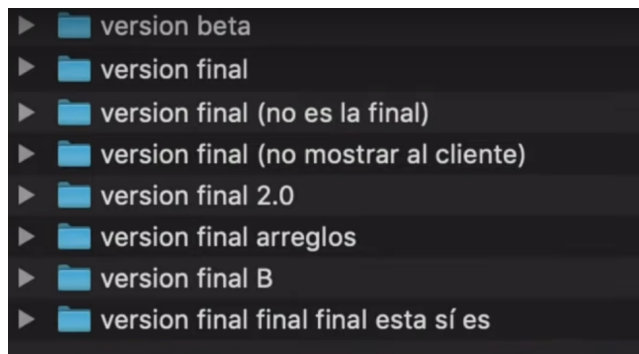


# Tecnicatura Universitaria en Inteligencia Artificial

U.N.R.

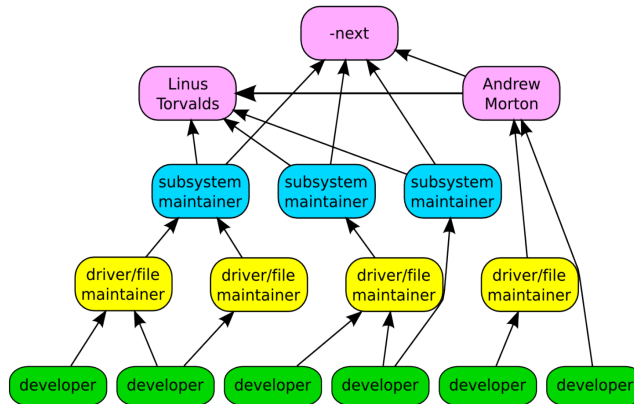
- Sistema de control de versiones (SCM). <https://git-scm.com/>
- Esencialmente, rastrea los cambios de nuestros archivos,  
y permite ir hacia atrás si hace falta.



- Creado originalmente por Linus Torvalds en 2005...



## Flexible, jerárquico, distribuido



- Programado en: C, Bourne Shell, Perl
- Es un proyecto de código abierto
- Una de sus principales características es que es distribuido.



## Config

```
$ git config --global user.name "Alan Turing"
$ git config --global user.email "aturing@princeton.edu"
$ git config --global credential.helper store
```

## ¿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
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

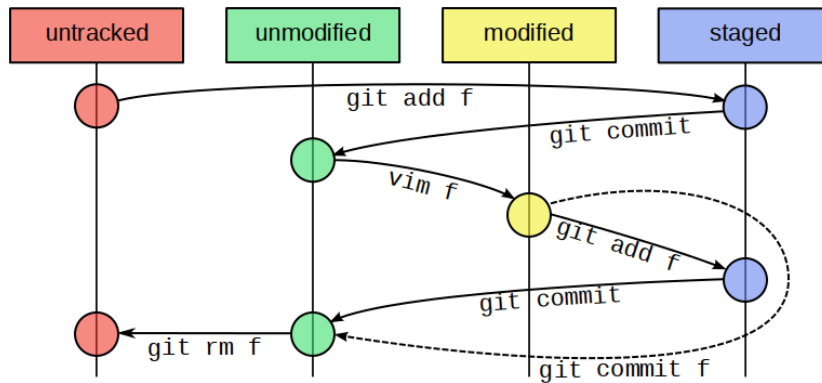
```
~/proyecto$ touch otro.sh
```

```
~/proyecto$ git status
En la rama master
Tu rama está actualizada con 'origin/master'.
```

```
Archivos sin seguimiento:
  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



## Stagin area (escenario)

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

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

En la rama master

Tu rama está actualizada con 'origin/master'.

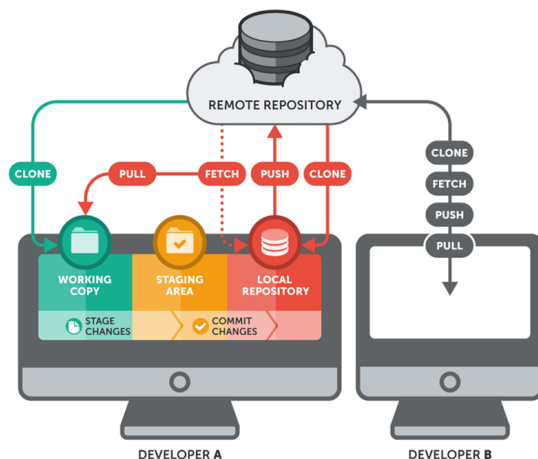
(usa `"git restore --staged <archivo>..."` para sacar del área de stage)

nuevos archivos: script.sh

Archivos sin seguimiento:

otro.sh

## Esquema general de uso de git



## Commit (compromiso)

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

```
[master (commit-raíz) 4ac0385] Inicio proyecto
create mode 100644 script.sh
```

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

En la rama master

Archivos sin seguimiento:  
    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:

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>
```

```
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>
```

```
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

```
~/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
```

```
delete mode 100644 main.c
```

## Buenas prácticas

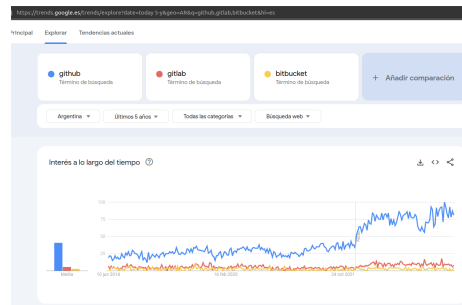
- Commits lo más pequeños posibles (“atómicos”): permite revertir fácilmente
- Mensajes descriptivos: “cambios” vs “Agrego tal funcionalidad”

## Remotes

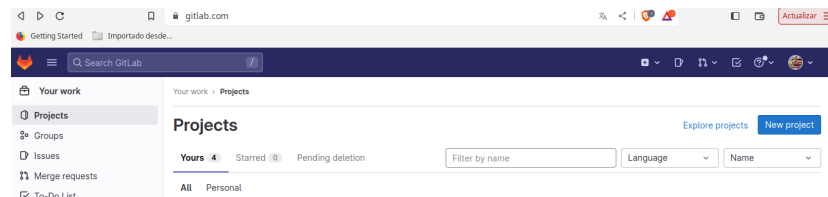
Ahora nos falta compartir con otros los cambios hechos:

- Push: actualiza el repositorio remoto desde el local (sólo cambios comiteados)
- Pull: actualiza el repositorio local desde el remoto

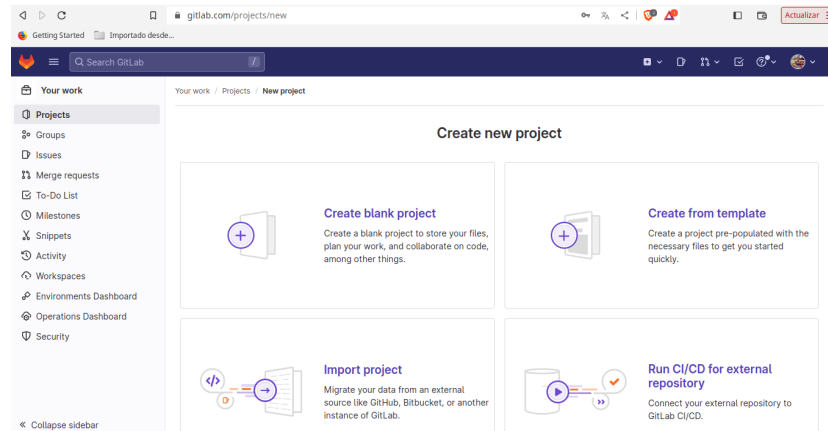
## Remotes



## 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](#).

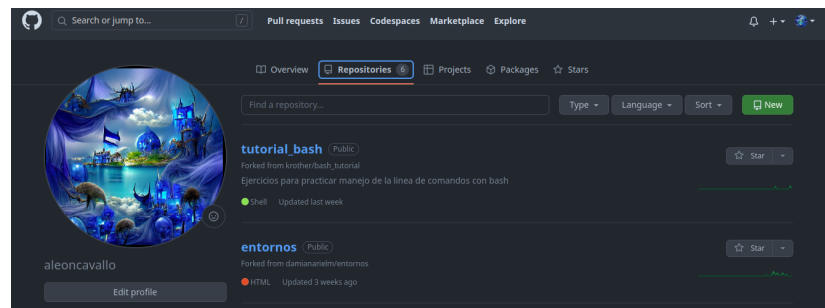
**Project deployment target (optional)**

**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



## 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 \***

/

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.

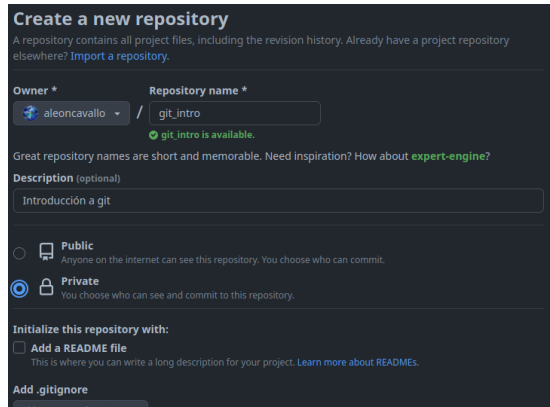
**Initialize this repository with:**

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

**Add .gitignore**



## 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 is available.](#)

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.

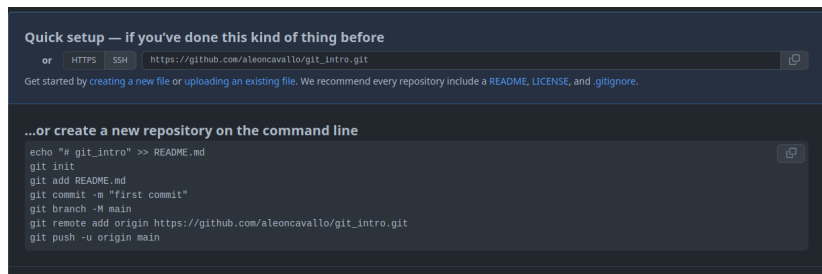
☒ Private  
You choose who can see and commit to this repository.

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

## Remotes



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

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 "# 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

Para el caso de la autenticación por ssh será necesario generar la clave siguiendo las instrucciones en:

Para simplificar nos manejaremos con GitLab, usando el link con https

Lo más básico ya está

## Revirtiendo cambios

- `git checkout <file>`: revierte cambios locales.
- `git reset`: vacía el staging area.
- `git reset <commit>`: vuelve al commit, sin modificar archivos.
- `git reset --hard \<commit>`: vuelve al commit, descartando todo.

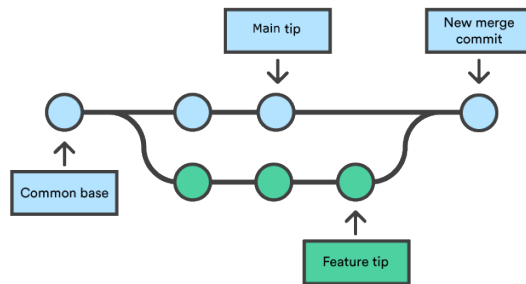
## Branches (Ramas)

- `git checkout <b>`: cambiar de branch
- master suele ser la rama principal

## Merge (Unir ramas)

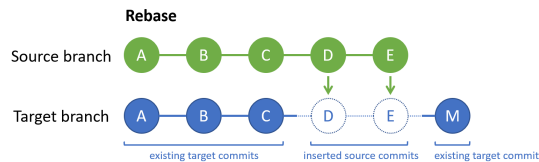
- En branch source `git merge <commit>`.

Pull tiene merge implícito, push sólo permite “fast-forwards”.



Posiblemente haya que corregir conflictos

## Rebase



- Generalmente sólo se hace en ramas privadas

## Bisect

`git bisect` ir dividiendo toda la pila de commits en dos partes,

## 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
- Sitio oficial de git: <https://git-scm.com/>
- Guía de git: <https://rogerdudler.github.io/git-guide/>

- Video Aprende GIT de HolaMudo <https://www.youtube.com/watch?v=VdGzPZ31ts8>
- <https://diegobersano.wordpress.com/2017/06/13/introduccion-a-git-repaso-a-los-conceptos-generales/>