Control de versiones (git)

Álvaro González Sotillo

5 de febrero de 2026

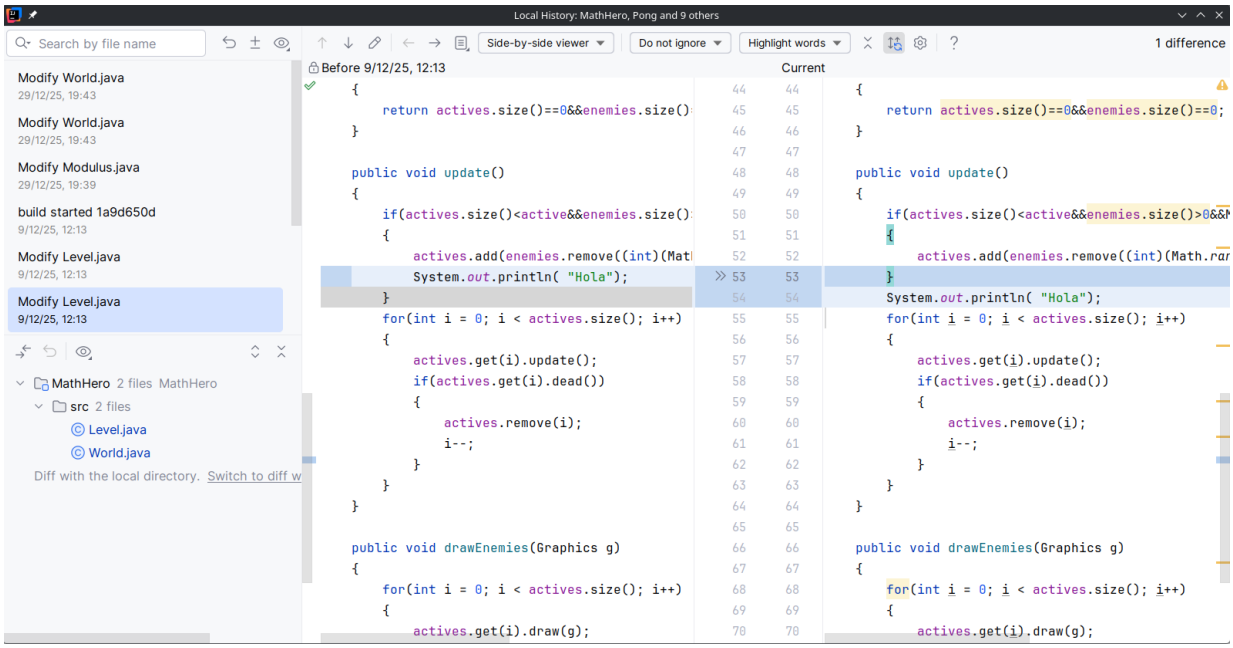
Índice

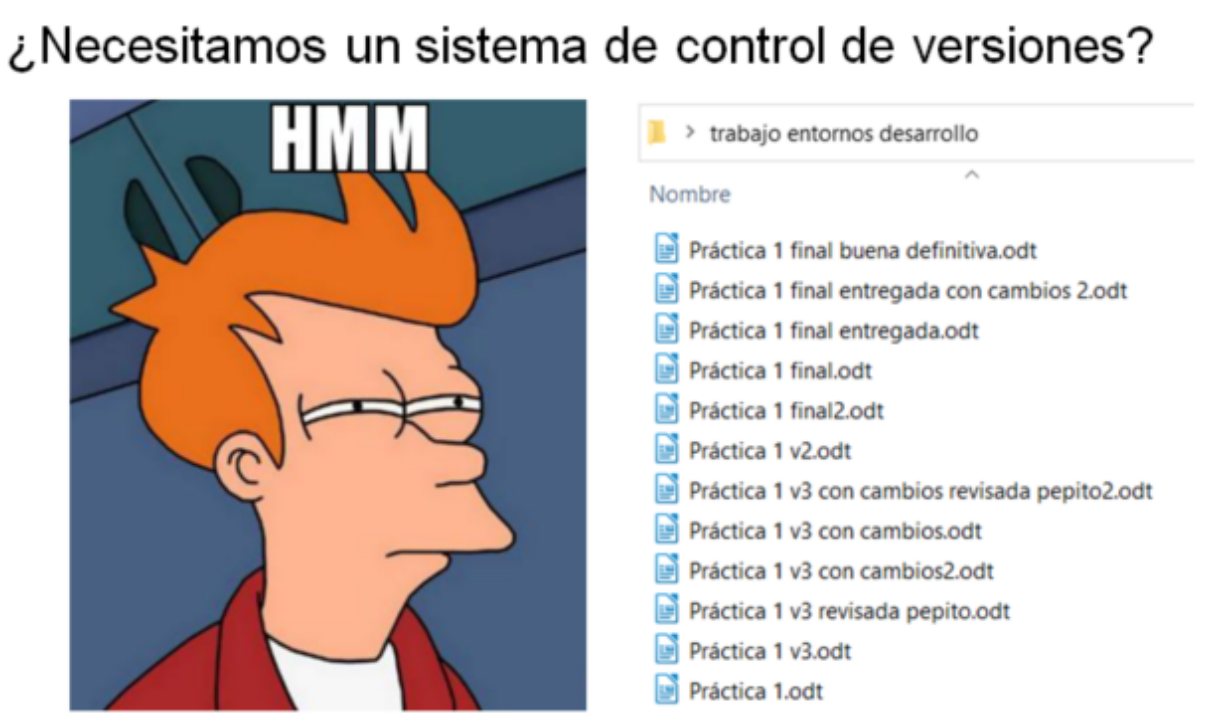
1. Introducción

- Problemas típicos en el desarrollo de aplicaciones
  - ¿Cómo trabajan varios desarrolladores sobre el mismo código?
  - ¿Qué cambios se han realizado?
  - ¿Quién ha realizado los cambios?

2. Trabajo en solitario

- Los IDE suelen tener *historia local*
- Indica los cambios que ha tenido el código en la máquina local





3. Trabajo colaborativo

3.1. Integración de código

- Cada miembro del equipo realiza su parte del trabajo
- Si hay errores
  - ¿Cómo se aislan los cambios que dieron lugar a un *bug*?
- Si hay más de una versión
  - ¿Cómo se recupera una versión antigua?
  - ¿Cómo se fusionan dos versiones?

3.2. Control de versiones

- Un control de versiones (de código) mantiene una base de datos con todos los cambios producidos sobre el código fuente y otros ficheros necesarios para generar una aplicación
- Pueden ser
  - Centralizados/distribuidos
  - Basados en ficheros/Basados en *changesets*
  - Bloqueantes/No bloqueantes
- Ejemplos
  - GIT
  - Subversion
  - Mercurial

  Control de versiones en Wikipedia

3.3. Conceptos de control de versiones (I)

- **Repositorio**: Base de datos con las sucesivas versiones
- **Copia de trabajo**: Ficheros sobre los que trabaja el programador
- ***Commit***: Confirmación de una nueva versión (de la copia de trabajo al repositorio)
- ***Checkout***: (Re)generación de la copia de trabajo a partir de la información del repositorio
- ***Changeset***: Conjunto de cambios en el repositorio, considerado atómico

3.4. Conceptos de control de versiones (II)

- **Rama (*Branch*)**: Versión paralela (ni anterior, ni posterior)
- **Rama `master` o `main`**: Rama principal del desarrollo
- **Fusión (*Merge*)**: Acumular cambios en una misma versión
  - Tras un *checkout*, mezclando la copia de trabajo con la versión del repositorio
  - Entre diferentes *branchs* del repositorio
- **Comentario**: Cada *commit* debería documentar sus efectos y motivaciones.

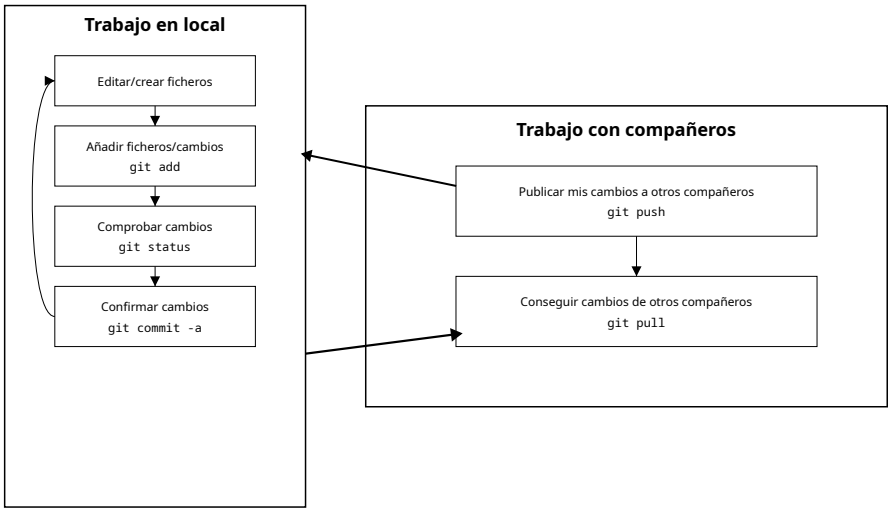
4. GIT

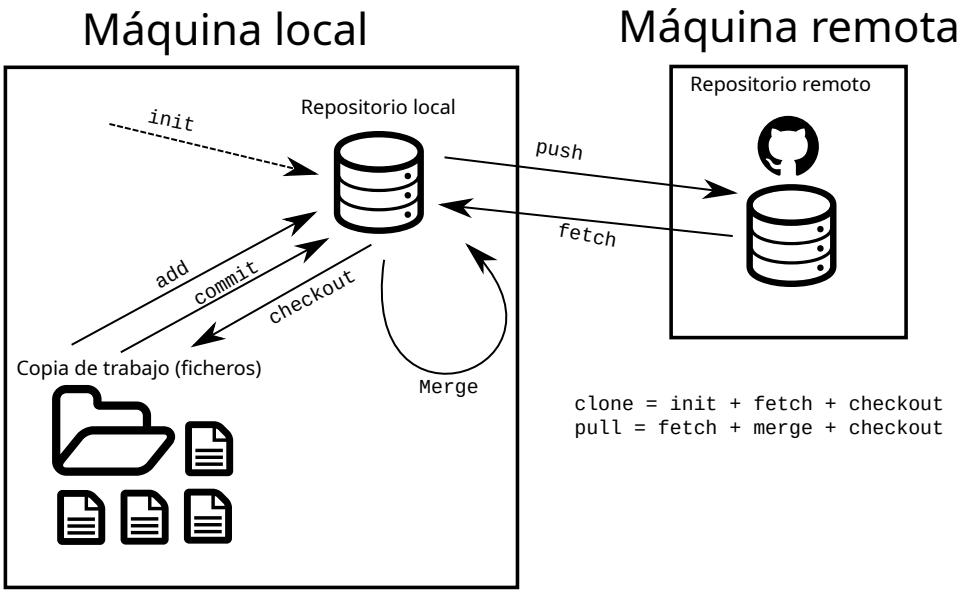
- Gratis, de código abierto
- Muy popular, en múltiples plataformas
- Distribuido
- Basado en *changesets*
- No bloqueante

4.1. GIT como cliente-servidor

- Git tiene una arquitectura peer-to-peer
  - Todos los repositorios son igual de "*importantes*"
- Por convenio, suele haber un repositorio "*central*"
  - Como servicio: github, gitlab, codeberg...
  - Autoalojado: gitea, forjego
  - O solo con Git y linux

4.2. Flujo de trabajo básico





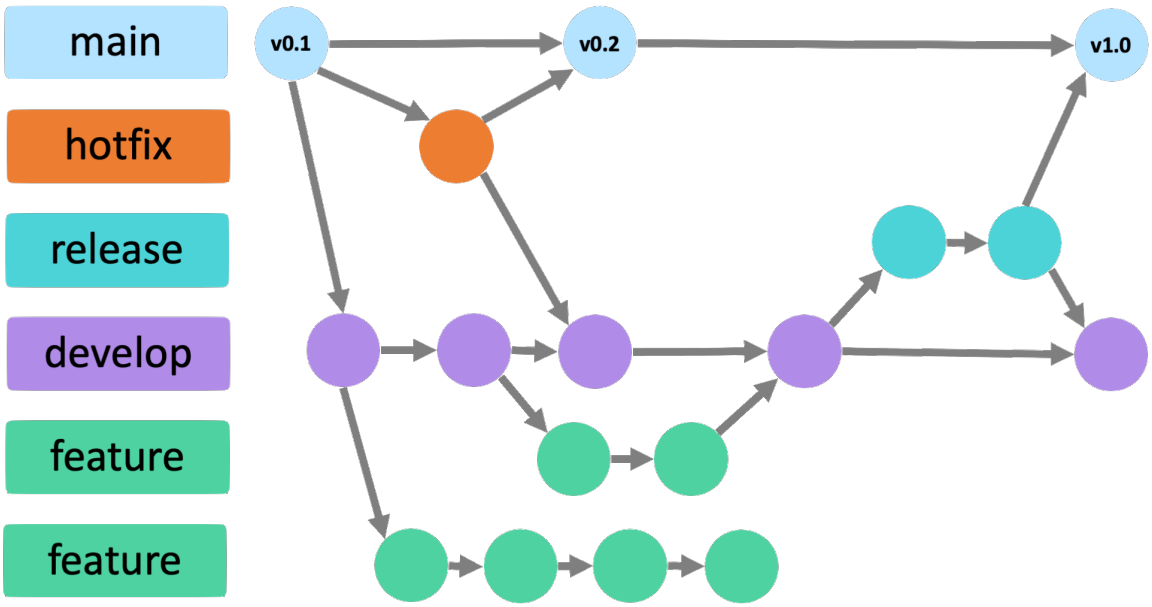
4.3. Ramas

- Cada *commit* genera una nueva versión del código
- Generalmente, una versión "*desciende*"de otra versión
  - En un *merge*, se puede "*descender*"de más de una versión
- Las versiones se identifican por un *hash*

- Una **rama** (`branch`) es un nombre que se da a un *hash*, y que va avanzando al hacer nuevas versiones
- Un `tag` es como un `branch`, pero no avanza al hacer nuevas versiones

4.3.1. Ramas típicas

- `master` o `main`: Versiones oficiales del código
  - Se garantiza que funcionan
  - Se entregan al usuario final
- `develop`: Trabajo para las siguientes versiones oficiales
  - Se acaban fusionando (*merge*) en `master`
- `feature`: Trabajo para una sola funcionalidad
  - Se acaban fusionando en `develop`
  - Pueden ser personales: solo las utiliza un desarrollador
- `hotfix`: Un arreglo urgente en `master`



5. Comandos de Git

5.1. `init`

- Crea un nuevo repositorio en el directorio
- Inicialmente vacío
- Se almacena en el directorio oculto `.git`

```
alvaro@debian8-64-alvarogonzalez:~/aplicacion-web$ git init
Initialized emp
```

5.2. clone

- Crea un nuevo repositorio, copia de uno ya existente
- El repositorio puede ser local o remoto, dependiendo de la URL
- La copia de trabajo será la de la rama `master`

```
alvaro@debian8-64-alvarogonzalez:~/aplicacion-web$ git clone https://github.com/alvarogonzalezsotillo/aplicacion-php.
    ↪ git
Cloning into 'aplicacion-php'...
remote: Counting objects: 50, done.
remote: Compressing objects: 100%
....
```

Comando *git clone*

5.3. add

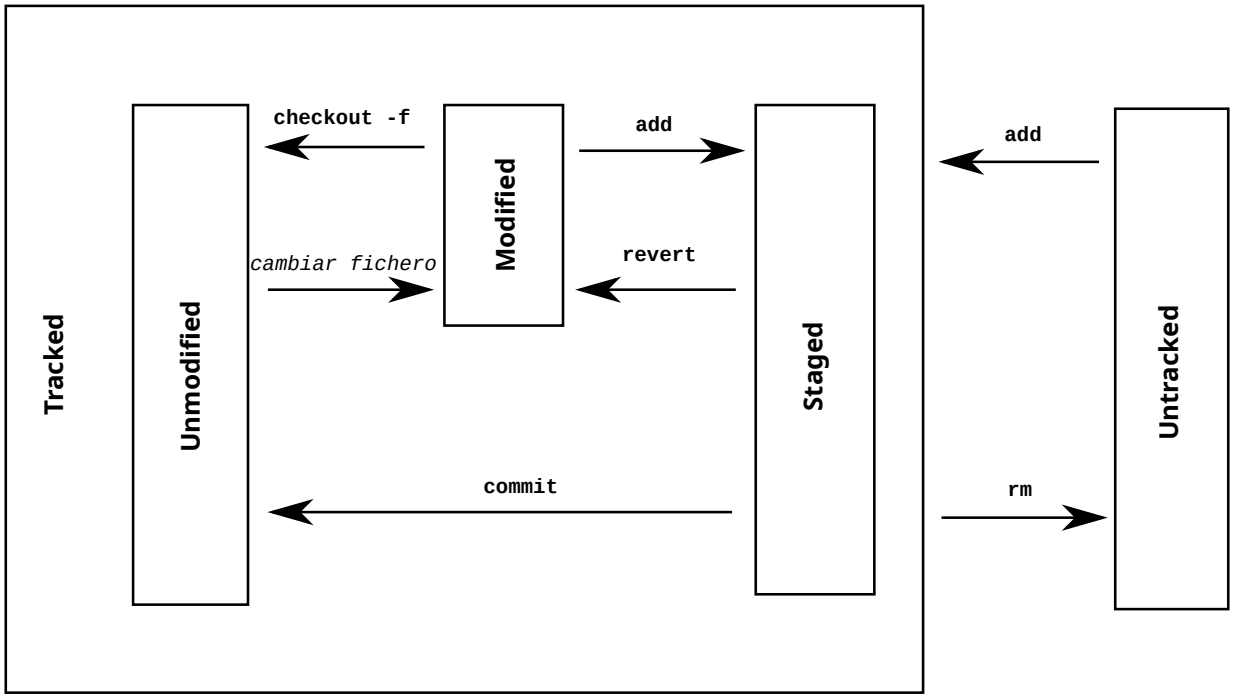
- Git no incluye todos los ficheros de la copia de trabajo en el repositorio
- Si se desea un fichero en el repositorio, se debe incluir con `add`
- Cada cambio posterior debe indicarse también con `add`
- Se puede quitar con `rm` (las versiones antiguas siguen en el repositorio)

Comando *git add*

5.4. status

- Imprime un resumen del estado de los ficheros de la copia de trabajo
  - **Untracked**: No guardados en el repositorio. Git ignora el fichero
  - **Unmodified**: Igual que en la rama activa del repositorio
  - **Modified**: Con cambios respecto al repositorio, pero que no está previsto guardar

Comando *git status*



5.5. diff

Muestra los cambios de los ficheros en estado **modified**

- `@@ -lineO,nlinesO +lineD,nlinesD @@`
  - Se muestra a partir de `lineO` del fichero antiguo y `lineD` del fichero actual
  - Se muestran `nlinesO` líneas del fichero antiguo y `nlinesD` líneas del fichero actual
- – Línea borrada
- + Línea añadida

Comando *git diff*

5.6. commit

```
[master 906fa20] Cambiados varios permisos de ejecución
 5 files changed, 411 insertions(+), 2 deletions(-)
 create mode 100644 09/estados-fichero-git.svg
 mode change 100755 => 100644 borrar-temporales-latex.sh
 mode change 100755 => 100644 generar-pdf.sh
 mode change 100755 => 100644 generar-pdfs.sh
 mode change 100755 => 100644 publicar-pdfs.sh
```

Comando *git commit*

5.7. push

- Sube las versiones del repositorio local a un repositorio remoto
- El repositorio remoto puede establecerse:
  - con `git clone` (en este caso se denomina **origin**)
  - con `git remote add`
- El repositorio debe tener todas las versiones del repositorio remoto
  - En otro caso, deberá realizarse un `git pull` previo

Comando *git push* Comando *git remote*

,numbers=none] alvaro@debian8-64-alvarogonzalez: /aplicacion-web$git push
Password for 'https://alvarogonzalezsotillo@b
To https://alvarogonzalezsotillo@bitbucket.org/alvarogonzalezsotillo/aplicacion-web.git
 ![rejected] master -> master (fetch first)
error: failed to push some refs to 'https://alvarogonzalezsotillo@bitbucket.org/alvarogonzalezsotillo/apli
web.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by anot
to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note a
forwards' in 'git push --help' for details.

5.8. pull

- Consigue la última versión de la rama actual

5.9. checkout

- Extrae versiones del repositorio a la copia de trabajo
  - **Un fichero**: `git checkout <fichero>`
  - **Una versión**: `git checkout <hashdeversión>`
  - **Un branch**: `git checkout <branch>`
- También crea nuevas ramas, con `git checkout -b <nombrerama>`
- Si coinciden una versión/nombre de fichero
  - Utilizar Referencias a objetos en Git
  - Es **mejor** evitar que coincidan

Comando *git checkout* Comando *git branch*

5.10. `merge`

- Fusiona la rama actual con otra rama
- En el caso de que haya conflictos:
  - Los ficheros se quedan con marcas de conflictos
  - Para aceptar el fichero completo de la otra rama: `git checkout --theirs path/file`
  - Para aceptar el fichero completo de la rama actual: `git checkout --ours path/file`
  - Para casos más complicados, editar los ficheros y confirmarlos con `git add`
- Recomendación: `git mergetool`, o usar el IDE

Comando *git merge*: Fusión de dos ramas

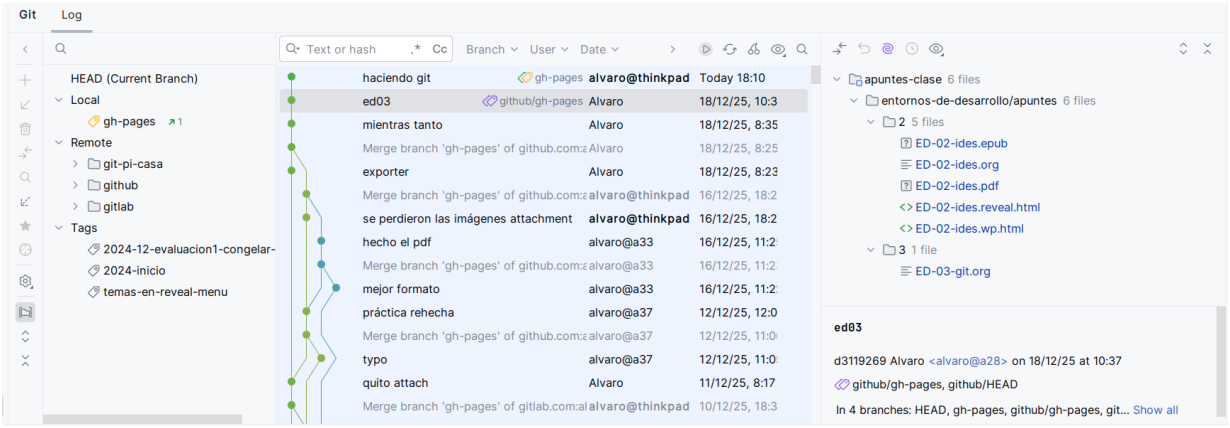
5.11. Otros comandos

- Comando *git blame*: Lista un fichero con el autor de cada línea
- Comando *git log*: Historia de cambios de un repositorio
- Comando *git gui*: Interfaz gráfica básica para utilizar git
- Comando *git switch*: Cambia rápidamente entre ramas
- Comando *git reset*: Deshace cambios (con precaución)
- Comando *git tag*: Da un nombre legible (no un *hash*) a una versión

6. Ejercicio GIT

- Clona el repositorio `https://codeberg.org/alvarogonzalezsotillo/Java-Games.git`
- Crea la rama `translate` a partir de `master` (comando `checkout` o `branch`)
- Traduce a otro idioma las interfaces de los juegos:
  - Tetris
  - MathHero
  - FallDown
- Incorpora los cambios a la rama `translate` (comando `add` y `commit`)
- Fusiona los cambios con la rama `master`
  - `git checkout master`
  - `git merge --no-ff translate`
- Enseña los resultados al profesor
  - `git log --graph --oneline`

7. Git desde Intellij



8. Ejercicio GIT (II)

- Crea una cuenta en Codeberg
  - Usa tu cuenta de correo de educamadrid
- Clona el repositorio `https://codeberg.org/alvarogonzalezsotillo/ejercicio-listado-alumnos.git`
  - Comando `clone`
- Extrae la rama `curso-2025-2026`
  - Comando `checkout`
- Modifica el fichero `listado-alumnos.md` y añade tu nombre y apellidos en el orden correcto
- Confirma tus cambios en local (`add`,`commit`)
- Sube tus cambios con `push`
  - Si algún compañero ha cambiado el fichero antes que tú, consigue sus cambios con `pull`, y vuelve a intentar subir tus cambios

9. ¿Qué es un *pull request* y un *fork*?

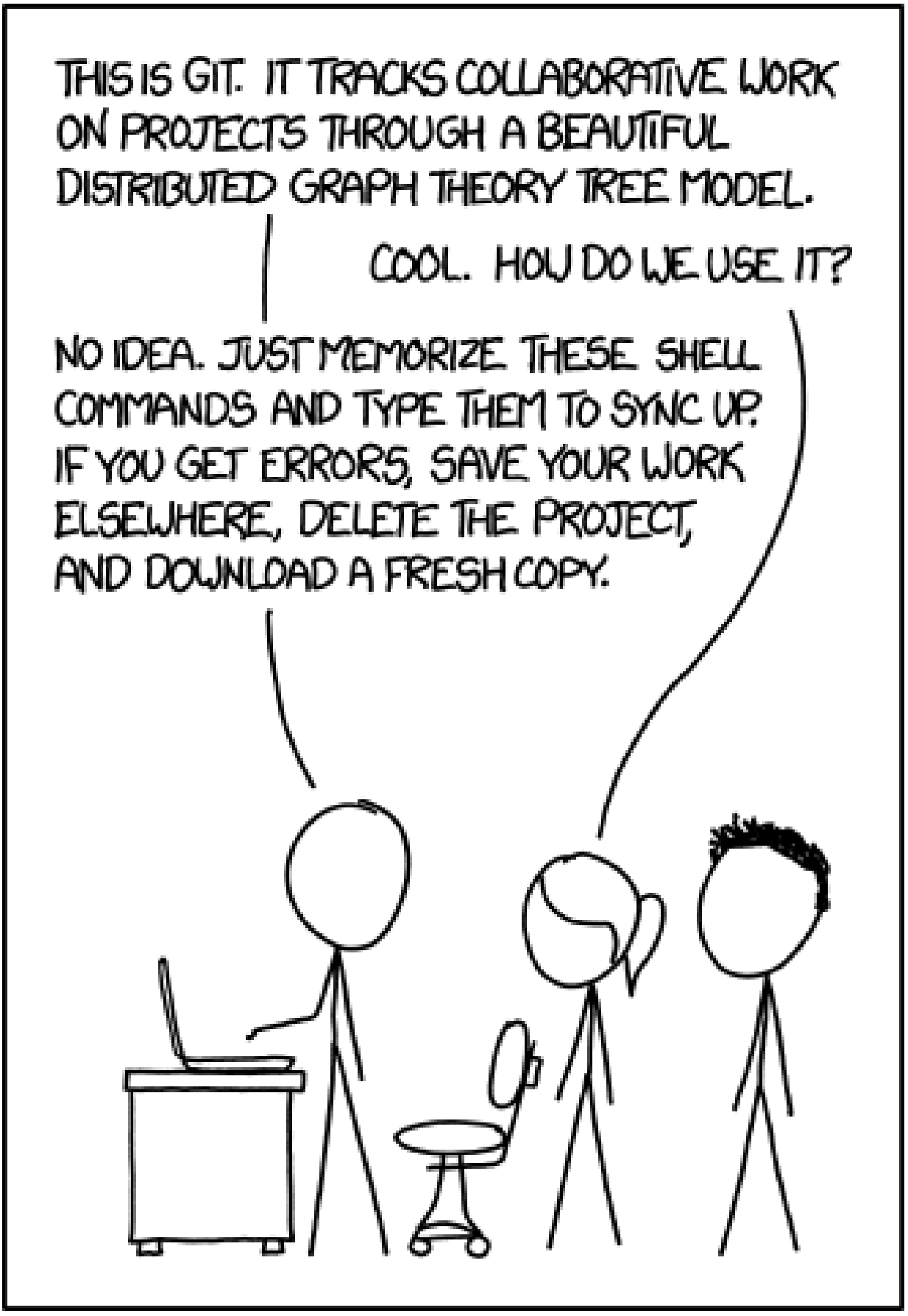
- Un PR es un concepto ajeno a Git
  - El equivalente de Git a un PR es un *push*
- En servicios de Git (como GitHub)
  - A veces se quieren hacer cambios en un repositorio donde no tenemos permisos de escritura
  - Se hace un *fork* (copia) de ese repositorio
  - Se realizan los cambios que se considere en esa copia
  - Finalmente, se pide que el dueño del repositorio original haga un *pull*
- Guía de Github

10. Codeberg en *selfhosting*: Forgejo

- Esta es una instalación no apta para producción, solo para pruebas
  - Basada en Podman
  - Con la imagen oficial
- En la página de instalación (`http:/localhost:3000/`)
  - seleccionar `SQLite3`
  - no deshabilitar autoregistro
- Entre otras cosas no tiene:
  - Una base de datos *seria*
  - Una forma de reiniciar el servidor
  - Permisos adecuados para el acceso a los datos en `forgejo-data`

```
podman run -d \
  --replace \
  --name=forgejo \
  -v forgejo-data:/data \
  -v /etc/localtime:/etc/localtime:ro \
  -p 3000:3000 \
  -p 2222:22 \
  codeberg.org/forgejo/forgejo:14
```

11. Si todo se complica...



Git cheat sheet

12. Referencias

- Formatos:
  - Transparencias
  - PDF
  - Página web
  - EPUB
- Alojado en Github