

Shell y Git: siempre, en todas partes, al mismo tiempo

Herramientas viejas para problemas actuales

Laboratorio de Datos - Primer Cuatrimestre 2024 - FCEyN, UBA

Tabla de Contenidos

1. [Introducción](#)
2. [Nudo](#)
3. [\(Un\) Desenlace: Sistemas de Control de Versiones](#)
4. [Manos a la obra](#)
5. [Manos a la obra \(ajena\): los *remotos*](#)
6. [Juntándolo todo: Pull Requests](#)



Introducción

Acá se presenta el ponente, si no le da pudor. Más vale que la audiencia crea que uno sabe de lo que les habla. De tomar

Academia

- Unas clases que di
- Otra cosa que hice

Industria

- Mi primer trabajo
- Otro re piola que hice una vez
- Y últimamente, por acá

Academia

Industria

Academia

- Ayudante de 2da, Estadística
- Lic. en Economía
- Mg. en Estadística Matemática
- Docente de Aprendizaje Automático (ML)
- Ayudante de 1ra, a su servicio

Industria

Academia

- Ayudante de 2da, Estadística
- Lic. en Economía
- Mg. en Estadística Matemática
- Docente de Aprendizaje Automático (ML)
- Ayudante de 1ra, a su servicio

Industria

- Analista de Negocios
- Asesor en Análisis de Datos
- Científico de Datos
- Gerente de Ingeniería
- Ingeniero AA/ML autónomo

Perfil del Egresado de LCD

Tomado del [plan de estudios](#) de la carrera:

Perfil del Egresado de LCD

Tomado del [plan de estudios](#) de la carrera:

... cuenta con formación en un conjunto de disciplinas, enfocadas tanto en sus aspectos teóricos como **prácticos**, que le otorgan un profundo conocimiento en matemática y ciencias de la computación, fundamentalmente en los aspectos de modelado y **programación**, capacidad de abstracción, razonamiento lógico y pensamiento crítico.

Perfil del Egresado de LCD

Tomado del [plan de estudios](#) de la carrera:

... cuenta con formación en un conjunto de disciplinas, enfocadas tanto en sus aspectos teóricos como **prácticos**, que le otorgan un profundo conocimiento en matemática y ciencias de la computación, fundamentalmente en los aspectos de modelado y **programación**, capacidad de abstracción, razonamiento lógico y pensamiento crítico.

El/La licenciado/a en Ciencias de Datos se desempeñará en ámbitos **públicos y privados**, (...). A su vez, estará preparado para iniciar estudios académicos de posgrado y realizar investigaciones en distintas áreas de Matemática y Computación, así como en **grupos interdisciplinarios** que trabajen en áreas de la Física, la Química, la Biología...

Nudo



La $\begin{Bmatrix} \text{reproducibilidad} \\ \text{colaboración} \end{Bmatrix}$ es un problema peludo en el ámbito $\begin{Bmatrix} \text{científico} \\ \text{industrial} \end{Bmatrix}$

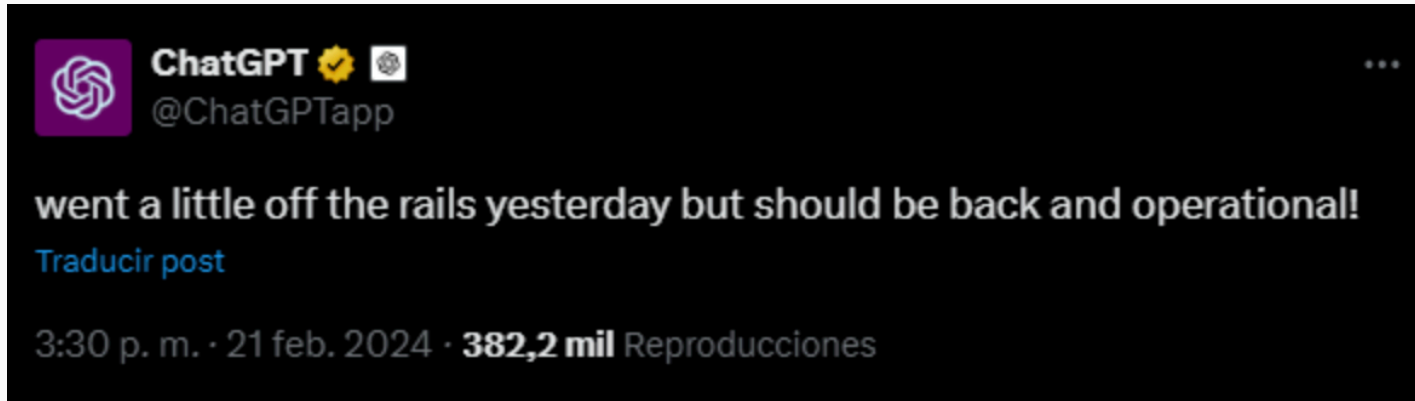
Reproducibilidad en el ámbito científico

[La experta en honestidad de Harvard fue acusada de inventar hallazgos en sus investigaciones](#)



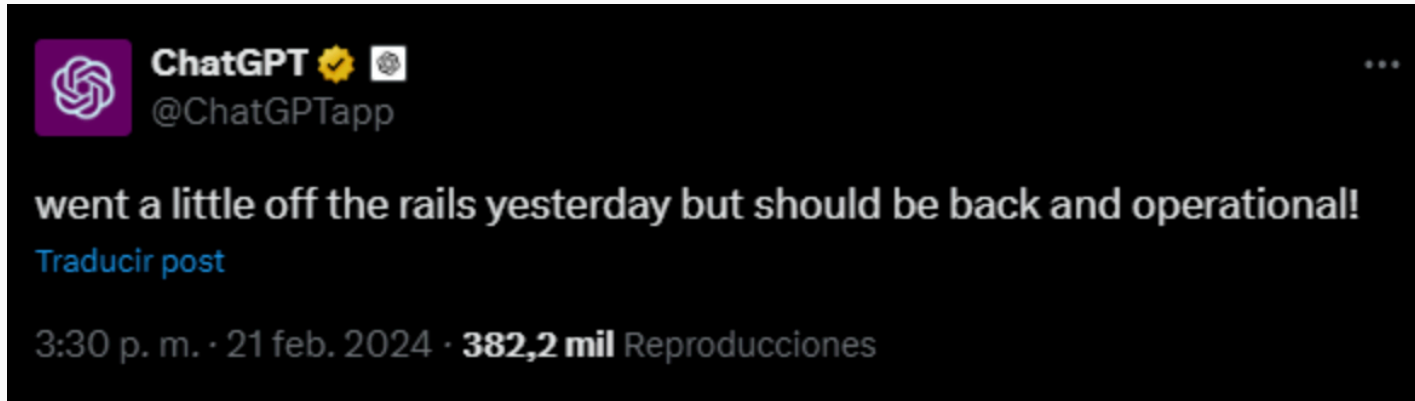
Reproducibilidad en el ámbito industrial

[ChatGPT se volvió loco y empezó a dar respuestas desubicadas](#)



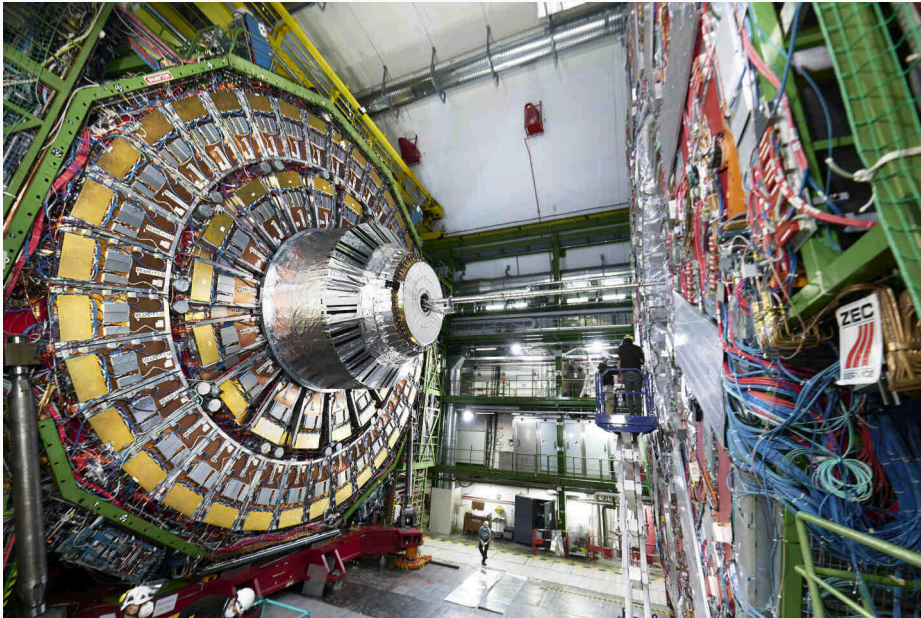
Reproducibilidad en el ámbito industrial

[ChatGPT se volvió loco y empezó a dar respuestas desubicadas](#)

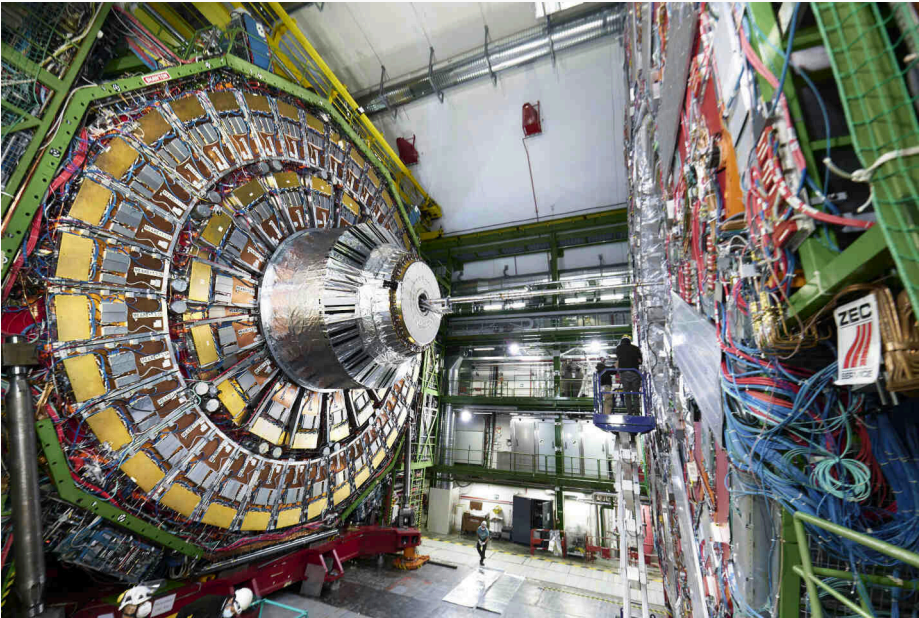


«El 20 de febrero de 2024, una optimización de la experiencia del usuario **introdujo un error** en la forma en que el modelo procesa el lenguaje. (...) Tras identificar la causa del incidente, **lanzamos una corrección** y confirmamos que el incidente estaba resuelto».

[¿Qué es el Solenoide Compacto de Muones del CERN?](#)



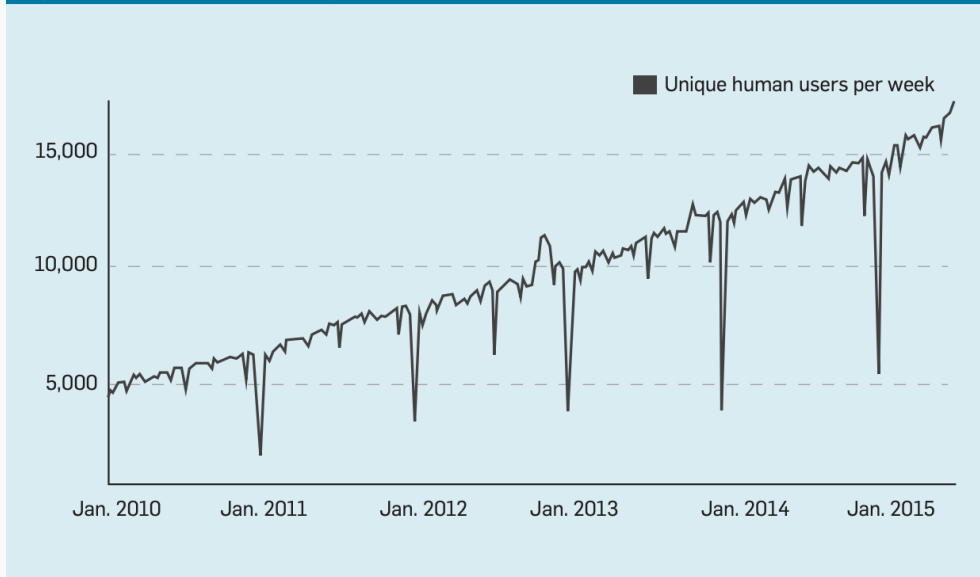
¿Qué es el Solenoide Compacto de Muones del CERN?



El CMS es un detector multipropósito construido alrededor de un solenoide superconductor gigante (...). La colaboración en el CMS que llevó al «descubrimiento» del bosón de Higgs es una de las más grandes colaboraciones científicas de la historia, alcanzando a 5.000 personas en 200 universidades e institutos de 50 países.

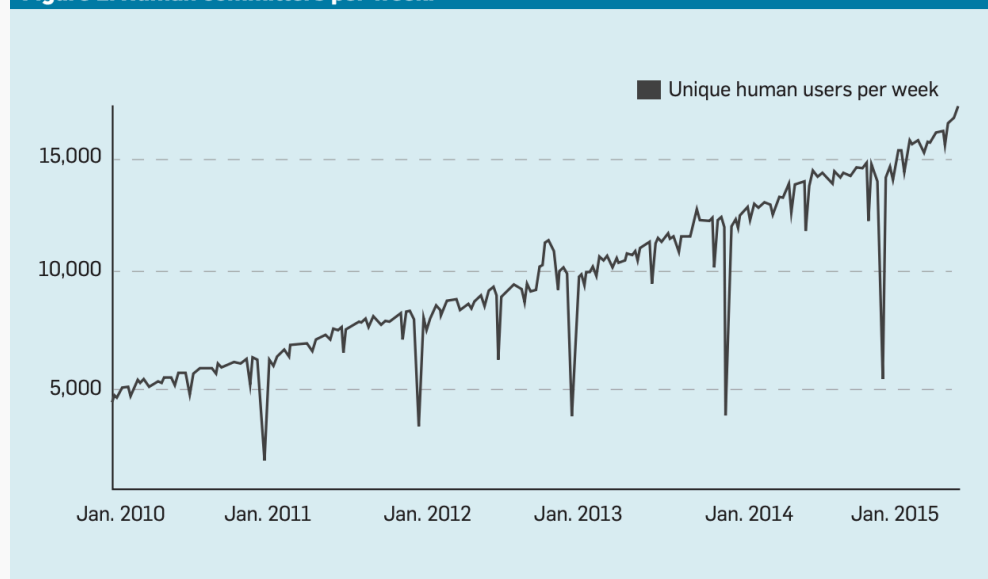
¿Por qué Google guarda billones de líneas de código en un único repositorio?

Figure 2. Human committers per week.



¿Por qué Google guarda billones de líneas de código en un único repositorio?

Figure 2. Human committers per week.



Google repository statistics, January 2015.

- Archivos Totales: 1.000 millones
- Archivos Fuente: 9 millones
- Líneas de código fuente: 2.000 millones
- Historial: 35 millones de «commits»
- Tamaño en disco: 86TB
- Commits por día laboral: 40.000

(Un) Desenlace: Sistemas de Control de Versiones

CVSes: hay un montón

CVSes: hay un montón

Sucesivas copias de un archivo



tp0-ldd-v12_final.docx

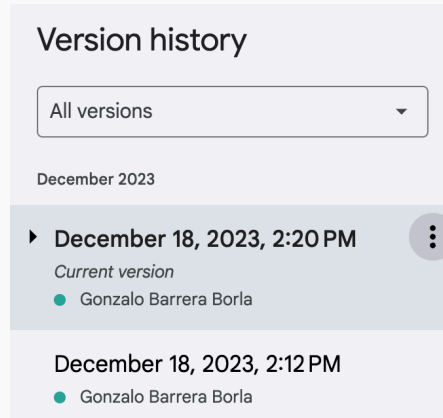
CVSes: hay un montón

Sucesivas copias de un archivo



tp0-ldd-v12_final.docx

Editores con historial de versiones (e.g. [Google Docs](#))



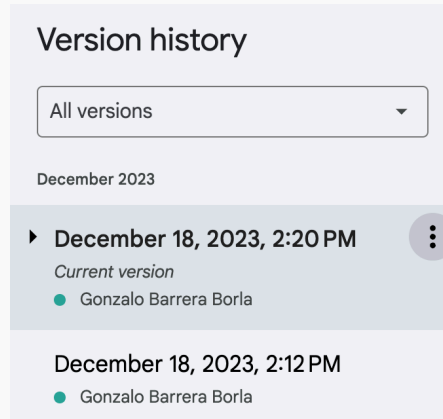
CVSes: hay un montón

Sucesivas copias de un archivo



tp0-ldd-v12_final.docx

Editores con historial de versiones (e.g. [Google Docs](#))



[DVCS: Sistemas de Control de Versiones Distribuidos](#)



Una breve historia de Git

El kernel de Linux es un proyecto de software de código abierto con un alcance bastante amplio. Durante la mayor parte del mantenimiento del kernel de Linux (1991-2002), los cambios en el software se realizaban a través de parches y archivos. En el 2002, el proyecto del kernel de Linux empezó a usar un DVCS propietario llamado BitKeeper.

En el 2005, la relación entre la comunidad que desarrollaba el kernel de Linux y la compañía que desarrollaba BitKeeper se vino abajo y la herramienta dejó de ser ofrecida de manera gratuita. Esto impulsó a la comunidad de desarrollo de Linux (y en particular a Linus Torvalds, el creador de Linux) a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron mientras usaban BitKeeper.

Desde su nacimiento en el 2005, Git ha evolucionado y madurado para ser fácil de usar y conservar sus características iniciales. Es tremendamente rápido, muy eficiente con grandes proyectos y tiene un increíble sistema de ramificación (branching) para desarrollo no lineal.

[Descargar](#)

[Pro Git - Libro Oficial en español](#)

Interludio: la Consola

Interludio: la Consola

(AKA «línea de comandos», «terminal», «shell» et cetera)

Manos a la obra



```
git --help
```

```
gonzalo@ohana ldd % git --help
```

git --help

```
gonzalo@ohana ldd % git --help
```

```
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>] (...)
```

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

'git help -a' and 'git help -g' list available subcommands and some concept guides. See 'git help <command>' or 'git help <concept>' to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

```
git --version
```

```
gonzalo@ohana ldd % git --version
```

```
git --version
```

```
gonzalo@ohana ldd % git --version
```

```
git version 2.39.2 (Apple Git-143)
```


git init

```
gonzalo@ohana ldd % FRUTA=melon  
gonzalo@ohana ldd % COLOR=purpura  
gonzalo@ohana ldd % PROFESION=judoca  
gonzalo@ohana ldd % mkdir $FRUTA-$COLOR-$PROFESION  
gonzalo@ohana ldd % cd !!$  
cd $FRUTA-$COLOR-$PROFESION
```

git init

```
gonzalo@ohana ldd % FRUTA=melon
```

```
gonzalo@ohana ldd % COLOR=purpura
```

```
gonzalo@ohana ldd % PROFESION=judoca
```

```
gonzalo@ohana ldd % mkdir $FRUTA-$COLOR-$PROFESION
```

```
gonzalo@ohana ldd % cd !!$
```

```
cd $FRUTA-$COLOR-$PROFESION
```

```
gonzalo@ohana melon-purpura-judoca % git init
```

```
Initialized empty Git repository in /Users/gonzalo/Git/ldd/melon-purpura-judoca/.git/
```

git config

```
--system: $(prefix)/etc/gitconfig # nivel sistema, rara vez usado  
--global: ~/.gitconfig # nivel usuario, `~` es "la home" del usuario  
--local: .git/config # nivel repositorio, relativo a su raíz
```

git config

```
--system: $(prefix)/etc/gitconfig # nivel sistema, rara vez usado  
--global: ~/.gitconfig # nivel usuario, `~` es "la home" del usuario  
--local: .git/config # nivel repositorio, relativo a su raíz
```

```
git config --local user.name "Gonzalo Barrera Borla"
```

```
git config --local user.email gonzalobb@gmail.com
```

```
git config --global core.editor vim
```

git config

```
--system: $(prefix)/etc/gitconfig # nivel sistema, rara vez usado  
--global: ~/.gitconfig # nivel usuario, `~` es "la home" del usuario  
--local: .git/config # nivel repositorio, relativo a su raíz
```

```
git config --local user.name "Gonzalo Barrera Borla"
```

```
git config --local user.email gonzalobb@gmail.com
```

```
git config --global core.editor vim
```

Si están en su PC personal, pueden usar --global;

git config

```
--system: $(prefix)/etc/gitconfig # nivel sistema, rara vez usado  
--global: ~/.gitconfig # nivel usuario, `~` es "la home" del usuario  
--local: .git/config # nivel repositorio, relativo a su raíz
```

```
git config --local user.name "Gonzalo Barrera Borla"
```

```
git config --local user.email gonzalobb@gmail.com
```

```
git config --global core.editor vim
```

Si están en su PC personal, pueden usar `--global`; si están en una computadora pública, con un usuario compartido -e.g., los labos-, usen `--local`.

Interludio: La poesía

Abra su editor de texto favorito y complete con dos versos la siguiente copla:

Abra su editor de texto favorito y complete con dos versos la siguiente copla:

Ayer pasé por tu aula
Me tiraste con un teorema
...

Abra su editor de texto favorito y complete con dos versos la siguiente copla:

Ayer pasé por tu aula
Me tiraste con un teorema
...

Guárdelo como `POEMA.txt` dentro del directorio del repo(sitorio).

git status

```
gonzalo@ohana melon-purpura-judoca % git status
```

git status

```
gonzalo@ohana melon-purpura-judoca % git status
```

```
On branch main
```

```
No commits yet
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
POEMA.txt
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

```
git add POEMA.txt
```

```
gonzalo@ohana melon-purpura-judoca % git add POEMA.txt
```

```
gonzalo@ohana melon-purpura-judoca % git status
```

```
git add POEMA.txt
```

```
gonzalo@ohana melon-purpura-judoca % git add POEMA.txt
```

```
gonzalo@ohana melon-purpura-judoca % git status
```

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: POEMA.txt

Interludio: Más poesía

Interludio: Más poesía

Abra `POEMA.txt` con su editor de texto favorito.

Agregue una segunda entrofa, de su propia invención, y guárdelo.

git status (cont.)

```
gonzalo@ohana melon-purpura-judoca % vim POEMA.txt # Aquí edité el archivo
```

```
gonzalo@ohana melon-purpura-judoca % git status
```

```
On branch main
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   POEMA.txt
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

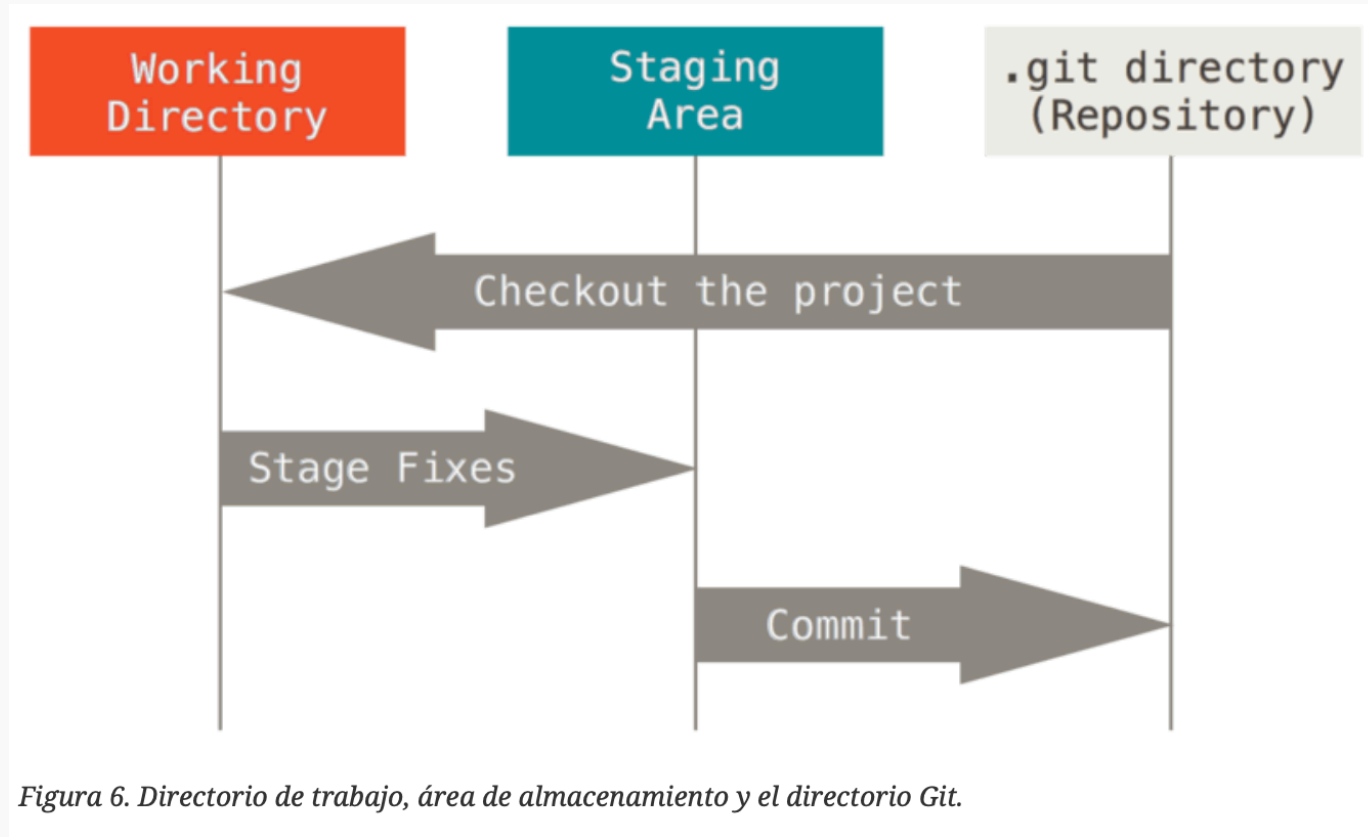
```
modified:   POEMA.txt
```

git diff

```
gonzalo@ohana melon-purpura-judoca % git diff POEMA.txt
diff --git a/POEMA.txt b/POEMA.txt
index e7316a6..7ff70d5 100644
--- a/POEMA.txt
+++ b/POEMA.txt
@@ -3,3 +3,8 @@ Me tiraste con un teorema
    La verdad, no había estudiado
    la demostración, qué pena!

+No es de vago, entiéndame
+es mi perro, el rengo Fefe
+que le encanta, yo lo sé
+devorar los pe de efe.
+
```

Archivos en git: sus tres estados y el ciclo de vida



Archivos en git: su ciclo de vida

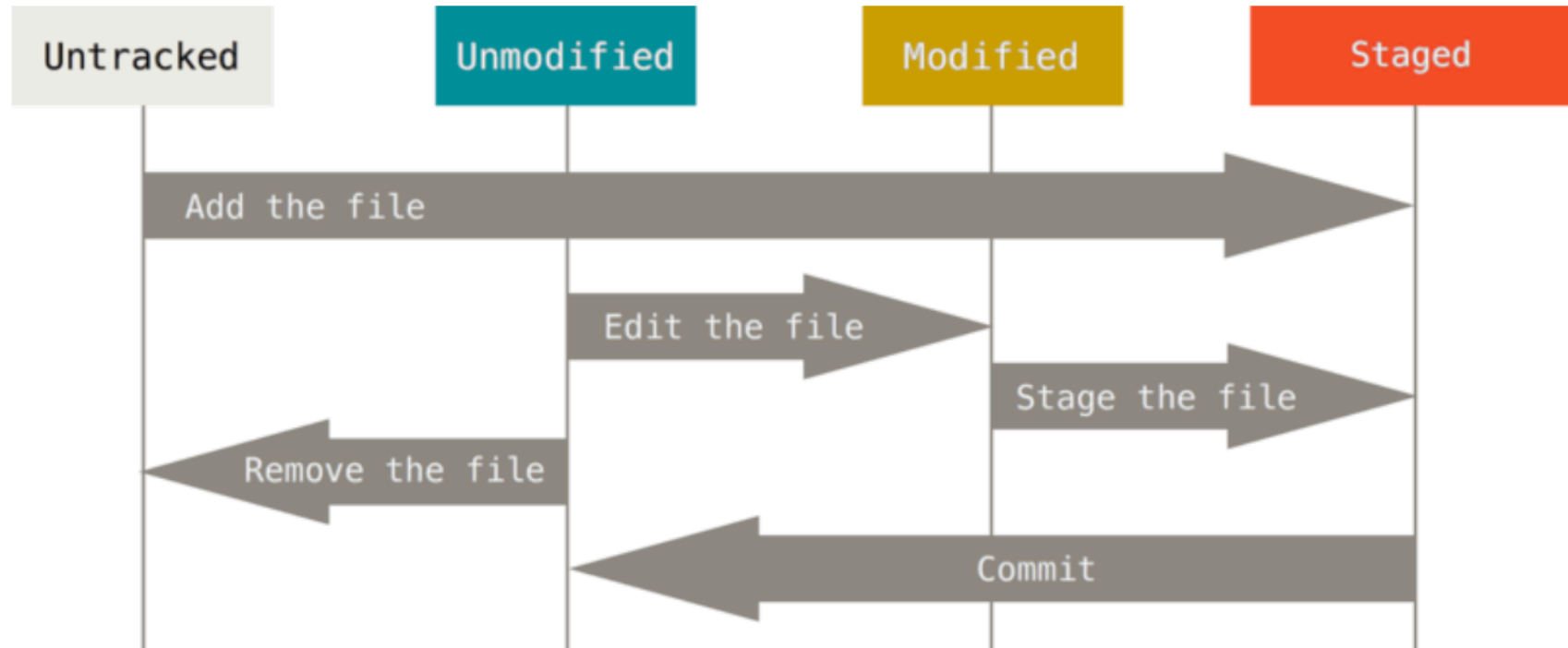


Figura 8. El ciclo de vida del estado de tus archivos.

```
git commit
```

```
gonzalo@ohana melon-purpura-judoca % git commit -m "Primer estrofa"
```

git commit

```
gonzalo@ohana melon-purpura-judoca % git commit -m "Primer estrofa"
```

```
[main (root-commit) 1ef4efe] Primer estrofa
```

```
1 file changed, 5 insertions(+)
```

```
create mode 100644 POEMA.txt
```

```
gonzalo@ohana melon-purpura-judoca % git status
```

```
On branch main
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

```
modified:   POEMA.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

```
gonzalo@ohana melon-purpura-judoca % git add POEMA.txt
```

```
gonzalo@ohana melon-purpura-judoca % git commit -m "Segunda estrofa"
```

```
gonzalo@ohana melon-purpura-judoca % git add POEMA.txt
gonzalo@ohana melon-purpura-judoca % git commit -m "Segunda estrofa"

[main 2b625f9] Segunda estrofa
 1 file changed, 5 insertions(+)
gonzalo@ohana melon-purpura-judoca % git status
On branch main
nothing to commit, working tree clean
```



```
gonzalo@ohana melon-purpura-judoca % git add POEMA.txt
gonzalo@ohana melon-purpura-judoca % git commit -m "Segunda estrofa"

[main 2b625f9] Segunda estrofa
 1 file changed, 5 insertions(+)
gonzalo@ohana melon-purpura-judoca % git status
On branch main
nothing to commit, working tree clean
```

Lean siempre los mensajes de la terminal!

```
git log
```

```
gonzalo@ohana melon-purpura-judoca % git log
```

```
gonzalo@ohana melon-purpura-judoca % git log
```

```
commit 2b625f9c5823f1b43d583e7366a6dfce9803af57 (HEAD -> main)
```

```
Author: Gonzalo Barrera Borla <gonzalobb@gmail.com>
```

```
Date: Tue Mar 26 07:20:59 2024 -0300
```

Segunda estrofa

```
commit 1ef4efe798fc46699f3399c6065e0fbe7b13d664
```

```
Author: Gonzalo Barrera Borla <gonzalobb@gmail.com>
```

```
Date: Tue Mar 26 07:20:39 2024 -0300
```

Primer estrofa


Manos a la obra (ajena): los *remotos*

Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar **repositorios remotos**. Los repositorios remotos son versiones de tu proyecto que están **hospedadas en Internet** o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura.

Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar **repositorios remotos**. Los repositorios remotos son versiones de tu proyecto que están **hospedadas en Internet** o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura.

Un *servidor de git* no es *git*, tanto como un *navegador web* no es la web en sí: se puede usar cualquiera! El más popular es [Github](#), pero existen otros como [Bitbucket](#) y [Gitlab](#).

Github Login




Sign in to GitHub

Username or email address

Password [Forgot password?](#)

Sign in

Or

 Sign in with a passkey

New to GitHub? [Create an account](#)

Nuestro repositorio sólo existe localmente por ahora. Démosle entidad en un *remoto*.

1. Botón verde «[+New](#)»

Nuestro repositorio sólo existe localmente por ahora. Démosle entidad en un *remoto*.

1. Botón verde «[+New](#)»
2. Configuren:
 - Exacto mismo nombre que al repo local
 - Descripción breve
 - Sin README.md
 - Sin .gitignore

Nuestro repositorio sólo existe localmente por ahora. Démosle entidad en un *remoto*.

1. Botón verde «[+New](#)»
2. Configuren:
 - Exacto mismo nombre que al repo local
 - Descripción breve
 - Sin README.md
 - Sin .gitignore
3. Click a «Create Repository»

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

```
git remote add origin https://github.com/capitantoto/melon-purpura-judoca.git  
git branch -M main  
git push -u origin main
```

Github - Nuevo Repositorio (cont.)

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

```
git remote add origin https://github.com/capitantoto/melon-purpura-judoca.git
git branch -M main
git push -u origin main
```

```
Username for 'https://github.com': capitantoto
```

```
Password for 'https://capitantoto@github.com':
```

```
remote: Support for password authentication was removed on August 13, 2021.
```

```
remote: Please see https://docs.github.com/get-started/getting-started-with-git/
about-remote-repositories#cloning-with-https-urls for information on currently
recommended modes of authentication.
```

```
fatal: Authentication failed for 'https://github.com/capitantoto/melon-purpura-
judoca.git/'
```

Github Personal Access Token

Los «Tokens de Acceso Personal» (PAT) son una alternativa al uso de contraseñas para la autenticación en GitHub cuando se usa la API de GitHub o la línea de comandos. El Personal access token está diseñado para acceder a los recursos de GitHub en tu nombre.

Github Personal Access Token

Los «Tokens de Acceso Personal» (PAT) son una alternativa al uso de contraseñas para la autenticación en GitHub cuando se usa la API de GitHub o la línea de comandos. El Personal access token está diseñado para acceder a los recursos de GitHub en tu nombre.

Fuente: [Administración de tokens de acceso personal](#)

Github Personal Access Token

Los «Tokens de Acceso Personal» (PAT) son una alternativa al uso de contraseñas para la autenticación en GitHub cuando se usa la API de GitHub o la línea de comandos. El Personal access token está diseñado para acceder a los recursos de GitHub en tu nombre.

Fuente: [Administración de tokens de acceso personal](#)

Advertencia: Trate los tokens de acceso como si fueran contraseñas, y guárdelos en su administrador de contraseñas predilecto.

Desde el desplegable del «avatar», elijan «Settings» > «Developer Settings» > «Personal Access Tokens» > «Generate New Token»

«Generate New Token»

... si eligen «Fine-grained»

- Scope («envergadura»): «All Repositories» (podrán ser más estrictos luego)
- Permisos «Read and Write» para
 - Contents
 - Issues
 - Pull Requests

... si eligen «Classic»

Tilden los permisos de «repo», que selecciona a toda la subsección correspondiente.

git push

```
gonzalo@ohana melon-purpura-judoca % git push -u origin main  
Username for 'https://github.com': capitantoto  
Password for 'https://capitantoto@github.com':
```

git push

```
gonzalo@ohana melon-purpura-judoca % git push -u origin main
Username for 'https://github.com': capitantoto
Password for 'https://capitantoto@github.com':

Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 613 bytes | 613.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/capitantoto/melon-purpura-judoca.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Ahora, ingresen otra vez a

[https://github.com/](https://github.com/{usuario}/{repositorio}){usuario}/{repositorio}

Ahora, ingresen otra vez a

[https://github.com/](https://github.com/{usuario}/{repositorio}){usuario}/{repositorio}

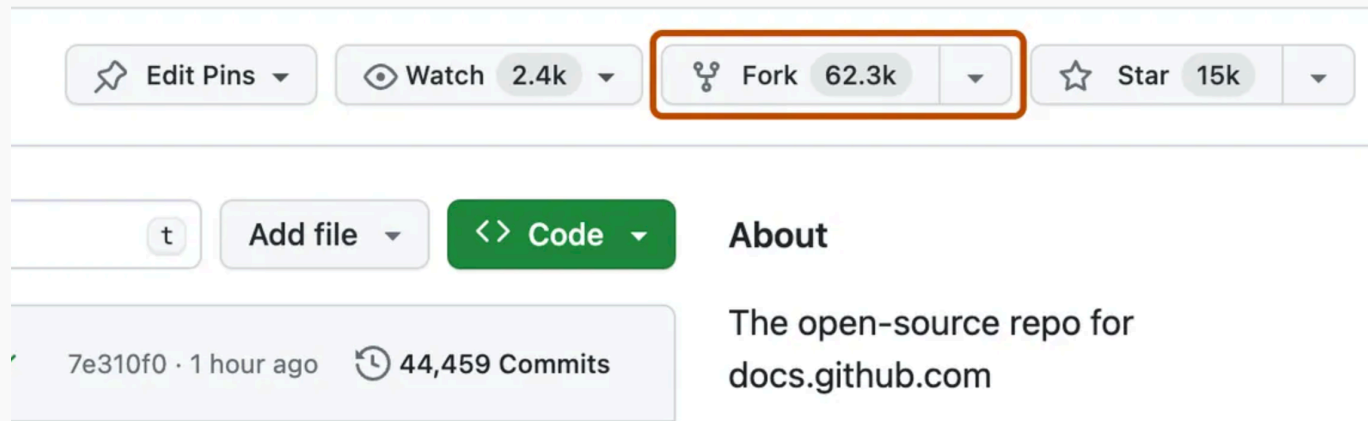
Voilà!

Tarea para el hogar: [Autenticación con clave SSH](#)

Tenedores! («forks»)

Una bifurcación («fork») es un nuevo repositorio que comparte la configuración de visibilidad y código con el repositorio acendente («upstream») original.

[Bifurcar un repositorio](#)



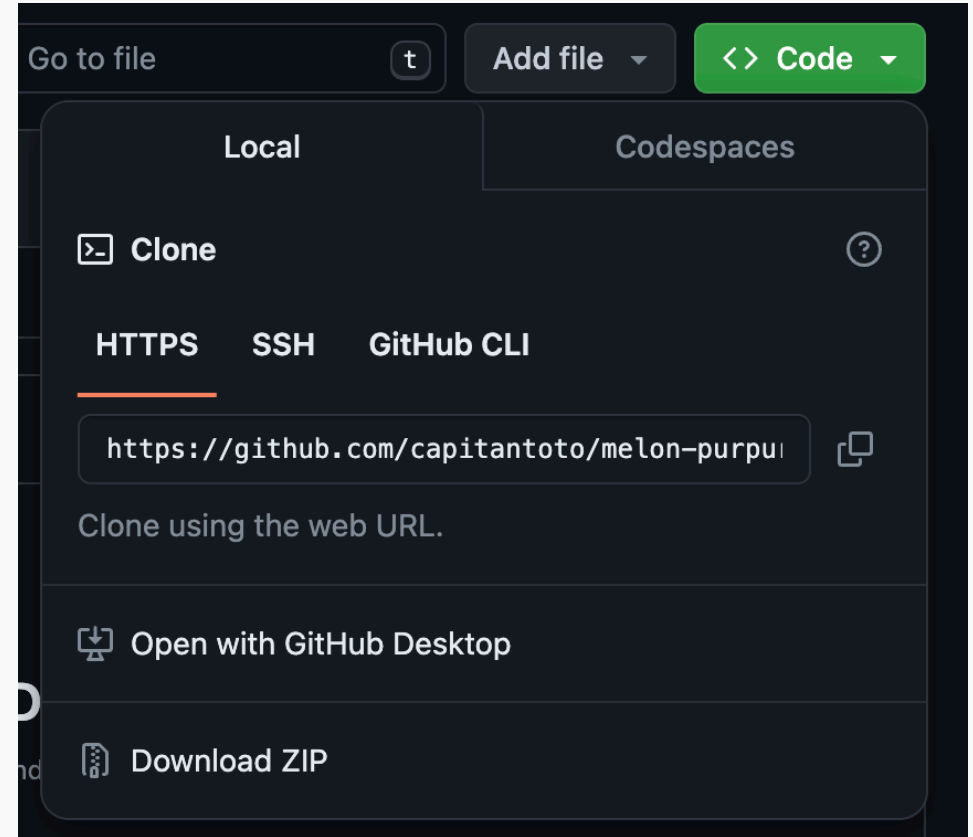
Ordenando el trabajo colaborativo

Pónganse en parejas. Pídanle a su *partenaire* su nombre de usuario y repositorio, y *forkéenlo* a su cuenta.

Ordenando el trabajo colaborativo

Pónganse en parejas. Pídanle a su *partenaire* su nombre de usuario y repositorio, y *forkéenlo* a su cuenta.

Abran el sitio web del *fork*, y busquen el botón verde [`<> Code`]

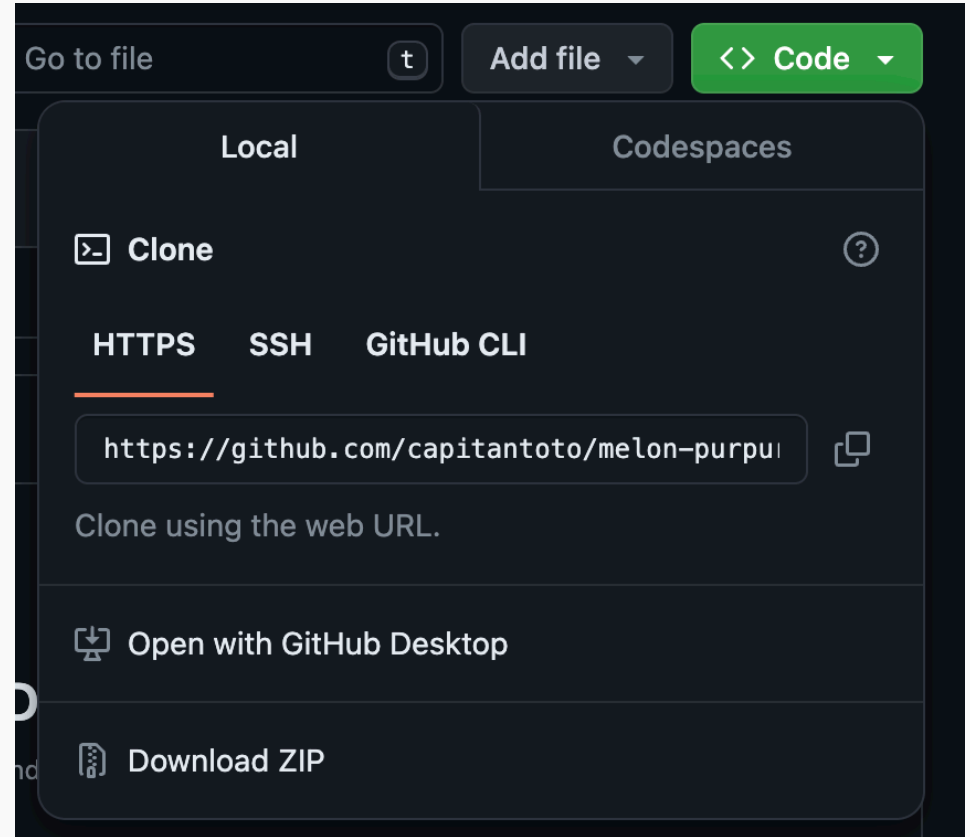


Ordenando el trabajo colaborativo

Pónganse en parejas. Pídanle a su *partenaire* su nombre de usuario y repositorio, y *forkéenlo* a su cuenta.

Abran el sitio web del *fork*, y busquen el botón verde [`<> Code`]

Elijan «HTTPS» (o «SSH» si ya lo saben usar) como método de clonado, y copien la URL visible justo debajo.



¿Y si en lugar de *crear* un repositorio, queremos trabajar con uno ya existente?

¿Y si en lugar de *crear* un repositorio, queremos trabajar con uno ya existente?

```
gonzalo@ohana Git % git clone https://github.com/capitantoto/sandia-verde-  
maquinista.git
```

¿Y si en lugar de *crear* un repositorio, queremos trabajar con uno ya existente?

```
gonzalo@ohana Git % git clone https://github.com/capitantoto/sandia-verde-  
maquinista.git
```

```
Cloning into 'sandia-verde-maquinista'...
```

```
remote: Enumerating objects: 3, done.
```

```
remote: Counting objects: 100% (3/3), done.
```

```
remote: Compressing objects: 100% (2/2), done.
```

```
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

```
Receiving objects: 100% (3/3), done.
```

Ramas! («branches»)

Una **rama** (*branch*) es «otra versión posible» del repositorio, que *diverge* del tronco (u otras ramas) en un cierto **commit**.

Ramas! («branches»)

Una **rama** (*branch*) es «otra versión posible» del repositorio, que *diverge* del tronco (u otras ramas) en un cierto **commit**.

Cuando colaboramos, es de mala educación usar directamente

- ramas compartidas (`main`, `dev`, `etc.`),
- ramas «personales» (`un-feature-copado`, a cargo de Cosme Fulanito, que no soy yo)

Ramas! («branches»)

Una **rama** (*branch*) es «otra versión posible» del repositorio, que *diverge* del tronco (u otras ramas) en un cierto **commit**.

Cuando colaboramos, es de mala educación usar directamente

- ramas compartidas (`main`, `dev`, etc.),
- ramas «personales» (`un-feature-copado`, a cargo de Cosme Fulanito, que no soy yo)

```
gonzalo@ohana sandia-verde-maquinista % git branch
```

```
* main
```

```
gonzalo@ohana sandia-verde-maquinista % git branch modesta-mejora
```

```
gonzalo@ohana sandia-verde-maquinista % git checkout modesta-mejora
```

```
Switched to branch 'modesta-mejora'
```

```
gonzalo@ohana sandia-verde-maquinista % git checkout -b inenarrable-mejora
```

```
Switched to a new branch 'inenarrable-mejora'
```

Ramas! («branches»)

```
gonzalo@ohana sandia-verde-maquinista % git branch
* inenarrable-mejora
  main
  modesta-mejora
```


Ramas! («branches»)

```
gonzalo@ohana sandia-verde-maquinista % git branch
```

```
* inenarrable-mejora
```

```
main
```

```
modesta-mejora
```

```
gonzalo@ohana sandia-verde-maquinista % git add POEMA.txt
```

```
gonzalo@ohana sandia-verde-maquinista % git commit --message "Agrega tercera estrofa"
```

```
[inenarrable-mejora a75fcb5] Agrega tercera estrofa
```

```
1 file changed, 5 insertions(+)
```

```
gonzalo@ohana sandia-verde-maquinista % git push origin inenarrable-mejora
```

Juntándolo todo: Pull Requests

Pedidos de tire (AKA: *pull requests*)

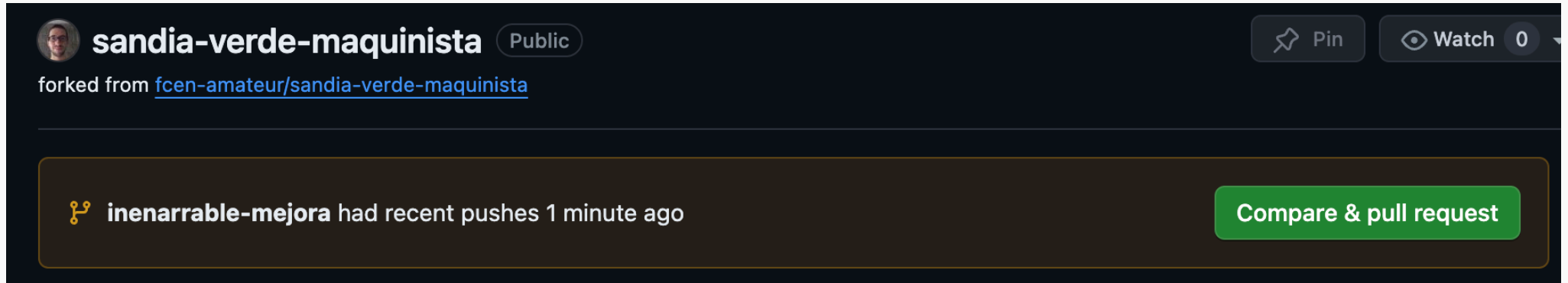
```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 371 bytes | 371.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'inenarrable-mejora' on GitHub by visiting:
remote:      https://github.com/capitantoto/sandia-verde-maquinista/pull/new/
inenarrable-mejora
remote:
To https://github.com/capitantoto/sandia-verde-maquinista.git
 * [new branch]      inenarrable-mejora -> inenarrable-mejora
```

Pedidos de tire (AKA: *pull requests*)

Vayan nuevamente a Github, y abran el repositorio que forkearon. Van a ver un aviso similar al siguiente:

Pedidos de tire (AKA: *pull requests*)

Vayan nuevamente a Github, y abran el repositorio que forkearon. Van a ver un aviso similar al siguiente:



Cliqueen en «Compare & pull request» para «crear un PR». Exploren el sitio que se abre.

Pedidos de tire (AKA: *pull requests*)

A saber:

- Se puede comparar entre «mis repos» y «across forks» (a través de bifurcaciones). Aquí, elijan «across forks» para crear un PR contra el repo del colega como base.

Pedidos de tire (AKA: *pull requests*)

A saber:

- Se puede comparar entre «mis repos» y «across forks» (a través de bifurcaciones). Aquí, elijan «across forks» para crear un PR contra el repo del colega como base.
- Elijan `main` como rama contra la cual «mergear» su «rama tónica» (*feature branch*).

Pedidos de tire (AKA: *pull requests*)

A saber:

- Se puede comparar entre «mis repos» y «across forks» (a través de bifurcaciones). Aquí, elijan «across forks» para crear un PR contra el repo del colega como base.
- Elijan `main` como rama contra la cual «mergear» su «rama tónica» (*feature branch*).
- Como Revisor («*Reviewer*»): elijan al dueño del repositorio original si es posible (deben ser «colaboradores» del repo original. Si no, dejen que el autor original del repo se auto-asigne más tarde.

Pedidos de tire (AKA: *pull requests*)

A saber:

- Se puede comparar entre «mis repos» y «across forks» (a través de bifurcaciones). Aquí, elijan «across forks» para crear un PR contra el repo del colega como base.
- Elijan `main` como rama contra la cual «mergear» su «rama tónica» (*feature branch*).
- Como Revisor («*Reviewer*»): elijan al dueño del repositorio original si es posible (deben ser «colaboradores» del repo original. Si no, dejen que el autor original del repo se auto-asigne más tarde.
- Denle un título breve (si es que el automático no sirve), y en la descripción, justifiquen el por qué de los cambios propuestos.

Pedidos de tire (AKA: *pull requests*)

A saber:

- Se puede comparar entre «mis repos» y «across forks» (a través de bifurcaciones). Aquí, elijan «across forks» para crear un PR contra el repo del colega como base.
- Elijan `main` como rama contra la cual «mergear» su «rama tónica» (*feature branch*).
- Como Revisor («*Reviewer*»): elijan al dueño del repositorio original si es posible (deben ser «colaboradores» del repo original. Si no, dejen que el autor original del repo se auto-asigne más tarde.
- Denle un título breve (si es que el automático no sirve), y en la descripción, justifiquen el por qué de los cambios propuestos.
- Observen que el PR (Pull Request) se puede crear como borrador (*draft*).

Pedidos de tire (AKA: *pull requests*)

A saber:

- Se puede comparar entre «mis repos» y «across forks» (a través de bifurcaciones). Aquí, elijan «across forks» para crear un PR contra el repo del colega como base.
- Elijan `main` como rama contra la cual «mergear» su «rama tónica» (*feature branch*).
- Como Revisor («*Reviewer*»): elijan al dueño del repositorio original si es posible (deben ser «colaboradores» del repo original. Si no, dejen que el autor original del repo se auto-asigne más tarde.
- Denle un título breve (si es que el automático no sirve), y en la descripción, justifiquen el por qué de los cambios propuestos.
- Observen que el PR (Pull Request) se puede crear como borrador (*draft*).
- **Creerlo!**

Vayan a su repositorio original, <https://github.com/{usuario}/{repo}/pulls>

Vayan a su repositorio original, <https://github.com/{usuario}/{repo}/pulls>

¿Ven el PR de su compañero que los tiene como revisor? Busquen la opción para comenzar la revisión si ya los asignaron. Si no, asigne como revisor a ud. mismo, y recargue la página; ahora debería ver el botón (verde) de «Start Review».

Vayan a su repositorio original, <https://github.com/{usuario}/{repo}/pulls>

¿Ven el PR de su compañero que los tiene como revisor? Busquen la opción para comenzar la revisión si ya los asignaron. Si no, asigne como revisor a ud. mismo, y recargue la página; ahora debería ver el botón (verde) de «Start Review».

Ahora, pueden insertar comentarios en cualquier lugar del código, tantos como quieran, y hasta hacer directamente sugerencias de cambios a «commitear».

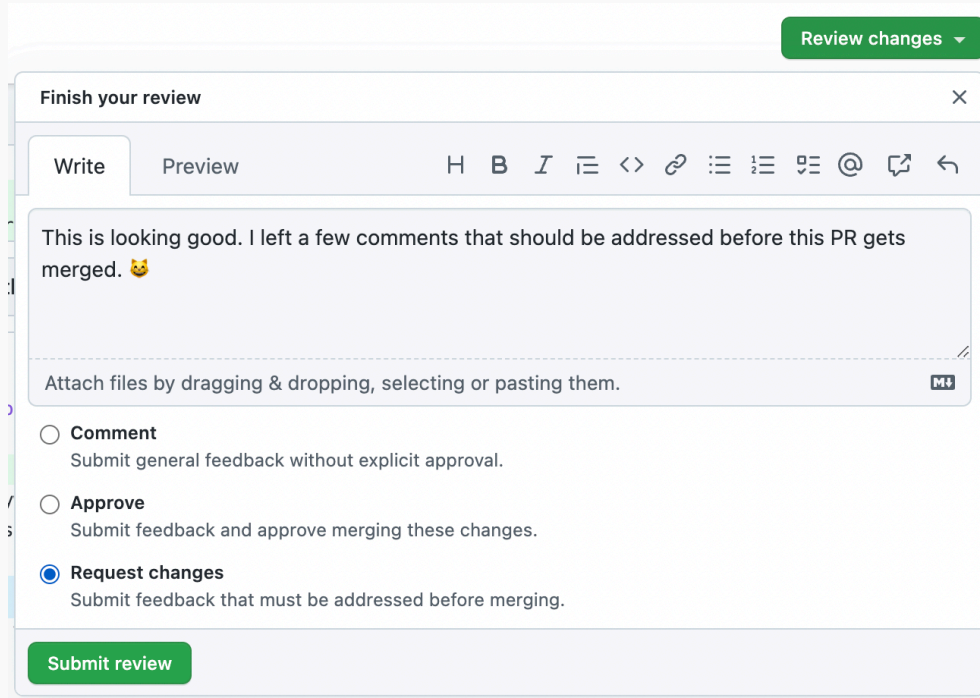
Vayan a su repositorio original, <https://github.com/{usuario}/{repo}/pulls>

¿Ven el PR de su compañero que los tiene como revisor? Busquen la opción para comenzar la revisión si ya los asignaron. Si no, asigne como revisor a ud. mismo, y recargue la página; ahora debería ver el botón (verde) de «Start Review».

Ahora, pueden insertar comentarios en cualquier lugar del código, tantos como quieran, y hasta hacer directamente sugerencias de cambios a «commitear».

Finalmente, cierren la revisión con una evaluación global, y elijan uno de los tres estados: «Comment», «Approve», «Request Changes»

Revisiones de Código



The screenshot shows a 'Finish your review' dialog box with a close button (X) in the top right corner. At the top right of the dialog is a green button labeled 'Review changes' with a dropdown arrow. Below this is a tabbed interface with 'Write' and 'Preview' tabs. The 'Write' tab is active, showing a rich text editor with a toolbar containing icons for bold (B), italic (I), text color (A), link (link icon), list (bulleted), list (numbered), quote (quote icon), mention (@), link (external), and undo (curved arrow). The text area contains the message: 'This is looking good. I left a few comments that should be addressed before this PR gets merged. 🙄'. Below the text area is a dashed line and the text 'Attach files by dragging & dropping, selecting or pasting them.' with a small 'M+' icon. At the bottom, there are three radio button options: 'Comment' (Submit general feedback without explicit approval.), 'Approve' (Submit feedback and approve merging these changes.), and 'Request changes' (selected, Submit feedback that must be addressed before merging.). A green 'Submit review' button is at the bottom left.

- «**Comentar**»: Dar una devolución general sin una aprobación explícita.
- «**Aprobar**»: Dar una evolución y aprobar la fusión («merging») de los cambios.
- «**Requerir Cambios**»: Proveer feedback que se debe incorporar al PR antes de *mergear*.

Para el revisor

Para el revisor

- **La aprobación es solidaria**: no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.

Para el revisor

- **La aprobación es solidaria:** no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.

Para el revisor

- **La aprobación es solidaria:** no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.
- GitHub también es una red social, seamos civilizados: *sean buena gente*.

Para el revisor

- La aprobación es solidaria: no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.
- GitHub también es una red social, seamos civilizados: *sean buena gente*.
- Al pedir cambios que requerirán nuevas revisiones, **no colgarla**.

Revisiones de Código, algunos tips

Para el revisor

- **La aprobación es solidaria**: no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.
- GitHub también es una red social, seamos civilizados: *sean buena gente*.
- Al pedir cambios que requerirán nuevas revisiones, **no colgarla**.

Para el autor

Revisiones de Código, algunos tips

Para el revisor

- **La aprobación es solidaria:** no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.
- GitHub también es una red social, seamos civilizados: *sean buena gente*.
- Al pedir cambios que requerirán nuevas revisiones, **no colgarla**.

Para el autor

- Antes de pedir una revisión, **abogado del diablo**: revisen ustedes mismos a fondo.

Revisiones de Código, algunos tips

Para el revisor

- **La aprobación es solidaria:** no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.
- GitHub también es una red social, seamos civilizados: *sean buena gente*.
- Al pedir cambios que requerirán nuevas revisiones, **no colgarla**.

Para el autor

- Antes de pedir una revisión, **abogado del diablo:** revisen ustedes mismos a fondo.
- Sean corteses: usen la descripción del PR, o comentarios sobre el mismo, para dar contexto al revisor.

Revisiones de Código, algunos tips

Para el revisor

- **La aprobación es solidaria:** no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.
- GitHub también es una red social, seamos civilizados: *sean buena gente*.
- Al pedir cambios que requerirán nuevas revisiones, **no colgarla**.

Para el autor

- Antes de pedir una revisión, **abogado del diablo**: revisen ustedes mismos a fondo.
- Sean corteses: usen la descripción del PR, o comentarios sobre el mismo, para dar contexto al revisor.
- Expliquen «por qué» antes que «cmo»: código bien escrito se documenta solo, pero los pensamientos son privados.

Revisiones de Código, algunos tips

Para el revisor

- **La aprobación es solidaria**: no vale aprobar sin asumir la responsabilidad del cambio conjuntamente con el autor.
- Si hay problemas serios, eviten marcar nimiedades. Mejor *hablar personalmente* para destrabar la situación.
- GitHub también es una red social, seamos civilizados: *sean buena gente*.
- Al pedir cambios que requerirán nuevas revisiones, **no colgarla**.

Para el autor

- Antes de pedir una revisión, **abogado del diablo**: revisen ustedes mismos a fondo.
- Sean corteses: usen la descripción del PR, o comentarios sobre el mismo, para dar contexto al revisor.
- Expliquen «por qué» antes que «cmo»: código bien escrito se documenta solo, pero los pensamientos son privados.
- **Tengan piel de rinoceronte**: están evaluando *su trabajo*, no su carácter.

Gracias!