

# Control de Versiones



**git**



**GitHub**



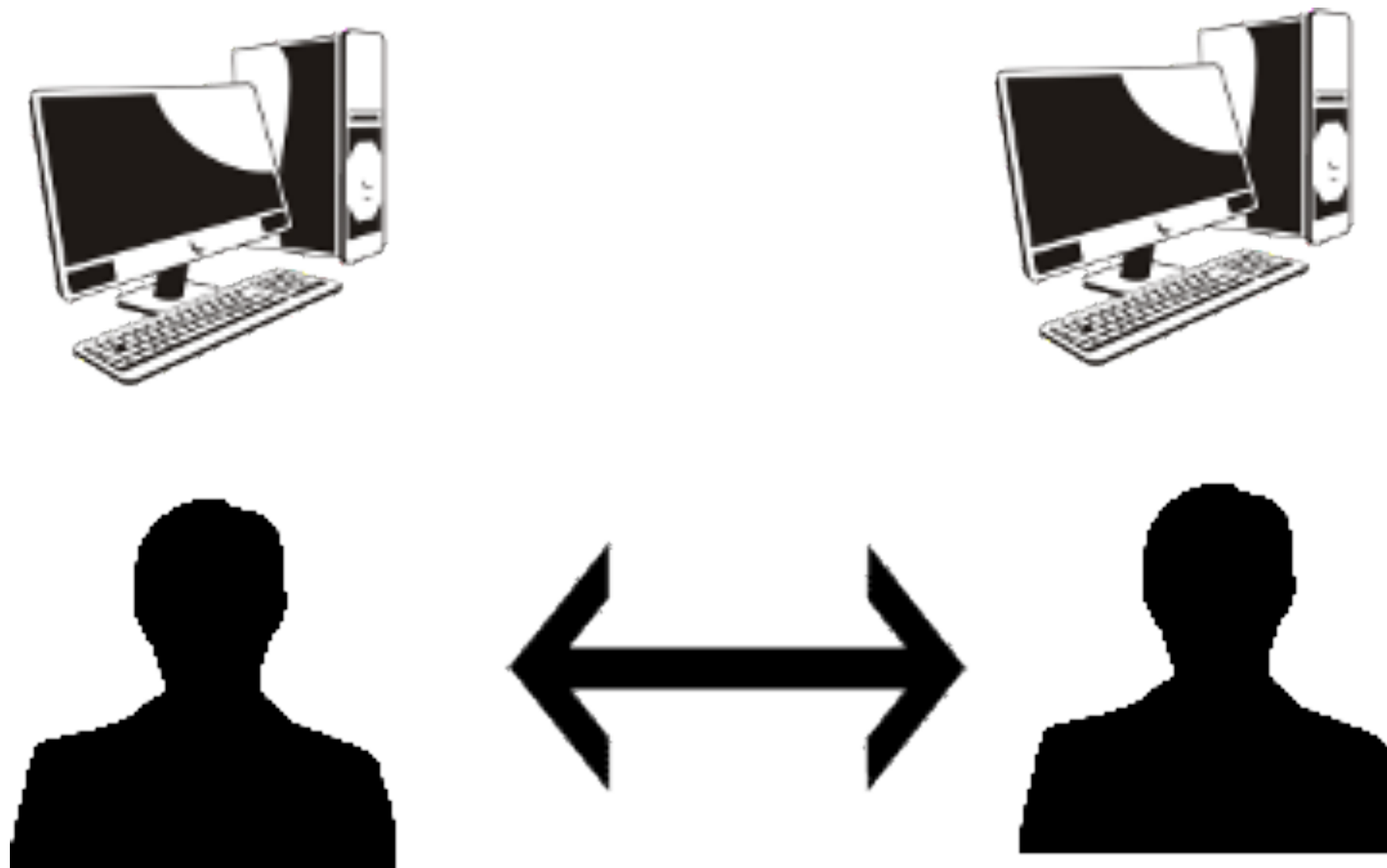
[emmanuel.lagarrigue@cs.uns.edu.ar](mailto:emmanuel.lagarrigue@cs.uns.edu.ar)

Atlassian  
**SourceTree**

# Guardando información

- ¿Qué tan importantes son sus datos/archivos?
- ¿Tienen archivos que quisieran no perder nunca? ¿Pueden asegurar que nunca vaya a pasar?
- ¿Les gustaría conocer la historia de los cambios de dichos archivos?
- Cuando los archivos son compartidos... ¿Qué pasa con los cambios por separado?

- Supongamos que dos programadores trabajan juntos en un mismo código, cada uno en su computadora.
- Qué sucede cuando ambos cambian el mismo archivo (clase), ya sean distintos métodos o los mismos.

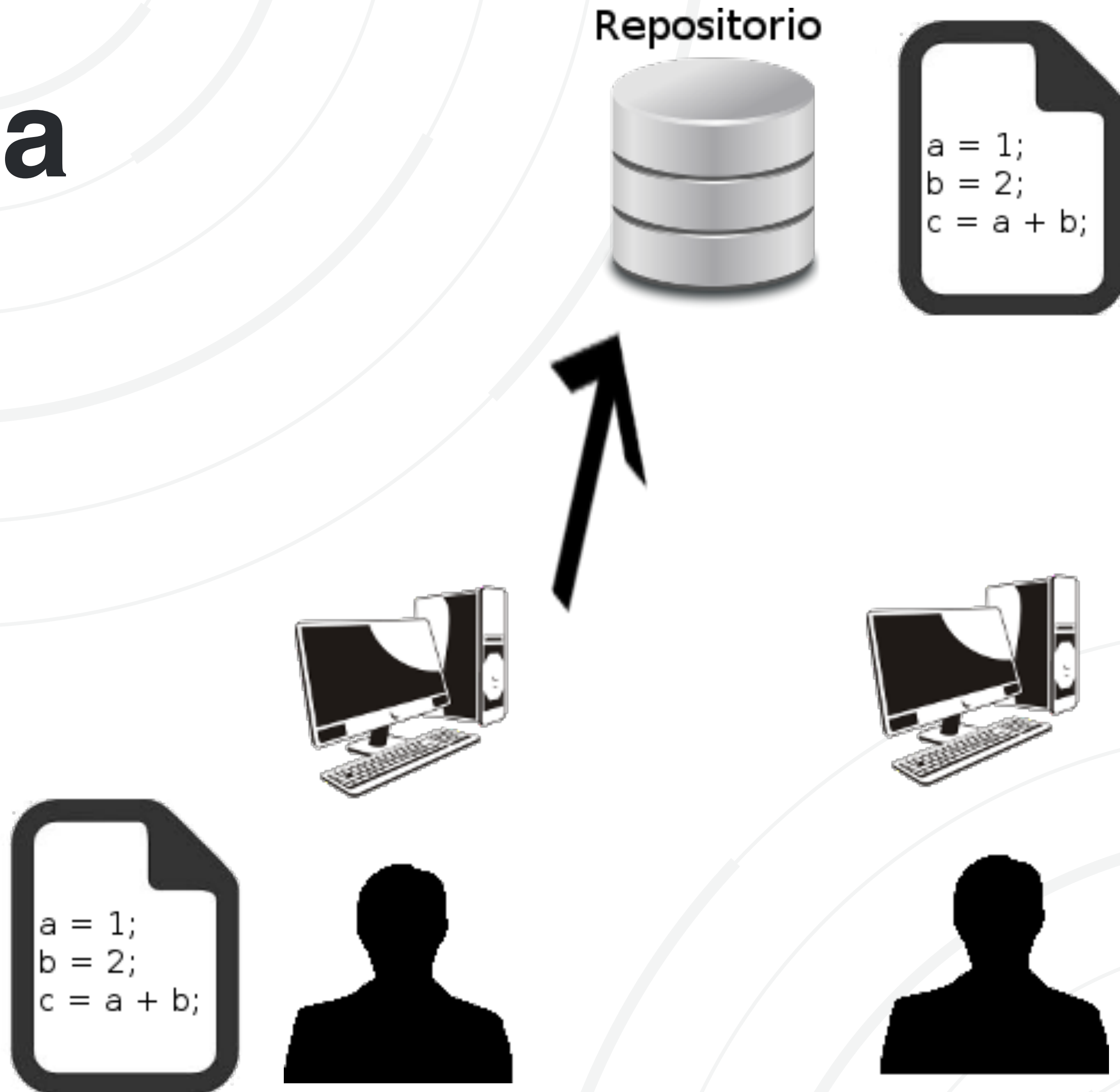


- Trabajan en distintos archivos, y se los van pasando para ir “mezclando” el código.
- Además, ¿cómo volver a versiones anteriores?
- Supongamos que la versión de hace una semana tenía solucionado un issue que ahora resurgió, ¿cómo recuperarla?
- ¿Y si se rompe el disco?!!!!!!!

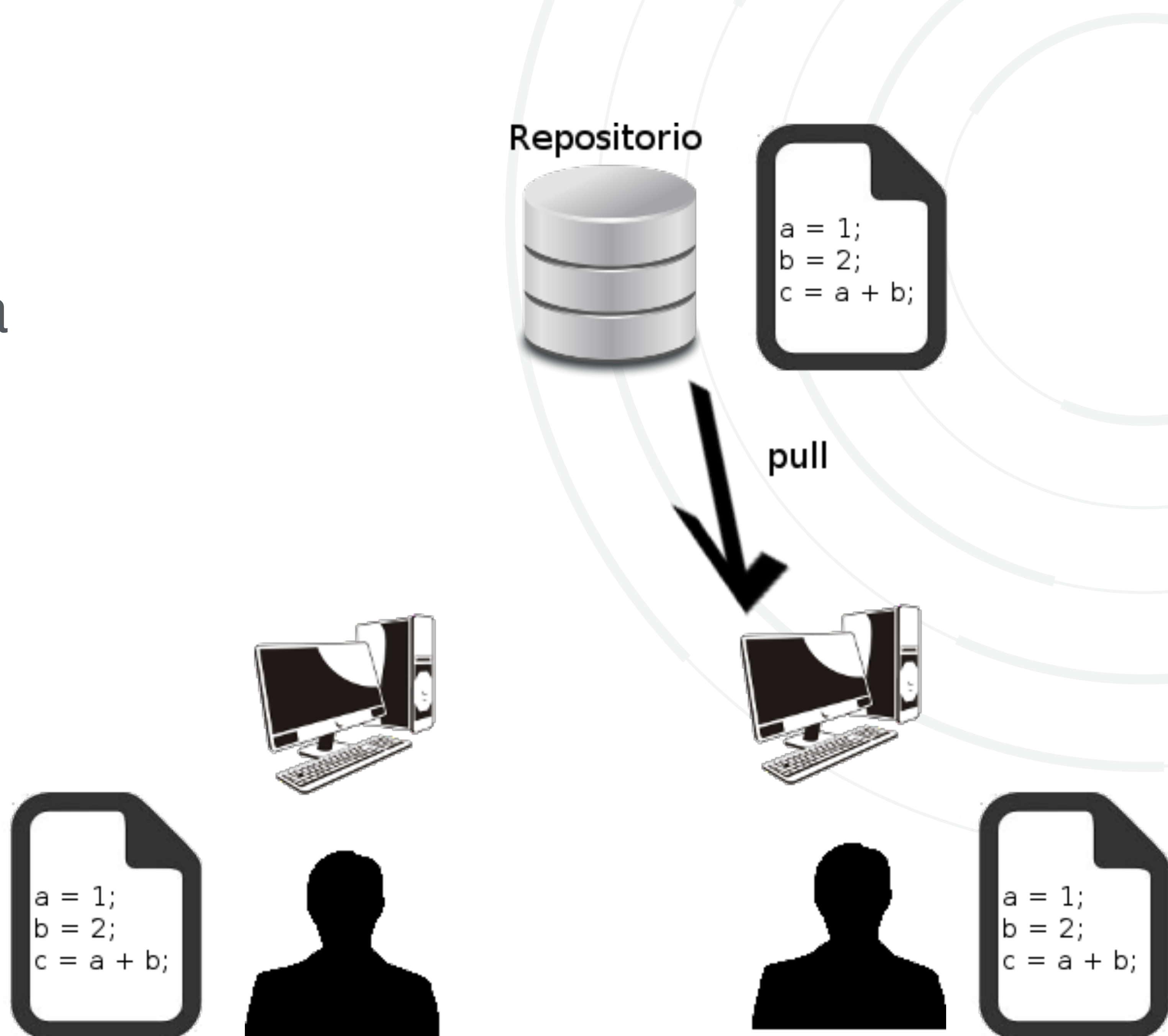
- Mientras más gente esté involucrada en el código fuente, los problemas crecen.
- Para solucionar este dilema, existen los **Sistemas de Control de Versión**, o Controladores de Versión.
- El control de versiones es la **gestión de los cambios** que se realizan sobre los elementos de un producto, en nuestro caso, código fuente.

# Cómo funciona

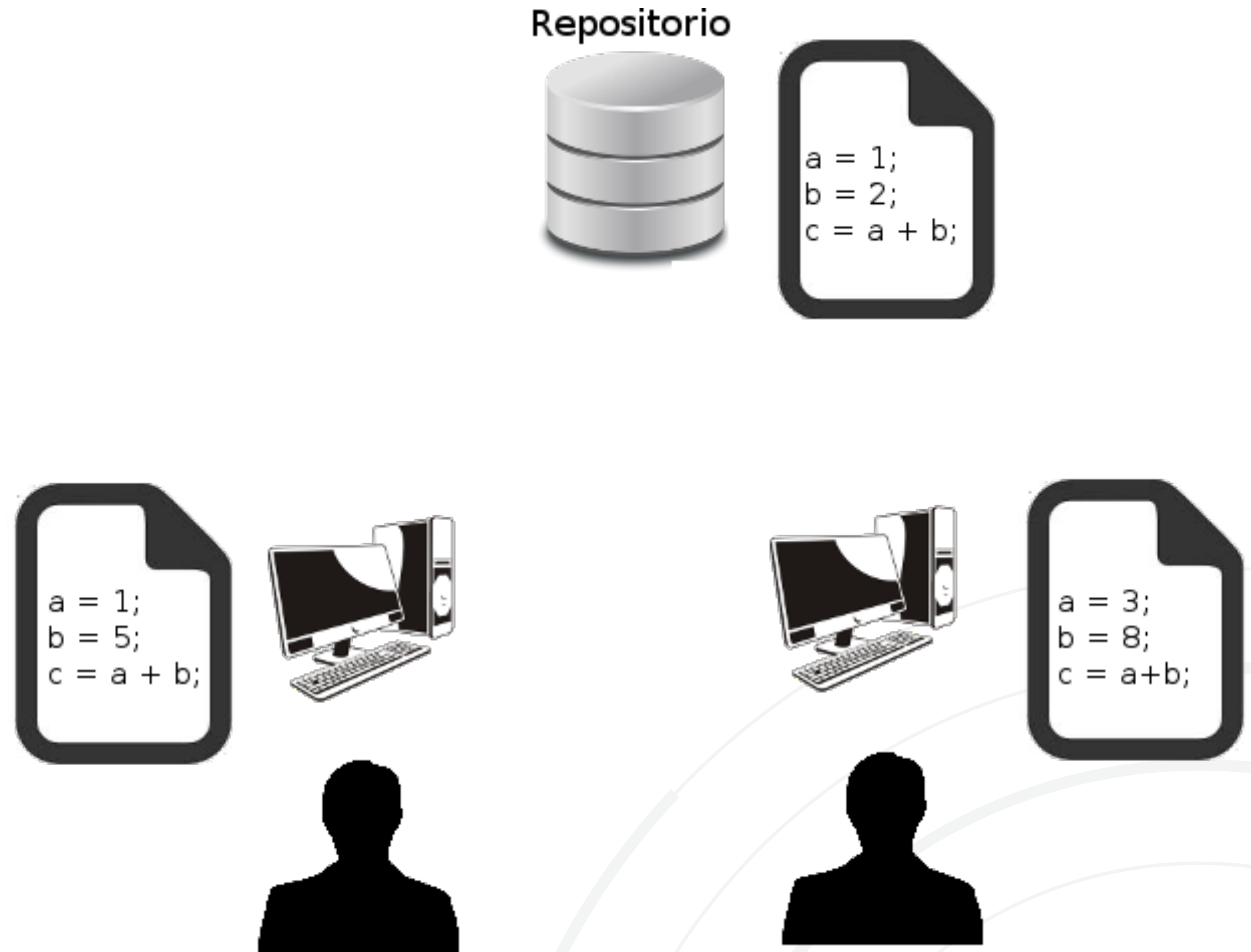
- El código se “sube” a un **“Repositorio”**.



- **“Bajamos”** una copia local (pull)

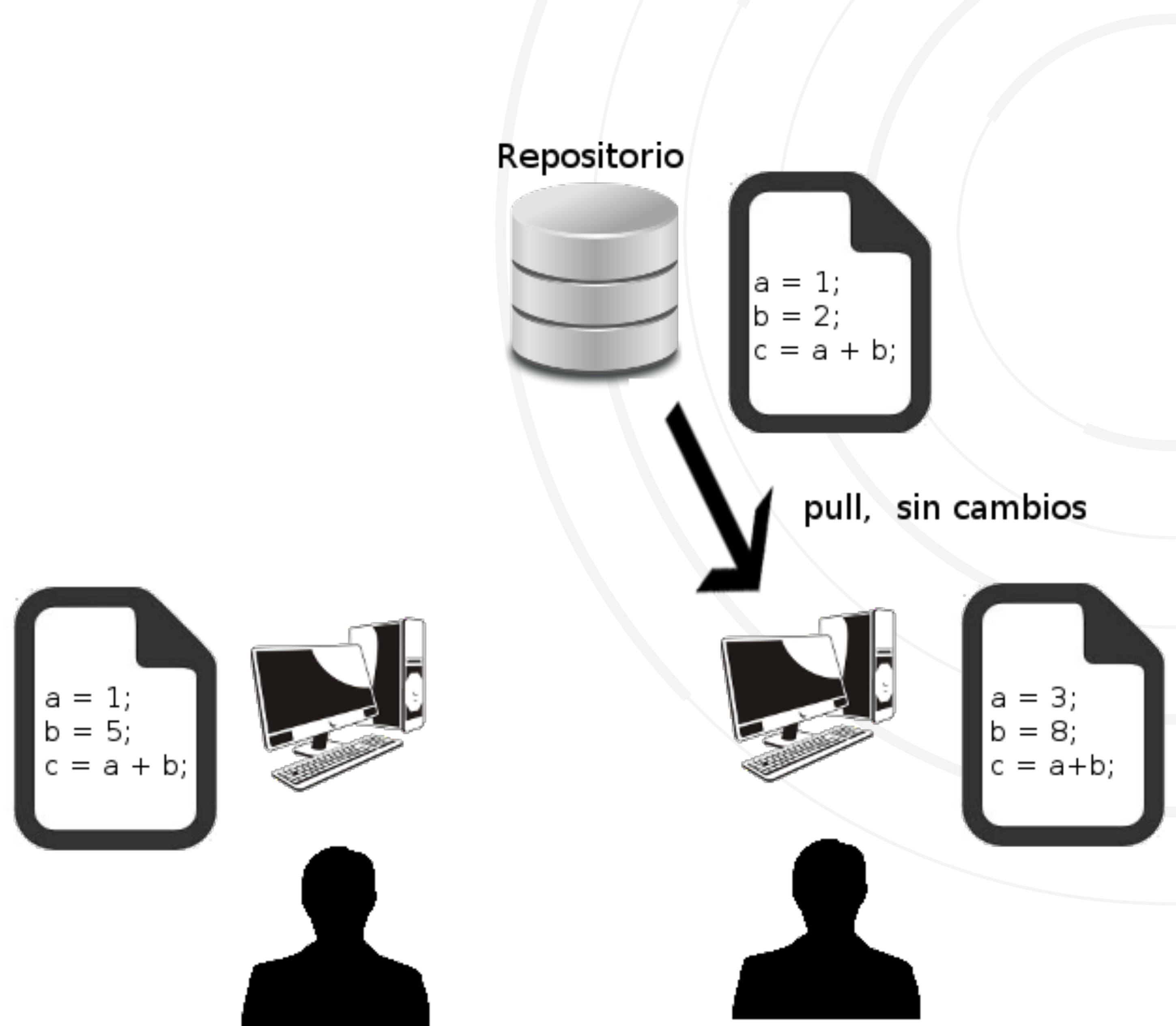


- Trabajamos en nuestra **copia local**, modificamos archivos y hacemos un **commit** (registro de historia de manera local).

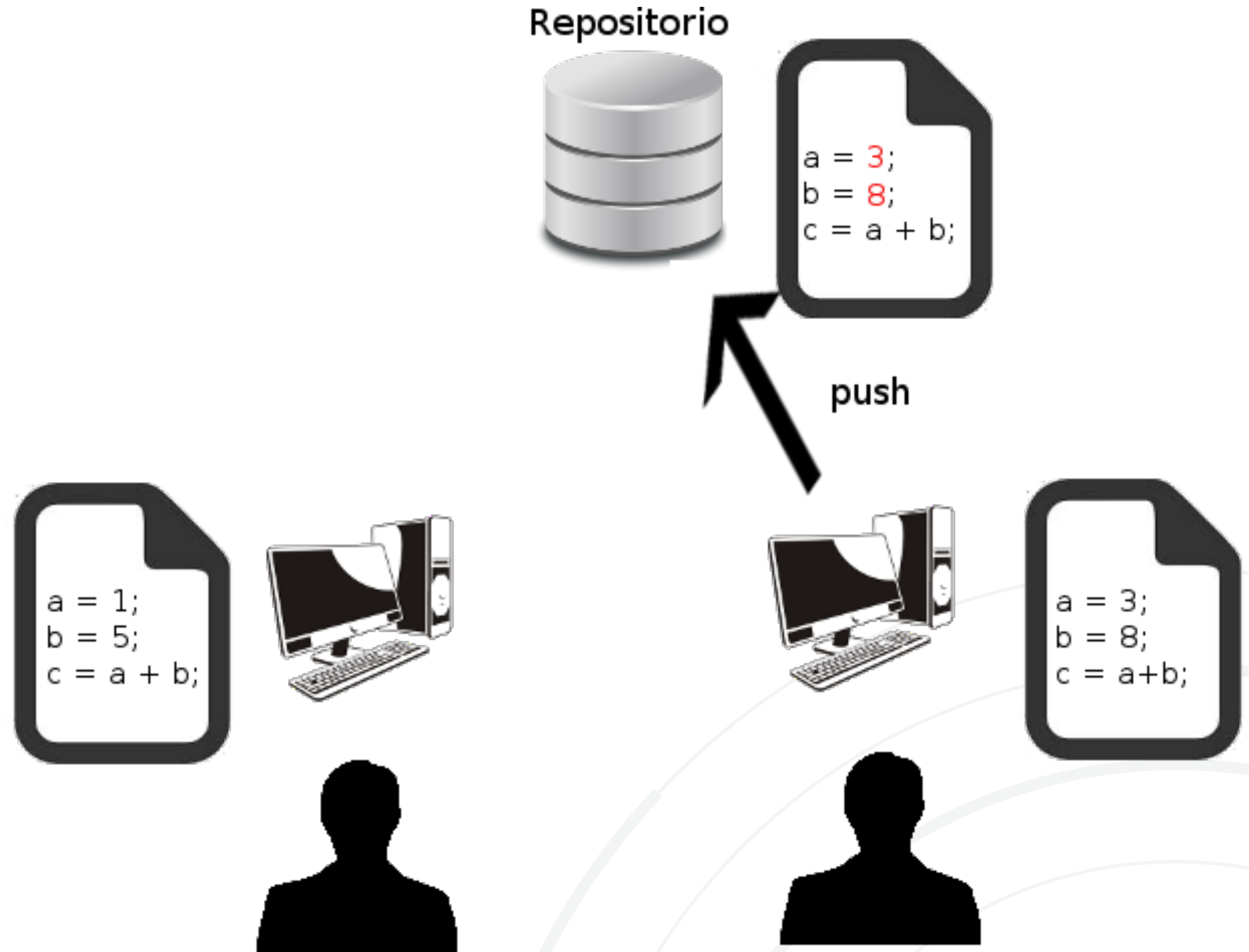




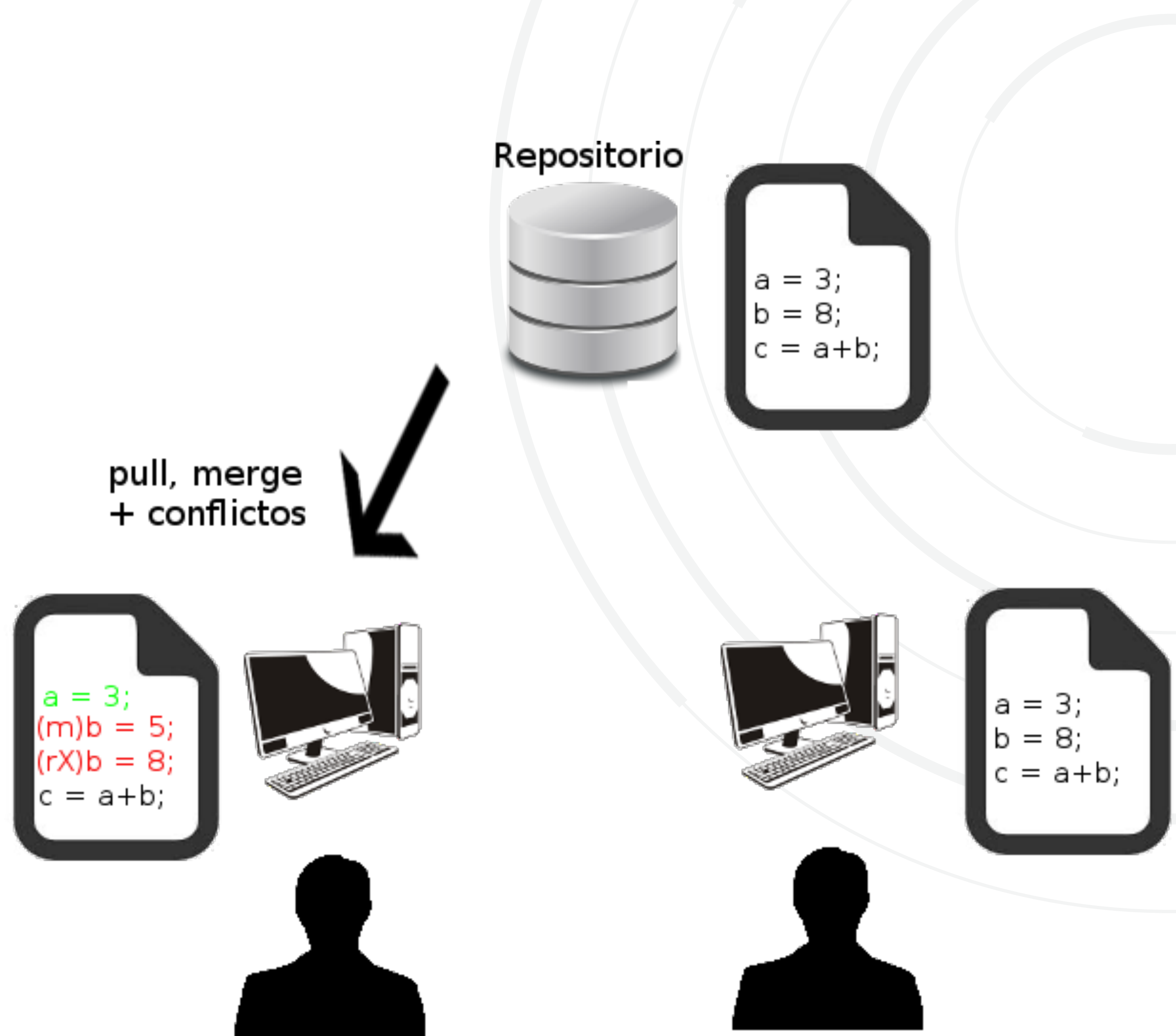
- Luego de hacer nuestros cambios, antes **subirlos**, actualizamos la copia local (**pull**).



- Ya que no había cambios en el repositorio, podemos subir (**push**) los cambios.



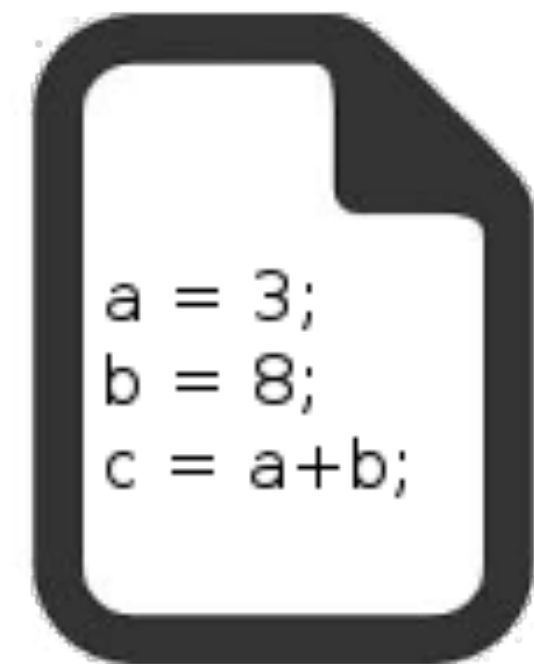
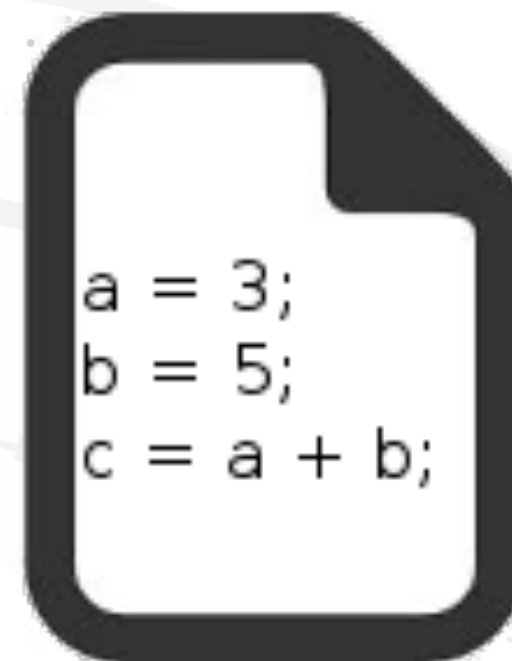
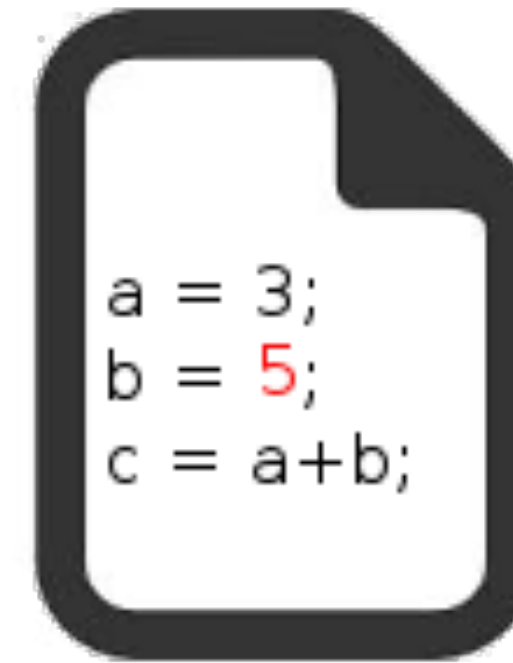
- Un 2do programador quiere subir sus cambios también, por lo que hace pull.
- El controlador de versiones **mezcla** la copia local con la del repositorio y reconoce un **conflicto**.



- Se resuelven los conflictos y se hace push

resolver conflictos,  
commit, push

Repositorio



# Resumen

- Cuando queremos que nuestros cambios se reflejen en una nueva versión, hacemos un “**commit**” (local).
- Antes de subir nuestros cambios al repositorio, hacemos “**pull**” de nuevo, para traer los cambios que hayan subido terceros.
- Hacemos “**merge**” de nuestro código con el traído del repositorio (automático).
- Solucionamos **conflictos** si es que los hay.
- Cuando queremos reflejar nuestros commits en el repositorio, hacemos “**push**”

# Conceptos

- **Repositorio:** Lugar donde se almacenan los datos y su historia. Generalmente, un servidor.
- **Revisión (versión):** Versión determinada que se gestiona. Es un estado recordado de nuestro código fuente. A la última versión se la llama “head”.
- **Línea Base:** Rama principal.
- **Branch:** Bifurcar, ramificar una rama. Es una copia de una versión para trabajar en ella de manera separada. En algún punto se debe hacer “merge” con la línea base.

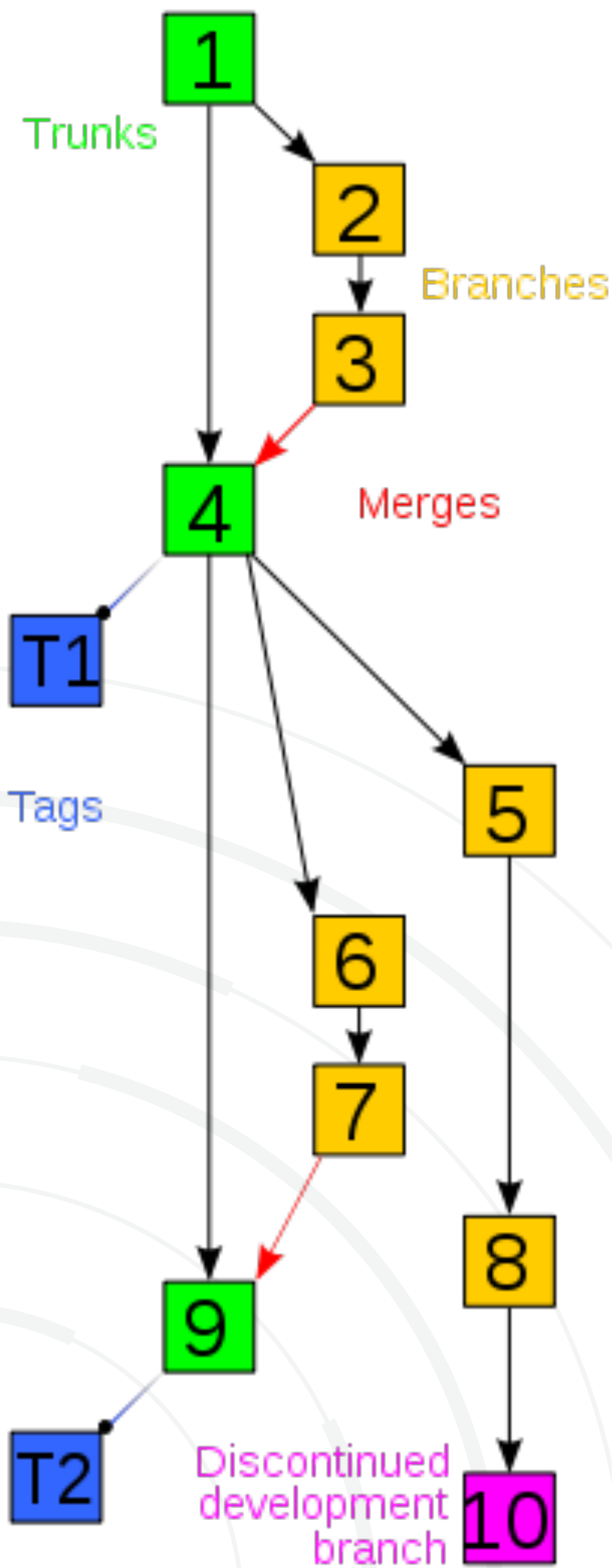
# Conceptos

- **Merge:** Integrar, mezclar, unificar el código fuente.
- **Clone:** Obtener una copia local del repositorio. A esa copia se la llama “workspace”.
- **Commit:** Guardar cambios en en repositorio. Se creará una nueva versión. (Si usamos git, el commit es local. Para actualizar el repositorio tenemos que hacer “push” de todos los commits locales)
- **Diff:** Cambio. Por ejemplo, diferencia entre la copia local y la última versión del repositorio.

# Conceptos

- **Conflicto:** Cuando una o más líneas modificadas localmente fueron modificadas y “commiteadas” por un tercero en el repositorio.
- **Export:** Genera una copia local (cómo el check out), pero fuera del controlador de versiones.
- **Update:** Actualizar el workspace, con la última versión del repositorio. En git, se le llama “pull”.





# Control de Versiones

- Tenemos principalmente dos componentes:
  - El **repositorio**: es el lugar donde estará alojado nuestro código fuente. Podría ser una base de datos local o un sistema externo, por ejemplo **GitHub**, Bitbucket, gitlab, Google Cloud Source Repositories.
  - El **sistema de control de versiones**: es la herramienta (software) que automatiza la gestión de los cambios. Ejemplos: **Git**, Subversion, Mercurial.

# Repositorio:



# GitHub

- Es uno de los repositorios más populares, sobre todo para proyectos open source.
- Gratis para repositorios públicos.
- Hay que crear un usuario, y permite crear y seguir repositorios.
- <https://github.com/>



Search GitHub

Pull requests Issues Marketplace Explore



elagarrigue

Browse activity

Discover repositories

thuutin pushed to tin/fix-readmore-show-only-first-alert in skedgo/tripgo-android a day ago  
2c724ae Fix test

thuutin pushed to tin/fix-readmore-show-only-first-alert in skedgo/tripgo-android a day ago  
f942d09 Show checkboxes and calculate alternative routes

pnuts2 pushed to experiments/svBasedMapButton in skedgo/tripkit-ios 2 days ago  
2c4555a Merge branch 'master' into experiments/svBasedMapButton  
79e3ab3 Fix for test failure  
2 more commits »

thuytrinh pushed to master in skedgo/tripgo-android 2 days ago  
cd6317a Release hot-fix v4.20  
de39c94 Fix: crash which happened when dismissing city picker (#2097)  
2 more commits »

thuytrinh pushed to dev-v4 in skedgo/tripgo-android 2 days ago  
cd6317a Release hot-fix v4.20

Repositories you contribute to 3

- skedgo/tripgo-android 1 ★
- skedgo/tripkit-android 2 ★
- skedgo/StringResourcesConv... 2 ★

Your repositories 0

New repository

Find a repository...

All Public Private Sources Forks

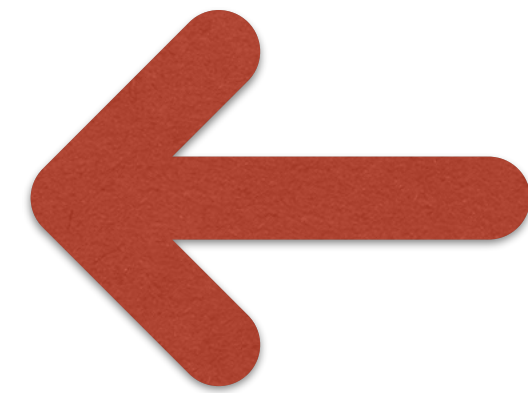
No repositories match the selected filters.

Your teams 2

Find a team...

- skedgo/mobile
- skedgo/mobile-admins

# GitHub

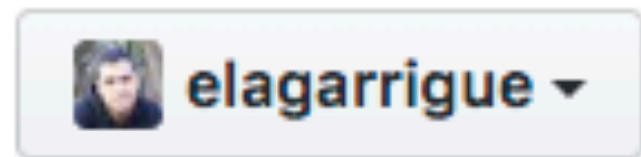


# Create a new repository

A repository contains all the files for your project, including the revision history.

Owner


Repository name




/ learning-git-hello-world ✓

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

Description (optional)

 **Public**  
Anyone can see this repository. You choose who can commit.

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

**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾ ⓘ

**Create repository**



# GitHub



This repository

Search

Pull requests

Issues

Marketplace

Explore



elagarrigue / learning-git-hello-world

Unwatch 1

Star 0

Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

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

Set up in Desktop or **HTTPS** **SSH** `git@github.com:elagarrigue/learning-git-hello-world.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

## ...or create a new repository on the command line

```
echo "# learning-git-hello-world" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:elagarrigue/learning-git-hello-world.git
git push -u origin master
```

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

```
git remote add origin git@github.com:elagarrigue/learning-git-hello-world.git
git push -u origin master
```

# GitHub

# Control de versiones:



# git

- Es la herramienta instalada en nuestra computadora que nos permite hacer pull, push, commit, etc en nuestros repositorios.
- Se ejecuta por línea de comandos, no tiene interfaz gráfica.
- <https://git-scm.com>

```
Emmanuels-MacBook-Pro:~ elagarrigue$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]
```

These are common Git commands used in various situations:

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

<code>clone</code>	Clone a repository into a new directory
<code>init</code>	Create an empty Git repository or reinitialize an existing one

work on the current change (see also: `git help everyday`)

<code>add</code>	Add file contents to the index
<code>mv</code>	Move or rename a file, a directory, or a symlink
<code>reset</code>	Reset current HEAD to the specified state
<code>rm</code>	Remove files from the working tree and from the index

examine the history and state (see also: `git help revisions`)

<code>bisect</code>	Use binary search to find the commit that introduced a bug
<code>grep</code>	Print lines matching a pattern
<code>log</code>	Show commit logs
<code>show</code>	Show various types of objects
<code>status</code>	Show the working tree status

grow, mark and tweak your common history

<code>branch</code>	List, create, or delete branches
<code>checkout</code>	Switch branches or restore working tree files
<code>commit</code>	Record changes to the repository
<code>diff</code>	Show changes between commits, commit and working tree, etc
<code>merge</code>	Join two or more development histories together
<code>rebase</code>	Reapply commits on top of another base tip
<code>tag</code>	Create, list, delete or verify a tag object signed with GPG

collaborate (see also: `git help workflows`)

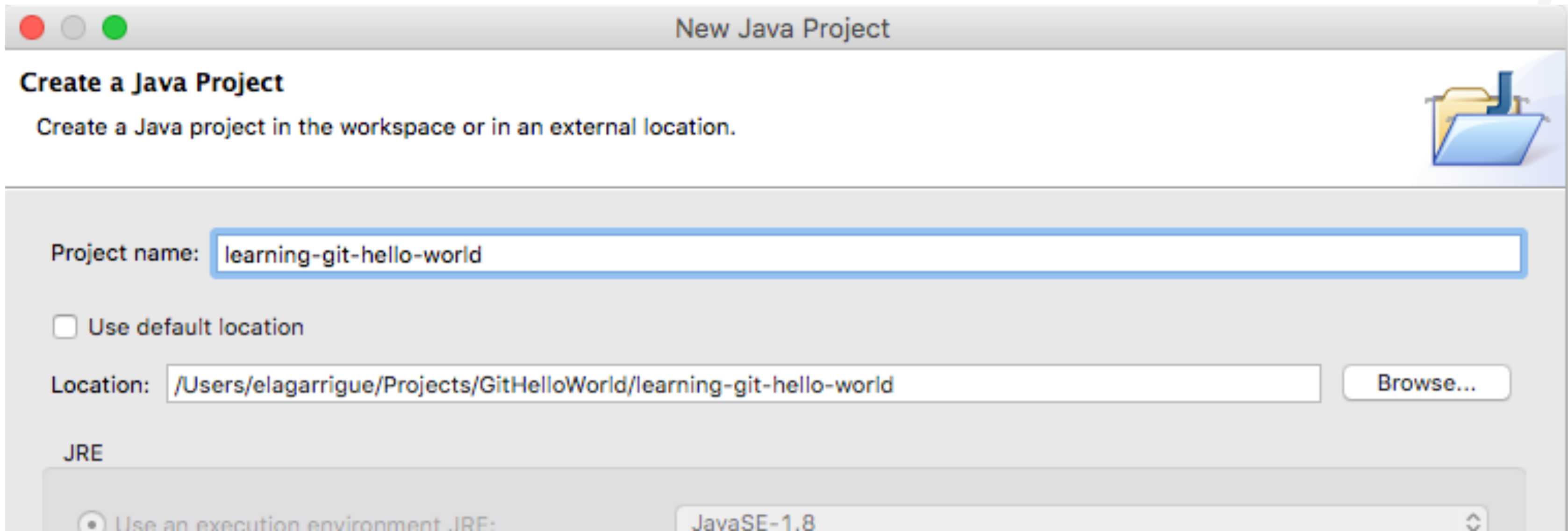
<code>fetch</code>	Download objects and refs from another repository
<code>pull</code>	Fetch from and integrate with another repository or a local branch
<code>push</code>	Update remote refs along with associated objects



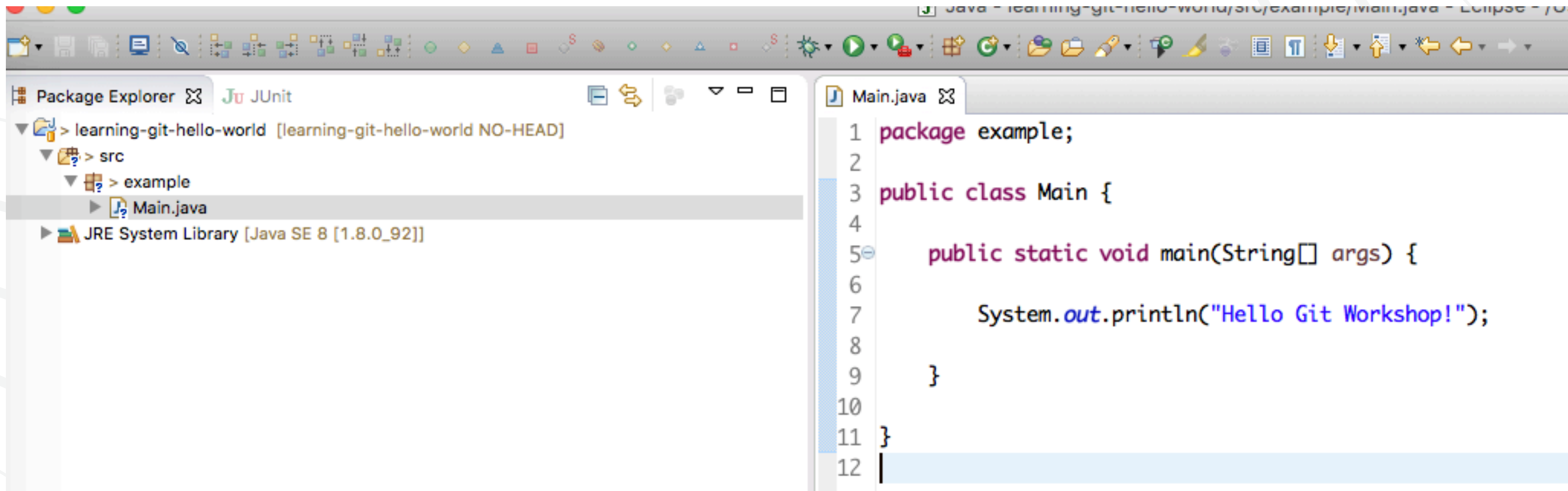




```
[Emmanuels-MacBook-Pro:Projects elagarrigue$ cd GitHelloWorld/  
[Emmanuels-MacBook-Pro:GitHelloWorld elagarrigue$ git clone https://github.com/elagarrigue/learning-git-hello-world.git  
Cloning into 'learning-git-hello-world'...  
[warning: You appear to have cloned an empty repository.  
[Emmanuels-MacBook-Pro:GitHelloWorld elagarrigue$ █
```



git





```
[Emmanuels-MacBook-Pro:learning-git-hello-world elagarrigue$ git add ./src/example/Main.java
```

```
[Emmanuels-MacBook-Pro:learning-git-hello-world elagarrigue$ git commit -m "initial commit"  
[master (root-commit) 701f34c] initial commit  
1 file changed, 11 insertions(+)  
create mode 100644 src/example/Main.java
```

```
[Emmanuels-MacBook-Pro:learning-git-hello-world elagarrigue$ git push  
Username for 'https://github.com': elagarrigue  
[Password for 'https://elagarrigue@github.com':  
Counting objects: 5, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (5/5), 429 bytes | 214.00 KiB/s, done.  
Total 5 (delta 0), reused 0 (delta 0)
```



This repository Search

Pull requests Issues Marketplace Explore



elagarrigue / learning-git-hello-world

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Git workshop repo example

Edit

Add topics

1 commit

1 branch

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

elagarrigue initial commit

Latest commit 701f34c 20 hours ago

src/example

initial commit

20 hours ago

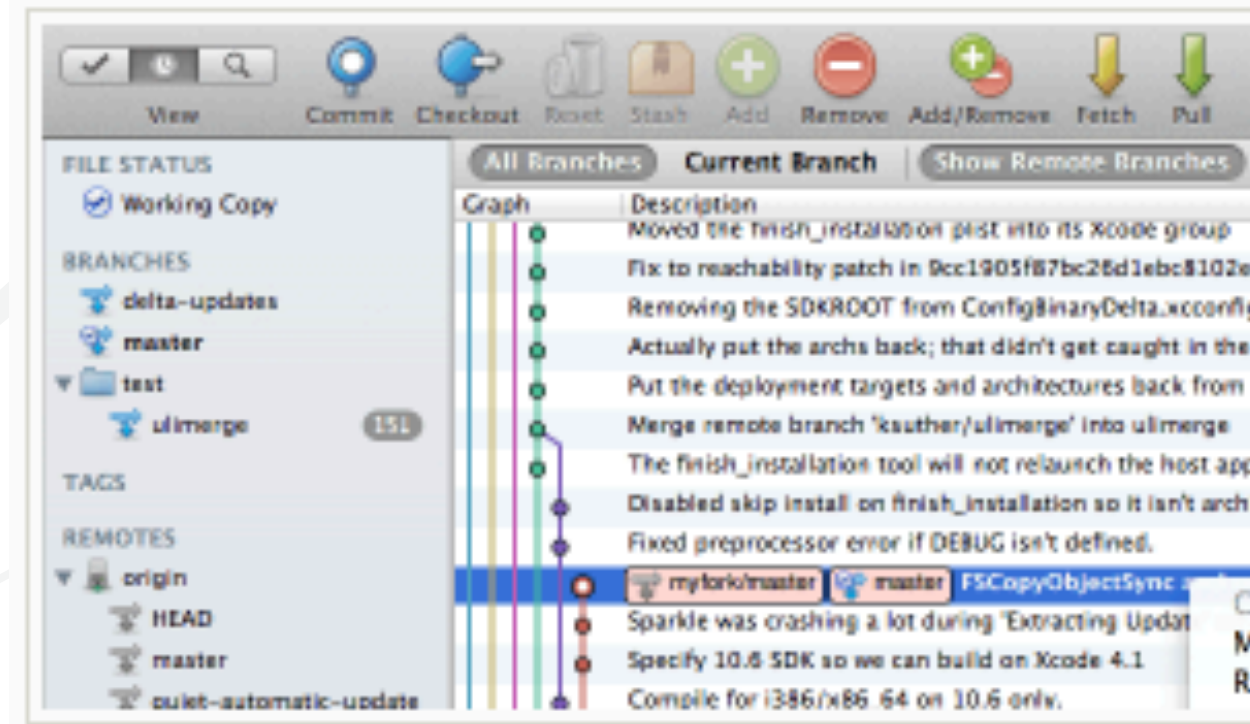
Help people interested in this repository understand your project by adding a README.

Add a README

git

# Git GUI Clients

- Existen varias herramientas GUI third party para manejo de git.
- Podemos realizar las mismas operaciones con una interfaz más amigable que la consola.
- <https://git-scm.com/>

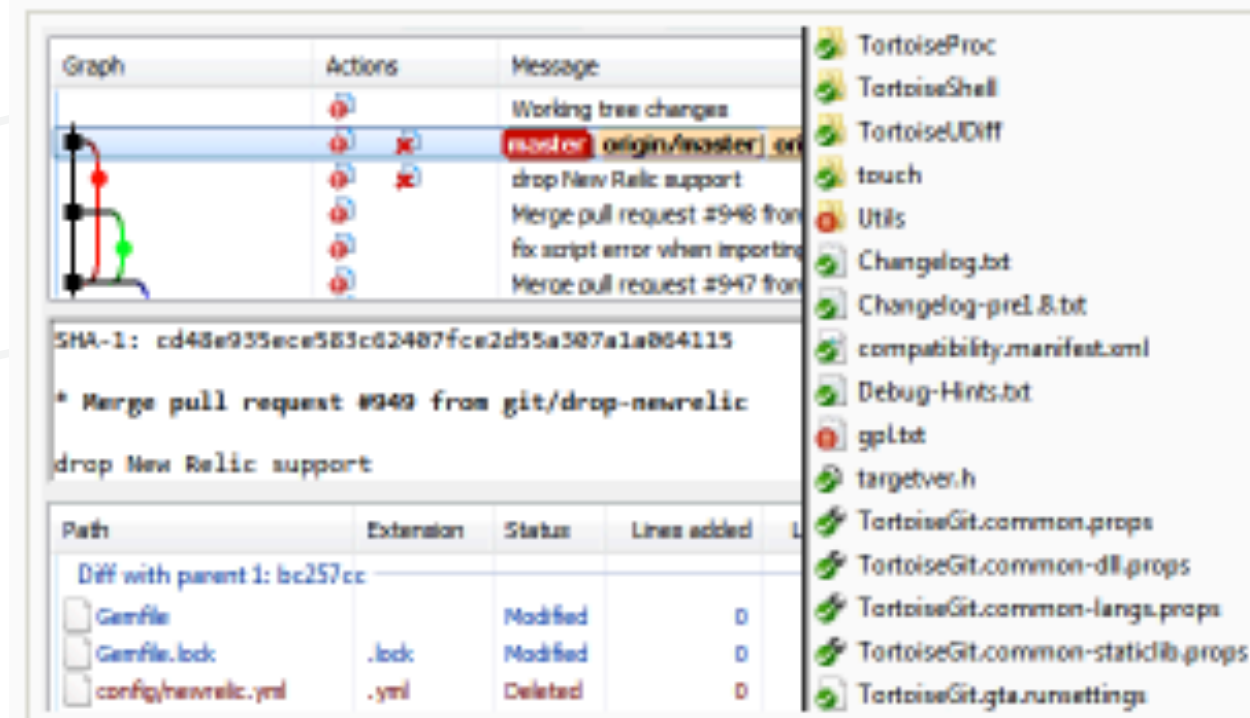


## SourceTree

Platforms: Mac, Windows

Price: Free

License: Proprietary

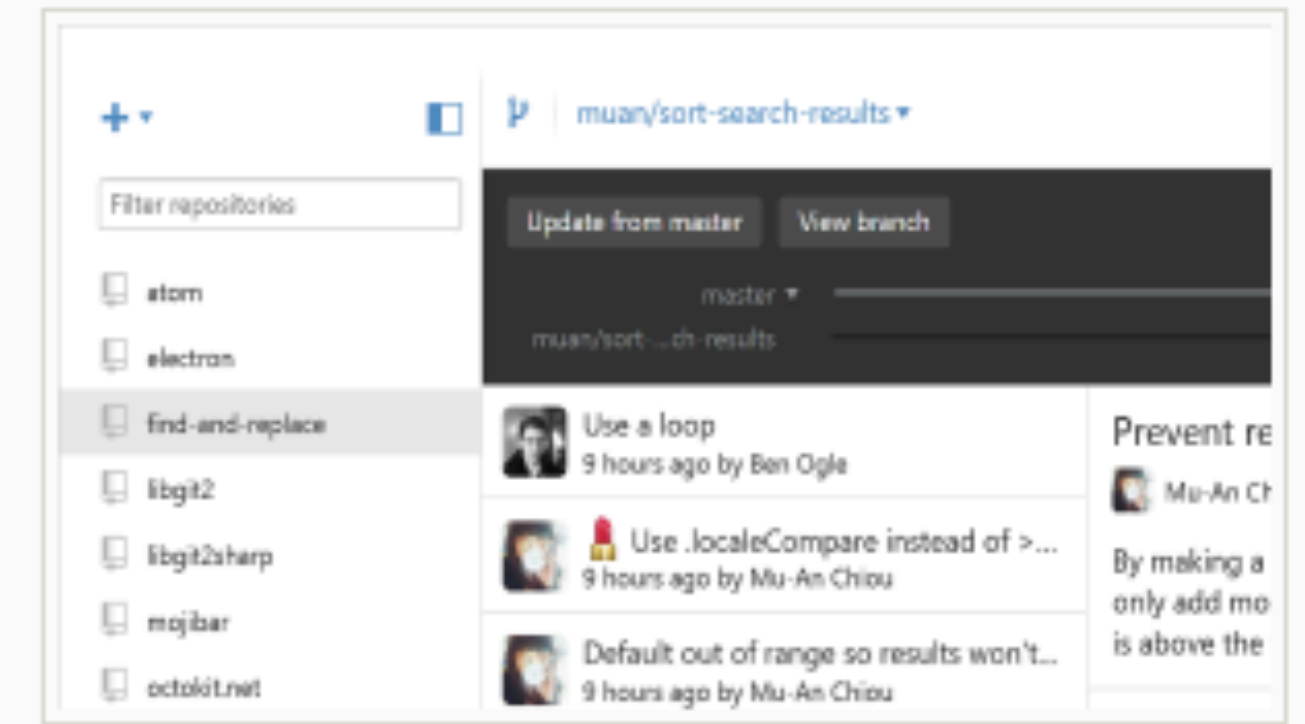
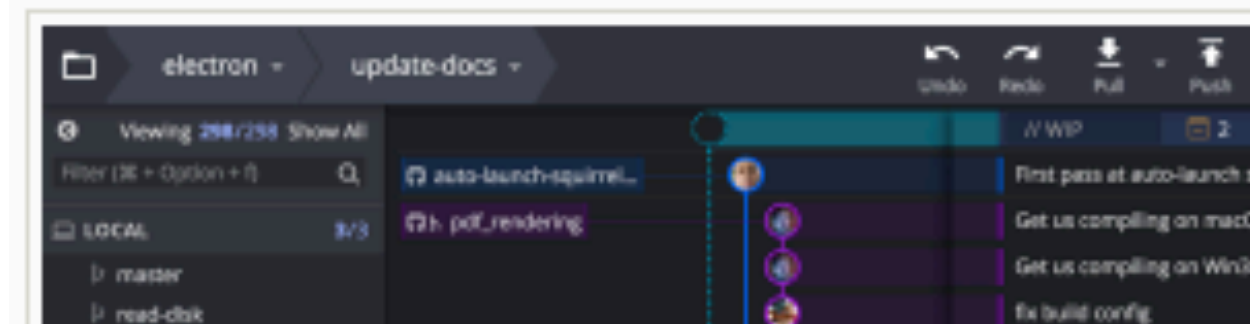


## TortoiseGit

Platforms: Windows

Price: Free

License: GNU GPL

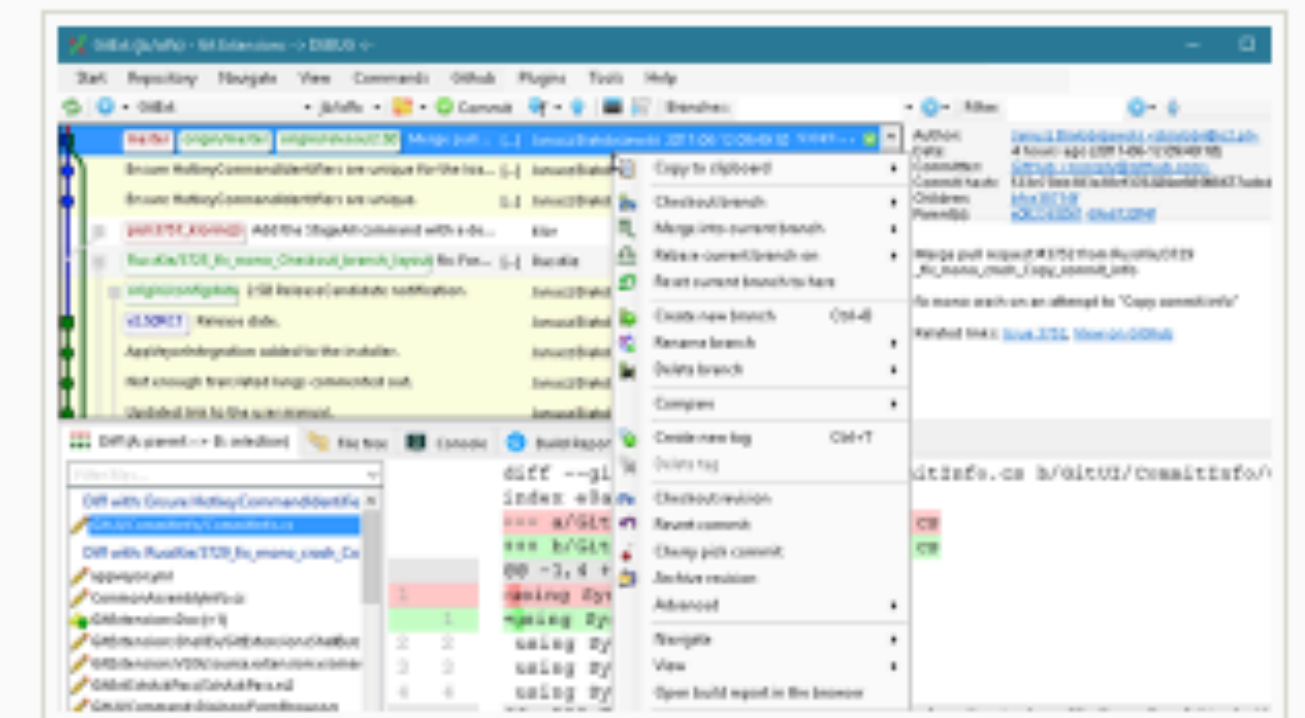


## GitHub Desktop

Platforms: Mac, Windows

Price: Free

License: MIT

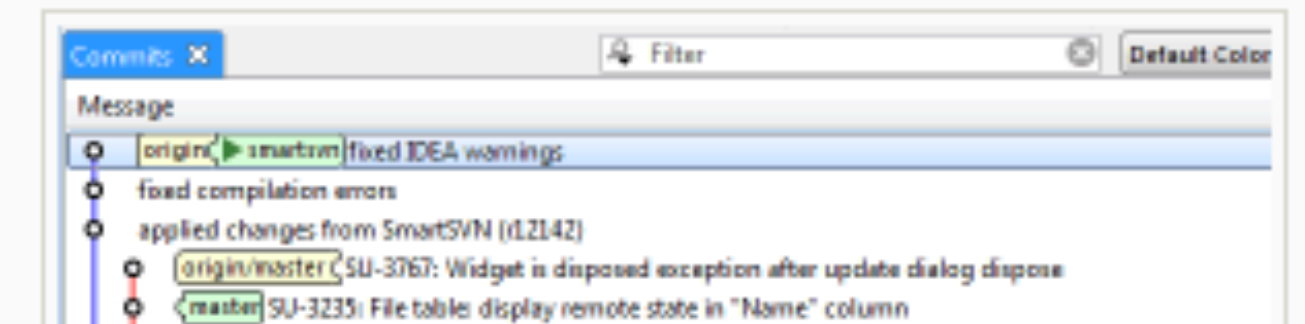


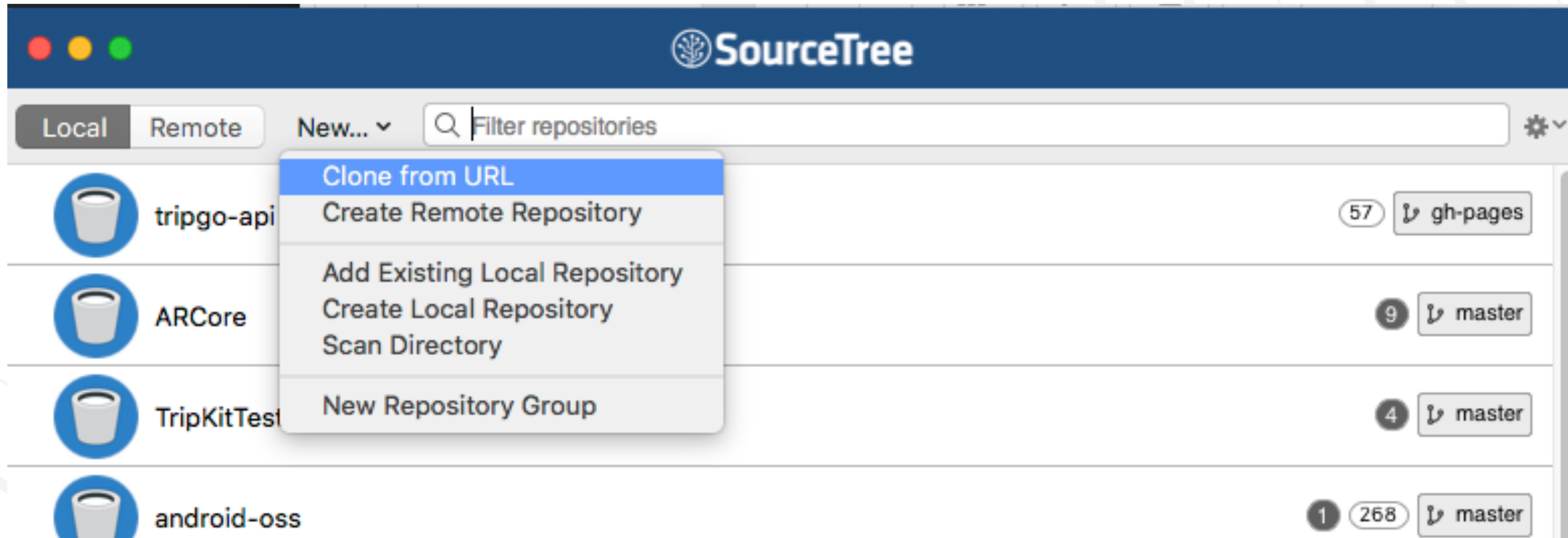
## Git Extensions

Platforms: Windows

Price: Free

License: GNU GPL





## Clone a repository

Source URL:

Destination Path:  ...

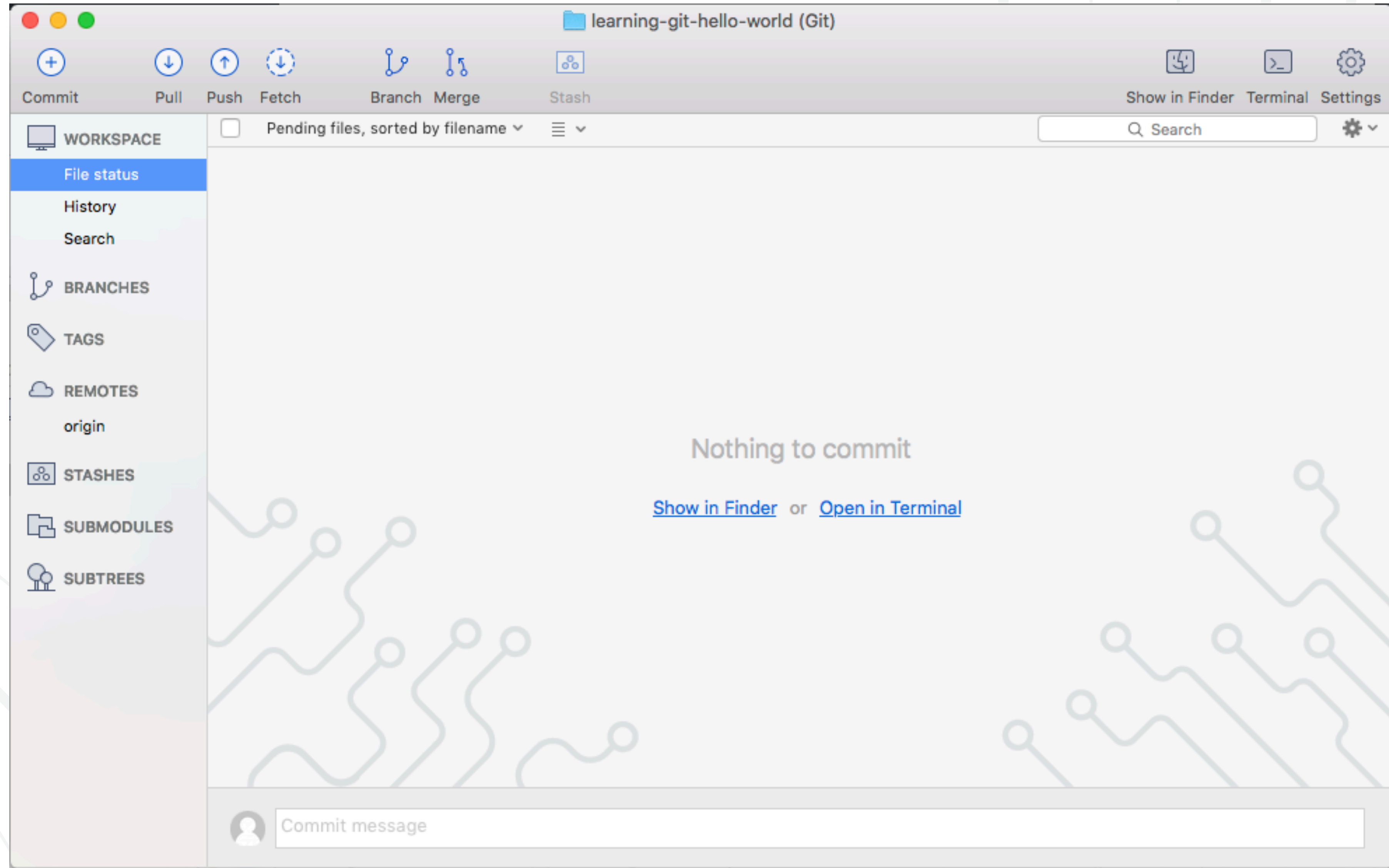
Name:

▶ Advanced Options

This is a Git repository

Cancel

Clone





New Java Project

**Create a Java Project**  
Create a Java project in the workspace or in an external location.

Project name:

Use default location

Location:

JRE

Use an execution environment JRE:

The screenshot displays the SourceTree application window titled "learning-git-hello-world (Git)". The interface is divided into several sections:

- Toolbar:** Contains icons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. On the right, there are icons for Show in Finder, Terminal, and Settings.
- Left Sidebar:** Lists workspace components: WORKSPACE, File status (4), History, Search, BRANCHES, TAGS, REMOTES (origin), STASHES, SUBMODULES, and SUBTREES.
- File List:** Shows pending files sorted by filename:
  - .classpath
  - .gitignore
  - .project
  - src/example/Main.java
- Code Editor:** Displays the contents of "src/example/Main.java":

```
1 package example;  
2  
3 public class Main {  
4     public static void main(String[] args) {  
5         System.out.println("Hello Git Workshop!");  
6     }  
7 }  
8  
9 }  
10  
11 }
```
- Bottom Bar:** A "Commit message" field with a user icon on the left and a "Stage hunk" button on the right.

The screenshot shows the SourceTree application window titled "learning-git-hello-world (Git)". The interface includes a top toolbar with icons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. A sidebar on the left lists workspace elements: WORKSPACE, File status (4), History, Search, BRANCHES, TAGS, REMOTES (origin), STASHES, SUBMODULES, and SUBTREES. The main workspace area displays a list of files: .classpath, .gitignore, and .project. A context menu is open over the .classpath file, listing actions such as Open, Show In Finder, Copy Path To Clipboard, Open In Terminal, Quick Look, External Diff, Create Patch..., Apply Patch..., Add to index, Unstage from index, Remove, Stop Tracking, Ignore... (highlighted), Commit Selected..., Reset..., Reset to Commit..., Resolve Conflicts, and Custom Actions. The right pane shows the content of the selected .classpath file, which is an XML configuration for an Eclipse project.

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
  <classpathentry kind="src" path="src"/>
  <classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER"/>
  <classpathentry kind="output" path="bin"/>
</classpath>
```

The screenshot shows the SourceTree application window titled "learning-git-hello-world (Git)". The interface includes a toolbar with icons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. On the right side of the toolbar are "Show in Finder", "Terminal", and "Settings" buttons. A search bar is located below the toolbar.

The left sidebar contains a "WORKSPACE" section with "File status" (4), "History", and "Search". Below this are "BRANCHES", "TAGS", "REMOTES" (with "origin" listed), "STASHES", "SUBMODULES", and "SUBTREES".

The main area shows a file tree with "src/example/Main.java" selected. A context menu is open over this file, listing the following options:

- Open
- Show In Finder
- Copy Path To Clipboard
- Open In Terminal
- Quick Look
- External Diff
- Create Patch...
- Apply Patch...
- Add to index** (highlighted)
- Unstage from index
- Remove
- Stop Tracking
- Ignore...
- Commit Selected...
- Reset...
- Reset to Commit...
- Resolve Conflicts ▶
- Custom Actions ▶
- Log Selected...
- Blame Selected...

The right pane displays the "File contents" for "src/example/Main.java" with a "Stage hunk" button. The code is as follows:

```
1 package example;  
2  
3 public class Main {  
4  
5     public static void main(String[] args) {  
6  
7         System.out.println("Hello Git Workshop!");  
8  
9     }  
10  
11 }
```

At the bottom of the window, there is a "Commit message" input field.

The screenshot shows the SourceTree application window for a project named "learning-git-hello-world". The interface includes a toolbar with icons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. Below the toolbar is a sidebar with navigation options: WORKSPACE, History, Search, BRANCHES, TAGS, REMOTES (with "origin" listed), and STASHES. The main workspace area displays "Pending files, sorted by filename" with a list containing one file: "src/example/Main.java". This file is selected, and its contents are shown in a preview pane on the right. The file contents are as follows:

	File contents
1	package example
2	
3	public class Ma
4	
5	public stat
6	
7	System.o
8	
9	}
10	
11	}

The screenshot shows the SourceTree application window titled "learning-git-hello-world (Git)". The interface includes a top toolbar with icons for commit, branch, merge, and stash. A sidebar on the left lists workspace and repository views. The main area shows a file "src/example/Main.java" with its contents: "package example;" and "public class Main {". A commit dialog is open, showing the user "Emmanuel Lagarrigue" and the commit message "initial commit".

learning-git-hello-world (Git)

Commit your current changes | Branch Merge Stash | Show in Finder Terminal Settings

WORKSPACE | Pending files, sorted by filename | Search

File status 1 | src/example/Main.java | src/example/Main.java | File contents | Stage hunk

```
1 package example;
2
3 public class Main {
4
```

Emmanuel Lagarrigue <emmanuel.lagarrigue@black-tobacco.com> | Commit Options...

initial commit

Push changes immediately to origin/master | Cancel | Commit

The screenshot shows the SourceTree application window for a repository named "learning-git-hello-world (Git)". The interface includes a top toolbar with icons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash, along with buttons for Show in Finder, Terminal, and Settings. A left sidebar contains navigation options: WORKSPACE, File status, History (selected), Search, BRANCHES, TAGS, REMOTES (origin), STASHES, SUBMODULES, and SUBTREES. The main workspace is divided into three sections: a commit history table, a file browser, and a commit details panel.

Graph	Description	Commit	Author	Date
	↵ master initial commit	748b267	Emmanuel Lagar...	Today at 4:50 PM

Sorted by path ▾

src/example/Main.java

File contents

```
1 + package example;
2 +
3 + public class Main {
4 +
5 +     public static void main(String[] args) {
6 +
7 +         System.out.println("Hello Git Workshop!");
8 +
9 +     }
10 +
11 + }
```

initial commit

Commit: 748b2672857e15bef84bc63  
Parents: Emmanuel Lagarrigue <emr...>  
Author: November 25, 2017 at 4:50:  
Date:  
Labels: HEAD -> master

learning-git-hello-world (Git)

Commit Pull Push Fetch Branch Merge Stash Show in Finder Terminal Settings

Push to repository:

Branches to push

Push?	Local branch	Remote branch	Track?
<input checked="" type="checkbox"/>	master	master	<input checked="" type="checkbox"/>
<input type="checkbox"/>			<input type="checkbox"/>
<input type="checkbox"/>			<input type="checkbox"/>

Select All



Push all tags

Cancel OK

WO  
File  
Hist  
Sea  
BRA  
TAG  
REM  
orig  
STA  
SUBMODULES



https://github.com/elagarrigue/learning-git-hello-world

 This repository Search Pull requests Issues Marketplace Explore + 

 **elagarrigue / learning-git-hello-world**

 Unwatch 1  Star 0  Fork 0

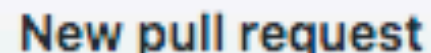
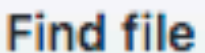
**<> Code**  Issues 0  Pull requests 0  Projects 0  Wiki  Insights  Settings


Git workshop repo example


Edit


Add topics

 1 commit  1 branch  0 releases  1 contributor

Branch: master     

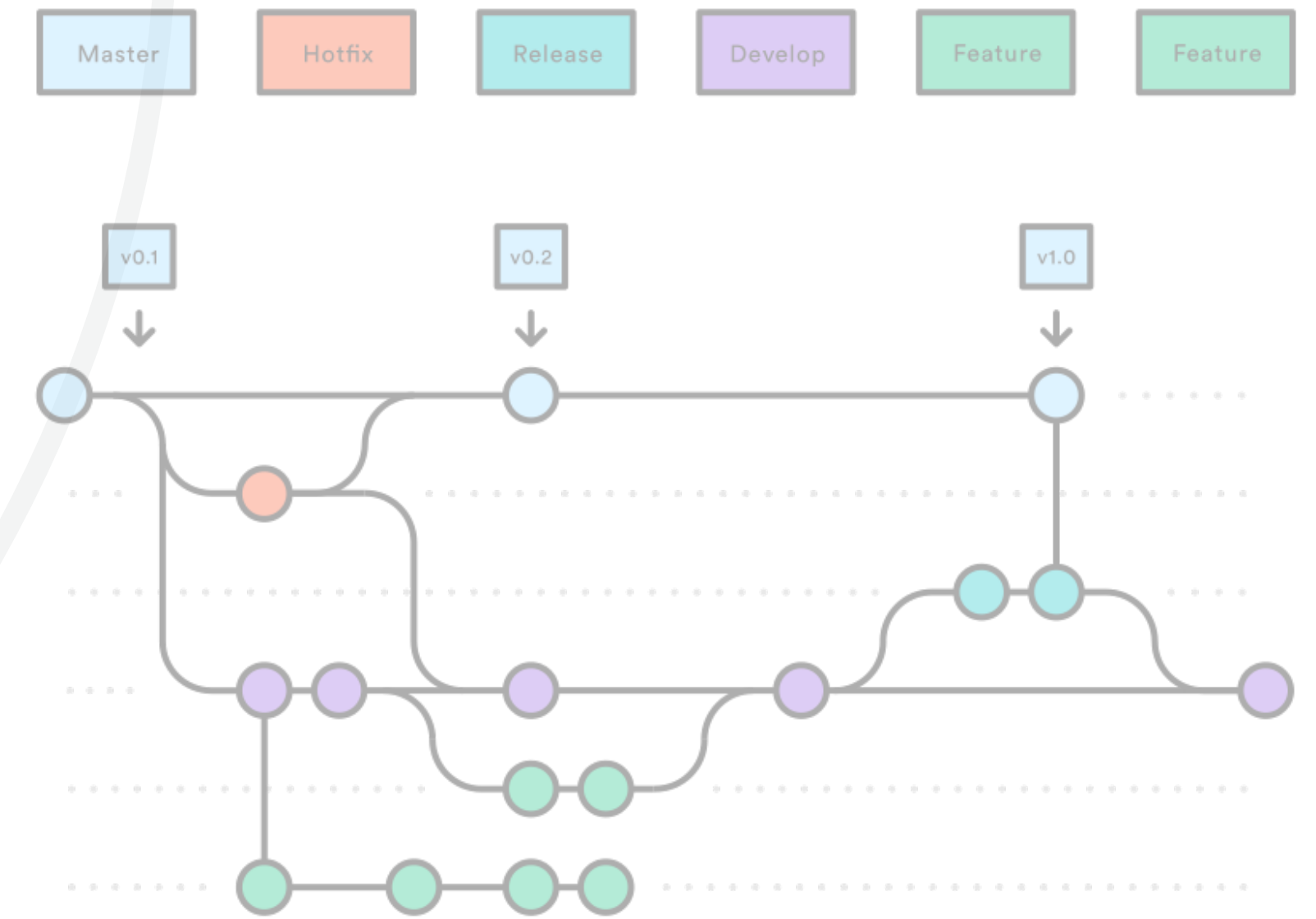
 **elagarrigue** initial commit Latest commit 701f34c 20 hours ago

 [src/example](#) initial commit 20 hours ago

Help people interested in this repository understand your project by adding a README. 

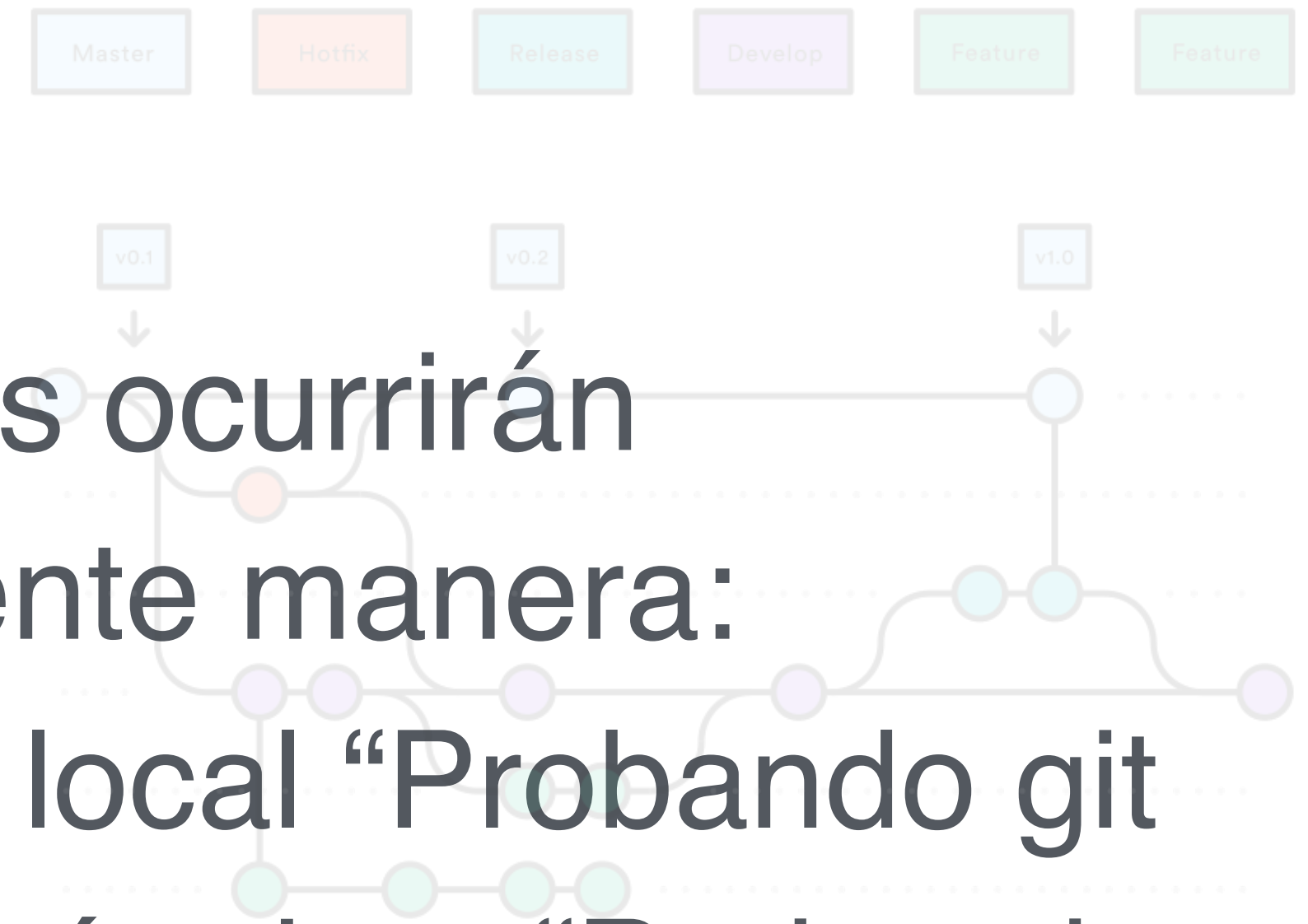
# Práctica:

- Armar equipos de al menos dos alumnos.
- Un miembro del equipo debe crear un programa “Hello World” en su lenguaje de programación favorito. La salida del programa debe ser el string “Probando git”.
- Cada miembro debe clonar el brach principal, y modificar el string de salida para que muestre “Probando git con Rick”.



# Práctica:

- A medida que se van subiendo los cambios ocurrirán conflictos. Debes solucionarse de la siguiente manera:
  - Supongamos que Rick realiza el cambio local “Probando git con Rick” y Morty hace lo mismo en su máquina, “Probando git con Morty”.
  - Si Rick hace push primero, cuando Morty quiera hacer push, git obligará a hacer pull antes, ya que hubo cambios.
  - Al hacer pull, se mezclará el código y surgirá el conflicto.
  - Debe solucionarlo agregando su nombre al final: “Probando git con Rick y con Morty”.



# Buenas Prácticas:



- Usar mensajes de commit descriptivos
- Cada commit tiene que ser una unidad lógica
- Mantener la copia local actualizada
- Mantener el repositorio actualizado

