

% Entorno de Programación - Introducción a scm y git

% Tecnicatura Universitaria en Inteligencia Artificial

% U.N.R.

## Introducción a git

---

...

- Sistema de control de versiones (SCM). <https://git-scm.com/>

...

- Esencialmente, rastrea los cambios de nuestros archivos,

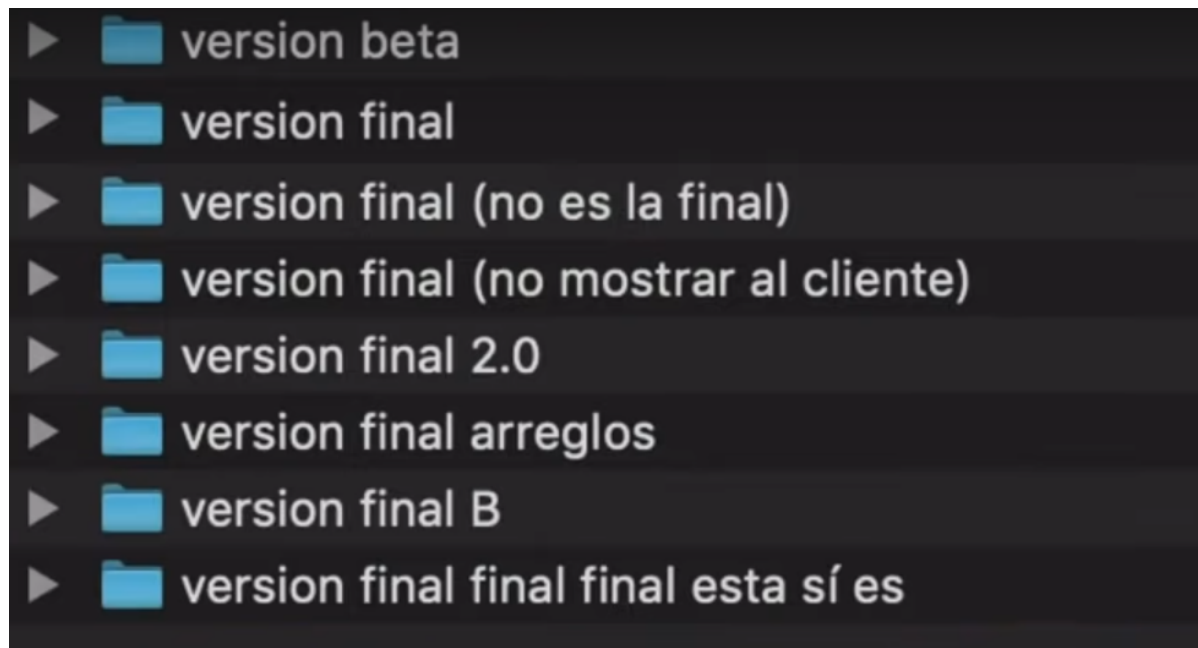
...

y permite ir hacia atrás si hace falta.

## Nuestra motivación a esta altura

---

vamos a tratar de evitar esto



## Introducción a git

---

- Creado originalmente por Linus Torvalds en 2005...

...

en semanas.

...



# git

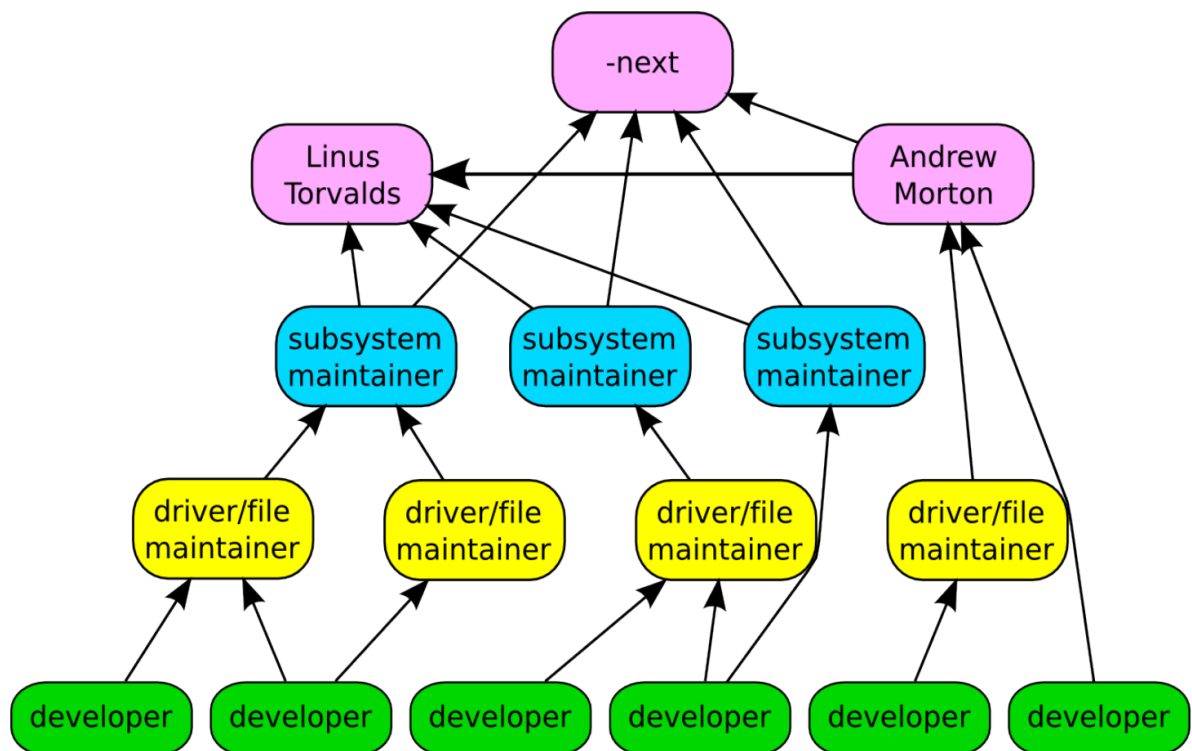
/git/

noun

**DEROGATORY • INFORMAL**

an unpleasant or contemptible person.  
"that mean old git"

## Flexible, jerárquico, distribuido



## Introducción a git

- Programado en: C, Bourne Shell, Perl

...

- Es un proyecto de código abierto

...

- Una de sus principales características es que es distribuido.

...

- Cada desarrollador tiene todo el código y toda la historia.

# Aprender entender y usar apropiadamente git



## ¿Cómo arranca un repositorio?

1. clonando:

```
$ git clone https://github.com/aleoncavallo/tutorial_bash
```

...

```
Clonando en 'tutorial_bash'...
remote: Enumerating objects: 186, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 186 (delta 18), reused 10 (delta 4), pack-reused 151
Recibiendo objetos: 100% (186/186), 821.02 KiB | 5.20 MiB/s, listo.
Resolviendo deltas: 100% (64/64), listo.
```

# ¿Cómo arranca un repositorio?

---

2. inicializando:

...

```
~/ $ git init proyecto
```

...

```
Inicializado repositorio Git vacío en /home/aleoncavallo/proyecto/.git/
```

...

```
~$ cd proyecto/
```

## Mirando el estado

---

```
~/tutorial_bash$ git status
```

...

```
En la rama master
Tu rama está actualizada con 'origin/master'.

nada para hacer commit, el árbol de trabajo está limpio
```

## Mirando el estado

---

```
~/proyecto$ touch otro.sh
```

...

```
~/proyecto$ echo 'echo hola mundo!' > script.sh
```

...

```
~/proyecto$ git status
```

...

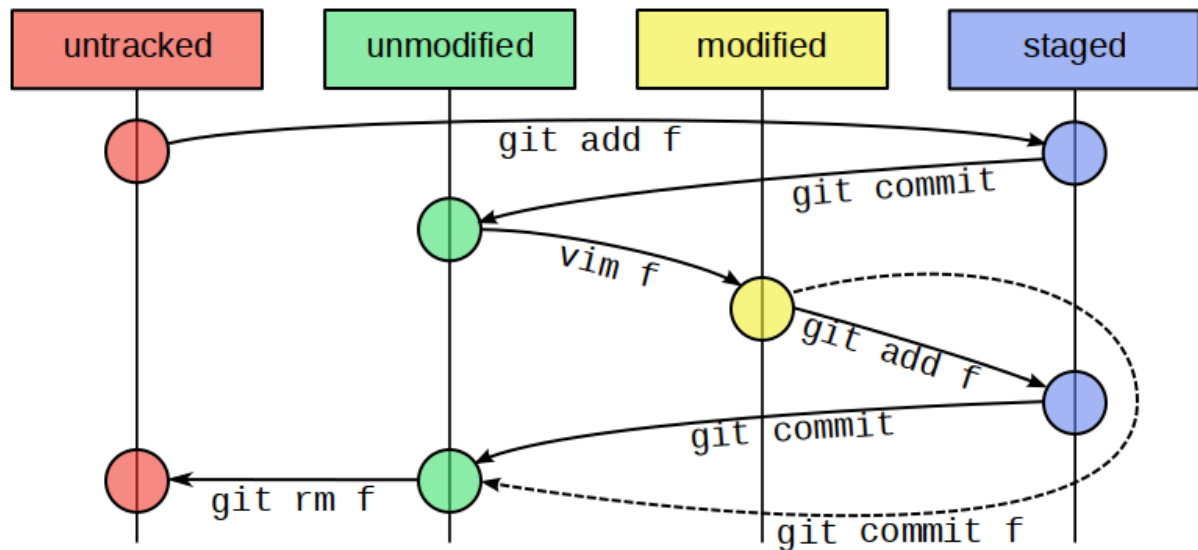
```
En la rama master
Tu rama está actualizada con 'origin/master'.
```

Archivos sin seguimiento:

```
(usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
otro.sh
script.sh
```

no hay nada agregado al commit pero hay archivos sin seguimiento presentes  
(usa "git add" para hacerles seguimiento)

## Estados de un archivo



## Staging area (escenario)

```
~/proyecto$ git add script.sh
```

...

```
~/proyecto$ git status
```

...

```
En la rama master
Tu rama está actualizada con 'origin/master'.
```

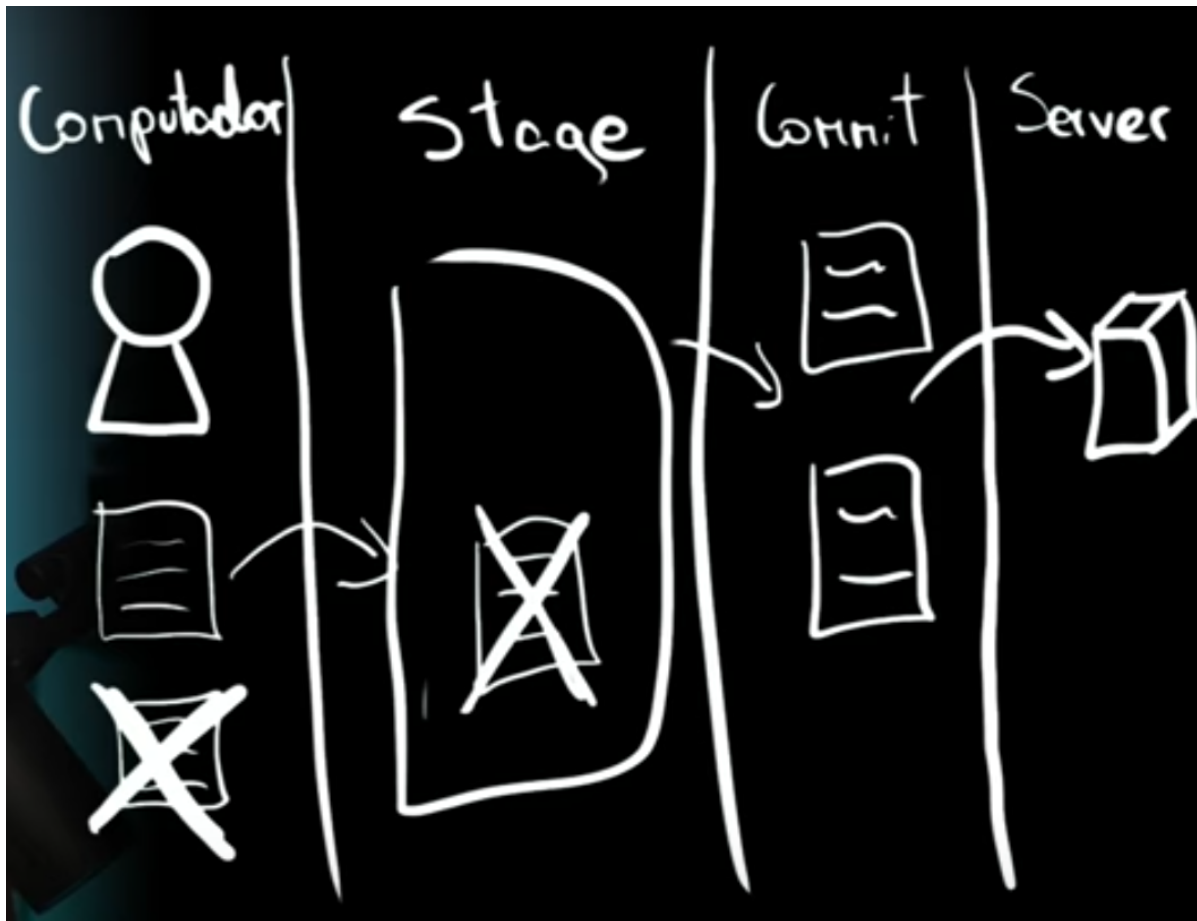
Cambios a ser confirmados:

```
(usa "git restore --staged <archivo>..." para sacar del área de stage)
nuevos archivos: script.sh
```

Archivos sin seguimiento:

```
(usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
otro.sh
```

## Estados repositorio



## Commit (compromiso)

```
~/proyecto$ git commit -m "Inicio proyecto"
```

...

```
[master (commit-raíz) 4ac0385] Inicio proyecto
1 file changed, 1 insertion(+)
create mode 100644 script.sh
```

...

```
~/proyecto$ git status
```

...

```
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
  otro.sh

no hay nada agregado al commit pero hay archivos sin seguimiento presentes
  (usa "git add" para hacerles seguimiento)
```

# Commit (compromiso)

---

- Es una fotografía de una situación de nuestro código/proyecto  
Se puede pensar como la foto completa del árbol de los estados hasta el momento

...

- Es un hash criptográfico de:
  - Todos los archivos
  - Mensaje de commit
  - Autor, fecha, etc
  - Commit padre

...

Hash criptográfico = es un identificador, se hace con una operación matemática que no se puede invertir, es muy difícil encontrar colisiones

Lo calculan con optimizaciones para no computar el hash desde cero cada vez

## Log

---

```
~/proyecto$ git log -p
```

...

```
commit 4ac03851baed8e79c19ba3c2e3707d0f8477abc8 (HEAD -> master)
Author: Andrea Leon Cavallo <aleoncavallo@gmail.com>
Date:   Tue May 30 17:16:28 2023 -0300
```

Inicio proyecto

```
diff --git a/script.sh b/script.sh
new file mode 100644
index 0000000..34cae35
--- /dev/null
+++ b/script.sh
@@ -0,0 +1 @@
+echo hola mundo!
```

## Diff Unificado

---

```
~/proyecto$ git show
```

```
commit 06031abc6fd5794e7b5e1bc6b941d0d3984408df (HEAD -> master)
Author: Andrea Leon Cavallo <aleoncavallo@gmail.com>
Date:   Tue May 30 17:26:00 2023 -0300
```

corregir énfasis en español

```
diff --git a/script.sh b/script.sh
index 34cae35..baaa505 100644
--- a/script.sh
```

```
+++ b/script.sh
@@ -1,1 @@
-echo hola mundo!
+echo ;hola mundo!
```

## Borrar archivos

- `git rm` borra un archivo (y anota el cambio en la staging area). Es lo mismo que hacer `rm` y `git add`.

```
~/proyecto$ git rm script.sh
```

...

```
rm 'script.sh'
```

...

```
~/proyecto$ git status
```

...

```
On branch master
Your branch is up to date with 'origin/master'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
   deleted:
    script.sh
```

...

```
$ git commit -m "Borrar main"
```

...

```
[master f068e5f] Borrar main
1 file changed, 1 deletions(-)
delete mode 100644 main.c
```

## Buenas prácticas

- Commits lo más pequeños posibles (“atómicos”): permite revertir fácilmente

...

- Mensajes descriptivos: “cambios” vs “Agrego tal funcionalidad”



# Config

```
### Establecer el nombre de usuario y email
$ git config --global user.name "Alan Turing"
$ git config --global user.email "aturing@princeton.edu"
```

```
### Guardar la clave y no la pregunte continuamente
$ git config --global credential.helper store
```

...

El `--global` es solo si lo están usando en un equipo que no comparten con otros y que no usan para otro servidor de git.

## Remotes

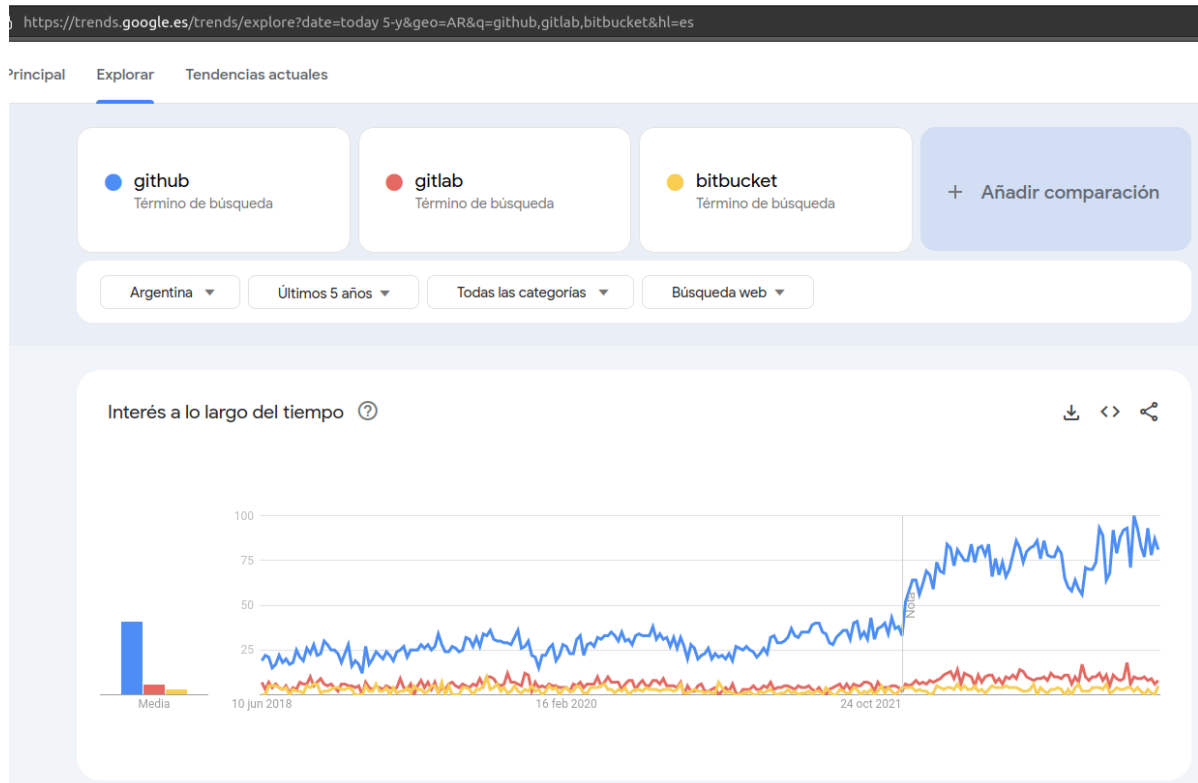
Hasta ahora, nada requirió internet.

...

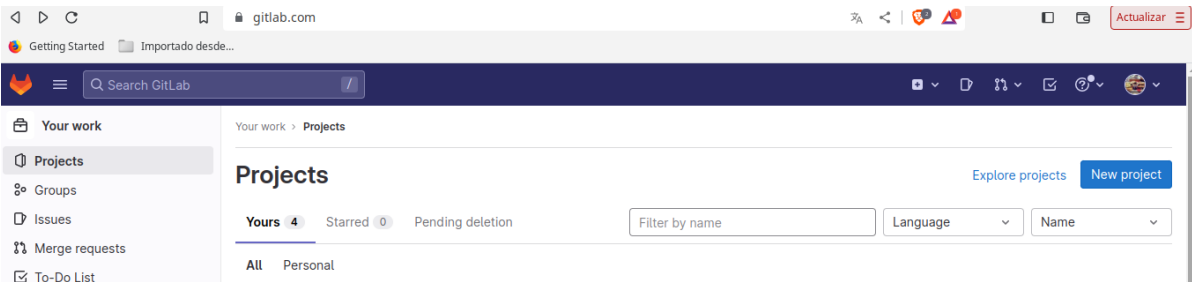
De ahí viene lo de distribuido,  
todos los cambios que fuimos haciendo y registrando  
los podemos hacer sin conexión a internet/red.

## Remotes

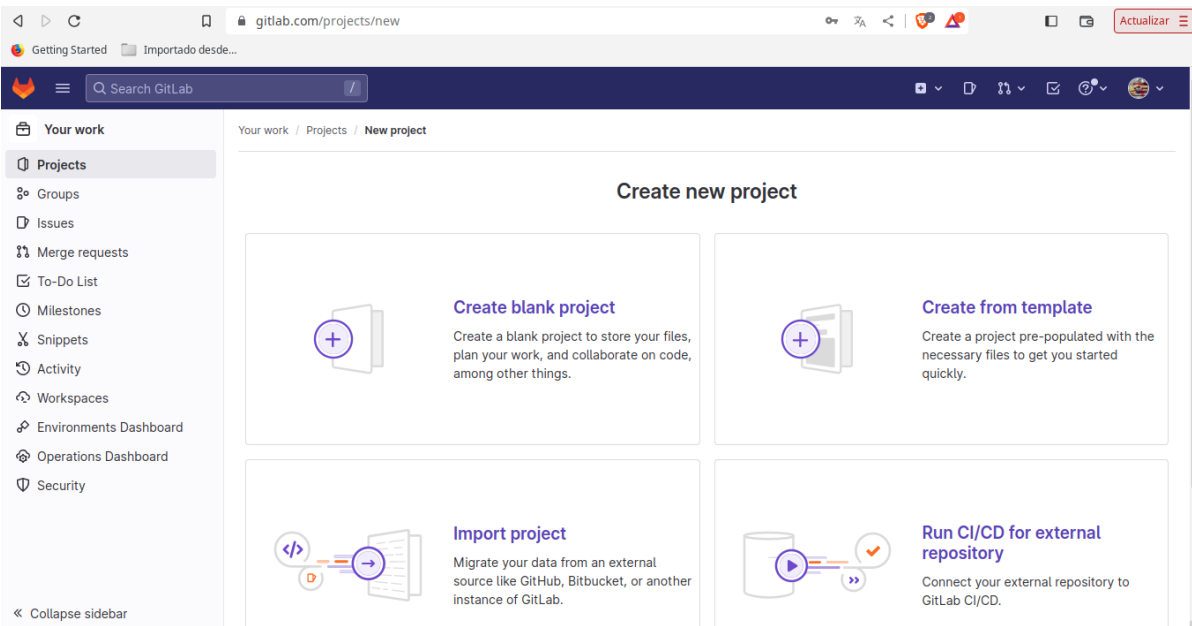
GitLab y GitHub, y Bitbucket son sitios que ofrecen la posibilidad de alojar en ellas nuestros proyectos.



# Remotes




# Remotes



# Remotes

Your work / Projects / New project / **Create blank project**



## Create blank project

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

**Project name**

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


**Project URL**

**Project slug**

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

**Project deployment target (optional)**

Select the deployment target

**Visibility Level** 

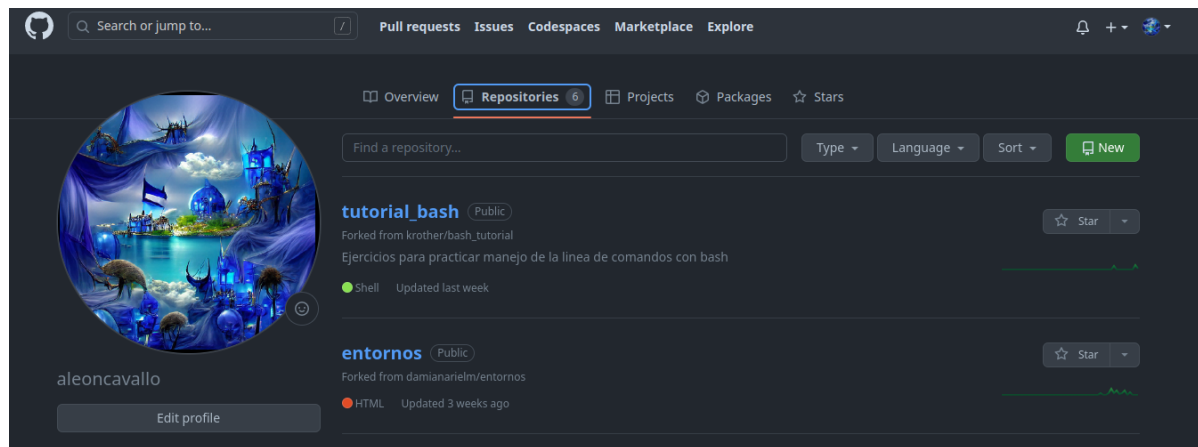
☒ Private

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.

# Remotes



The screenshot shows the GitHub profile of user 'aleoncavallo'. The profile picture is a circular image of a fantastical island with a lighthouse and a flag. The user's name 'aleoncavallo' is displayed below the picture, with an 'Edit profile' button. The 'Repositories' tab is selected, showing a list of repositories. The first repository is 'tutorial\_bash', which is public, forked from 'krother/bash\_tutorial', and is a Shell repository updated last week. The second repository is 'entornos', which is public, forked from 'damianarielm/entornos', and is an HTML repository updated 3 weeks ago. The top navigation bar includes links for Pull requests, Issues, Codespaces, Marketplace, and Explore. A search bar is located at the top left.


# Remotes

## Create a new repository

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

---


**Owner \*** **Repository name \***


 aleoncavallo /

Great repository names are short and memorable. Need inspiration? How about **expert-engine**?

**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:**

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs](#).

**Add .gitignore**


.gitignore template: None ▾

# Remotes

## Create a new repository

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

Owner \*

 aleoncavallo ▾

/

Repository name \*


git\_intro

✔ **git\_intro** is available.


Great repository names are short and memorable. Need inspiration? How about **expert-engine**?

Description (optional)

Introducción a git

☐  **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:

☐ **Add a README file**

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

Add .gitignore

gitignore template: None

# Remotes


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

or

HTTPS

SSH


https://github.com/aleoncavallo/git\_intro.git



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



# Remotes

```
git remote add origin https://github.com/aleoncavallo/git_intro.git
```

...

o desde gitlab

```
git remote add origin git@gitlab.com:aleoncavallo/git_init.git
```

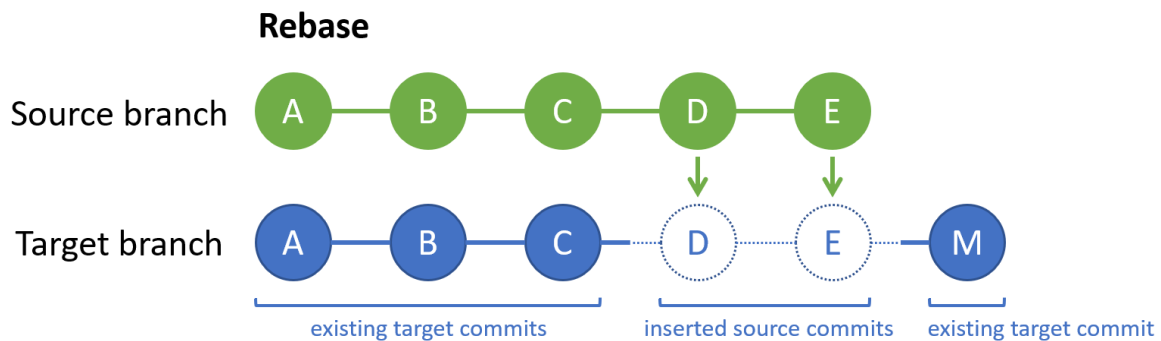


...

Posiblemente haya que corregir conflictos

## Rebase

- Similar a un merge... pero "reescribe la historia"



- Generalmente sólo se hace en ramas privadas

## Bisect

```
git bisect
```

Bisect es partir por la mitad y es justamente lo que va a hacer este comando, ir dividiendo toda la pila de commits en dos partes, una parte de la pila contendrá el error y otra parte no.

## Clean

Usar con mucho cuidado...

- `git clean -dfx`: borra todo lo que no esté trackeado/staged
- `git clean -x`: borra sólo archivos ignorados (suele ser seguro)
- Flag -n: no hacer nada, imprimir lo que haría

## Fuentes y links recomendados

- Sitio oficial de git: <https://git-scm.com/>
- Guía de git: <https://rogerdudler.github.io/git-guide/>
- Aprender a hacer reamas: [https://learngitbranching.js.org/?locale=es\\_AR](https://learngitbranching.js.org/?locale=es_AR)
- Video Aprende GIT de HolaMudo <https://www.youtube.com/watch?v=VdGzPZ31ts8>