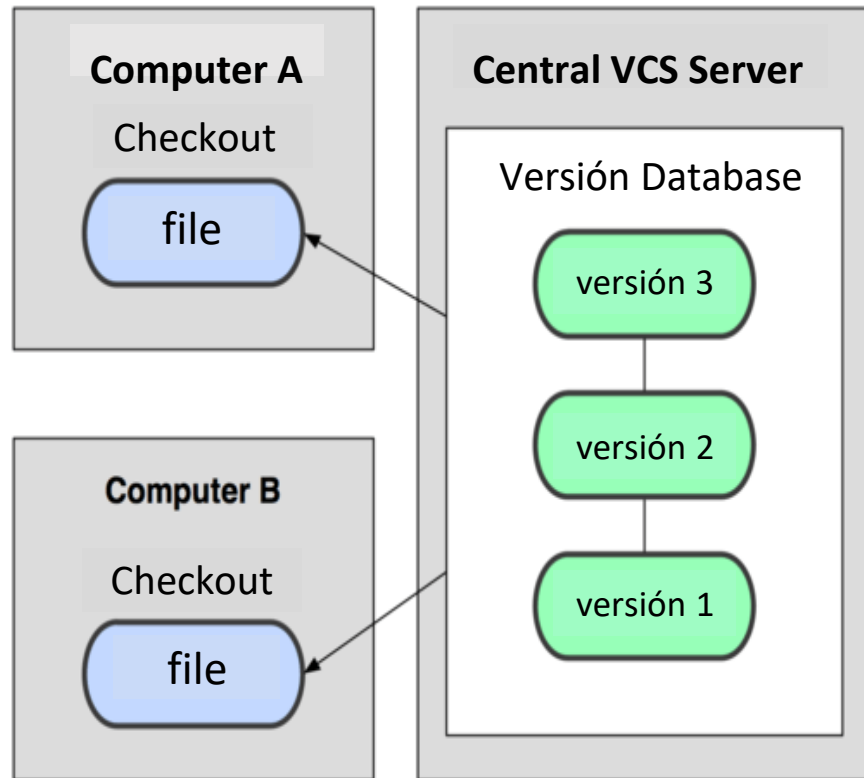
- VENTAJAS:
- Es simple.
- DESVENTAJAS:
- No permite trabajar en conjunto con otros desarrolladores.
- Propenso a errores.



SCV CENTRALIZADOS:

Tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central.

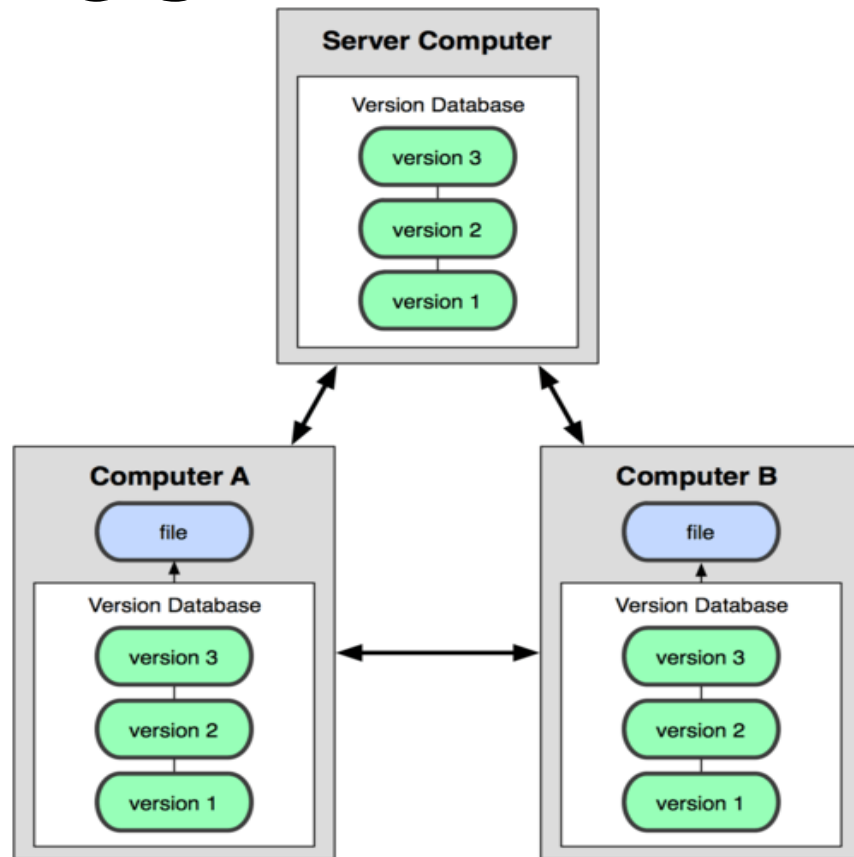
- VENTAJAS:
- Es más fácil de administrar.
- Sirve para trabajos en conjunto con otros desarrolladores.
- DESVENTAJAS:
- Si el servidor se cae o la base de datos se corrompe pierdes todo.



SCV DISTRIBUIDOS:

Los clientes no sólo descargan la última instantánea de los archivos: replican completamente el repositorio con cada descarga.

- VENTAJAS:
- Existen copias para restaurar el servidor en caso de ser necesario.
- DESVENTAJAS:
- Posibilidad de error al modificar un mismo archivo desde lugares diferentes.



Qué es? git

Git es un **sistema de control de versiones** que amplía la noción de otros VCS (Version Control System) por su capacidad de ofrecer casi todas sus funciones para usar **sin conexión y sin un servidor central**.

Hoy, Git es efectivamente el **estándar** para el control de versiones de software y es una herramienta esperada para todo framework de desarrollo. Los usuarios destacan su excelente rendimiento, flexibilidad de uso, capacidades fuera de línea y funciones de colaboración como su motivación para cambiar.

Balance

- + Se puede seguir trabajando offline. Incluso si se cae el servidor.
- + Cada repositorio tiene toda la información histórica (Backups replicados).
- + Repositorios más limpios ([Dictador Benevolente](#)).
- + Server de Git consume menos recursos.
- + Permite hacer pruebas locales versionadas y subir solo lo relevante.
- + Branching más sencillo.
- Curva de aprendizaje mayor.

TERMINOLOGÍA:

El **Remote repository** o repositorio remoto es donde se envían los cambios cuando desea compartirlos con otras personas, y de dónde obtienen sus cambios.

El **Development environment** o ambiente de desarrollo es todo lo que tenemos en nuestra computadora local

El **Local repository** o repositorio local es una copia del remoto más nuestros commits.

El **Staging area** o área de ensayo es donde se encuentran todos los cambios que se desea poner en el repositorio local.

El **Working directory** o directorio de trabajo es donde están todos los archivos. Aquí hay dos tipos de archivos: archivos rastreados que git conoce y archivos no rastreados que git no conoce

TERMINOLOGÍA:

Branches o ramas, se utilizan para desarrollar características aisladas unas de otras. El branch master es el default cuando se crea un repositorio. Usamos otras ramas para el desarrollo y las unimos (merge) nuevamente a la rama maestra al finalizar.

Tags o etiquetas, se usan para realizar marcas entre commits. Se recomienda crear etiquetas para lanzamientos de software.

Pull requests (PR), te permite informar a otros sobre los cambios que has introducido en un repositorio remoto. Una vez que se envía un PR, las partes interesadas pueden revisar el conjunto de cambios, discutir posibles modificaciones y demás.

Comandos básicos de GIT

[GIT most useful commands](#)

Practicamos GIT

[Learn GIT branching](#)

Instalación en Windows

Debido a que el Command Prompt (o consola de Windows) no es muy amigable y no soporta comandos de Unix, la web de Git nos proporciona una herramienta muy buena llamada Git Bash y que nos permitirá escribir comandos como si estuviésemos en Linux o Mac OS X.

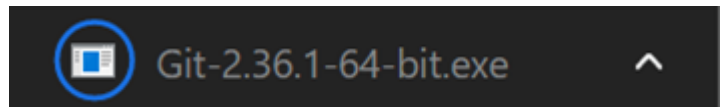
Instalación en Windows

Para empezar, entra a la página de Git ([git windows](#)) y presionar en el link que se muestra a continuación:



Instalación en Windows

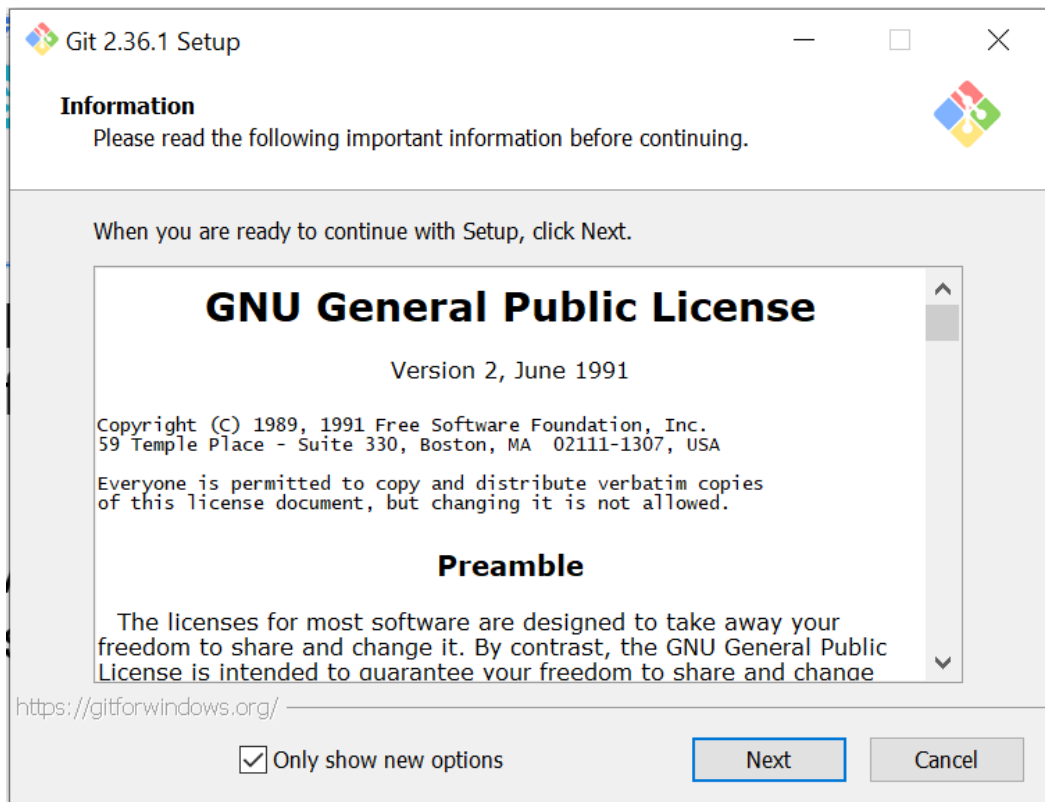
Esto realiza la descarga de un archivo con el formato Git-version.exe.



Al finalizar la descarga, hacemos doble click sobre el archivo y se inicia el instalador.

Al inicio preguntará por permisos para ejecutar el instalador, a lo cual debemos responder que sí.

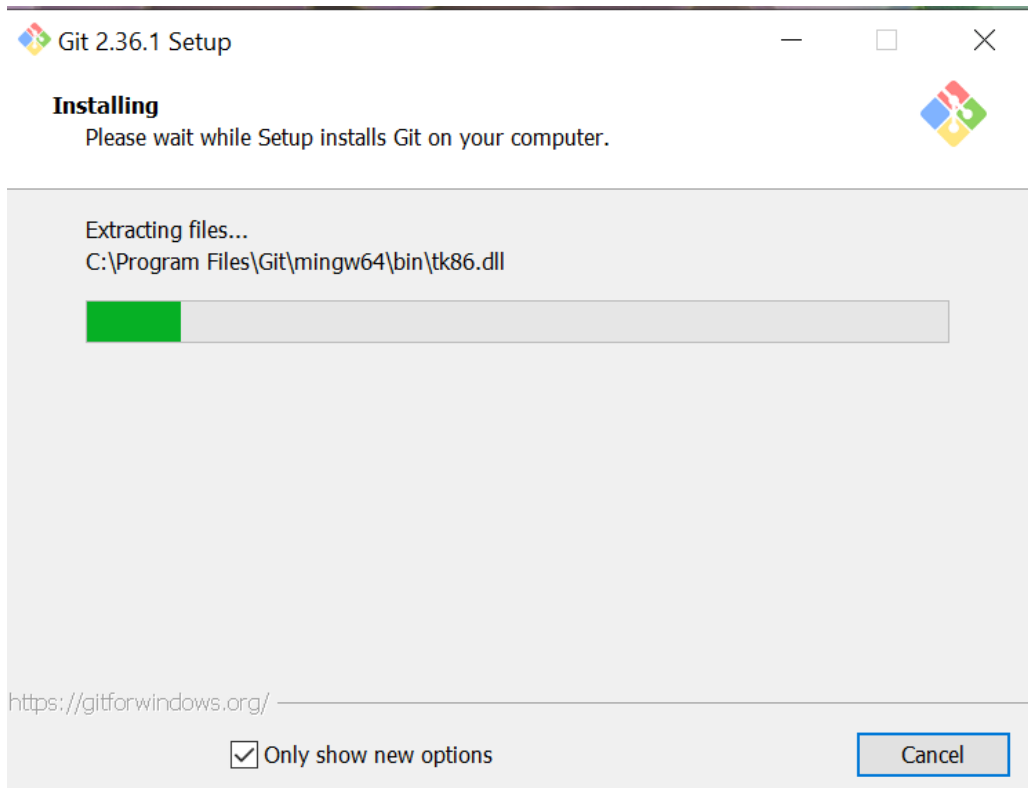
Instalación en Windows



Instalación en Windows

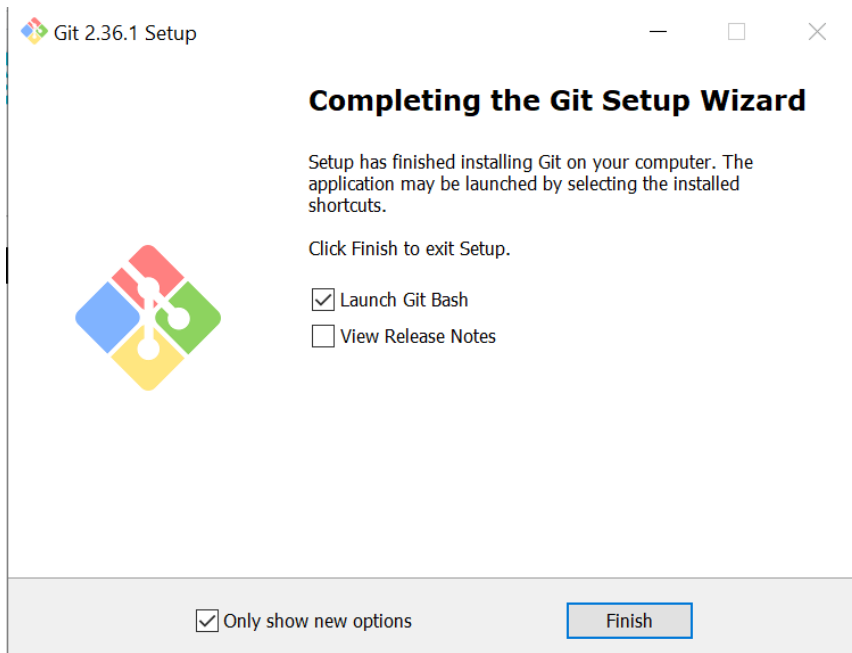
Como muchos de los instaladores en Windows, debemos de aceptar las opciones por defecto y darle Next (siguiente) a todo hasta que nos salga el botón de instalar, presionamos y comienza la instalación:

Instalación en Windows



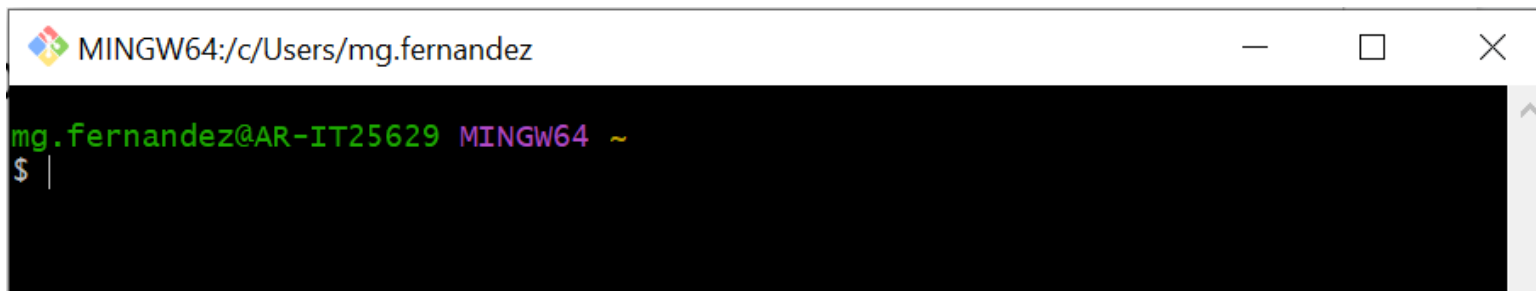
Instalación en Windows

Una vez finalizada la instalación presionamos finish.



Instalación en Windows

Seleccionando en el paso anterior la opción “Launch Git Bash”, abre la terminal para ingresar comandos.



```
MINGW64:/c/Users/mg.fernandez
mg.fernandez@AR-IT25629 MINGW64 ~
$ |
```

Formación Integral

Carrera Programador full-stack

GITHUB

Servicios de hosting para **git**

Para hostear/almacenar sus repositorios Git, necesitará algún servicio de hosting. Hay diferentes variantes, aquí tienes algunos ejemplos.



GitHub



GitLab



Bitbucket



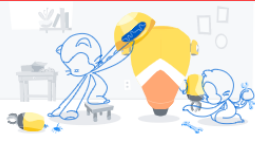
- Plataforma de desarrollo colaborativo, que utiliza Git.
- Ofrece GIT y más cosas juntas
- Gratuito
- Tiene facetas de red social. Se suele usar como CV de proyectos **propios**.
- Existen otras alternativas (GitLab, BitBucket, etc).

Crear un repositorio

Your email was verified.


What do you want to do first?

Every developer needs to configure their environment, so let's get your GitHub experience optimized for you.



Start a new project
Start a new repository or bring over an existing repository to keep contributing to it.

Create a repository



Collaborate with your team
Improve the way your team works together and get access to more features with an organization.

Create an organization



Learn how to use GitHub
Get started with an "Introduction to GitHub" course in our Learning Lab.

Start Learning

Overview Repositories 4 Projects Packages Stars

Find a repository...

Type Language Sort

New



Crear un repositorio

Create a new repository

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

Owner *



cflFullstackDocente ▾

Repository name *

CFL2021 ✓

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

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.](#)

Create repository



Crear un repositorio

Vamos a crear un repositorio específico para este cuatrimestre: CFS2022II




Crear un repositorio

cflFullstackDocente / CFL2021

Unwatch 1 Star 0 Fork 0

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


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

```
echo "# CFL2021" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/cflFullstackDocente/CFL2021.git
git push -u origin main
```



...or push an existing repository from the command line


```
git remote add origin https://github.com/cflFullstackDocente/CFL2021.git
git branch -M main
git push -u origin main
```



...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

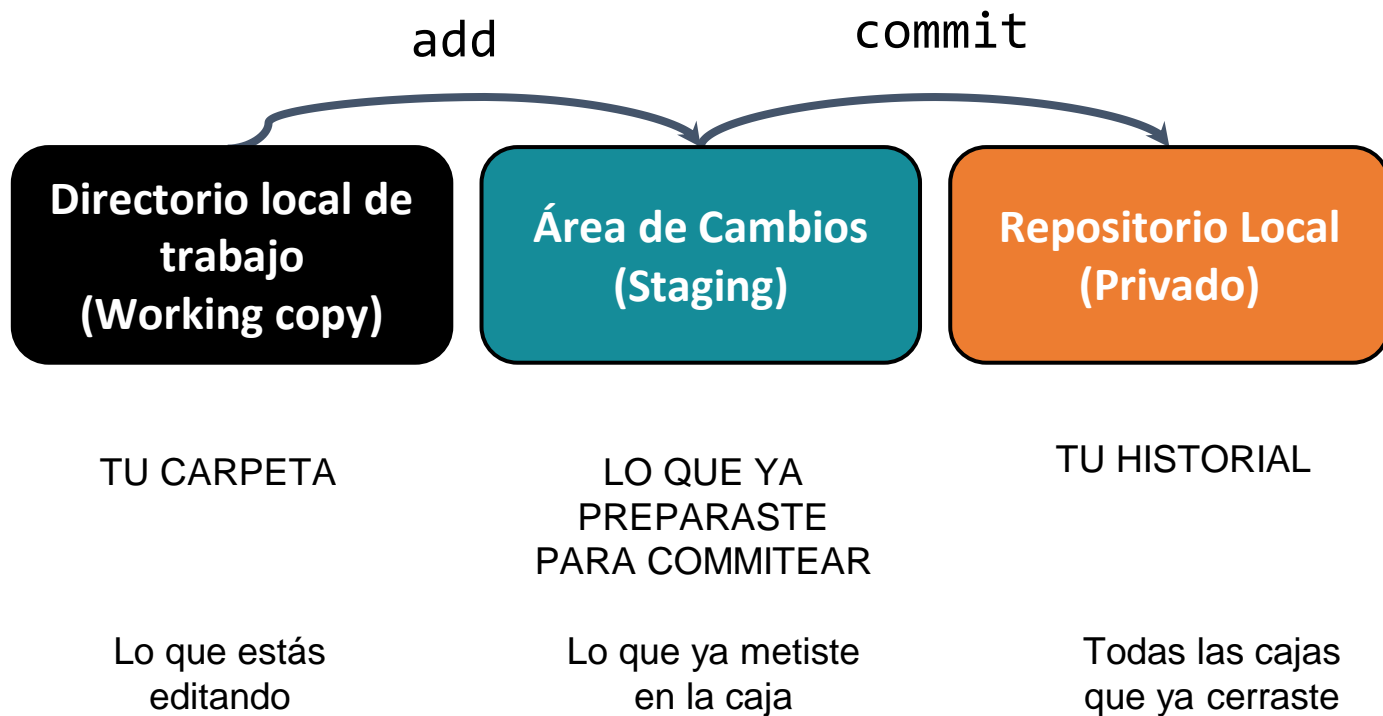
[Import code](#)

 **ProTip!** Use the URL for this page when adding GitHub as a remote.



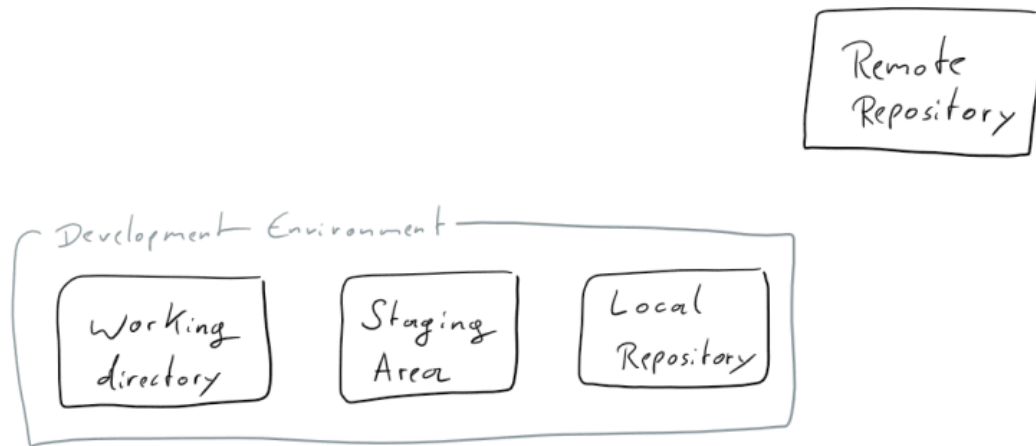
Metafora

- Cada versión es una caja
- Lo que estamos editando es una caja abierta (llamada stage)
- Y tenemos nuestra versión de los archivos (working copy)
- A esa caja le agregamos los archivos modificados
 - `git add` archivos
- Al cerrar la caja congelamos la versión
 - `git commit -m "Mensaje de la versión"`
 - Se abre otra caja nueva para la próxima versión (nuevo stage)
- Una vez lista la versión podemos pasarle esa caja al servidor
 - `git push` servidor branch
 - Si se rompe nuestra PC tenemos una copia
 - Podemos interactuar ahí con otra gente, copiarnos sus cajas y mezclarlas



Agregando cambios en mi Repo
local

El Repositorio Remoto es donde envíamos los cambios cuando queremos compartirlos con otras personas, y de dónde obtenemos sus cambios.



El entorno de desarrollo es lo que tenemos en nuestra máquina local. Las tres partes son nuestro directorio de trabajo, el área de ensayo y el repositorio local.

Elijamos un lugar en el que deseamos colocar nuestro entorno de desarrollo.

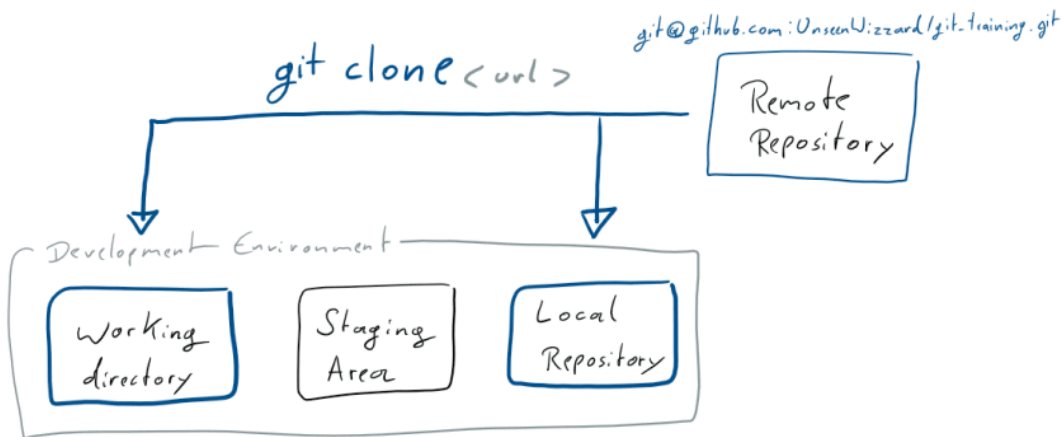
Simplemente vamos a la carpeta de inicio, o donde quieran poner sus proyectos. No es necesario crear una nueva carpeta para su entorno de desarrollo.

Ahora queremos tomar un Repositorio Remoto y poner lo que contiene en su máquina.

Para hacer eso usamos `git clone` https://github.com/my_repo

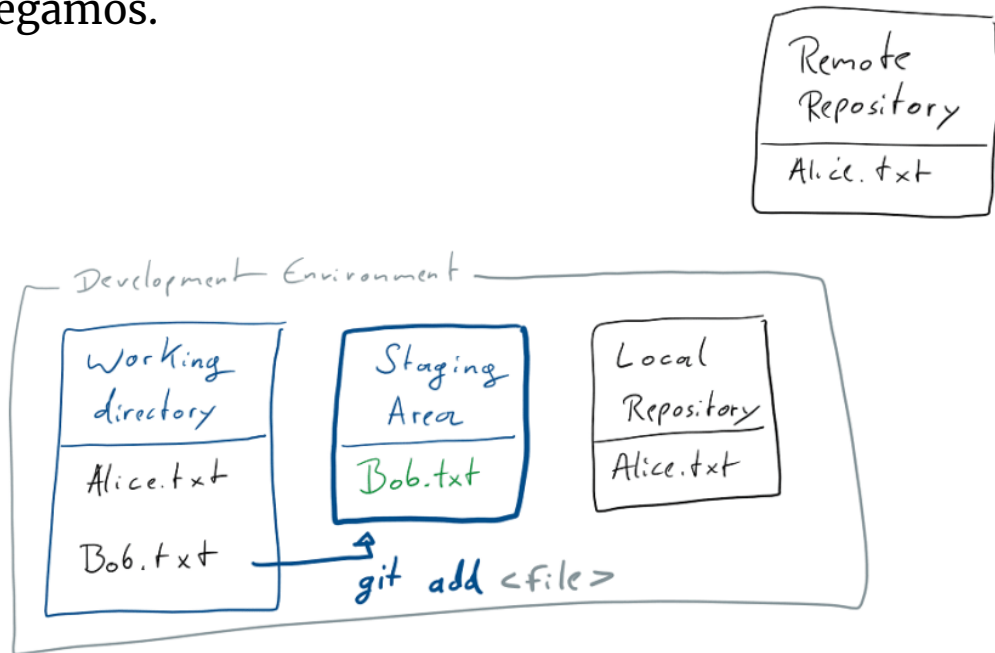
Como podemos ver en el diagrama a continuación, esto copia el repositorio remoto en dos lugares, tu directorio de trabajo y el repositorio local.

Ahora ves cómo se distribuye el control de versiones de git. El repositorio local es una copia del remoto y actúa como tal. La única diferencia es que no lo compartis con nadie.

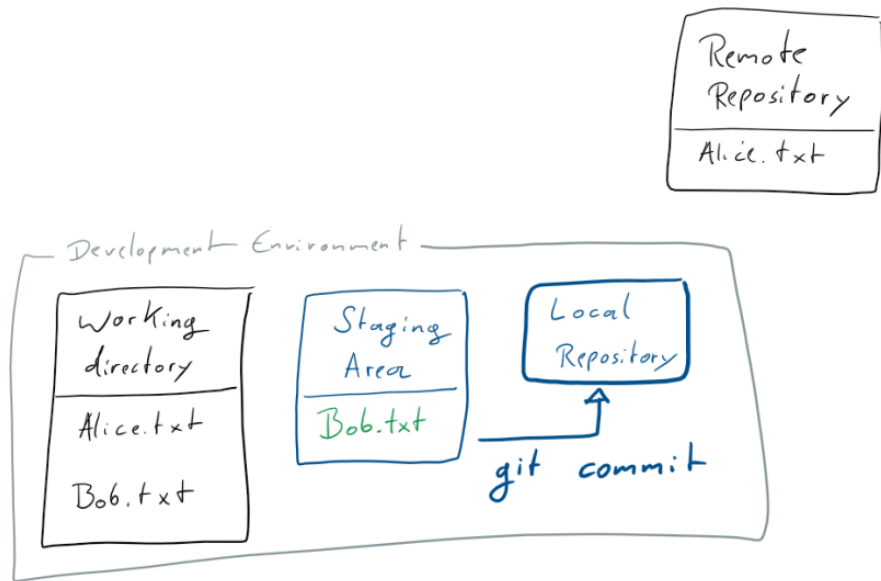


A continuación creamos un nuevo file y lo vamos a agregar:

Usamos **git status** para verificar qué cambios tenemos para agregar y con **git add tu_file** lo agregamos.

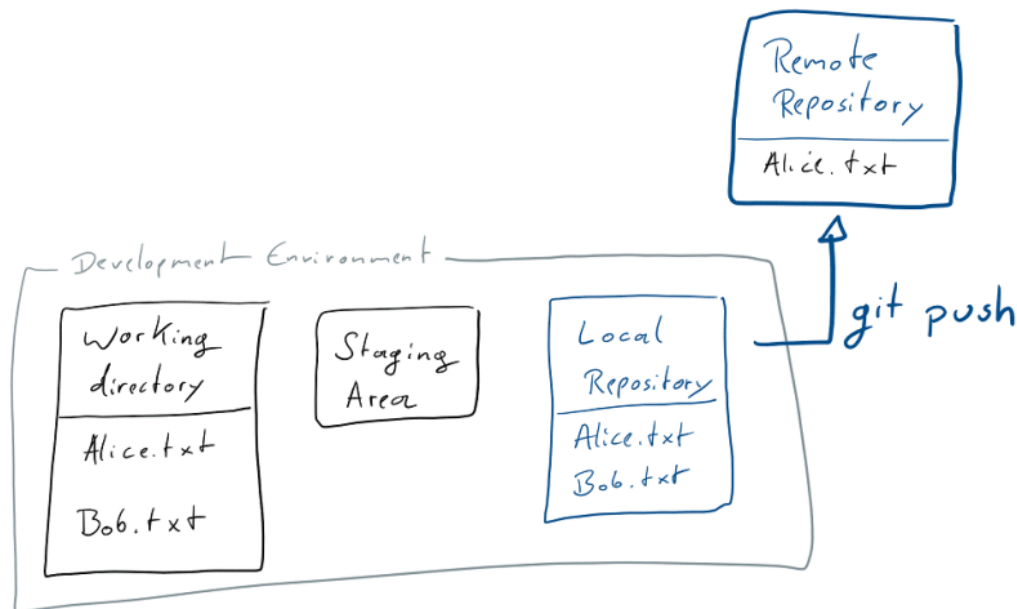


Cuando hayas agregado todos tus cambios (que en este momento solo es tu file nuevo), estas listo para commitear lo que acabas de hacer en el Repositorio local. Los cambios recopilados que commiteas son una parte importante del trabajo, por lo que cuando ejecutes `git commit -m "Mensaje"`, donde podés escribir un mensaje que diga todo lo que acabas de hacer. Cuando guardas y cerrás el archivo de mensajes, tu commit se agrega al Repositorio local.

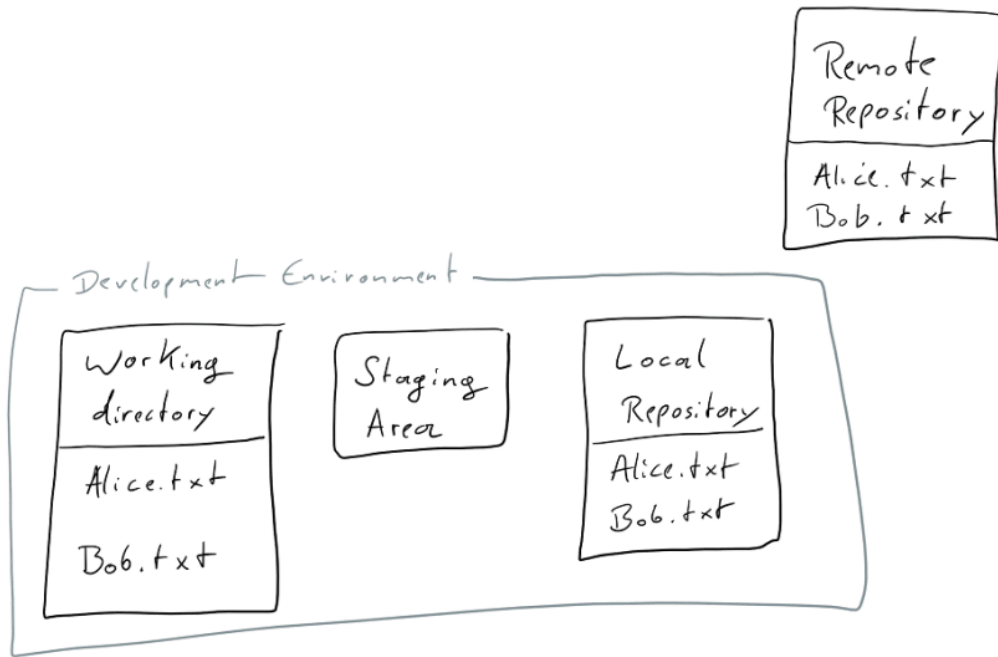


Ahora sus cambios están en su repositorio local, que es un buen lugar para estar mientras nadie más los necesite o aún no esté listo para compartirlos.

Para compartir sus confirmaciones con el repositorio remoto, debe enviarlas, usando el comando **git push**.



Una vez que ejecutes git push, los cambios se enviarán al repositorio remoto. En el siguiente diagrama, podés ver el estado después del push.



Este gato ya hizo

```
git add gatoblanco.txt
```

En la imagen se ve como hace

```
git commit -m "Callate"
```



Trabajar solo + Backup

- Por ahora solo trabajamos de a uno en el repositorio.
- Un SCV nos permite trabajar solos y tener un tracking de todo lo que fuimos haciendo.
- Cuando hacemos **push** estamos haciendo un backup en otro repositorio (en nuestro caso GitHub).

Trabajar en equipo

- Podemos trabajar en equipo.
- Resolver los conflictos de código
- Que todos estemos sincronizados con lo último que hay en el repositorio.



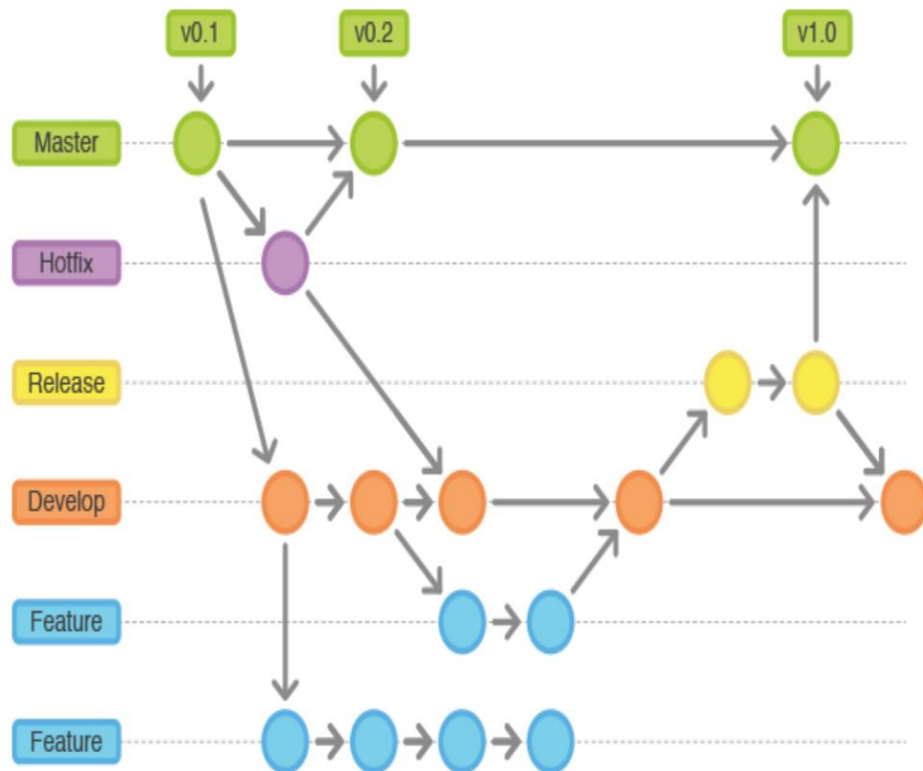
RAMAS:

- **Ramas de largo recorrido:**

Se tienen varias ramas siempre abiertas, que indican diversos grados de estabilidad del contenido.

- **Ramas puntuales:**

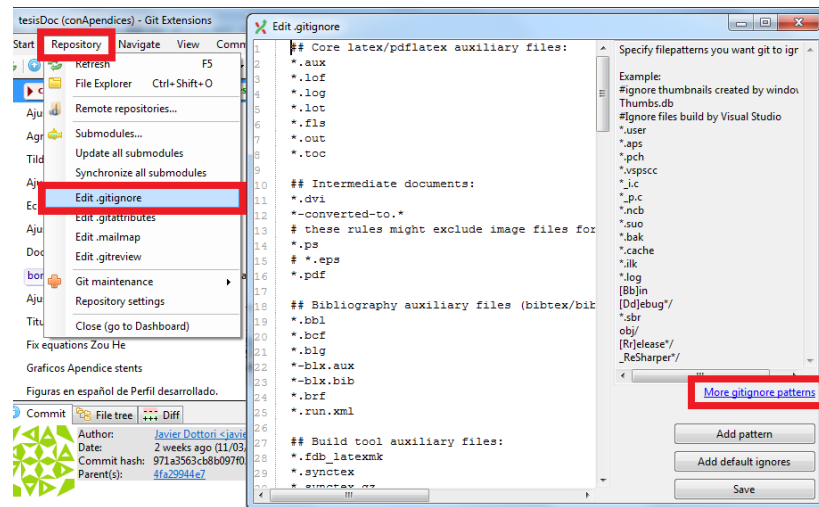
Se crean de forma puntual para realizar una funcionalidad muy concreta.



Como creo una rama?

.gitignore

- Es un archivo del repositorio (se versiona también).
- Dice que archivos no se van a versionar.
- Permite excluir carpetas de archivos compilados, fotos pesadas, etc.
- Hay muchos ejemplos para cada lenguaje en la web para cada lenguaje (<https://github.com/github/gitignore>).



Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]: [file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
```

```
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

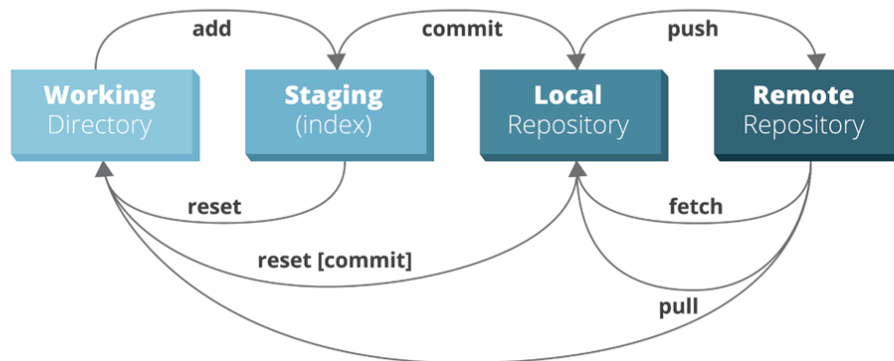
```
$ git push
```

Finally!

When in doubt, use git help

```
$ git command --help
```

Or visit <https://training.github.com/> for official GitHub training.



Configurar mi usuario como autor

A cada commit se le guarda el autor con un username y mail

```
git config --global user.name "Mi Nombre"
```

```
git config --global user.email "..."
```

--global hace que se guarde como configuración del usuario

Sino es solo para cada repositorio

Podes commitear en la misma PC con mail diferente (laboral/personal) en cada repositorio

Cómo subo mis cambios para que se vean en github?

Vamos a crear un PR

Como agrego reviewers?

Migracion a Github

A partir de este módulo vamos a empezar a hacer las entregas y consultas de los ejercicios a través de Pull requests en github.

- 1- Crear un nuevo repositorio para el segundo cuatrimestre (Si no lo hicieron en el repaso)
- 2- Crear un nuevo branch desde master/main llamado ej1-readme
- 3- En el archivo readme del repositorio, poner una descripción del curso y sus contenidos junto con el objetivo de su repositorio
- 4- Subir el branch nuevo con ese cambio a github
- 5- Generar un PR a master poniendo a los profesores como reviewers
- 6- Después de tener 2 approvals hacer merge.
- 7- En el repositorio local, volver a pararse en master/main
- 8- Traer los cambios de github que se mergearon en el punto 6 (desde master/main)