Introducción a la Ciencia de Datos
2023

Micro historia de git

¿Quién lo creó?

Linus Torvalds.

¿Qué significa?

> *"git" can mean anything, depending on your mood.*

> *e.g: "Global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.*

¿Para qué sirve?

> Es ~~LA~~ una herramienta de Control de Versiones (hay otras: CVS, Subversion, Mercurial, Bazaar)

¿Qué significa eso?

> Permite ver y controlar todos los cambios que se le hacen a un proyecto

> Saber quién escribió cada línea

> Combinar (mergear) archivos en conflicto

> Colaborar a cualquier escala: desde proyectos personales a miles de desarrolladores

> Veamos ejemplos! Linux, paperswithcode, introCD

git vs. GitHub y GitLab

- **git** es un programa que corre en nuestra computadora
  - Puede usarse sin conectarse jamás a internet.
  - Es descentralizado: todos los usuarios tienen una copia de todo el repo.
  - Cada usuario puede subir el repositorio a tantos *remotes* como quiera (incluyendo el cero)

- **GitHub** es una plataforma en la nube para colaborar sobre repositorios de git
  - Actualmente es la más usada en proyectos open source
  - ¿Es la red social de los desarrolladores? Tal vez.

- **GitLab** es un competidor de GitHub

Log (o historia) de commits

$ git log

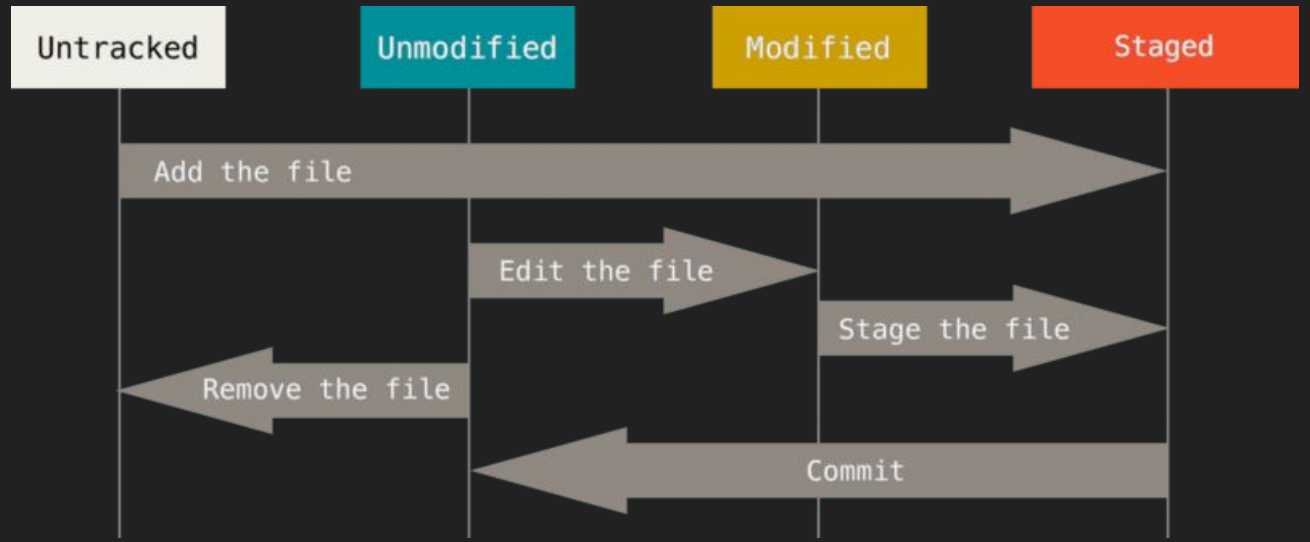
```
commit a42981bfdd711b231a325d47d20f6cea2a96cab6 (HEAD -> master, origin/master, origin/HEAD)
Author: Braulio Ríos <braulioriosf@gmail.com>
Date:   Thu May 4 14:44:14 2023 -0300

    Actualizados links en la Tarea 1

commit 21c9fd6616767f01902d0fd1ed12c4948523d043
Author: Braulio Ríos <braulioriosf@gmail.com>
Date:   Thu May 4 14:39:49 2023 -0300

    Added Tarea_1 (2023)

commit 7437f096ff5e6573b4b54c8f357ac50f440886f5
Author: Braulio Ríos <braulioriosf@gmail.com>
Date:   Thu May 4 14:39:08 2023 -0300

    README and .gitignore

commit 3fc090d80e7906aa750ea42dce9681afcbce623e
Author: Lorena Etcheverry <lorenae@fing.edu.uy>
Date:   Mon Apr 25 14:45:37 2022 -0300

    Cambio donde dice versión  3.7 a 3.X

commit bf0e4a681c907a72546e28b5894407eb511630ac
Author: Guillermo Moncecchi <gmonce@fing.edu.uy>
Date:   Sun Mar 29 18:20:56 2020 -0300

    Upload New File
```

$ git show 3fc090d

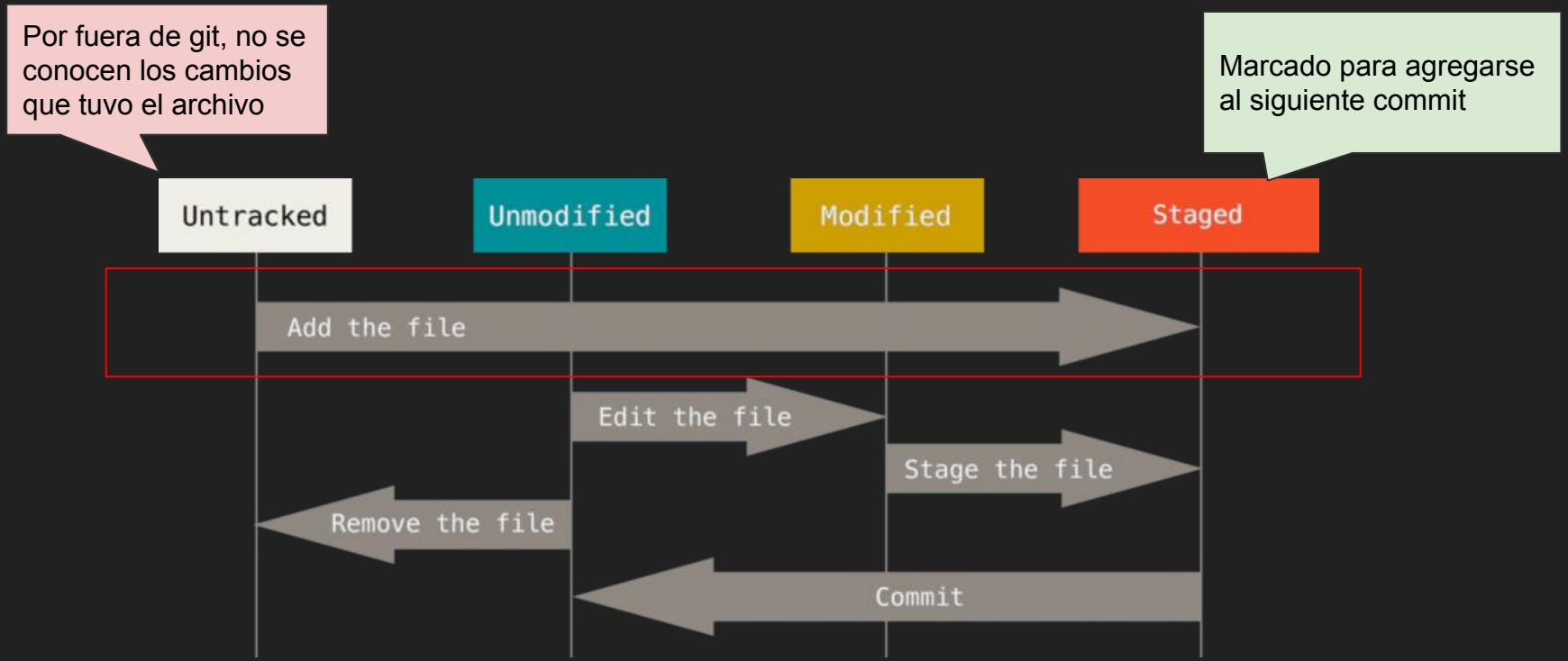
```
commit 3fc090d80e7906aa750ea42dce9681afcbce623e
Author: Lorena Etcheverry <lorenae@fing.edu.uy>
Date:   Mon Apr 25 14:45:37 2022 -0300

    Cambio donde dice versión  3.7 a 3.X

diff --git a/Ambiente.md b/Ambiente.md
index 73f6f90..bb1d747 100644
--- a/Ambiente.md
+++ b/Ambiente.md
@@ -14,7 +14,7 @@ Anaconda es una distribución de Python y R pensada especialmente para
 desarroll

 Anaconda y sus paquetes están disponible para todas las plataformas (Linux, Windows, M
ac OS X).

-Para instalar Anaconda, descárgue la versión para Python 3.7 correspondiente a su Sist
ema Operativo [aquí](https://www.anaconda.com/distribution/#download-section).  Una vez
 instalado, hay dos formas principales de acceder:
+Para instalar Anaconda, descárgue la versión para Python 3.X correspondiente a su Sist
ema Operativo [aquí](https://www.anaconda.com/distribution/#download-section).  Una vez
 instalado, hay dos formas principales de acceder:

 - Vía la línea de comandos, utilizando conda
 - Vía un navegador gráfico llamado Anaconda Navigator.
@@ -65,4 +65,4 @@ Un aspecto interesante de los Notebooks de Jupyter es que github los
visualiza a

 Para esto, utilice Jupyter Lab, y cree un nuevo documento. La interfaz deberá ofrecerl
e la opción de crear un notebook de R o de Python. En el mismo directorio que este docu
mento están los Notebooks [HelloR.ipynb](HelloR.ipynb) y [HelloPython.ipynb](HelloPytho
n.ipynb). Intente abrirlos y ejecutar el código. Si no tiene errores, usted tiene segur
amente el ambiente configurado para empezar a trabajar.

-**Bienvenidos al curso de Introducción a la Ciencia de Datos**
\ No newline at end of file
+**Bienvenidos al curso de Introducción a la Ciencia de Datos**
(END)
```

Ciclo de vida de un archivo

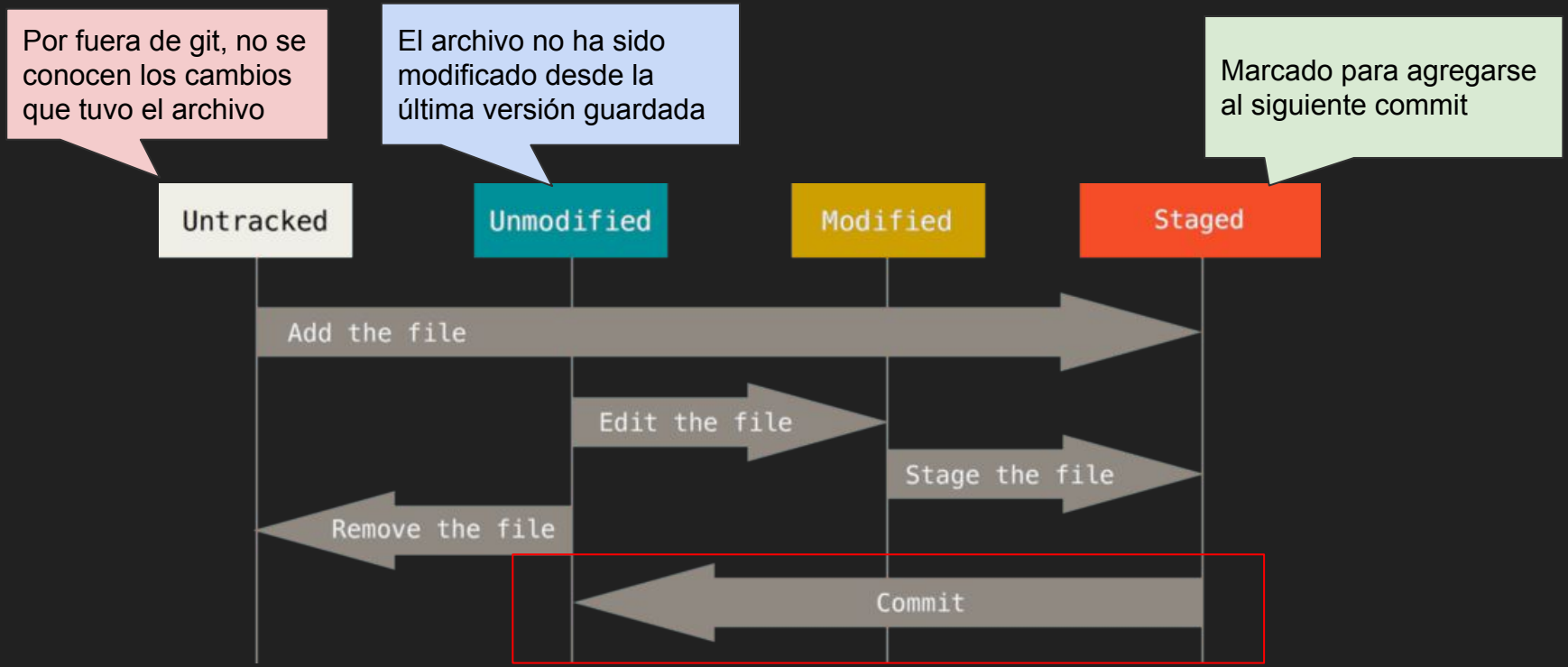
Por fuera de git, no se conocen los cambios que tuvo el archivo



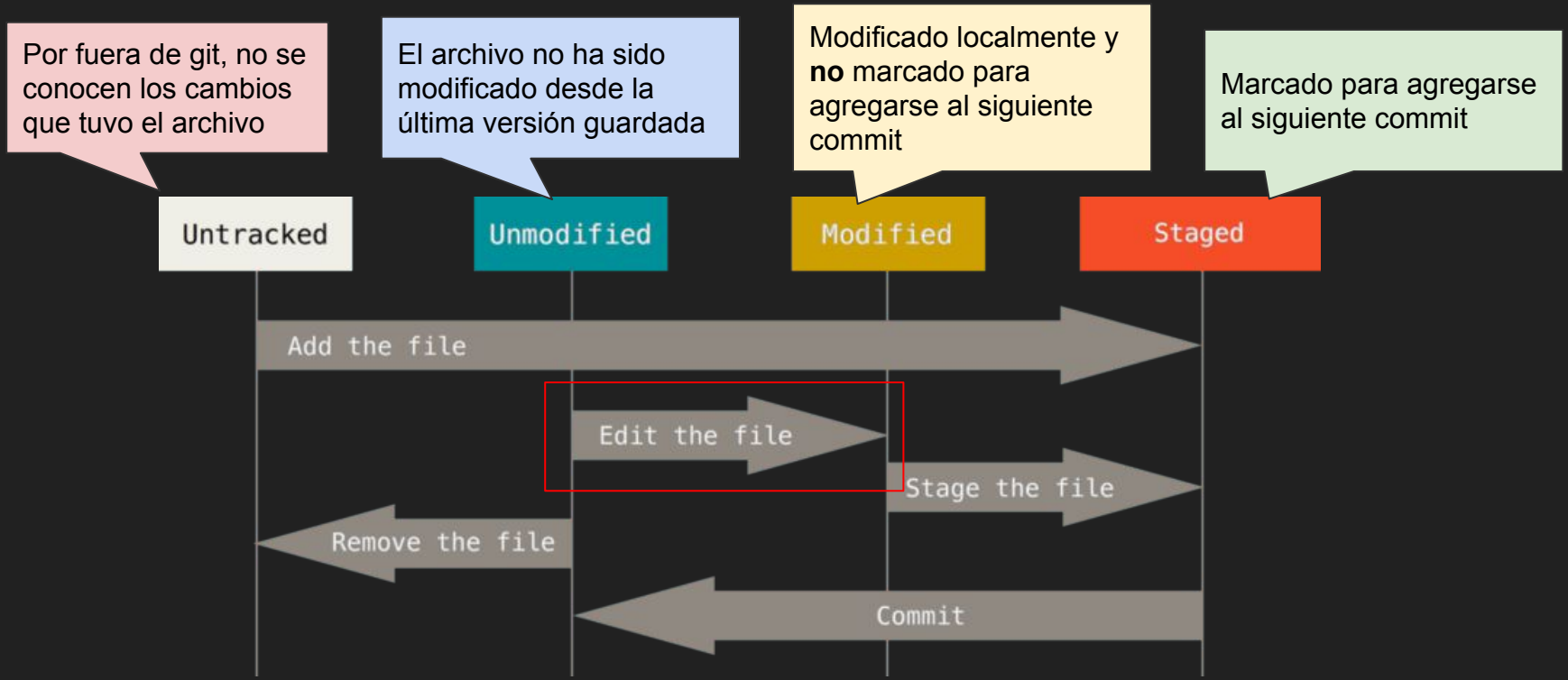
Ciclo de vida de un archivo



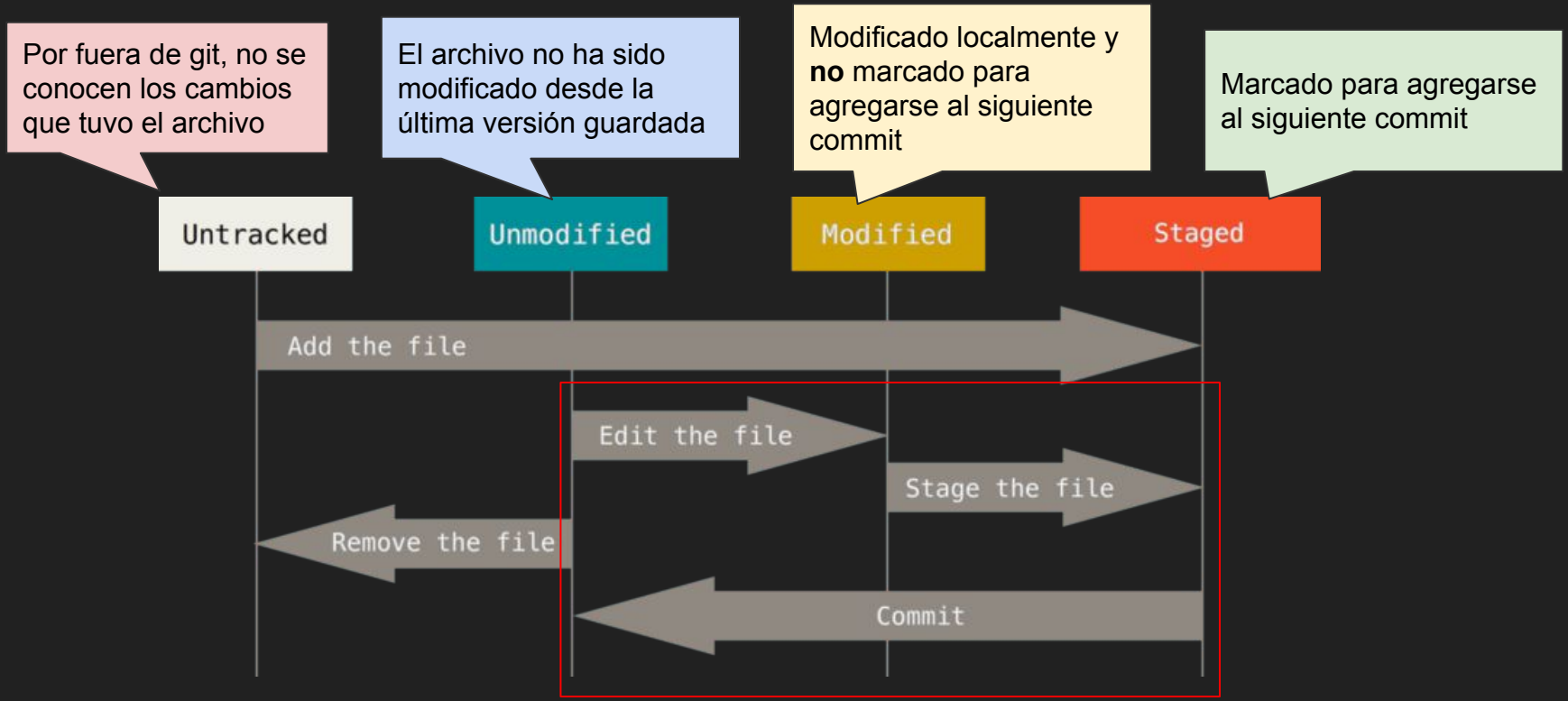
Ciclo de vida de un archivo



Ciclo de vida de un archivo



Ciclo de vida de un archivo



Ejercicio 1: Inicialización y ciclo de vida

Inicializar un repositorio y agregar un archivo

```
$ mkdir ejemplo
$ cd ejemplo

$ git status              # (falla)
$ git init                # para deshacer: $ rm -r .git/
$ git status              # (funciona)

<Crear un archivo.txt >   # <- untracked
$ git add archivo.txt     # <- staged
$ git commit              # <- unmodified
<Modificar archivo.txt>   # <- modified
```

Ejercicio 1: Inicialización y ciclo de vida

Inicializar un repositorio y agregar un archivo

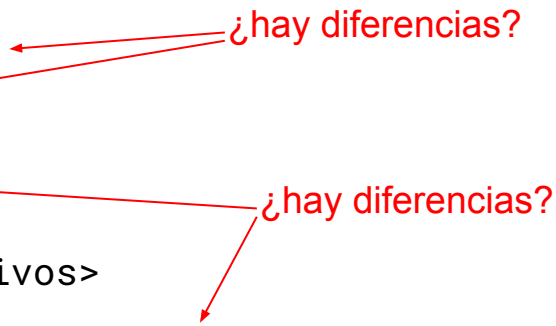
```
$ mkdir ejemplo            # No necesario en: $ git clone <url>
$ cd ejemplo

$ git status               # (falla)
$ git init                 # para deshacer: $ rm -r .git/
$ git status               # (funciona)

<Crear un archivo.txt >    # <- untracked
$ git add archivo.txt      # <- staged
$ git commit               # <- unmodified
<Modificar archivo.txt>    # <- modified
```

Importante: un archivo puede estar en **Modified** y **Staged** a la vez ¿por qué?

Ejercicio 2: viendo los cambios (log, show, diff)

Paso 1: Crear 2 commits y 2 archivos (uno por commit)
Paso 2: Crear un tercer commit modificando ambos archivos

```
$ git log   # Salir con: q
<copiar el hash de un commit>

$ git show <pegar hash>
$ git diff <hash>

$ git checkout <hash>
$ git status
<revisar contenido de archivos>
$ git checkout main
$ git checkout <hash> -- archivo.txt
$ git status
```



Ejercicio 3: subir y bajar de GitHub (remotes)

Paso 0: Crear una cuenta en GitHub
Paso 1: Crear un repositorio vacío en GitHub

```
$ git remote add un_remoto git@github.com:<Usuario>/example.git
$ git push un_remoto
<ir a GitHub y modificar el README>
$ git pull un_remoto
```

Genera un nuevo commit en GitHub

Verificar que se bajó la mofidicación

Importante: al hacer ***git clone <url>***, se crea un remoto de nombre ***origin***

Tipos de archivo

Caso ideal:

Texto plano, código fuente, markdown, etc.

Tipos de archivo

Caso ideal:

Texto plano, código fuente, markdown, etc.

Caso no ideal:

Archivos de texto grandes generados con algún programa

Al ser mucho texto, es más difícil visualizar bien los cambios.

Generalmente inviable mezclar cambios (**merge**).

Tipos de archivo

Caso ideal:

Texto plano, código fuente, markdown, etc.

Caso no ideal:

Archivos de texto grandes generados con algún programa

Al ser mucho texto, es más difícil visualizar bien los cambios.

Generalmente inviable mezclar cambios (**merge**).

Caso menos ideal:

Archivos binarios no legibles sin una aplicación específica

No se pueden ver los cambios.

Git **almacena todas las versiones** que alguna vez existieron, se puede volver pesado el repositorio.

¿Qué hacer con los notebooks, entonces?

Si quieren tener la historia de cambios más limpia y liviana, o trabajar en paralelo y resolver conflictos más fácilmente, hay dos opciones:

- Limpiar los outputs (clean outputs) antes de agregarlos a git.

- Trabajar con archivos .py en modo interactivo
  - Una vez terminado el trabajo o la versión, exportar .ipynb.
  - Se puede volver al .py desde el .ipynb.

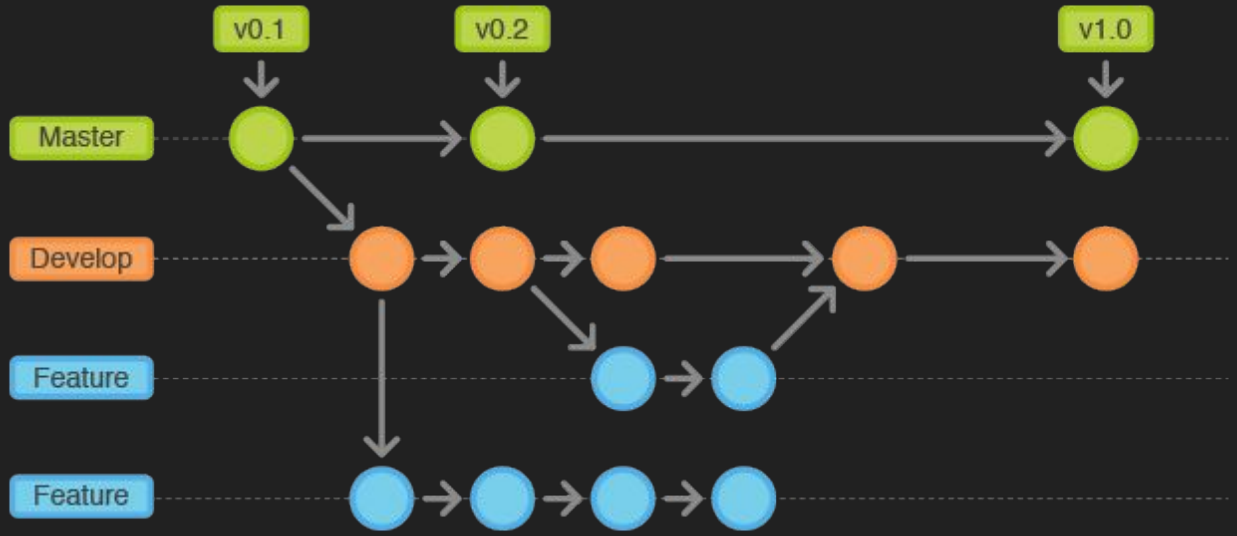
¿Y con los archivos binarios?

Sólo agregar a git las versiones importantes, no guardar todos los cambios. Sobre todo si son pesados (ej: >5MB) con muchas imágenes.

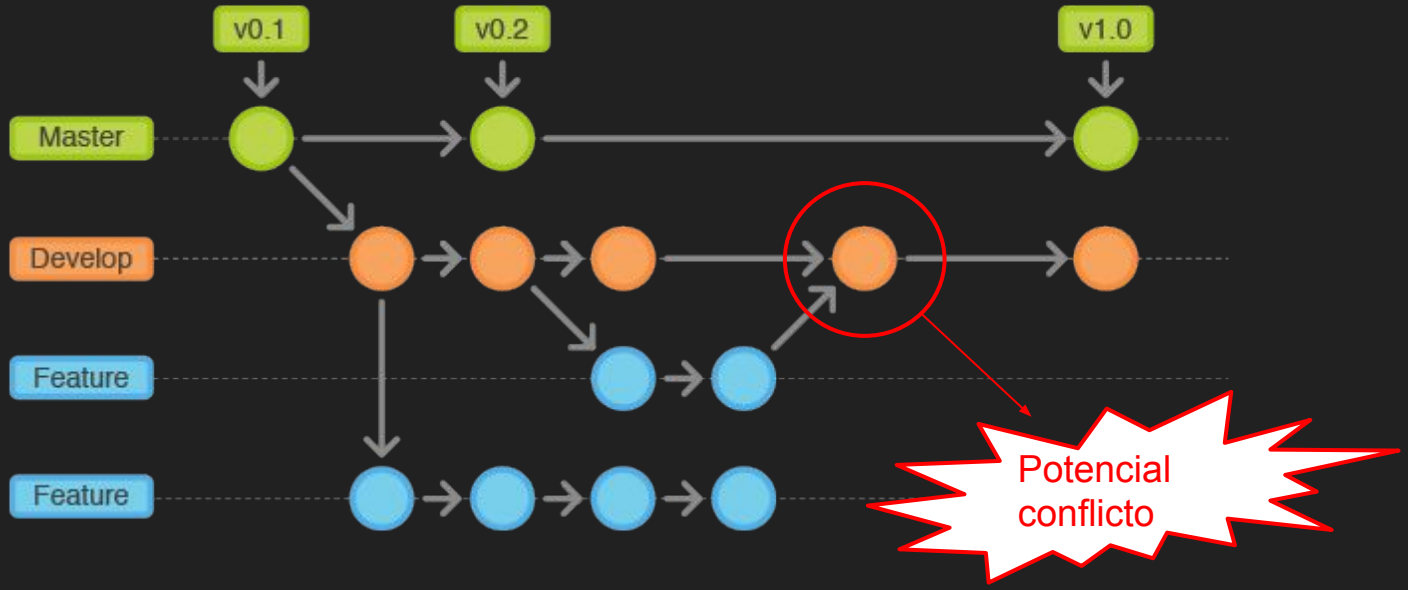
Por ejemplo, al trabajar en el informe, pueden utilizar Google Docs o LaTex durante el trabajo, y sólo agregan el PDF al final (y eventuales revisiones).

Nota: sí les recomendamos ir guardando en git las versiones intermedias del código (notebook y python).

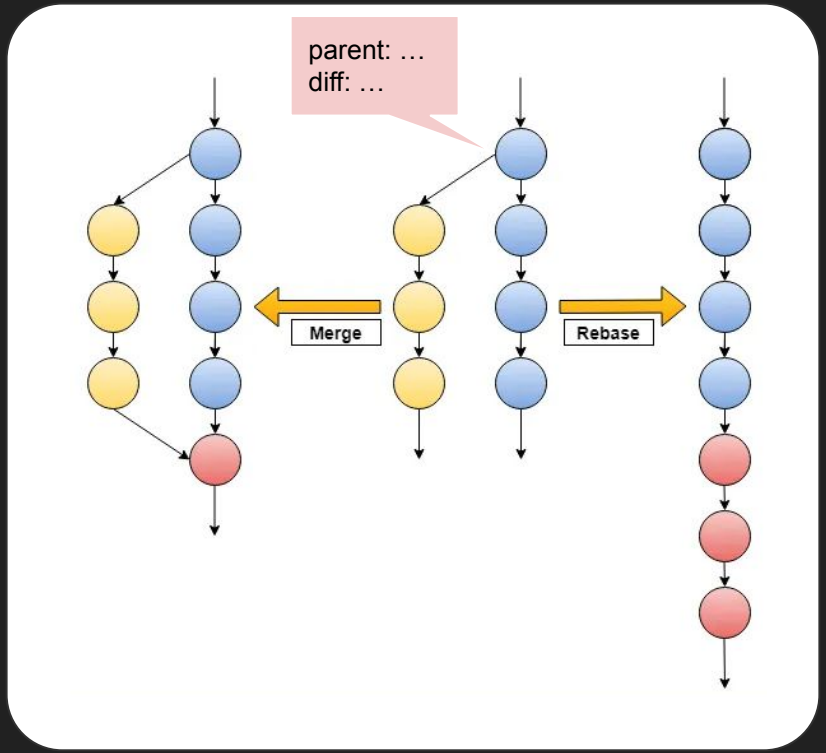
Branches: ¿para qué sirven?



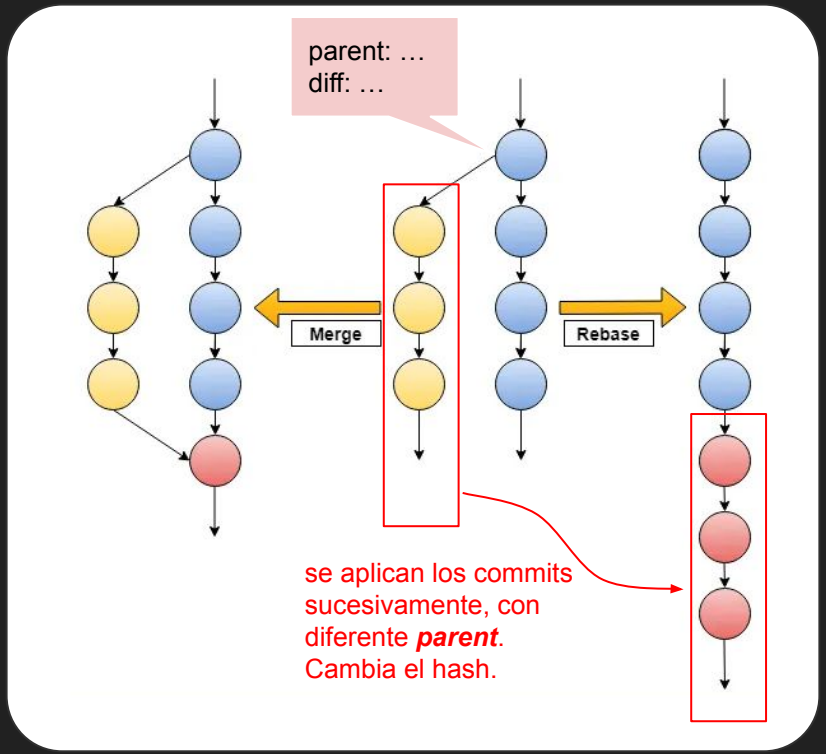
Branches y conflictos



Resolución de conflictos: rebase vs. merge



Resolución de conflictos: rebase vs. merge



Ventajas del rebase:
Historia de commits lineal.
No hay commits de merge.

Desventajas:
Los conflictos se van resolviendo a medida que se aplican commits.
Reescribe la historia (hay que hacer push force).

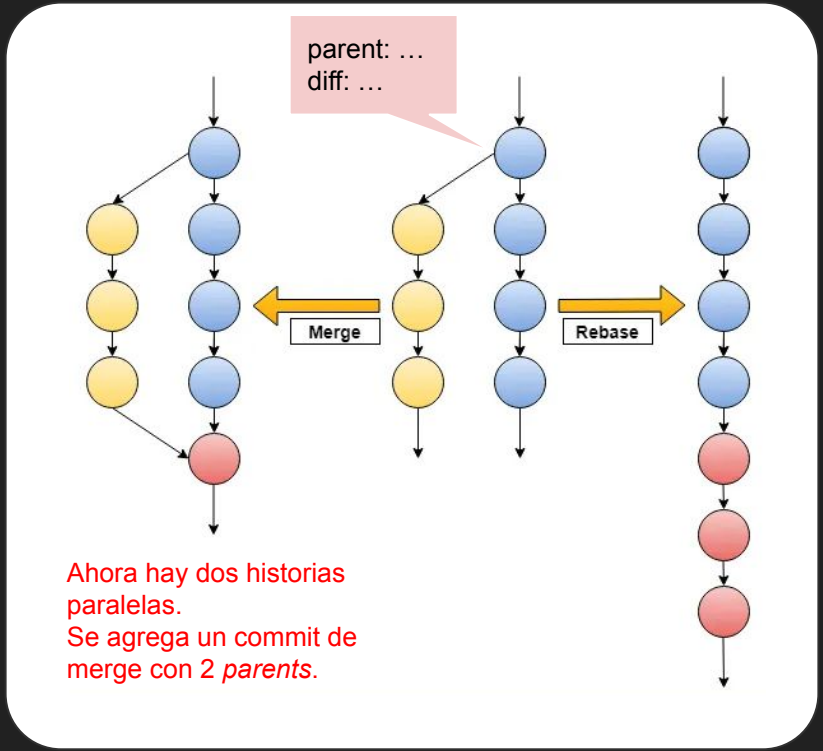
Cuando usarlo:
Branch privada con pocos cambios y donde no quiero generar commits de merge.

Resolución de conflictos: rebase vs. merge

Ventajas del merge:
Sólo resolvemos un commit
No hay que reescribir la historia.

Desventajas:
Ya no es tan fácil visualizar la historia (log)
Genera un commit de merge.

Cuando usarlo:
Cuando las branches divergieron demasiado o si estamos en una **branch compartida** que no se puede reescribir.



Branches y conflictos (rebase, merge)

```
$ git checkout -b nueva-branch   # borrar: git branch -D nueva-branch

$ git checkout main              # volver a main
<crear commits en main>

$ git checkout nueva-branch      # volver a nueva-branch (ya creada)
$ git log                        # no debería ver los últimos commits
<crear commits en nueva-branch>  # main y nueva-branch divergieron!

$ git rebase main                # puede haber conflictos
$ git checkout main
$ git merge nueva-branch         # ahora main tiene todos los cambios
```

Otros comandos interesantes

```
$ git stash              # Poner los cambios locales en el freezer
$ git stash list         # Ver lo que hay
$ git stash pop          # Sacar lo último que había puesto

$ git rebase -i HEAD~3   # Reescribir historia reciente (interactivo)

$ git pull --rebase      # Adivinanza

$ git add -i             # Agregar partes de archivos
```

Para bajarse el repo de la Tarea 1

- Opción 1: Crear un fork (en GitLab)
- Opción 2: clonar el repo y cambiar el ***remote*** (para usar GitHub o GitLab)

```
$ git clone git@gitlab.fing.edu.uy:maestria-cdaa/intro-cd.git

<crear un repo en su cuenta>
$ cd intro-cd/
$ git remote set-url origin <url de su repo>
$ git push
```

Recursos recomendados

- Practicar y practicar, consultar con Google y ChatGPT
- Documentación oficial (muy buena): https://git-scm.com/doc
  - Ejemplo: https://git-scm.com/docs/gittutorial
- Comandos rápidos: https://training.github.com/downloads/es_ES/github-git-cheat-sheet/
- Tutorial (tal vez demasiado) sencillo: https://rogerdudler.github.io/git-guide/
- Tutorial más completo: https://www.w3schools.com/git/
- Se aceptan sugerencias :-)

The End