
# Introducción a **git**

**Dr. Pedro J. Molina**

METADEV

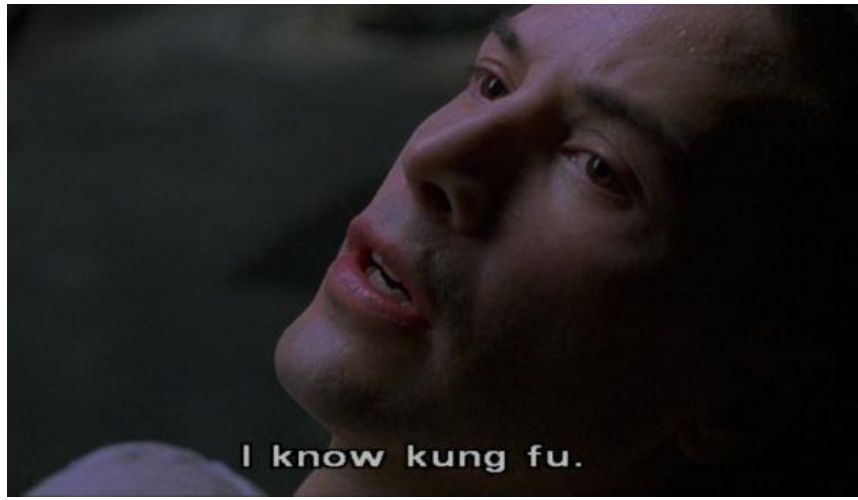
http://pjmolina.com

@pmolinam

# Objetivos

## Conocer **git** para

- poder usarlo en el día a día
- trabajo con R
- colaboración en equipo

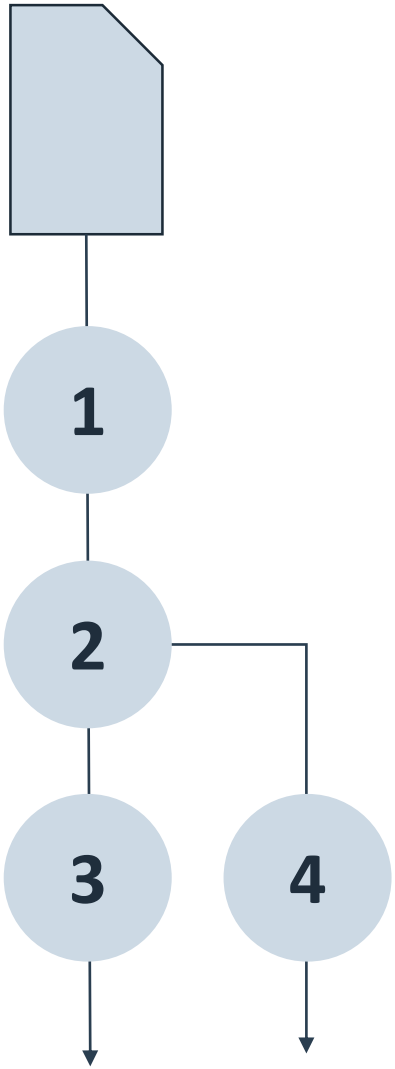


# Agenda

- Origen de git
- Conceptos básicos
- Línea de comando
- Herramientas gráficas
- Ignorar ficheros
- Estado de los cambios en Git
- Deshacer cambios
- Diferencias de ficheros

# Control de Versiones

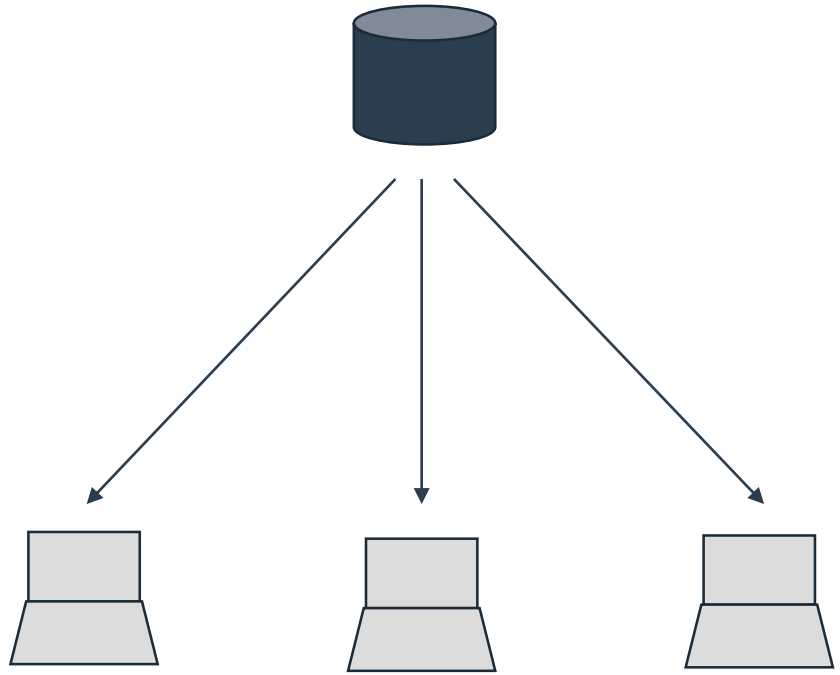
- Útiles para
  - Desarrollo de Software
  - Gestión de la configuración
  - Compartir y gestionar cambios a documentos (principalmente de texto)
  - Gestión de contenidos



# Control de Versiones Centralizados

- Los cambios se guardan en un repositorio central
- Los ficheros se bloquean en el servidor o se realizan mezclas ante cambios

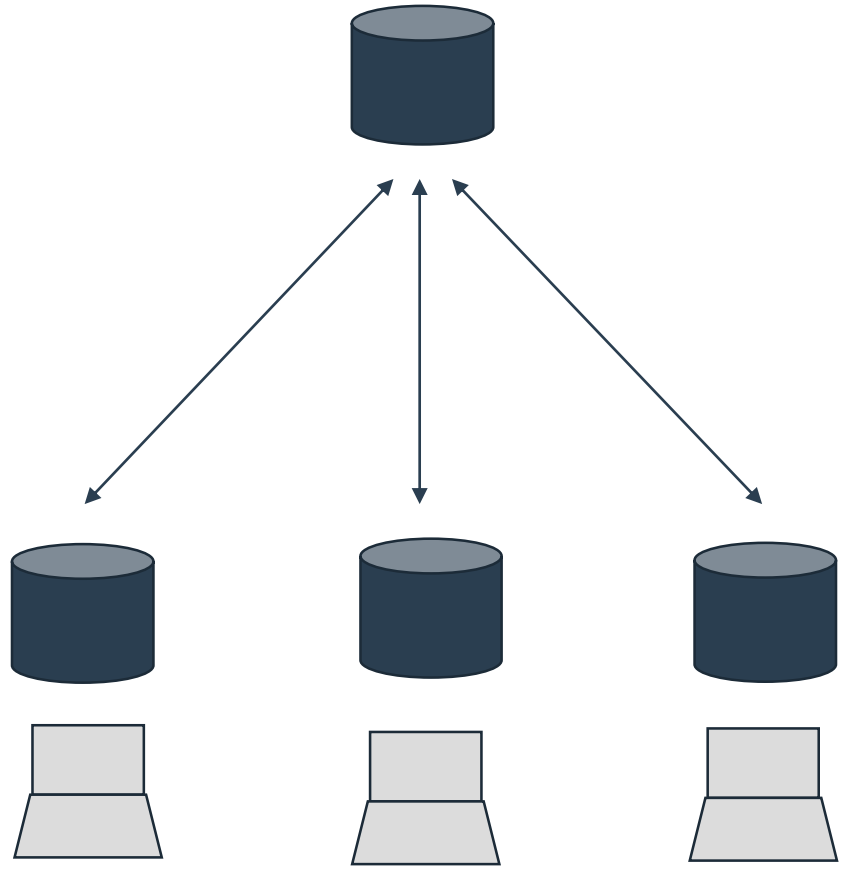
- Productos
  - SourceSafe
  - TFS
  - Subversion
  - CVS



# Control de Versiones Distribuidos

- Cada cliente trabajando con el repositorio tiene una copia local
- Se sincronizan bajo demanda
- Sin bloqueos
- Basados en ramas y mezclas

- Productos
  - Bazaar
  - Mercurial
  - Git
  - Plastic SCM

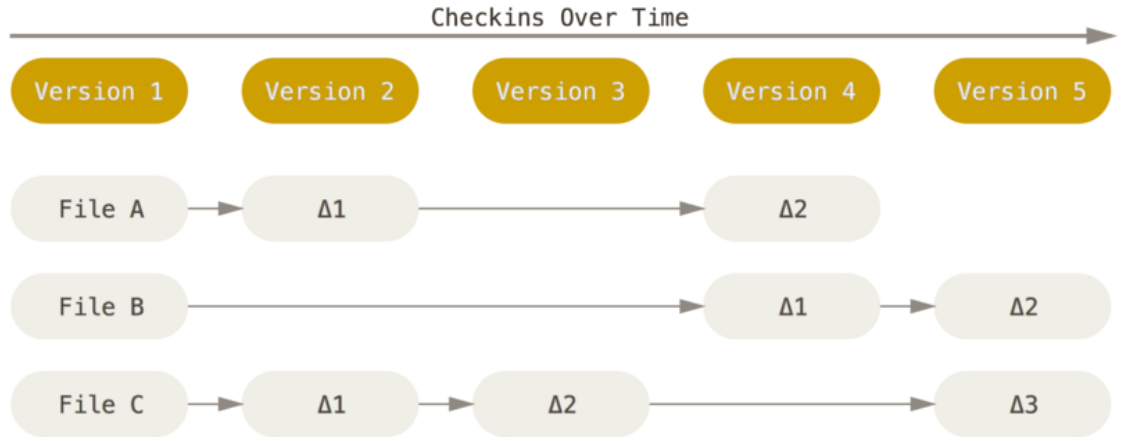


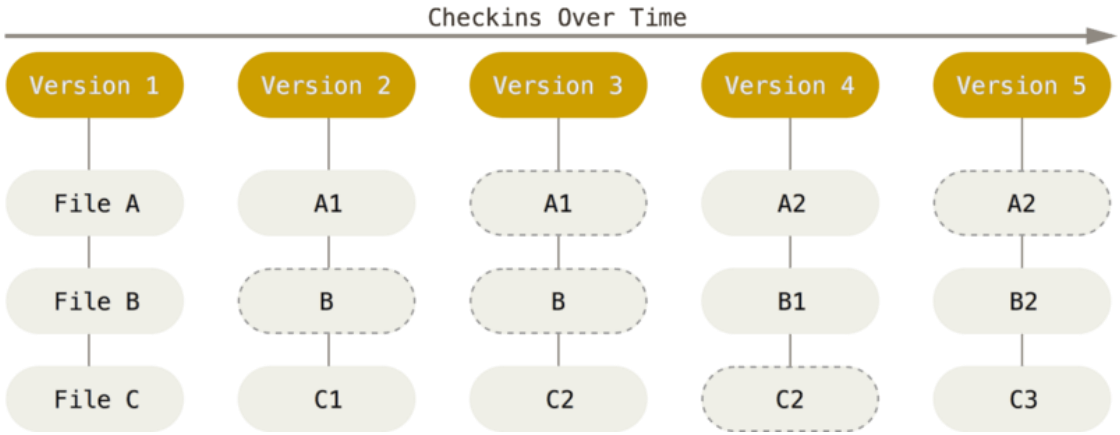
# Git

- Escrito por Linus Torvalds (el padre Linux) en 2005 para el desarrollo del kernel de Linux

# ¿Cómo funciona?

- Git almacena **deltas** (diferencias) entre ficheros
- De modo que con la versión inicial del fichero junto con la aplicación de los deltas en orden se puede recuperar cualquier versión del fichero.
- Emplea un almacenamiento indexado y eficiente para premiar la velocidad de acceso



# Instalación

- Windows

  https://git-scm.com/download/win

- Mac

  https://git-scm.com/download/mac

- Linux

  https://git-scm.com/download/linux

```
$ apt-get install git
```

# Conceptos básicos

- Repositorio .git/
- Versión/Revisión
- Commit
- Rama
- Instantánea/Snapshot
- Pull Request
- Push / Pull
- Merge
- Diff

# Herramientas

- Línea de comando (CLI)
  - Ejecución desde consola
- Herramienta visuales (GUI)
  - SourceTree (Win, Mac)
  - GitKraken (Win, Mac, Unix)
  - Github for Windows (Win)
  - Integraciones con IDEs como:
    - R Studio
    - Visual Studio
    - VS Code
    - Eclipse
  - Integracion con el SO / Explorer
    - TortoiseGit
  - Etc.

# Línea de comandos (CLI)

- Git surge como herramienta dentro de Linux y está influido por su filosofía:
  - Linux está orientado a pequeñas herramientas de bajo nivel
  - Donde capa pieza realiza una sola función
  - Por composición pueden realizar trabajos mas complejos

- Inicialmente git era una colección de librerías en C y scripts en Perl que han crecido de modo orgánico (poco organizado)

# Git - Comandos Básicos

## Configuración inicial. Nombre y correo.

```
# Set
$ git config --global user.name "Jesica Alba"
$ git config --global user.email jesica.alba@acme.com

# Query
$ git config --global user.name
Jesica Alba

$ git config --global user.email
jesica.alba@acme.com

git config --list
```

# Git - Comandos Básicos

## Ayuda básica

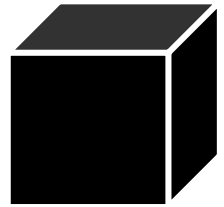
```
$ git



$ git help commit
```

# Git - Comandos Básicos

Crear un repositorio.



```
$ mkdir repo1
$ cd repo1
$ git init

Initialized empty Git repository
in /…/repo1/.git/
```

```
.git/
    hooks/
    info/
    objects/
    refs/
    config
    description
    HEAD
```

# Git - Comandos Básicos

Crear un nuevo fichero

```
$ echo Hola > ejemplo.txt
```

Ver el estado

```
$ git status
Untracked files:
        ejemplo.txt
```

# Git - Comandos Básicos

Añadir fichero al control de código fuente

```
$ git add ejemplo.txt
```

Ver el estado

```
$ git status
Changes to be committed:
    new file: ejemplo.txt
```

# Git - Comandos Básicos

## Commit

```
$ git commit -m “Primer fichero”
[master (root-commit) 3d39c01] Primer fichero
 1 file changed, 1 insertion(+)
 create mode 100644 ejemplo.txt
```

# Git - Comandos Básicos

## Clonar

```
$ git clone https://url.com:8443/repo1 directory
```

# Clientes Gráficos

- Normalmente casi todos los desarrolladores usan entornos de ventanas (Win, Mac, o X11 en Linux) para el desarrollo
- Visualizar la historia de un repositorio no es nada cómodo en modo texto (presentar grafos y arboles como ASCII art)
- Es muy habitual trabajar todo lo que se pueda con un cliente gráfico “sin bajar” a la línea de comando salvo cuando sea necesario
- Estos clientes integran con IDEs, el SO, o se ejecutan de modo aislado.

# Clientes Gráficos

- Github Desktop — Windows, Mac
- SourceTree — Windows, Mac
- GitKraken — Windows, Mac, Linux X11
- TortoiseGit — Windows

# Diferencias entre ficheros

- A la hora de gestionar versiones de ficheros es esencial determinar qué a cambiado de un fichero.

- En Unix existe el comando diff que permite hacer esto desde hace más de 20 años

# Diff

**file1**

```
1 A
2 B
3 C
4 D
```

**file2**

```
1 Inserto
2 A
3 B cambio
4 D
```

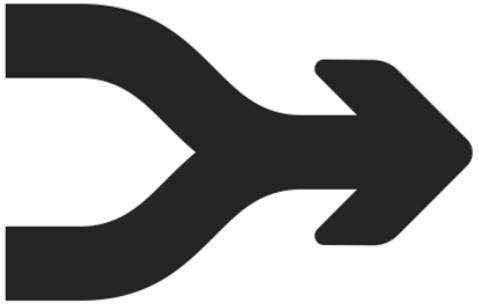
```
$ diff -u file1 file2
```

```
--- file1       2016-09-13 16:47:06.605417200 +0200
+++ file2       2016-09-13 16:45:43.444479900 +0200
@@ -1,4 +1,4 @@
+Inserto
 A
-B
-C
+B cambio
 D
\ No newline at end of file
```

```
$ diff -u file2 file1
```

```
--- file2       2016-09-13 16:45:43.444479900 +0200
+++ file1       2016-09-13 16:47:06.605417200 +0200
@@ -1,4 +1,4 @@
-Inserto
 A
-B cambio
+B
+C
 D
\ No newline at end of file
```

# Mezclas / Merge

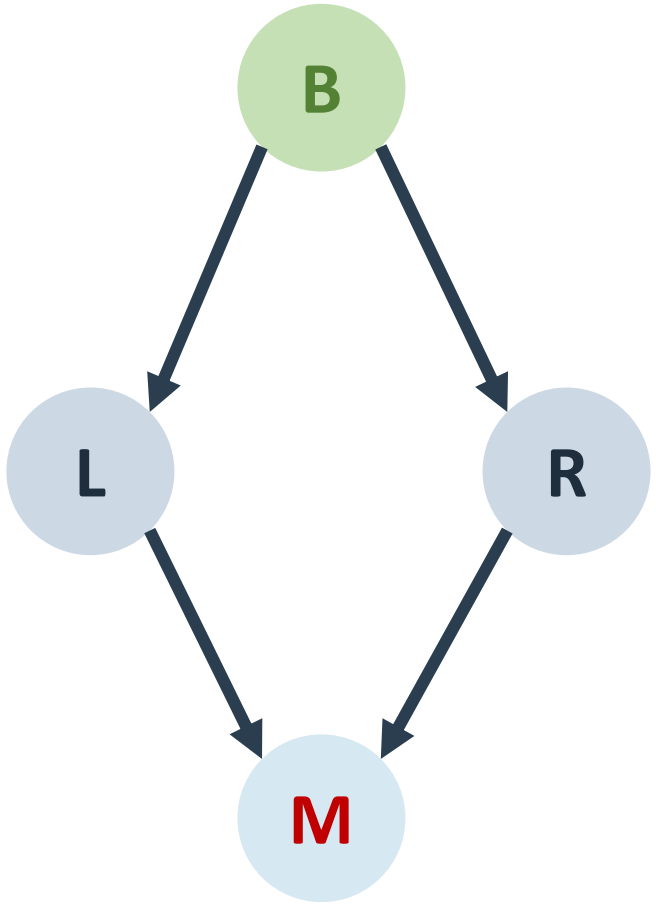


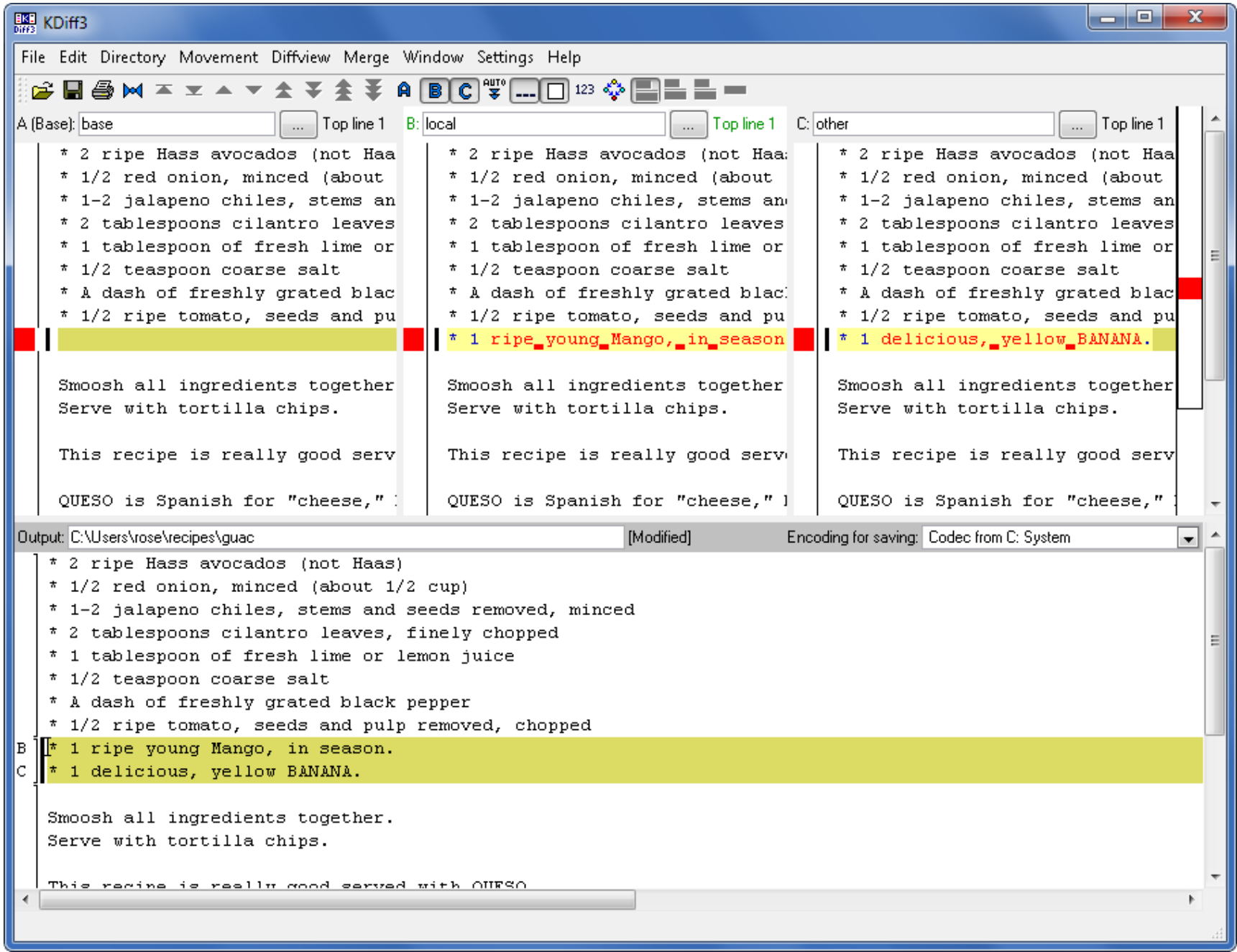
- Una mezcla es un commit que unifica cambios de **dos ramas en una.**
- La mezcla puede involucrar ramas, directorios o ficheros individuales.

- En muchos casos la mezcla es **trivial** y puede ser realizada **automáticamente**.
  - Añadir ficheros o carpetas
  - Borrar ficheros o carpetas (sin cambios en la otra rama)
  - Modificar ficheros o carpetas (sin cambios en la otra rama)
  - Cambios al mismo fichero triviales (en líneas diferentes)

- En otros casos no: y requiere de **intervención manual** para resolver la mezcla.
  - Cambios en las dos ramas no triviales (mismas líneas)

# 3-Way Diff


B
L
R
M


KDiff3
File Edit Directory Movement Diffview Merge Window Settings Help
A (Base): base
Top line 1
B: local
Top line 1
C: other
Top line 1
* 2 ripe Hass avocados (not Haas)
* 1/2 red onion, minced (about 1/2 cup)
* 1-2 jalapeno chiles, stems and seeds removed, minced
* 2 tablespoons cilantro leaves, finely chopped
* 1 tablespoon of fresh lime or lemon juice
* 1/2 teaspoon coarse salt
* A dash of freshly grated black pepper
* 1/2 ripe tomato, seeds and pulp removed, chopped
* 1 ripe young Mango, in season.
* 1 delicious, yellow BANANA.
Smoosh all ingredients together.
Serve with tortilla chips.
This recipe is really good served with QUESO
Output: C:\Users\rose\recipes\guac
[Modified]
Encoding for saving: Codec from C: System

# Trabajo con repositorios

- Inicializar

```
$ git init
```

- Clonar

```
$ git clone <url> [dir]
```

# Trabajo con ramas

- Una Rama (*branch*) es una secuencia versionada de cambios secuenciales
- Todo repositorio comienza con una rama única
- Se crea una rama cuando el árbol se bifurca → Y
- Crear una rama:

```
$ git branch rama1
```

# Trabajo con ramas

- Tras crear una rama, para trabajar con ella hay que **cambiar la rama de trabajo**:

```
$ git checkout rama1
```

- Las ramas se pueden mezclar entre ellas para reunificarlas con los comandos merge y rebase*:

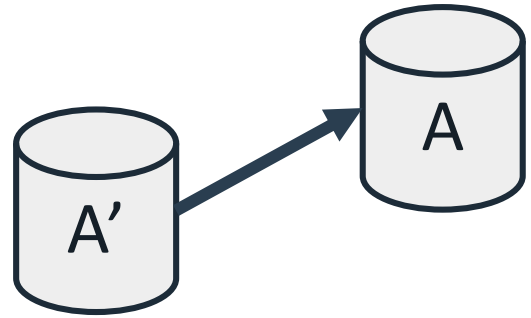
```
$ git merge master
```

```
$ git rebase master
```

# Proveedores de repositorios en la nube

- Github https://github.com
- Bitbucket https://bitbucket.org
- Gitlab https://about.gitlab.com
- Codeplex https://www.codeplex.com

# Trabajo con repositorios remotos

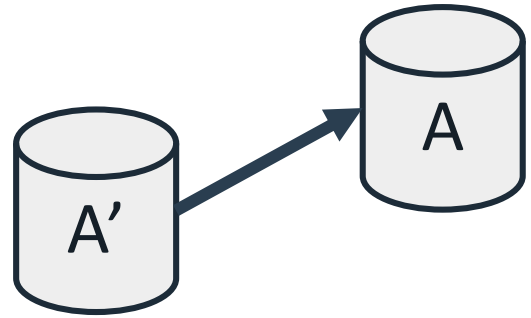



- La principal “gracia” de versionar cambios aparece cuando trabajamos en equipo.
- Debemos conocer que cambios han hecho los demás y añadir los nuestros de modo más limpio posible y distribuirlos.

- La operación para agregar miembros al equipo de trabajo comienza con una clonación del repositorio:

```
$ git clone <url>
```

# Trabajo con repositorios remotos



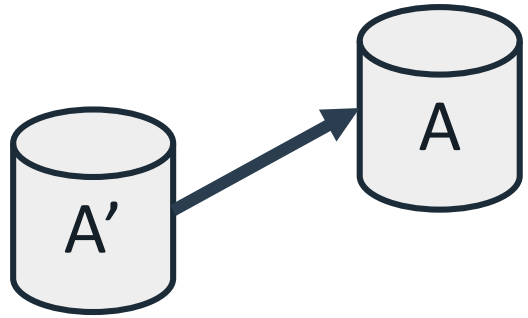

- Esquema más habitual: 1 repositorio central (github p.e.), N desarrolladores con clon local enlazado a github.

```
$ git clone <url>
```

- Tras clonar un repositorio, el clon recuerda (queda enlazado al padre con la denominación **origin**).

```
$ git remote -v
```

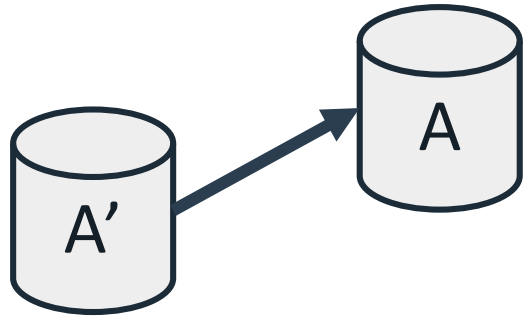
# Configuración de remotos




- Ver los remotos configurados

```
$ git remote -v
origin  https://github.com/pjmolina/baucis-swagger2 (fetch)
origin  https://github.com/pjmolina/baucis-swagger2 (push)
```

# Configuración de remotos




- Añadir repositorio remoto:

```
$ git remote add <alias> <url>
```

- Eliminar repositorio remoto:

```
$ git remote remove <alias>
```

# Ciclo de trabajo con repositorios compartidos

1. Traer cambios de los demás.  (PULL)

```
$ git pull
```

2. Realizar tus cambios. Mezclar si es necesario. (MERGE / COMMIT)

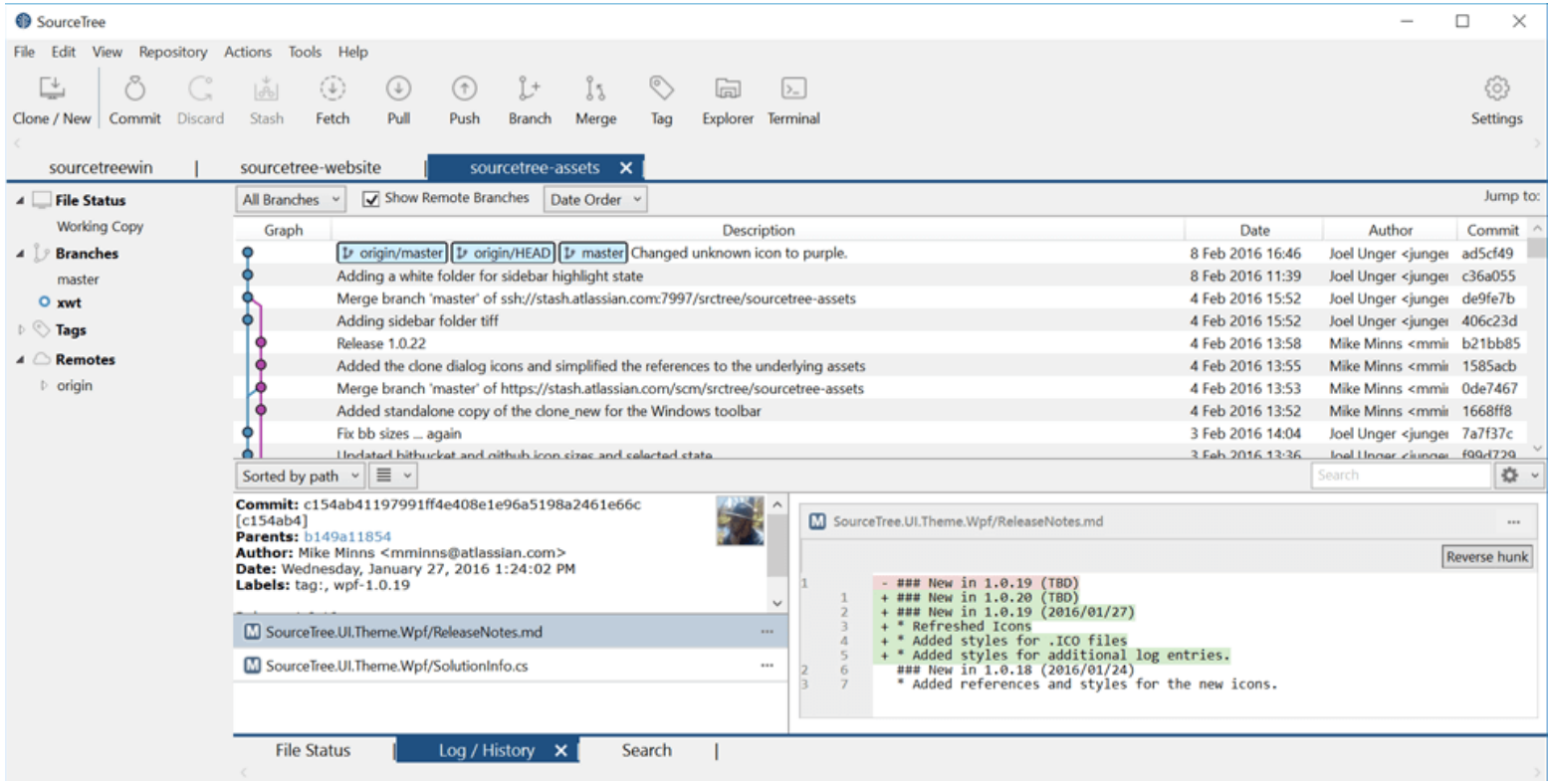
```
$ git merge / commit
```

3. Enviar cambios a los repositorio(s) remoto(s)  (PUSH)

```
$ git push origin master
```

# SourceTree

- Cliente de Atlassian
- Bitbucket




- https://www.sourcetreeapp.com

# Metodologías de gestión de ramas

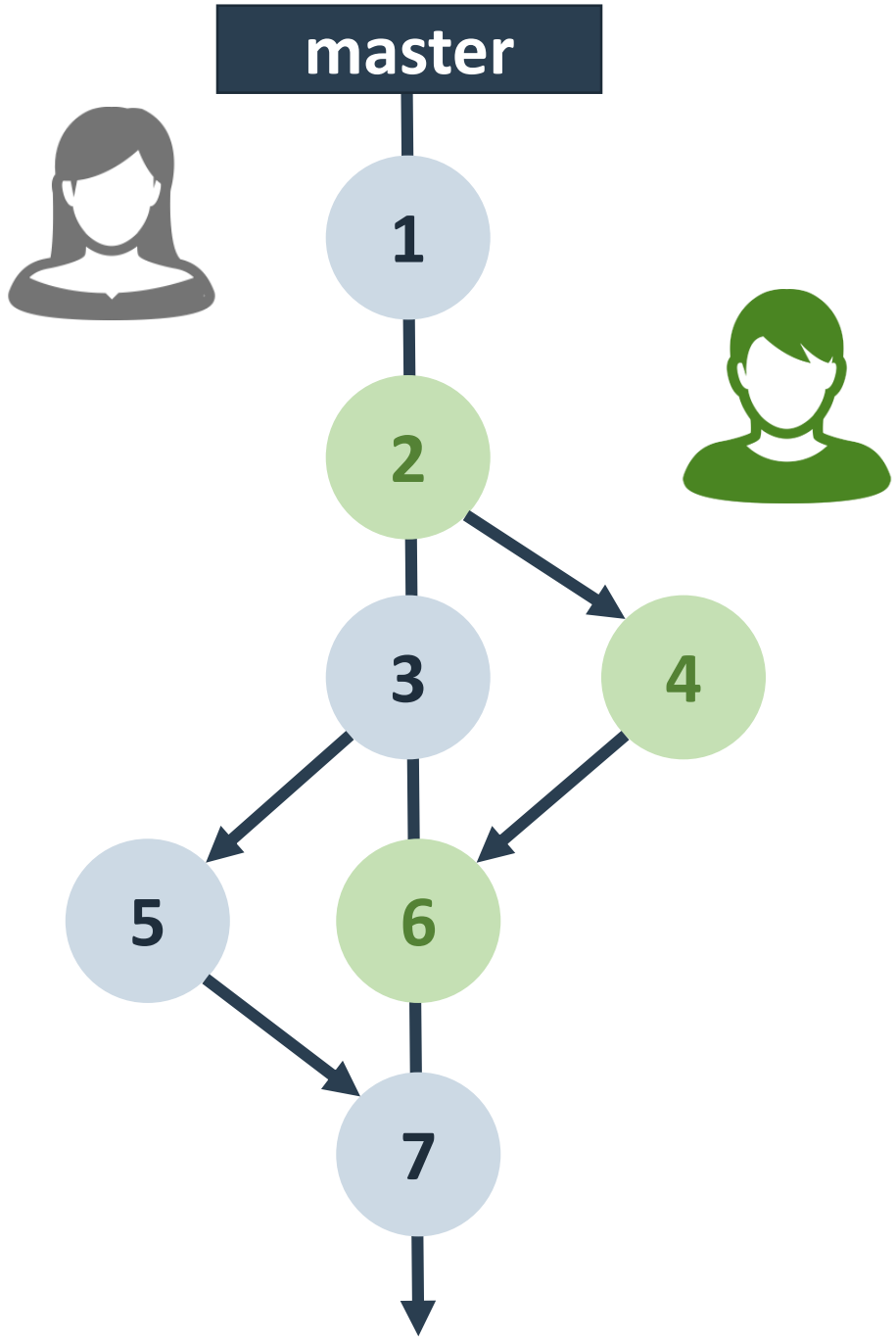
- Cuando se trabaja en equipo se necesita una disciplina y procedimiento para mantener el repositorio de código limpio y ordenado.

# Monorama

- Todos los commits se entregan en la rama **master**

- Pros
  - Sencillo, sin ramas
  - Cómodo para una o dos personas que trabajan juntas

- Contras
  - Merges muy frecuentes cuando el numero de desarrolladores crece.
  - Descontrol, caótico
  - Merges complejos

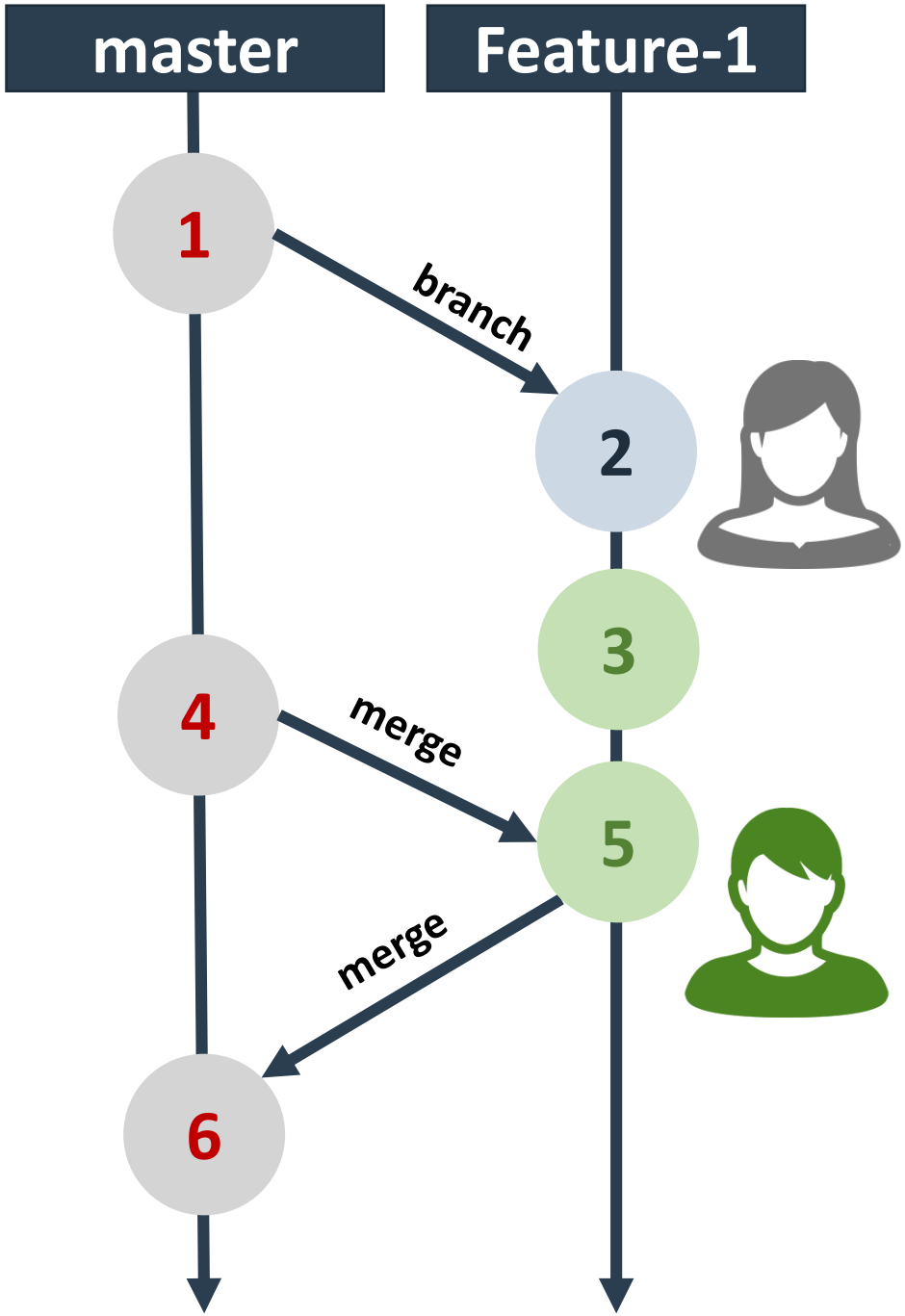


# Rama por característica (*feature*)

- Para cada feature se abre una rama.
- Se integra en master cuando esta completa.
- Prohibidos los commits **master**
- La rama se cierra al acabar la característica.

- Pros
  - Sencillo
  - Rico semánticamente (la rama contiene toda la *feature*)
  - Escala mejor

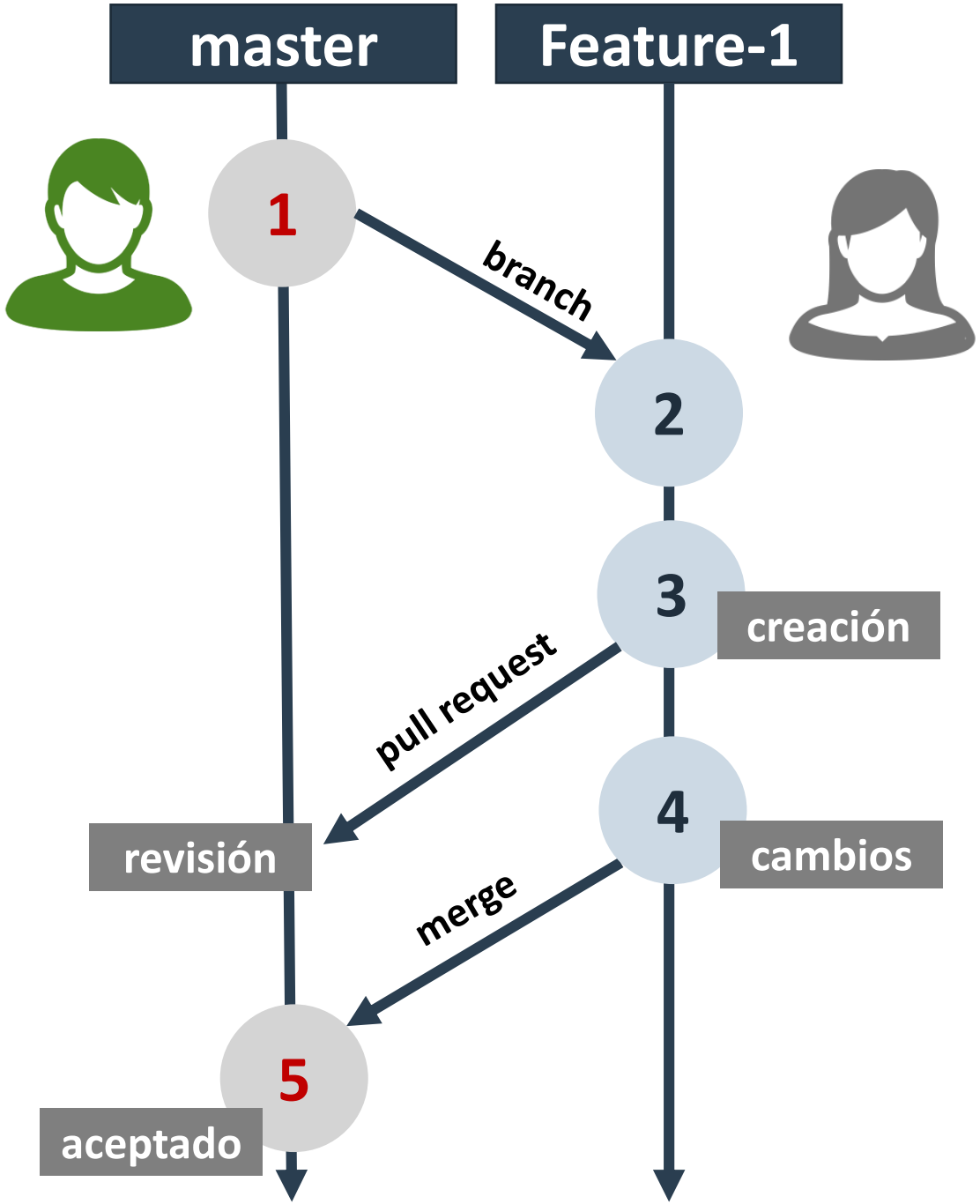
- Contras
  - Requiere disciplina
  - Merge al final de la característica más grande. Para remediarlo conviene ir *mezclando master en rama de feature* (ejemplo 4→5) con frecuencia.



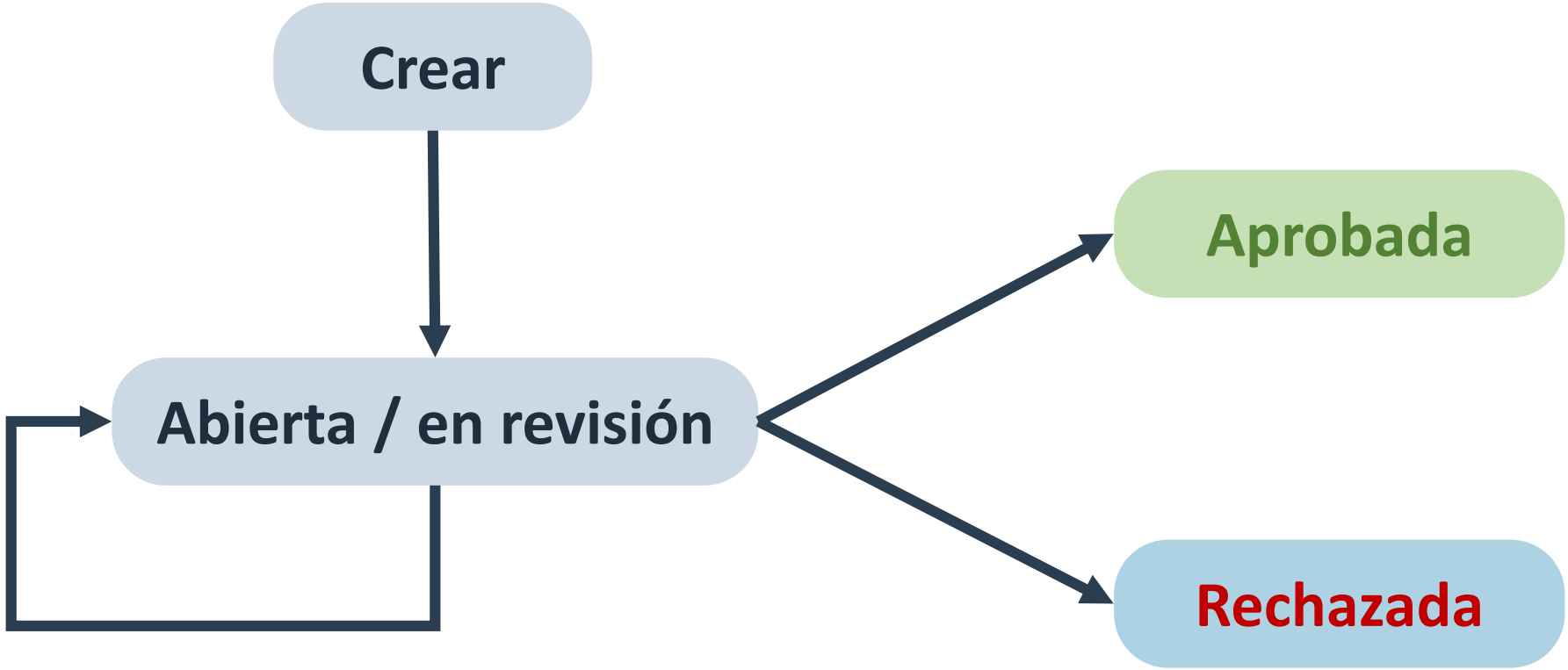
# Pull Request

- Revisión de código (*Peer Reviews*)

- Antes de permitir un merge en una rama principal
- Se revisa el código entregado
- Se hacen comentarios y sugerencias de mejora
- Finalmente se mezcla o se rechaza



# Pull Request. Estados



# Referencias

- Git https://git-scm.com
- Git Book (Español) https://git-scm.com/book/es/v2
- Git Book (Ingles) https://git-scm.com/book/en/v2
- Git Reference http://gitref.org
- GitHub https://github.com