

# Computación en Física

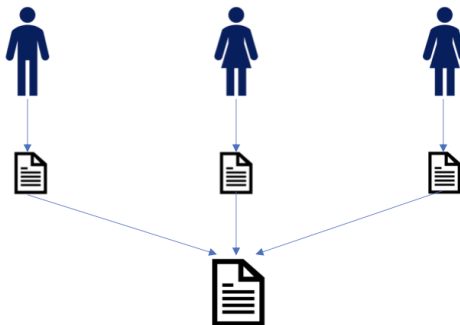
Harvey Rodriguez Gil

Universidad EIA

20 de Agosto de 2024

# ¿Por qué hablar de VCS?

Estamos en segundo semestre, y nos piden realizar un proyecto grupal. Dado que este proyecto es algo extenso, lo más recomendable es repartirse entre los compañeros del grupo diferentes partes del trabajo para finalmente unirlos. ¿Cómo logramos hacerlo?

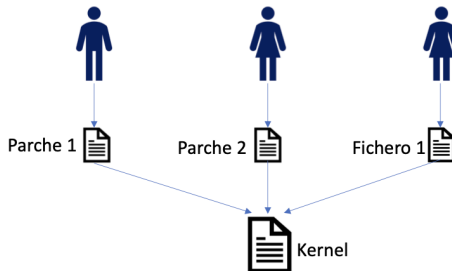


¿Trabajo final - versión 1?

# ¿Por qué hablar de VCS?

Este mismo problema se enfrentaron los ingenieros del Kernel de Linux:

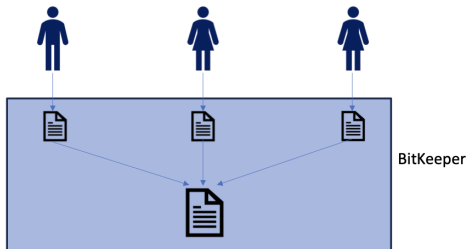
- ▶ Entre 1991-2002 los cambios en el software se realizaban a través de parches y archivos.
- ▶ Año 2002, BitKeeper es utilizado para el desarrollo del Kernel de Linux



# ¿Por qué hablar de VCS?

Este mismo problema se enfrentaron los ingenieros del Kernel de Linux:

- ▶ Richard Stallman expresó su preocupación por utilizar software propietario en el desarrollo de software libre.
- ▶ Alan Cox se negó al uso de esta herramienta argumentando su licencia propietaria.



# GIT

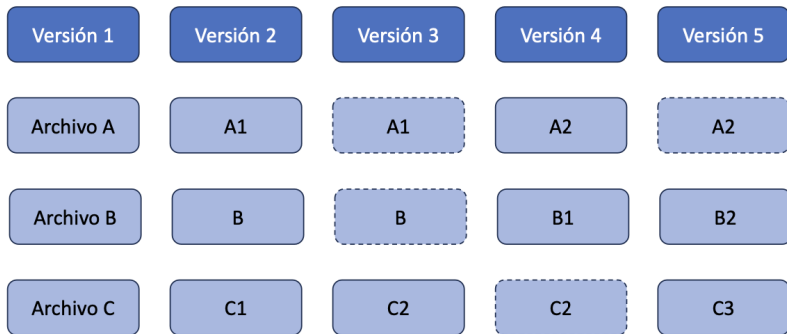
Algunos de los objetivos del nuevo sistema propuesto por Linus Torvalds en el año 2005 fueron los siguientes:

- ▶ Velocidad
- ▶ Diseño sencillo
- ▶ Gran soporte para desarrollo no lineal (miles de ramas paralelas)
- ▶ Completamente distribuido
- ▶ Capaz de manejar grandes proyectos (como el Kernel de Linux) eficientemente (velocidad y tamaño de los datos)

# ¿Cómo lo lograron?

Git maneja sus datos como conjunto de copias instantáneas de un sistema de archivos miniatura. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente toma una foto del aspecto de todos tus archivos en ese momento y guarda una referencia a esa copia instantánea. Para ser eficiente, si los archivos no se han modificado Git no almacena el archivo de nuevo, sino un enlace al archivo anterior idéntico que ya tiene almacenado. Git maneja sus datos como una secuencia de copias instantáneas.

# ¿Cómo lo lograron?



# ¿Cómo lo puedo hacer yo?

Para interactuar con git, primero deberemos realizar la instalación:

- ▶ Windows
  - ▶ <http://git-scm.com/download/win>
- ▶ Linux
  - ▶ `yum install git` (sistemas basados en RedHat)
  - ▶ `apt-get install git` (Sistemas basados en Debian)
- ▶ Mac OS
  - ▶ <http://git-scm.com/download/mac>



# Primeros Pasos

Configurar la identidad: lo primero que debemos hacer al instalar GIT es establecer el nombre de usuario y dirección de correo electrónico, ya que será utilizada esta información para ser introducida de manera inmutable en los cambios sobre los ficheros que modificamos:

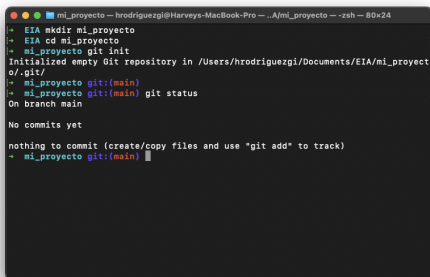
```
$ git config --global user.name "Harvey Rodriguez"  
$ git config --global user.email hrodriguez@gmail.com
```

Para validar que han quedado correctamente configuradas ejecutamos:

```
$ git config --list
```

# Creando proyectos

Ya con lo anterior configurado, podemos comenzar a trabajar con Git. Primero lo haremos de forma local, para esto crearemos una carpeta y ejecutaremos el comando `git init`. El comando `git status` nos ofrece una visión de nuestro proyecto.



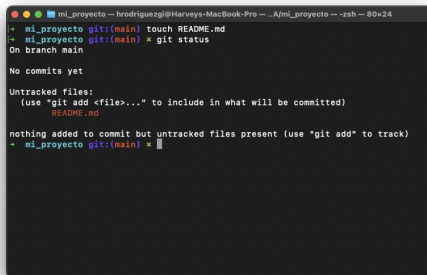
```
mi_proyecto — hrodriguezgi@Harveys-MacBook-Pro — ..A/mi_proyecto — zsh — 80x24
+ EIA mkdir mi_proyecto
+ EIA cd mi_proyecto
+ mi_proyecto git init
Initialized empty Git repository in /Users/hrodriguezgi/Documents/EIA/mi_proyecto/.git/
+ mi_proyecto git:(main)
+ mi_proyecto git:(main) git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
+ mi_proyecto git:(main)
```

# Creando un primer cambio

Cada vez que se realice un cambio como crear un archivo nuevo, modificar uno ya existente o incluso eliminarlo, se verá reflejado cuando ejecutamos el comando `git status`



```
mi_proyecto -- hrodriguezgi@Harveys-MacBook-Pro -- ../mi_proyecto -- zsh -- 80x24
+ mi_proyecto git:(main) touch README.md
+ mi_proyecto git:(main) * git status
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
+ mi_proyecto git:(main) *
```

# Guardando un primer cambio

Y si queremos persistir el cambio, entonces ejecutaremos el comando `git commit`

```
mi_proyecto -- hrodriguezgi@Harveys-MacBook-Pro -- ../mi_proyecto -- -zsh -- 80x24
+ mi_proyecto git:(main) * git add .
+ mi_proyecto git:(main) * git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md

+ mi_proyecto git:(main) * git commit -m "first change"
[main (root-commit) f78f317] first change
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
+ mi_proyecto git:(main) git status
On branch main
nothing to commit, working tree clean
+ mi_proyecto git:(main) █
```