



Gestión de proyectos con Git y GitHub

Juan Quemada, DIT - UPM

Objetivo del curso



Iniciar en el uso de
la herramienta **Git** y el portal
 **GitHub** para **desarrollar**
programas en equipo de
forma profesional

El curso

◆ Curso corto

- Pensado para realizarse en 2 semanas
 - ◆ Se puede realizar a ritmo lento en 4-6 semanas

◆ Curso práctico tipo PBL

- Basado en un proyecto publicado en GitHub

◆ El curso tiene 2 módulos

- Modulo 1: La herramienta Git y su uso
- Modulo 2: Colaborar utilizando de GitHub

◆ Estructura de cada módulo

- A) Tareas de **Aprendizaje** (varias):
 - ◆ un **video** del tema (3 y 15 minutos)
 - ◆ evaluado con un **test** o un **ejercicio P2P**
- B) Tarea **final** del modulo: Ejercicio P2P

Índice: Git y GitHub

MODULO 1: La herramienta Git y su uso

1.	<u>Introducción a Git y a sus comandos</u>	6
2.	<u>El repositorio local y el directorio de trabajo: add, checkout, diff, init, log, mv, reset, rm, status y commit</u>	12
3.	<u>Repositorios públicos en GitHub: new repository, push, import repository y Fork</u>	29
4.	<u>Ramas y grafo de commits: branch, checkout, diff, log, reset y show</u>	48
5.	<u>Integración de ramas: merge, commit y checkout y fast-forward</u>	70
6.	<u>Integración de ramas con rebase</u>	86

Modulo 2: Colaborar utilizando GitHub

7.	<u>Crear commit inicial en GitHub, clonar y actualizar: new repository, .gitignore, clone, remote y push</u>	101
8.	<u>Clonar con Fork en GitHub: Fork, clone y push</u>	121
9.	<u>Ramas locales, remotas, tracking y refsheets: branch, checkout, clone, fetch y pull</u>	146
10.	<u>Contribuir a un repositorio central con pull request: auto-merge, branch, clone, fetch, merge, pull y push</u>	165
11.	<u>Guía de los principales comandos de Git</u>	189



Final del tema



Git y GitHub

Introducción a Git y a sus comandos

Juan Quemada, DIT - UPM

Git



◆ Git es un gestor de **repositorios** de versiones software

- Desarrollado por Linus Torvalds en 2005 en código libre
 - Lo diseño para soportar el desarrollo de Linux con un modelo descentralizado y abierto

◆ git es un comando de UNIX/Linux

- Documentación: <https://git-scm.com/documentation>
 - Tutorial: <https://www.atlassian.com/git/tutorials/setting-up-a-repository>
- Instalación de git o actualización a la última versión
 - Instrucciones de GitHub: <https://help.github.com/articles/set-up-git/>
 - Instrucciones git-scm: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
 - Tutorial Atlassian: <https://www.atlassian.com/git/tutorials/install-git>

```
$ git --version # Indica la versión instalada. Si Git no esta instalado, lo indicará.
```

```
$ git --help # Equivale a git help y muestra lista de los comandos de git mas habituales.
```

```
venus:proy jq$ git --help
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>]
```

The most commonly used git commands are:

add
bisect
branch

Add file contents to the index

Find by binary search the change that introduced a bug

List, create, or delete branches

<https://git-scm.com/docs>

Git: manuales en línea y configuración

```
$ git init ... # git es un meta-comando, donde el primer parámetro (init) la operación solicitada  
# Los parámetros add, bisect, branch, checkout,... invocan distintas operaciones.
```

```
$ git init --help # Equivale a git help init, ayuda del comando git init, igual para: add, bisect, ..
```

```
# El comando git config configura git con las credenciales del desarrollador:
```

```
$ git config --global user.name "Pedro Ramirez"  
$ git config --global user.email pramirez@dit.upm.es
```

```
# Consultar el valor de todas las opciones configuradas:
```

```
$ git config --list  
user.name=Pedro Ramirez  
user.email=pramirez@dit.upm.es  
color.ui=true  
.....
```

```
# Consultar el valor de una opción:
```

```
$ git config user.name  
Pedro Ramirez  
$
```

El desarrollador debe firmar todos los commits que crea en la historia de un proyecto, porque Git es un software de colaboración. Por ello debe configurar sus credenciales antes de utilizar Git.

Estructura del directorio de un proyecto

◆ Directorio de un proyecto:

- Contiene **todos** los **ficheros** y **subdirectorios** del **proyecto**
 - Estructura el **contenido** con **convenios de nombres**
- Algunos nombres muy habituales
 - **README.md** fichero **resumen** (formato **GitHub markdown**)
 - GitHub markdown: <https://github.com/github/markup/blob/master/README.md>
 - **LICENSE** fichero con licencia de **distribución** del **proyecto**
 - **public**: directorio donde almacenar **recursos Web**
 - **bin**: directorio donde almacenar **programas ejecutables**
 - **lib**: directorio con las **librerías** de software utilizadas
 - **test**: directorio con las **pruebas** de funcionamiento **correcto**



◆ Herramientas de gestión del proyecto:

- Suelen utilizar **ficheros** y **subdirectorios** con nombres reservados, por ejemplo
 - **npm** (gestor de paquetes de Node.js)
 - **Fichero package.json**: información del paquete y dependencias de otros paquetes
 - **Directorio node_modules**: contiene los **dependencias** (paquetes) instaladas en el **proyecto**
 - **Git** (gestor de versiones)
 - **Directorio .git**: guarda el grafo de cambios guardados del proyecto del sistema se ven con -a
 - **Fichero .gitignore**: indica los ficheros a ignorar
 -

GitHub



◆ GitHub

- Portal donde programadores comparten repositorios con proyectos Git
 - Nos da acceso a ellos a través del navegador Web y a través de Git

◆ Este curso requiere tener cuenta en GitHub: <https://github.com>

- Al crearla nos da instrucciones claras y precisas sobre uso de GitHub y Git

The screenshot shows the GitHub homepage in a web browser. The URL in the address bar is <https://github.com>. The page features a prominent green button labeled "Read the guide". Below it, there's a message: "Learn Git and GitHub without any code! Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request." To the right of this message is a "Start a project" button. At the bottom of the page, there are several notifications: one from "jquemada" about being added to the "ging" organization, and another about restricting review dismissals with protected branches.



Final del tema

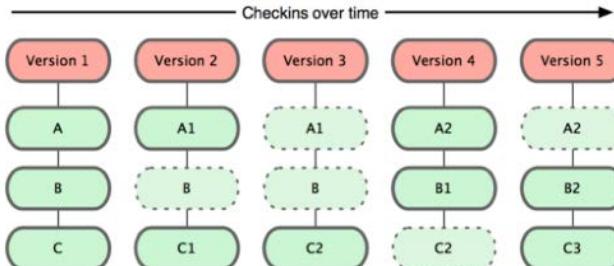


Git y GitHub

El repositorio local y el directorio de trabajo: add, checkout, diff, init, log, mv, reset, rm, status y commit

Juan Quemada, DIT - UPM

Proyecto Software



*S. Chacon, B. Straub: <https://git-scm.com/book/es/v1>

- ◆ Los proyectos software están activos durante largos periodos
 - Durante su vida generan muchas versiones y variantes diferentes
 - Unas corrigen errores, otras añaden funcionalidad o adaptan nuevo hardware/software, etc.
- ◆ Commit o versión
 - Instantánea del estado de los ficheros de un proyecto (puede restaurarse)
 - Algunos commits se etiquetan con tags especiales de versión, p.e. v1, v1.3, ..
- ◆ Rama
 - Secuencia de commits ordenada por fechas que soporta un desarrollo
 - Los nuevos commits se añaden al final de la rama de desarrollo
 - La rama de desarrollo principal se denomina habitualmente **master**
 - La rama master se crea automáticamente al crear el primer commit

Directorio de trabajo y repositorio de commits



◆ Directorio de trabajo

- **Directorio** donde se crean las **versiones** del proyecto: código fuente, datos, ...
 - Se denomina también **área o espacio de trabajo** (workspace)
 - **árbol de trabajo** (work-tree) por la estructura en árbol de los subdirectorios que agrupan ficheros
 - **base de código** (codebase)

◆ El comando: **git init**

- Transforma un directorio en un **directorio de trabajo Git**
 - Añadiendo el **repositorio de commits** al directorio, lo que lo convierte en un directorio de trabajo Git
 - Lo añade en el **subdirectorio oculto .git** que contiene la base de datos donde guardar los commits

◆ Muchos comandos **git** se invocan en el dir. de trabajo o un subdirectorio

- Es decir, si el **directorio de trabajo del terminal** es el **directorio de trabajo Git**

Índice o área de cambios

◆ Índice o área de cambios (staging area, index)

- **Registro de cambios** del directorio de trabajo a incluir en el próximo commit
 - Los cambios no registrados en el índice no se incluyen al generar un nuevo commit
- **Los ficheros no modificados** del commit anterior permanecen en el siguiente commit

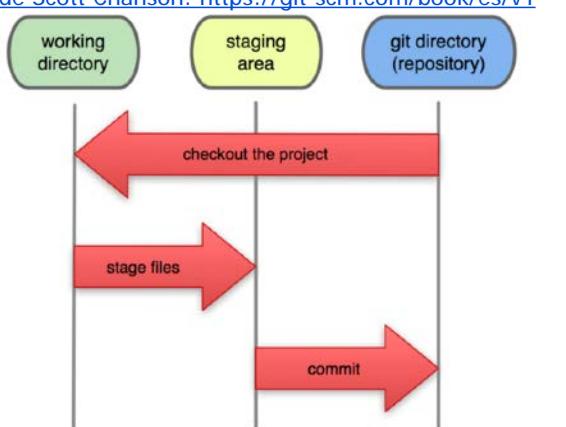
◆ git add ...

- registra en el índice los ficheros indicados
 - `git add .` registra en el índice todos los ficheros **nuevos o modificados**
 - `git add LICENSE README.md` registra los ficheros **LICENSE README.md** en el índice

◆ git reset ...

- extrae **<ficheros>** del índice (deshace `git add ..`)
 - `git reset .` extrae del índice todos los ficheros
 - `git reset LICENSE` extrae LICENSE del índice

*de Scott Chanson: <https://git-scm.com/book/es/v1>

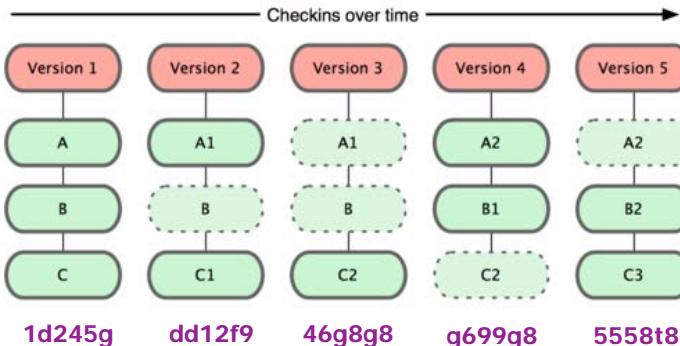


◆ git commit ...

- Genera un nuevo commit con lo registrado en el índice
 - `git commit -m "Descripción"` guarda nuevo commit con mensaje o título "**Descripción**"
 - `git commit` guarda nuevo commit y abre un editor para crear mensaje del commit
 - `git commit --amend -m "...."` modifica el último commit con lo registrado en el índice !OJO cambia commit

Identificador de commit

*S. Chacon, B. Straub: <https://git-scm.com/book/es/v1>



◆ git commit ... asigna un **identificador único** a cada nuevo commit

- El identificador actúa como nombre o referencia única del commit
 - Ningún otro commit en ningún otro repositorio poseerá el mismo identificador
 - Garantiza la integridad del commit: igualdad de identificadores implica igualdad de commits

◆ Identificador de commit

- Número hexadecimal de 40 dígitos generado como clave de hash SHA1
 - Ejemplo de identificador: **973751d21c4a71f13a2e729ccf77f3a960885682**

◆ Se suele utilizar el **formato corto** (formato largo es incómodo)

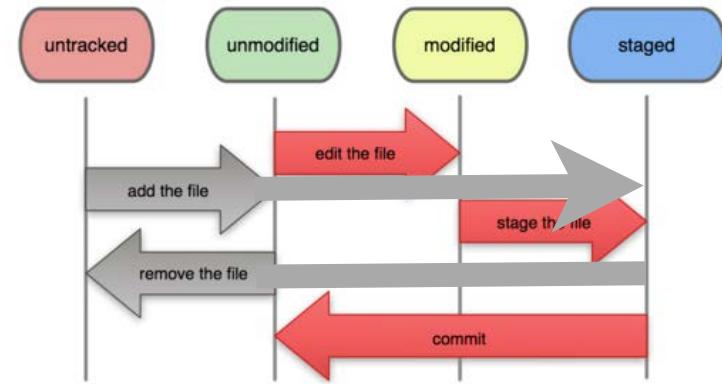
- 7-8 dígitos iniciales (únicos en un proyecto): **973751d2**
 - Los comandos git permiten **identificadores cortos o largos**
- **git log --oneline** muestra la historia de commits (e ids cortos) de una rama

Análisis del directorio de trabajo

- ◆ Un commit se genera por modificación del commit anterior
 - Añadiendo, eliminando o modificando ficheros del directorio de trabajo (y de sus subdir.)
 - Git controla los cambios en los ficheros respecto al commit anterior

◆ git status

- muestra estado de los ficheros del directorio:
 - **modified**: modificados respecto al commit anterior
 - **untracked**: no existentes en el commit anterior
 - **staged**: registrados para el próximo commit
- **git status -s**
 - Muestra estado en formato compacto muy cómodo y conciso



*de Scott Chanson: <https://git-scm.com/book/es/v1>

◆ git diff

- Mostrar diferencias en los ficheros **modified** respecto al commit anterior
 - **git diff** muestra los cambios en todos los ficheros **modified** del directorio de trabajo
 - **git diff README.md** muestra solo los cambios en **README.md**, pero solo si es **modified**
- Mostrar diferencias en los ficheros **staged** respecto al commit anterior
 - **git diff --cached** muestra los cambios en todos los ficheros **staged** del directorio de trabajo
 - **git diff --cached README.md** muestra solo los cambios en **README.md**, pero solo si es **staged**
- **git diff** muestra las diferencias en las líneas del código:
 - **Líneas añadidas**: en verde y comienzan por +
 - **Líneas eliminadas**: en rojo y comienzan por -

Más comandos

◆ git mv <old> <new>

- Cambia el nombre de un fichero en el directorio de trabajo (y en el índice)
 - `git mv file1.js file2.js` cambia el nombre de file1.js a file2.js en el directorio de trabajo y en el índice

◆ git rm <files>

- Borra <files> del directorio de trabajo y registra lo borrado en el índice
 - `git rm file1.js file2.js` borra file1.js y file2.js del directorio de trabajo y del índice

◆ git rm --cached <files>

- Borra <files> del índice, los ficheros pasan de staged a untracked
 - `git rm --cached file1.js file2.js` borra file1.js y file2.js solo del índice

◆ git checkout <files>

- Elimina cambios de <files> que pasan a **unmodified (Peligro! Cambios se pierden)**
 - `git checkout file1.js` elimina los cambios del fichero modified file.js

◆ git checkout .

- Elimina los cambios de todos los ficheros modified del directorio de trabajo que pasan a **unmodified (Peligro! Cambios se pierden)**
 - `git checkout .` elimina cambios en todos los ficheros modified del directorio de trabajo

Crear directorio del S.O.

```
venus:proy jq$ mkdir cal  
venus:proy jq$ cd cal  
venus:cal jq$ git init  
Initialized empty Git repository in /Users/jq/proy/cal/.git/  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit README.md & LICENSE  
venus:cal jq$  
venus:cal jq$ git status -s  
?? LICENSE  
?? README.md  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$  
venus:cal jq$ git status -s  
A LICENSE  
A README.md  
venus:cal jq$  
venus:cal jq$ git commit -m "Readme & License"  
[master (root-commit) 1096247] Readme & License  
2 files changed, 23 insertions(+)  
create mode 100644 LICENSE  
create mode 100644 README.md  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$
```

Crea el **directorio cal** y entra en él.



Transformar en directorio de trabajo Git

```
venus:proy jq$  
venus:proy jq$ mkdir cal  
venus:proy jq$ cd cal  
venus:cal jq$  
venus:cal jq$ git init  
Initialized empty Git repository in /Users/jq/proy/cal/.git/  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit README.md & LICENSE  
venus:cal jq$  
venus:cal jq$ git status -s  
?? LICENSE  
?? README.md  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$  
venus:cal jq$ git status -s  
A LICENSE  
A README.md  
venus:cal jq$  
venus:cal jq$ git commit -m "Readme & License"  
[master (root-commit) 1096247] Readme & License  
2 files changed, 23 insertions(+)  
create mode 100644 LICENSE  
create mode 100644 README.md  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$
```

Transforma el directorio **cal** en un directorio de trabajo Git con su repositorio en **.git**

directorio de trabajo Git



Muestra directorio de trabajo limpio (sin ningún cambio).

Crear ficheros del commit

```
venus:proy jq$  
venus:proy jq$ mkdir cal  
venus:proy jq$ cd cal  
venus:cal jq$  
venus:cal jq$ git init  
Initialized empty Git repository in /Users/jq/proy/cal/.git/  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit README.md & LICENSE  
venus:cal jq$  
venus:cal jq$ git status -s  
?? LICENSE  
?? README.md  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$  
venus:cal jq$ git status -s  
A LICENSE  
A README.md  
venus:cal jq$  
venus:cal jq$ git commit -m "Readme & License"  
[master (root-commit) 1096247] Readme & License  
2 files changed, 23 insertions(+)  
create mode 100644 LICENSE  
create mode 100644 README.md  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$
```

Llevar estos 2 ficheros al directorio de trabajo. Copiarlos o editarlos: vi, vim, sublime-text, Webstorm, Atom,

directorio de trabajo Git



cal
Educational Git project. Creates a simple calculator in HTML and JavaScript in short steps.

MIT License

Copyright (c) 2016 Juan Quemada

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

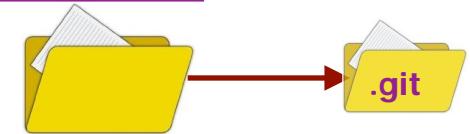
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Mostrar estado del directorio de trabajo

```
venus:proy jq$  
venus:proy jq$ mkdir cal  
venus:proy jq$ cd cal  
venus:cal jq$  
venus:cal jq$ git init  
Initialized empty Git repository in /Users/jq/proy/cal/.git/  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit README.md & LICENSE  
venus:cal jq$  
venus:cal jq$ git status -s  
?? LICENSE  
?? README.md  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$  
venus:cal jq$ git status -s  
A LICENSE  
A README.md  
venus:cal jq$  
venus:cal jq$ git commit -m "Readme & License"  
[master (root-commit) 1096247] Readme & License  
2 files changed, 23 insertions(+)  
create mode 100644 LICENSE  
create mode 100644 README.md  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$
```

directorio de trabajo Git

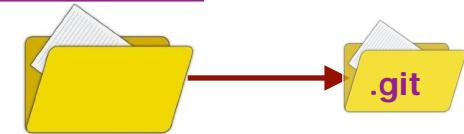


Muestra los 2 nuevos ficheros todavía sin registrar en el índice.

Registrar nuevos ficheros en el índice

```
venus:proy jq$  
venus:proy jq$ mkdir cal  
venus:proy jq$ cd cal  
venus:cal jq$  
venus:cal jq$ git init  
Initialized empty Git repository in /Users/jq/proy/cal/.git/  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit README.md & LICENSE  
venus:cal jq$  
venus:cal jq$ git status -s  
?? LICENSE  
?? README.md  
venus:cal jq$  
venus:cal jq$ git add .  
Registra ficheros en el índice  
venus:cal jq$  
venus:cal jq$ git status -s  
A LICENSE  
A README.md  
venus:cal jq$  
venus:cal jq$ git commit -m "Readme & License"  
[master (root-commit) 1096247] Readme & License  
2 files changed, 23 insertions(+)  
create mode 100644 LICENSE  
create mode 100644 README.md  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$
```

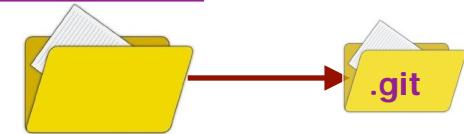
directorio de trabajo Git



Mostrar estado del directorio de trabajo

```
venus:proy jq$  
venus:proy jq$ mkdir cal  
venus:proy jq$ cd cal  
venus:cal jq$  
venus:cal jq$ git init  
Initialized empty Git repository in /Users/jq/proy/cal/.git/  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit README.md & LICENSE  
venus:cal jq$  
venus:cal jq$ git status -s  
?? LICENSE  
?? README.md  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$  
venus:cal jq$ git status -s  
A LICENSE  
A README.md  
venus:cal jq$  
venus:cal jq$ git commit -m "Readme & License"  
[master (root-commit) 1096247] Readme & License  
2 files changed, 23 insertions(+)  
create mode 100644 LICENSE  
create mode 100644 README.md  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$
```

directorio de trabajo Git



Muestra los ficheros ya registrados. Estos se incluirán en el próximo commit.

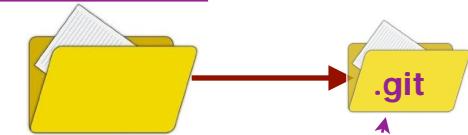
Crear nuevo commit y ver historia

```
venus:proy jq$  
venus:proy jq$ mkdir cal  
venus:proy jq$ cd cal  
venus:cal jq$  
venus:cal jq$ git init  
Initialized empty Git repository in /Users/jq/proy/cal/.git/  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit README.md & LICENSE  
venus:cal jq$  
venus:cal jq$ git status -s  
?? LICENSE  
?? README.md  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$  
venus:cal jq$ git status -s  
A LICENSE  
A README.md  
venus:cal jq$  
venus:cal jq$ git commit -m "Readme & License"  
[master (root-commit) 1096247] Readme & License  
2 files changed, 23 insertions(+)  
create mode 100644 LICENSE  
create mode 100644 README.md  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$
```

Genera nuevo commit.

Muestra nuevo commit
en formato 1 línea

directorio de
trabajo Git



master

Readme & License

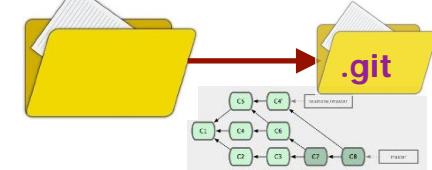
Inspeccionar historia y área de trabajo

```
venus:cal jq$  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit calculator.html  
venus:cal jq$  
venus:cal jq$ git status -s  
?? calculator.html  
venus:cal jq$  
venus:cal jq$ git add calculator.html  
venus:cal jq$  
venus:cal jq$ git commit -m "x^2 button"  
[master b0e63ad] x^2 button  
 1 file changed, 17 insertions(+)  
 create mode 100644 calculator.html  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$
```

Muestra el primer commit ya generado.

Muestra el directorio de trabajo Git limpio.

directorio de trabajo



master

Readme & License

Añadir nuevo fichero

```
venus:cal jq$  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit calculator.html  
venus:cal jq$  
venus:cal jq$ git status -s  
?? calculator.html  
venus:cal jq$  
venus:cal jq$ git add calculator.html  
venus:cal jq$  
venus:cal jq$ git commit -m "x^2 button"  
[master b0e63ad] x^2 button  
 1 file changed, 17 insertions(+)  
 create mode 100644 calculator.html  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$
```

Se añade el fichero **calculator.html** al directorio de trabajo con un editor, copiando (cp), ...

```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
function square() {  
    var num = document.getElementById("n1");  
    num.value = num.value * num.value;  
}  
</script>  
</head>  
<body>  
    Number:  
        <input type="text" id="n1"><p>  
  
        <button onclick="square()"> x<sup>2</sup> </button>  
</body>  
</html>
```

directorio de trabajo

Muestra el fichero **calculator.html** en el directorio de trabajo todavía sin registrar.

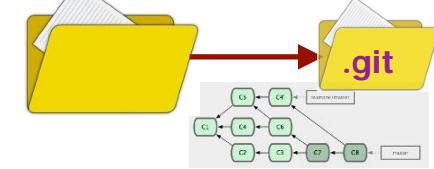


Añadir un nuevo commit a master

```
venus:cal jq$  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git status -s  
venus:cal jq$  
venus:cal jq$ # Edit calculator.html  
venus:cal jq$  
venus:cal jq$ git status -s  
?? calculator.html  
venus:cal jq$  
venus:cal jq$ git add calculator.html  
venus:cal jq$  
venus:cal jq$ git commit -m "x^2 button"  
[master b0e63ad] x^2 button  
1 file changed, 17 insertions(+)  
create mode 100644 calculator.html  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$
```

Registra fichero **calculator.html** en el índice.

directorio de trabajo



Genera nuevo commit con mensaje "**x^2 button**"

Muestra los dos commits ya generados en la rama master (formato 1 línea).

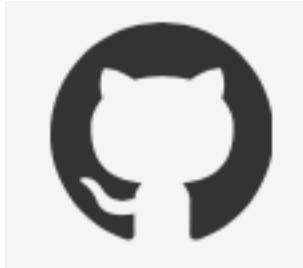
master

x^2 button

Readme & License

Ejercicio opcional

- ◆ Crear un repositorio Git en un directorio **mi_repo** con
 - **git init mi_repo**
 - crea el directorio mi_repo (si no existe) y lo inicializa como repositorio Git
- ◆ Añadir un fichero de nombre **mi_fichero** al directorio **mi_repo**
 - Contenido: “El primer fichero del primer repositorio de <nombre apellido>”
 - Mostrar el estado del directorio de trabajo con **git status** y luego con **git status -s**
 - Inspeccionar las diferencias con **git diff** y luego con **git diff --cached**
- ◆ Añadir con **git add .** el fichero al índice
 - Volver a mostrar el estado del directorio de trabajo con **git status** y luego con **git status -s**
 - Volver a inspeccionar las diferencias con **git diff** y luego con **git diff --cached**
- ◆ Crear commit con **git commit -m “mi primer commit”**
 - Mostrar la historia de la rama master **git log** y luego con **git log --oneline**
- ◆ Renombrar el fichero **mi_fichero** a **README.md** con
 - **git mv mi_fichero README.md**
 - Mostrar el estado del directorio de trabajo con **git status ..** y con **git diff ..**
- ◆ Registrar los cambios en el índice con **git add .**
- ◆ Crear commit con **git commit -m “mi segundo commit”**
 - Mostrar la historia de la rama master **git log** y luego con **git log --oneline**

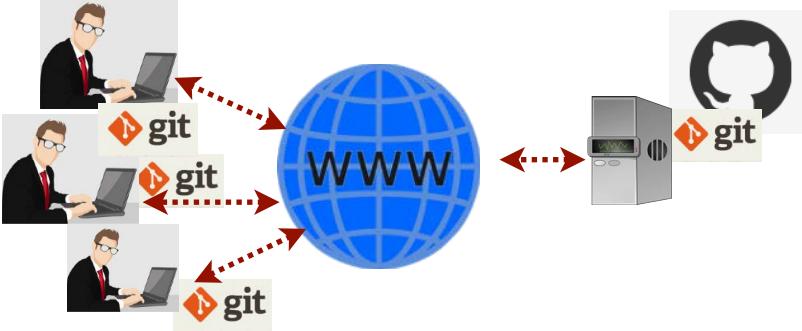


Git y GitHub

Repositorios públicos en GitHub:
new_repository, push, import_repository y Fork

Juan Quemada, DIT - UPM

Tipos de repositorio



◆ Repositorio **sin** directorio de trabajo (bare)

- Repositorio para **compartir desarrollos** con otros a través de Internet
 - Suele estar alojado en un servidor de Internet y se sincroniza con uno local, con
 - **git push ...** sube ramas de desarrollo a un repositorio bare
 - **git clone ...** clona un repositorio en un repositorio local
 - **git fetch ...** trae ramas de otro repositorio al repositorio local
 - **git pull ...** integra una rama de otro repositorio con una rama local
- Se crea con: **git init --bare**
 - Crea solo un fichero con el **repositorio de commits** y sin directorio de trabajo



◆ Repositorio **con** directorio trabajo (local o de trabajo)

- Repositorio para **desarrollar** en el ordenador local
 - Git **no** permite hacer **push** hacia este tipo de repositorios



◆ Portales Web de repositorios bare: **GitHub**, **Bitbucket**, ..

- Equipos u organizaciones alojan en ellos repositorios compartidos con terceros
 - Permiten tanto acceso web, como acceso con comandos Git



GitHub

◆ Github → lema "Social coding"

- Red social donde programadores comparten repositorios remotos Git
 - Nos da acceso a ellos a través del navegador Web (además de Git)

◆ Repositorios **públicos** son gratis, los privados de pago

- Algunos proyectos libres en Github: Linux, Eclipse, jQuery, RoR, ...

◆ Este curso requiere tener cuenta en GitHub: <https://github.com>

- Al crearla nos da instrucciones claras y precisas sobre uso de GitHub y **Git**

◆ Otro repositorio se identifica en un repositorio local con un **URL**, p. e.

- <https://github.com/jquemada/cal>
- <https://github.com/jquemada/cal.git>
- <git@github.com:jquemada/cal.git>

URL del rep. jquemada/cal en GitHub
con extensión .git explícita (equivalente)
URL Git (equivalente, poco utilizado)



Funciones principales de GitHub

- ◆ La función principal de **GitHub** es **compartir** repositorios con terceros
- ◆ Las operaciones principales de un **usuario registrado** son
 - **Crear repositorio remoto** inicial nuevo para albergar un proyecto
 - Utilizando el botón: **New repository**
 - **Copia** un repositorio albergado en GitHub a otra cuenta (para contribuir)
 - Utilizando el botón: **Fork**
 - **Importa** un repositorio identificado por su URL a GitHub, incluso en otro formato
 - Utilizando el botón: **Import repository**
 - Equivale a crear repositorio vacío (New_repository) e importar en él otro repositorio con un URL
 - **Crear una organización** para albergar múltiples proyectos relacionados
 - Utilizando el botón: **New organisation**
 - Organización de asignatura CORE: <https://github.com/CORE-UPM>
 - Y otras operaciones de compartición, gestión y mantenimiento
- ◆ Permite operaciones Git de **sincronización de repositorios bare**
 - **push** (subir rama), **clone** (clonar repositorio), **fetch** (traer rama), **pull** ..

Tres formas de crear repositorios en GitHub

◆ Crear un repositorio vacío con **New_repository**:

- <https://github.com/jquemada/cal>
 - GitHub lo crea en sus servidores invocando: `git init --bare`



◆ Copiar un repositorio a través de su URL con **Import_repository**:

- https://github.com/jquemada/cal_2com
 - El repositorio puede importarse de otro servidor en Internet o de GitHub, incluso cambiar el formato

◆ Copiar un repositorio con **Fork** a otra cuenta u organización:

- <https://github.com/CORE-UPM/cal>
 - se copia de la cuenta **jquemada** a la organización **CORE-UPM**

A screenshot of a web browser window showing a GitHub repository page for 'jquemada/cal'. The browser's address bar shows the URL. A yellow callout box labeled 'Crear nuevos objetos en GitHub' points to a context menu that has appeared over the repository header. The menu items listed are: New repository, Import repository, New gist, New organization, This repository, and New issue. The main repository page shows basic stats like 0 issues, 0 pull requests, 0 projects, and 0 wiki pages. At the bottom, there's a note about no description, website, or topics provided, and buttons for 'New' and 'Add topics'.

Crear jquemada/cal vacío en GitHub

The screenshot shows the GitHub 'Create a New Repository' interface. A red dashed arrow points from the 'New repository' button in the sidebar to the 'Repository name' field. A yellow box highlights the 'Repository name' field containing 'jquemada / cal'. Another yellow box highlights the 'Owner' dropdown set to 'jquemada'. A third yellow box highlights the 'Repository name' field again. A fourth yellow box highlights the 'Create repository' button at the bottom.

Crear nuevo repositorio vacío en GitHub

New repository

Import repository

New gist

New organization

Create a new repository

A repository contains all the files for your project, including

Owner Repository name

jquemada / cal

Great repository names are short and memorable. Need ins

Nombre del repositorio

Repository público sin .gitignore, LICENSE y README

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.

Add .gitignore: None Add a license: None

Create repository

Nuevo repositorio creado con URL:
<https://github.com/jquemada/cal>

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

Set up in Desktop or HTTPS SSH https://github.com/jquemada/cal.git

We recommend every repository include a README, LICENSE, and .gitignore.

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

```
echo "# cal" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin https://github.com/jquemada/cal.git  
git push -u origin master
```

Instrucciones de uso del repositorio.

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

```
git remote add origin https://github.com/jquemada/cal.git  
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS

Import code

© Juan Quemada

Subir el repositorio local a `jquemada/cal` en GitHub con `git push`

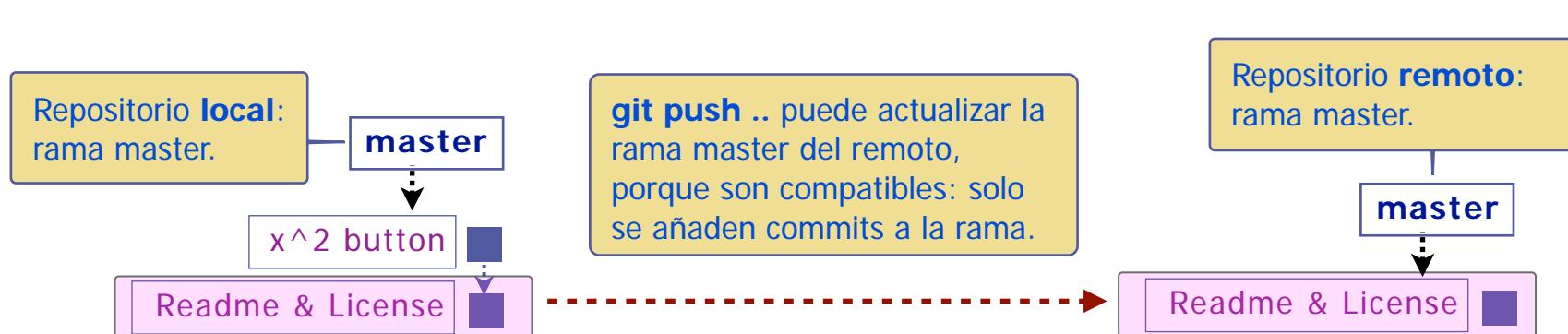
Actualizar un repositorio en GitHub con push

◆ git push ...

- Actualiza el repositorio remoto con los nuevos commits de una rama local
 - `git push https://github.com/jquemada/cal master`
 - actualiza los nuevos commits de la rama master en el repositorio GitHub `jquemada/cal`
 - `git push https://github.com/jquemada/cal_2com master`
 - actualiza los nuevos commits de la rama master en el repositorio GitHub `jquemada/cal_2com`

◆ git push ... necesita 2 condiciones para finalizar con éxito

- Se debe tener credenciales de acceso al repositorio remoto
 - Por ejemplo, un repositorio en una cuenta u organización del usuario que lo actualiza
- La actualización de commits debe ser compatible con la rama actualizada en el remoto
 - Solo debe **añadir nuevos commits al final de la rama remota** o actualizar un **repositorio vacío**
 - **Peligroso!** La opción `-f` permite actualizar una rama incompatible, pero se pierden commits

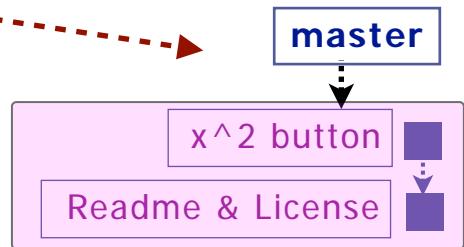


Historia del repositorio local



```
cal -- bash -- 67x15
venus:cal jq$ git log --oneline
b0e63ad x^2 button
1096247 Readme & License
venus:cal jq$ git push https://github.com/jquemada/cal master
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.40 KiB | 0 bytes/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/jquemada/cal
 * [new branch]      master -> master
venus:cal jq$
```

git log --oneline muestra los commits de la rama master del repositorio local cal.



Screenshot of a GitHub repository page for "jquemada / cal".

Repository details:

- Owner: jquemada
- Name: cal
- Last commit: x^2 button (by Juan, 1 day ago)
- Branch: master

Repository stats:

- Code: 1 file
- Issues: 0
- Pull requests: 0
- Projects: 0
- Wiki
- Pulse
- Graphs
- Settings

Quick setup instructions:

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

Set up in Desktop or HTTPS <https://github.com/jquemada/cal.git>

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

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

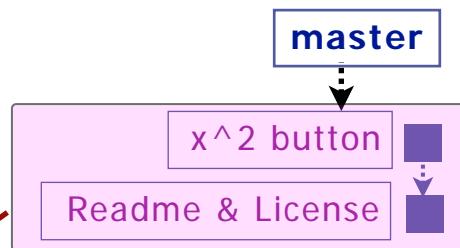
```
echo "# cal" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/jquemada/cal.git
```

Sincronizar rama master remota con la local



```
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git push https://github.com/jquemada/cal master  
Counting objects: 7, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (7/7), done.  
Writing objects: 100% (7/7), 1.40 KiB | 0 bytes/s, done.  
Total 7 (delta 1), reused 0 (delta 0)  
remote: Resolving deltas: 100% (1/1), done  
To https://github.com/jquemada/cal  
 * [new branch]      master -> master  
venus:cal jq$
```

Sube la rama **master** del repositorio **local** al repositorio **remoto** en GitHub con URL <https://github.com/jquemada/cal>. Como el repositorio remoto está vacío, ambos se sincronizan sin ningún problema.



jquemada/cal

This repository Search

Pull requests Issues Gist

jquemada / cal

No description, website, or topics provided.

New Add topics

2 commits 1 branch 0 releases 1 contributor MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

El repositorio en GitHub



This repository

Search

Repository en GitHub I

Issues Gist



jquemada / cal

Es un repositorio público accesible con el URL
<https://github.com/jquemada/cal> a
 cualquier persona través de Internet.

Code

Issues 0

Unwatch 1

Star 0

Fork 0

Pulse

Graphs

Settings

No description, website, or topics provided.

New Add topics

Hay 2 commits (versiones)

2 commits

1 branch

0 releases

1 contributor

MIT

Branch: master ▾

New pull request

new file

Upload files

Find file

Clone or download ▾

El proyecto: último commit de la rama master con los 3 ficheros indicados.

jquemada x^2 button

Latest commit b0e63ad 5 days ago

LICENSE

Readme & License

5 days ago

README.md

Readme & License

5 days ago

calculator.html

x^2 button

5 days ago

README.md

cal

Educational Git project. Creates a simple calculator in HTML and JavaScript in short steps.

Fichero README.md se ve aquí. Es muy conveniente incluirlo en un fichero en GitHub describiendo el proyecto o repositorio.

GitHub, Inc. [US] https://github.com/jquemada/cal Apps Bancos MiTwitter DIT Google Drive This repository Search Pull requests Is

Repository en GitHub II

jquemada / cal

No description, website, or topics provided.

New Add topics

2 commits

Branch: master New pull request

jquemada x² button

LICENSE README.md calculator.html

Último commit de la rama master con los 3 ficheros indicados.

Readme Readme

README.md

cal

Educational Git project. Creates a simple calculator in HTML

This repository Search

jquemada / cal

Code Issues 0 Pull requests 0 Projects 0

Branch: master cal / calculator.html

jquemada x² button

1 contributor

18 lines (15 sloc) 350 Bytes

```
1 <!DOCTYPE html><html><head>
2 <title>Calculator</title><meta charset="utf-8">
3 <script type="text/javascript">
4
5 function square() {
6     var num = document.getElementById("n1");
7     num.value = num.value * num.value;
8 }
9 </script>
10 </head>
11 <body>
12 Number:
13 <input type="text" id="n1"><p>
14
15 <button onclick="square()"> x2 </button>
16 </body>
17 </html>
```

Number: 2

x²

A red dashed arrow points from the 'calculator.html' file in the GitHub repository sidebar to the 'x²' button in the screenshot of the browser window.

A screenshot of a GitHub repository page for 'jquemada/cal'. The repository title 'Repository en GitHub III' is displayed prominently at the top. Below it, there are tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', and 'Wiki 0'. A green 'New' button with 'Add topics' is visible. On the left, there's a summary: '2 commits', '1 branch', and '0 releases'. A red dashed arrow points from the '2 commits' button to a commit entry. The commit history shows two entries:

- x² button** by jquemada committed 3 hours ago
- Readme & License** by jquemada committed 3 days ago

The commit for 'x² button' is highlighted with a pink box and has a red arrow pointing to the 'Number: 2' input field in a floating window.

The GitHub repository page continues below the commit history. It shows a list of files:

- jquemada x² button
- LICENSE
- README.md
- calculator.html

Below the file list is the content of the README.md file:

```
cal
```

Educational Git project. Creates a simple calculator in HTML and JavaScript in short steps.

GitHub, Inc. [US] https://github.com/jquemada/cal Apps Bancos MiTwitter DIT Google Drive g Miria

Repository en GitHub III

jquemada / cal

Code Issues 0 Pull requests 0 Projects 0

No description, website, or topics provided

New Add topics

2 commits 1 branch

Branch: master New pull request

jquemada x² button

LICENSE README.md calculator.html

cal

Educational Git project. Creates a simple calculator in

Verde: código añadido

No hay rojo, ni negro porque al ser un fichero nuevo solo se añade.

Branch: master

Commits on Feb 12, 2017

x² button jquemada committed 3 hours ago

Commits on Feb 9, 2017

Readme & License jquemada committed 3 days ago

calculator.html

```
17 calculator.html
1 +<!DOCTYPE html><html><head>
2 +<title>Calculator</title><meta charset="utf-8">
3 +<script type="text/javascript">
4 +
5 +function square() {
6 +  var num = document.getElementById("n1");
7 +  num.value = num.value * num.value;
8 +
9 +</script>
10 +</head>
11 +<body>
12 + Number:
13 + <input type="text" id="n1"><p>
14 +
15 + <button onclick="square()"> x2 </button>
16 +</body>
17 +</html>
```

Number: 2 x²

Fork

Crear repositorio jquemada/cal_2com

Crear jquemada/cal_2com

The screenshot illustrates the steps to import a repository from GitHub to a new repository.

Left Panel (Importar un repositorio a GitHub):

- The URL <https://github.com/new/import> is shown in the address bar.
- The "Import repository" button is highlighted with a pink box.
- The "Name" field contains "cal_2com".
- A note at the bottom states: "Your new repository will be public. In order to make this repository private you'll need to upgrade your account."
- The "Begin import" button is at the bottom right.

Right Panel (Repository Overview):

- The repository "jquemada / cal_2com" is listed.
- Statistics: 2 commits, 1 branch.
- Branch: master
- Files listed: LICENSE, README.md, calculator.html, README.md.
- Description: "Educational Git project. Creates a simple calculator".

Annotations:

- A yellow box highlights the "Import repository" button in the left sidebar.
- A yellow box highlights the "URL de identificación del repositorio a importar." input field in the center.
- A yellow box highlights the "Nombre del nuevo repositorio" input field in the center.
- A blue box highlights the "Begin import" button in the bottom right of the import form.
- A blue box highlights the repository name "jquemada / cal_2com" in the top right of the right panel.
- A red dashed arrow points from the "Import repository" button to the "Import your project to GitHub" section.
- A red dashed arrow points from the "Import your project to GitHub" section to the "URL de identificación del repositorio a importar." input field.
- A red dashed arrow points from the "Nombre del nuevo repositorio" input field to the "Name" field in the import form.

Crear repositorio CORE-UPM/cal con Fork



Fork: clonar un proyecto de GitHub

Repository cal del usuario jquemada en: <https://github.com/jquemada/cal>

Copia el repositorio a otra cuenta u organización del usuario en GitHub pulsando el botón Fork.

Copia de jquemada/cal creada en la organización CORE-UPM. El nuevo repositorio estará accesible en: <https://github.com/CORE-UPM/cal>

En el momento del Fork el repositorio **jquemada/cal** tiene estos **2 commits**. A partir de este momento cada repositorio evolucionara por separado a partir de estos 2 commits

2 commits

1 branch

0 releases

Create new file Upload files Find file

Unwatch 1 Star 0 Fork 0

This repository Search Pull requests Issues Gist

Where should we fork this repository?

@ging @CORE-UPM...

Can't find what you're looking for?

You already have a fork of this repository: [jquemada/cal_2com](https://github.com/jquemada/cal_2com)

Apps Bancos MiTwitter DIT Cursos

Wiki Pulse Graphs Settings

Code Issues Add topics

No description, website, or topi

New Add topics

2 commits

Branch: master New pull request

jquemada x^2 button

Commits on Feb 12, 2017

x^2 button jquemada committed 3 hours ago

Commits on Feb 9, 2017

Readme & License jquemada committed 3 days ago

Unwatch 6 Star 0

This repository Search Pull requests Issues Gist

Settings

2 commits

1 branch

0 releases

Create new file Upload files Find file

Unwatch 1 Star 0 Fork 0

This repository Search Pull requests Issues Gist

Wiki Pulse Graphs Settings

Code Issues Add topics

No description, website, or topi

New Add topics

2 commits

Branch: master New pull request

jquemada x^2 button

Commits on Feb 12, 2017

x^2 button jquemada committed 3 hours ago

Commits on Feb 9, 2017

Readme & License jquemada committed 3 days ago

Unwatch 6 Star 0

Ejercicio opcional

- ◆ Crear una cuenta en GitHub que incluya su nombre y apellidos (o sus iniciales) en el nombre de la cuenta. Si ya tiene una cuenta con estos requisitos puede utilizarla.
- ◆ Crear a continuación un repositorio vacío **mi_repo** en GitHub
 - Utilizando el botón **New_repository** para crearlo
 - El repositorio debe crearse vacío sin ningún fichero, ni .gitignore, ni LICENSE, ni README.md
- ◆ Subir la rama master local al repositorio **mi_repo** en GitHub
 - Subir el repositorio local con el comando **git push**
 - Inspeccionar el repositorio subido a GitHub con el navegador Web
- ◆ Copiar https://github.com/jquemada/cal_2com a su cuenta GitHub
 - Utilizando el botón de Fork para realizar la copia
 - Inspeccionar ambos repositorios en GitHub (original y copiado) con el navegador Web
 - Comprobar que tienen los mismos commits con exactamente los mismos identificadores

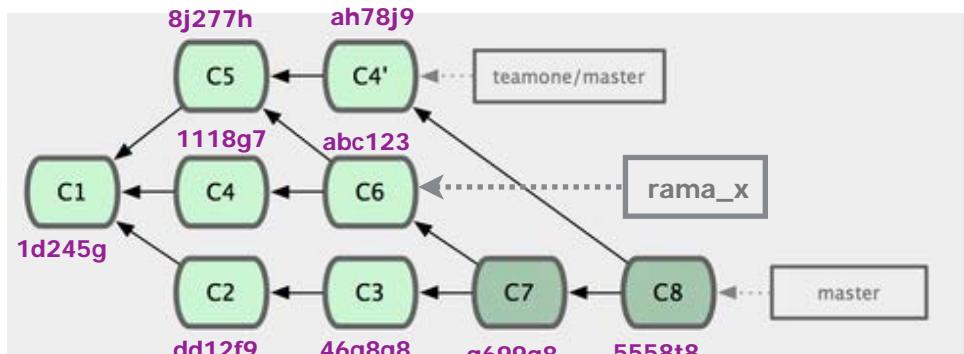


Git y GitHub

Ramas y grafo de commits: branch, checkout, diff, log, reset y show

Juan Quemada, DIT - UPM

Ramas



*S. Chacon, B. Straub: <https://git-scm.com/book/es/v1>

◆ La **rama master** es la rama principal de un repositorio

- Es una rama predefinida que se crea al crear el primer commit

◆ Las **ramas** soportan desarrollos separados de master

- Una rama puede comenzar en cualquier commit del repositorio
 - Los nuevos **commits** de una rama **se deben añadir al final** (la rama siempre crece)
 - La flecha indica el commit o los commits a partir de los se ha generado

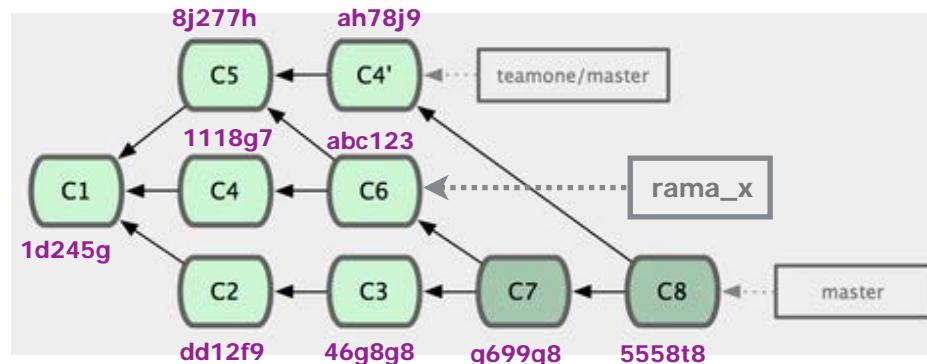
◆ El nombre de la rama es un puntero a su último commit, por ej.

- **master** identifica el commit **5558t8**
- **rama_x** identifica el commit **abc123**

◆ **git branch**

- **git branch rama_y 46g8g8** crea la rama_y con base en C3 (46g8g8)
- **git branch [-v]** muestra las ramas existentes

Grafo de commits



*S. Chacon, B. Straub: <https://git-scm.com/book/es/v1>

◆ El grafo de commits de un repositorio

- Grafo con la relación de generación de todos los commits de las ramas de un repositorio

◆ Las flechas de salida de un commit indican su relación con los anteriores

- Una flecha: commit **generado por modificación** del commit anterior
- Dos flechas: commit **generado por integración** de una rama en otra

◆ Historia de un commit

- Secuencia ordenada (por fechas) de commits utilizados para generar dicho commit

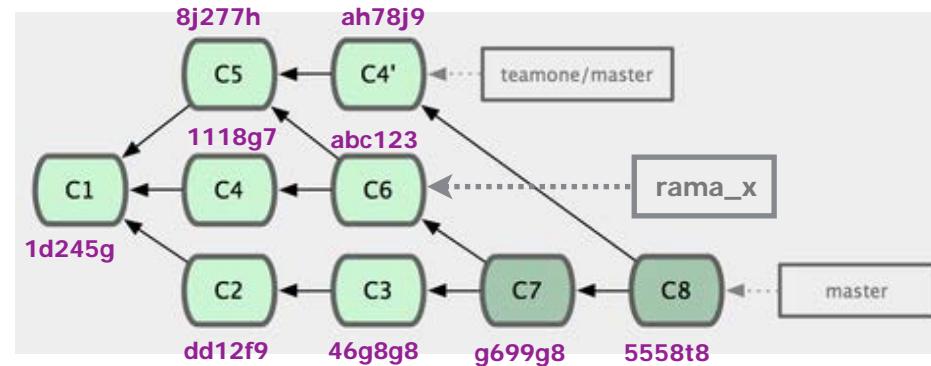
◆ Padres y ancestros de un commit

- **<commit>ⁿ:** representa el número (n) de **padre** de un commit de integración
 - Por ejemplo: master¹=g699g8 o master²=ah78j9
- **<commit>~n:** **ancestro n** de su historia
 - Por ejemplo: 46g8g8~1=dd12f9 o master~2=46g8g8
- **La notación ancestro sigue la línea del primer parent (Cx¹)**
 - Es decir, master~1=master¹ y por lo tanto master~1=g699g8 y master~3=dd12f9

Comandos show, log y diff

◆ git log

- muestra la historia de **un commit o una rama**
 - `git log -3` muestra los 3 últimos commits de la historia del commit actual (HEAD)
 - `git log --oneline --graph rama_x` muestra el grafo de integración de commits de **rama_x** (formato 1 línea)
 - `git log --oneline master~2` historia del 2º ancestro de **master** (formato 1 línea)
- muestra la historia de commits de **todo el repositorio**
 - `git log --oneline --all -8` muestra los 8 últimos commits del repositorio ordenados por fechas
- muestra el **grafo** de commits del repositorio
 - `git log --oneline --all --graph` muestra el **grafo completo** de commits del repositorio



*S. Chacon, B. Straub: <https://git-scm.com/book/es/v1>

◆ git diff

- Muestra diferencias de código entre ficheros de diferentes commits
 - `git diff rama_x master` muestra los cambios en ficheros entre **rama_x** y **master**
 - `git diff master~2` muestra los cambios en ficheros entre **master~2** y HEAD
 - `git diff rama_x master -- LICENSE` muestra los cambios en el fichero **LICENSE** entre **rama_x** y **master**
 - `git diff rama_x master LICENSE` similar al anterior si no hay ambigüedad en los nombres

◆ git show

- muestra metadatos de commit y diferencias con el commit anterior
 - `git show rama_x` muestra los metadatos del último commit de **rama_x** y las diferencias con el commit anterior
 - `git show` muestra los metadatos del commit actual (HEAD) y las diferencias con el commit anterior

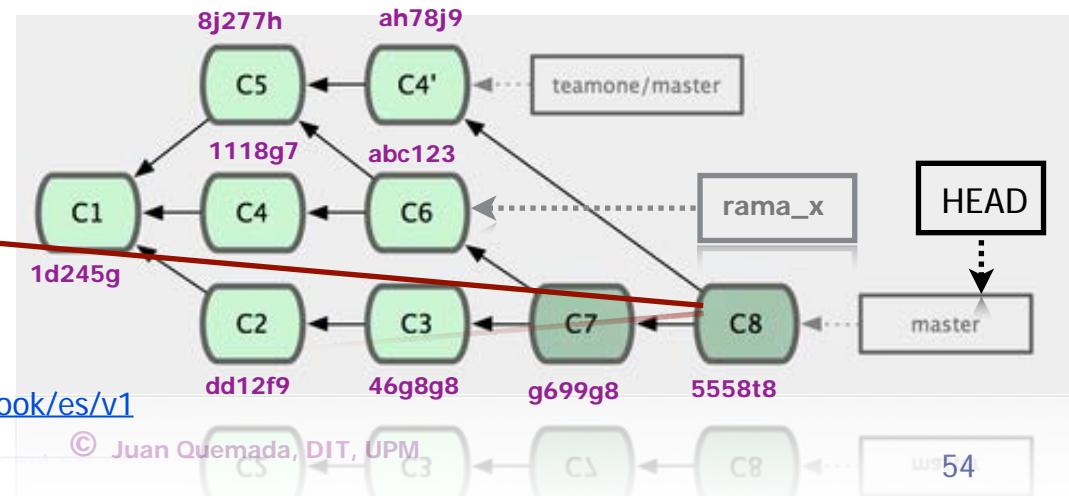
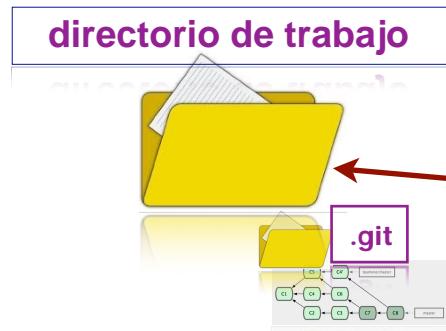
Restaurar commits en el directorio de trabajo

◆ **git checkout <commit>** restaura <commit> en el **directorio de trabajo**

- El contenido anterior se debe haber guardado previamente
 - **!OJO** Los cambios no guardados se perderán
- Ejemplos
 - **git checkout rama_x** restaura rama_x (abc123) en el directorio de trabajo y actualiza HEAD
 - **git checkout master** restaura la rama master (5558t8) en el dir. de trabajo y actualiza HEAD
 - **git checkout dd12f9** restaura el commit 5558t8 en el dir. de trabajo y actualiza HEAD

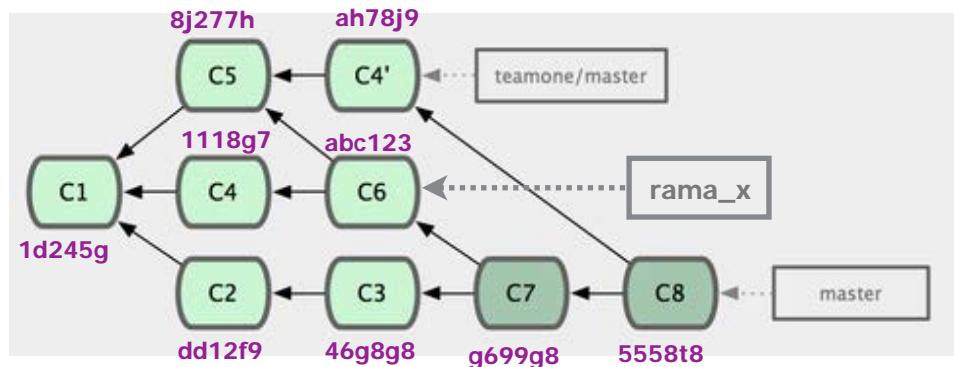
◆ **HEAD** es un puntero al commit que está restaurado en el directorio de trabajo

- **HEAD** se actualiza automáticamente al hacer el checkout



*S. Chacon, B. Straub: <https://git-scm.com/book/es/v1>

Comando reset



*S. Chacon, B. Straub: <https://git-scm.com/book/es/v1>

◆ git reset <commit>

- Cambia el puntero de rama y HEAD a <commit>. Deja las diferencias entre HEAD y <commit> en el directorio de trabajo. Es decir, restaura el contenido de commit, añadiendo las diferencias con <commit>
 - **Peligro!** Se pueden perder commits del grafo, si los eliminados no están guardados en otra rama.
- Ejemplos de uso
 - `git reset master~2` mueve HEAD y el puntero de rama al 2º ancestro, dejando las diferencias en el dir. de trabajo
 - **Nota:** Los cambios introducidos en los 2 commits que desaparecen de la rama quedan en los ficheros modified

◆ git reset --hard <commit>

- Cambia el puntero de rama y HEAD a <commit>.
 - **Muy peligroso!** Los commits y su código se pueden perder, si los eliminados no están guardados en otra rama
- Ejemplos de uso
 - `git reset --hard master~2` mueve HEAD y el puntero de rama al 2º ancestro
 - **Muy peligroso!** Los últimos 2 commits de la rama desaparecerían si no estuviesen en otra rama

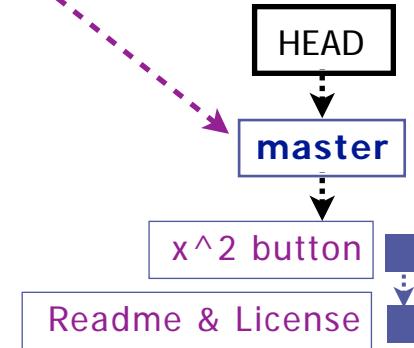
◆ El comando git reset permite compactar o eliminar commits del grafo

- normalmente se utiliza para eliminar, compactar o rehacer la última parte de la historia de una rama
 - Por ejemplo, si HEAD estuviese en master, `git reset --hard master~1` eliminaría el commit C8 (5558t8)
 - Como los cambios han quedado en el directorio de trabajo, estos se pueden integrar en un commit solo de master

Mostrar ramas

Muestra las ramas existentes después de haber creado los dos primeros commits.
Solo existe la rama master, que Git creo automáticamente con el primer commit.
El asterisco (*) indica que HEAD está en la rama master, que es la rama activa.

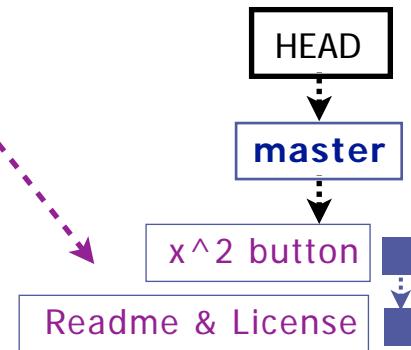
```
venus:cal jq$ git branch
* master
venus:cal jq$ git log --oneline
b0e63ad x^2 button
1096247 Readme & License
venus:cal jq$ git branch inverse 1096247
venus:cal jq$ git branch
  inverse
* master
venus:cal jq$ git branch -v
  inverse 1096247 Readme & License
* master b0e63ad x^2 button
venus:cal jq$ git checkout inverse
Switched to branch 'inverse'
venus:cal jq$ git branch
* inverse
  master
venus:cal jq$
```



Mostrar historia

```
venus:cal jq$  
venus:cal jq$ git branch  
* master  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git branch inverse 1096247  
venus:cal jq$  
venus:cal jq$ git branch  
    inverse  
* master  
venus:cal jq$ git branch -v  
    inverse 1096247 Readme & License  
* master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout inverse  
Switched to branch 'inverse'  
venus:cal jq$  
venus:cal jq$ git branch  
* inverse  
    master  
venus:cal jq$
```

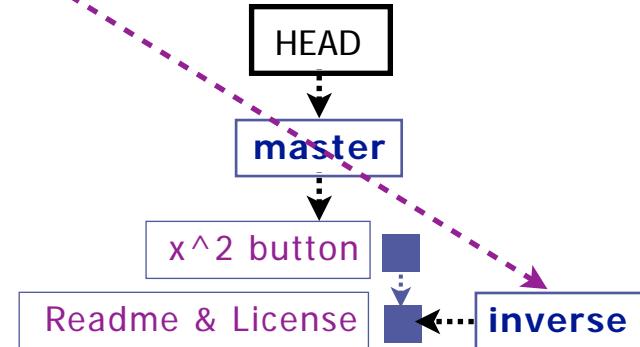
git log --oneline es equivalente a
git log --oneline HEAD que muestra
la historia de commits hasta HEAD.



Crear la rama inverse

```
venus:cal jq$  
venus:cal jq$ git branch  
* master  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git branch inverse 1096247  
venus:cal jq$  
venus:cal jq$ git branch  
    inverse  
* master  
venus:cal jq$ git branch -v  
    inverse 1096247 Readme & License  
* master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout inverse  
Switched to branch 'inverse'  
venus:cal jq$  
venus:cal jq$ git branch  
* inverse  
  master  
venus:cal jq$
```

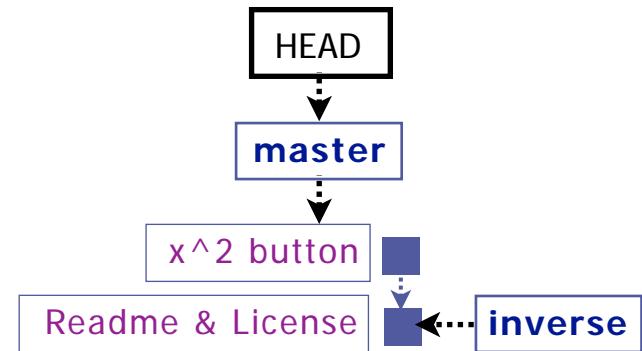
Crea la rama de nombre inverse con base en el commit **1096247** (equivalente a **HEAD~1**).



Mostrar ramas

```
venus:cal jq$  
venus:cal jq$ git branch  
* master  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git branch inverse 1096247  
venus:cal jq$  
venus:cal jq$ git branch  
  inverse  
* master  
venus:cal jq$ git branch -v  
  inverse 1096247 Readme & License  
* master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout inverse  
Switched to branch 'inverse'  
venus:cal jq$  
venus:cal jq$ git branch  
* inverse  
  master  
venus:cal jq$
```

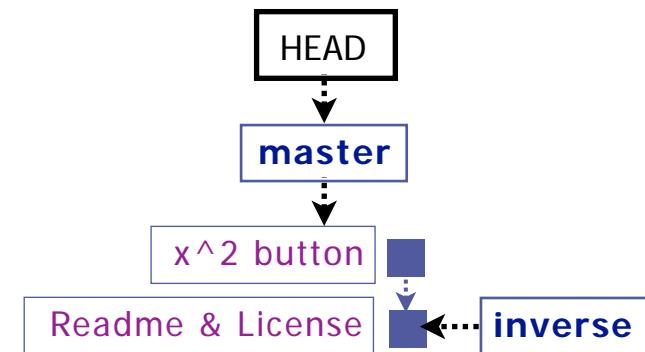
Muestra que ahora hay dos ramas, aunque la rama activa (*) sigue siendo master.



Mostrar ramas en modo verbose

```
venus:cal jq$  
venus:cal jq$ git branch  
* master  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git branch inverse 1096247  
venus:cal jq$  
venus:cal jq$ git branch  
    inverse  
* master  
venus:cal jq$ git branch -v  
  inverse 1096247 Readme & License  
* master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout inverse  
Switched to branch 'inverse'  
venus:cal jq$  
venus:cal jq$ git branch  
* inverse  
  master  
venus:cal jq$
```

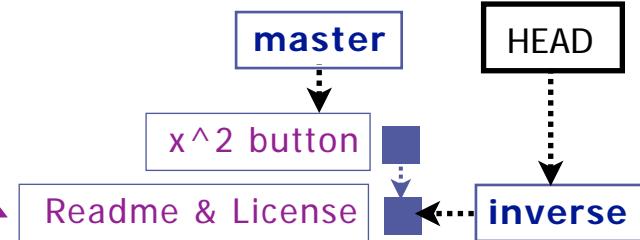
Muestra las dos ramas en modo verbose (**-v**), dando no solo el nombre de la rama, sino el id corto y el mensaje/título de su último commit.



Restaurar rama inverse

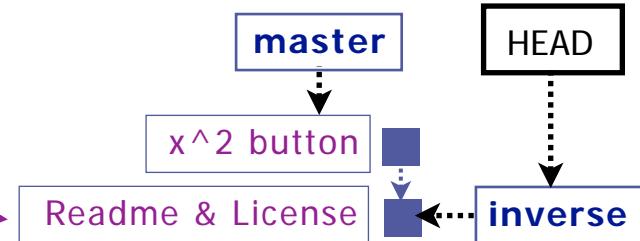
```
venus:cal jq$  
venus:cal jq$ git branch  
* master  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git branch inverse 1096247  
venus:cal jq$  
venus:cal jq$ git branch  
    inverse  
* master  
venus:cal jq$ git branch -v  
    inverse 1096247 Readme & License  
* master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout inverse  
Switched to branch 'inverse'  
venus:cal jq$  
venus:cal jq$ git branch  
* inverse  
  master  
venus:cal jq$
```

Restaura la rama inverse en el directorio de trabajo, para poder trabajar en ella. HEAD referencia ahora rama inverse (el commit 1096247).



Mostrar ramas

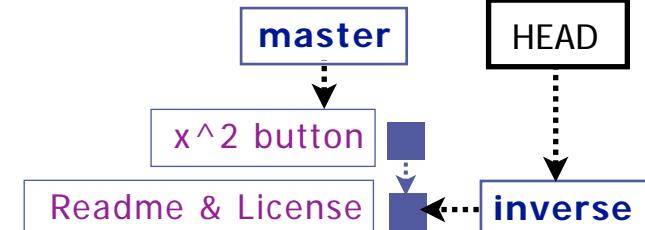
```
venus:cal jq$  
venus:cal jq$ git branch  
* master  
venus:cal jq$  
venus:cal jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git branch inverse 1096247  
venus:cal jq$  
venus:cal jq$ git branch  
    inverse  
* master  
venus:cal jq$ git branch -v  
    inverse 1096247 Readme & License  
* master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout inverse  
Switched to branch 'inverse'  
venus:cal jq$  
venus:cal jq$ git branch  
* inverse  
  master  
venus:cal jq$
```



Mostrar ramas en modo verbose

```
venus:cal jq$  
venus:cal jq$ git branch -v  
* inverse 1096247 Readme & License  
  master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ ls  
LICENSE README.md  
venus:cal jq$  
venus:cal jq$ # Edit calculator.html  
venus:cal jq$  
venus:cal jq$ ls  
LICENSE README.md calculator.html  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$ git commit -m "1/x button"  
[inverse e868dc4] 1/x button  
 1 file changed, 17 insertions(+)  
  create mode 100644 calculator.html  
venus:cal jq$  
venus:cal jq$ git log --oneline --all --graph  
* e868dc4 1/x button  
| * b0e63ad x^2 button  
|/  
* 1096247 Readme & License  
venus:cal jq$
```

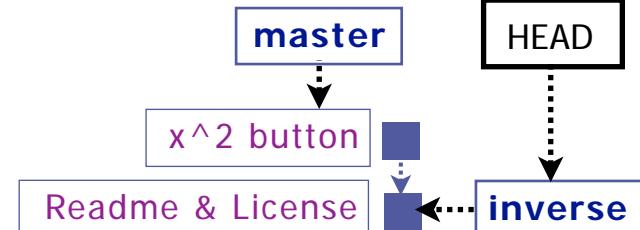
git branch -v muestra las dos ramas, incluyendo el id de commit asociado.
El asterisco indica que **inverse** es la rama activa.



Mostrar historia

```
venus:cal jq$ git branch -v
* inverse 1096247 Readme & License
  master b0e63ad x^2 button
venus:cal jq$ git log --oneline
1096247 Readme & License
venus:cal jq$ ls
LICENSE README.md
venus:cal jq$ # Edit calculator.html
venus:cal jq$ ls
LICENSE README.md calculator.html
venus:cal jq$ git add .
venus:cal jq$ git commit -m "1/x button"
[inverse e868dc4] 1/x button
 1 file changed, 17 insertions(+)
 create mode 100644 calculator.html
venus:cal jq$ git log --oneline --all --graph
* e868dc4 1/x button
| * b0e63ad x^2 button
|
* 1096247 Readme & License
venus:cal jq$
```

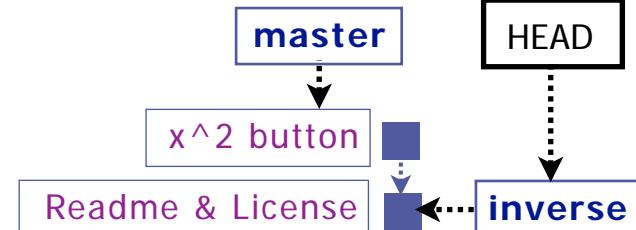
historia tiene ahora 1 commit.



Mostrar ficheros del directorio de trabajo

```
venus:cal jq$ git branch -v
* inverse 1096247 Readme & License
  master b0e63ad x^2 button
venus:cal jq$ git log --oneline
1096247 Readme & License
venus:cal jq$ ls
LICENSE          README.md
venus:cal jq$ # Edit calculator.html
venus:cal jq$ venus:cal jq$ ls
LICENSE          README.md      calculator.html
venus:cal jq$ git add .
venus:cal jq$ git commit -m "1/x button"
[inverse e868dc4] 1/x button
 1 file changed, 17 insertions(+)
  create mode 100644 calculator.html
venus:cal jq$ git log --oneline --all --graph
* e868dc4 1/x button
| * b0e63ad x^2 button
|
* 1096247 Readme & License
venus:cal jq$
```

calculator.html no aparece en el directorio de trabajo, porque el commit 1096247 no lo incluyó.



Crear fichero: calculator.html

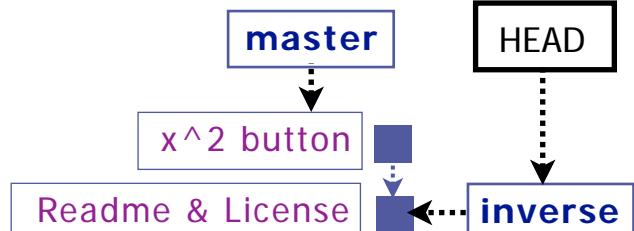


```
venus:cal jq$ git branch -v
* inverse 1096247 Readme & License
  master b0e63ad x^2 button
venus:cal jq$ git log --oneline
1096247 Readme & License
venus:cal jq$ ls
LICENSE          README.md
venus:cal jq$ # Edit calculator.html
venus:cal jq$ ls
LICENSE          README.md      calculator.html
venus:cal jq$ git add .
venus:cal jq$ git commit -m "1/x button"
[inverse e868dc4] 1/x button
  1 file changed, 17 insertions(+)
   create mode 100644 calculator.html
venus:cal jq$ git log --oneline --all --graph
* e868dc4 1/x button
| * b0e63ad x^2 button
|/
* 1096247 Readme & License
venus:cal jq$
```

```
<!DOCTYPE html><html><head>
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">

function inverse() {
  var num = document.getElementById("n1");
  num.value = 1/num.value;
}
</script>
</head>
<body>
  Number:
  <input type="text" id="n1" onclick="empty()"><p>
  <button onclick="inverse()"> 1/x </button>
</body>
</html>
```

Editamos la nueva calculadora en el dir. de trabajo.

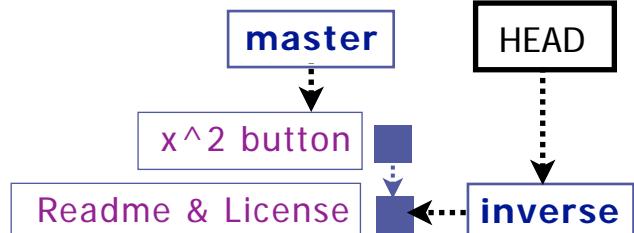


Mostrar fich. del dir. de trabajo

```
venus:cal jq$  
venus:cal jq$ git branch -v  
* inverse 1096247 README & License  
  master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git log --oneline  
1096247 README & License  
venus:cal jq$  
venus:cal jq$ ls  
LICENSE          README.md  
venus:cal jq$  
venus:cal jq$ # Edit calculator.html  
venus:cal jq$  
venus:cal jq$ ls  
LICENSE          README.md  
calculator.html  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$ git commit -m "1/x button"  
[inverse e868dc4] 1/x button  
 1 file changed, 17 insertions(+)  
  create mode 100644 calculator.html  
venus:cal jq$  
venus:cal jq$ git log --oneline --all --graph  
* e868dc4 1/x button  
| * b0e63ad x^2 button  
|/  
* 1096247 README & License  
venus:cal jq$
```

```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
function inverse() {  
    var num = document.getElementById("n1");  
    num.value = 1/num.value;  
}  
</script>  
</head>  
<body>  
    Number:  
    <input type="text" id="n1" onclick="empty()"><p>  
  
    <button onclick="inverse()"> 1/x </button>  
</body>  
</html>
```

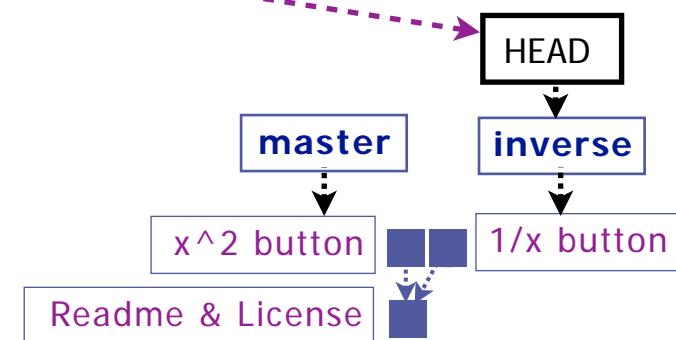
Después de crear **calculator.html** con el editor ls lo muestra, pero ahora tiene otro programa: el botón 1/x



Registrar en índice y crear commit

```
venus:cal jq$ git branch -v
* inverse 1096247 README & License
  master b0e63ad x^2 button
venus:cal jq$ git log --oneline
1096247 README & License
venus:cal jq$ ls
LICENSE          README.md
venus:cal jq$ venus:cal jq$ # Edit calculator.html
venus:cal jq$ venus:cal jq$ ls
LICENSE          README.md      calculator.html
venus:cal jq$ venus:cal jq$ git add .
venus:cal jq$ git commit -m "1/x button"
[inverse e868dc4] 1/x button
 1 file changed, 17 insertions(+)
 create mode 100644 calculator.html
venus:cal jq$ venus:cal jq$ git log --oneline --all --graph
* e868dc4 1/x button
| * b0e63ad x^2 button
|
* 1096247 README & License
venus:cal jq$
```

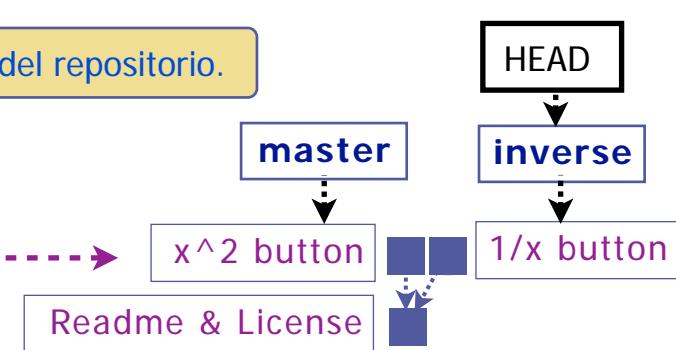
Registrar todos los cambios en el índice y generar el nuevo commit "1/x button".



Mostrar grafo de commits

```
venus:cal jq$ git branch -v
* inverse 1096247 README & License
  master b0e63ad x^2 button
venus:cal jq$ git log --oneline
1096247 README & License
venus:cal jq$ ls
LICENSE README.md
venus:cal jq$ # Edit calculator.html
venus:cal jq$ ls
LICENSE README.md calculator.html
venus:cal jq$ git add .
venus:cal jq$ git commit -m "1/x button"
[inverse e868dc4] 1/x button
 1 file changed, 17 insertions(+)
 create mode 100644 calculator.html
venus:cal jq$ git log --oneline --all --graph
* e868dc4 1/x button
| * b0e63ad x^2 button
|/
* 1096247 README & License
venus:cal jq$
```

Muestra el grafo de commits del repositorio.



Ejemplo de show

```
venus:cal jq$ git show master  
commit b0e63ad0a57510d65814ac8ff98c7eb16bf359c1  
Author: Juan Quemada <jquemada@dit.upm.es>  
Date: Tue Feb 14 19:23:10 2017 +0100  
x^2 button  
  
diff --git a/calculator.html b/calculator.html  
new file mode 100644  
index 000000..839b37e  
--- /dev/null  
+++ b/calculator.html  
@@ -0,0 +1,17 @@  
+<!DOCTYPE html><html><head>  
+<title>Calculator</title><meta charset="utf-8">  
+<script type="text/javascript">  
+  
+function square() {  
+  var num = document.getElementById("n1");  
+  num.value = num.value * num.value;  
+}  
+</script> ←-----→  
+</head>  
+<body>  
+ Number:  
+ <input type="text" id="n1"><p>  
+ <button onclick="square()"> x2 </button>  
+</body>  
+</html>  
venus:cal jq$
```

Identificador largo del commit

Autor del commit

Fecha del commit

Mensaje o título del commit

calculator.html tiene los cambios indicados

Todo el código esta en verde, porque el fichero es nuevo. Todo se ha añadido en este commit.



```
<!DOCTYPE html><html><head>  
<title>Calculator</title>  
<script type="text/javascript">  
  
function square() {  
  var num = document.getElementById("n1");  
  num.value = num.value * num.value;  
}  
</script>  
</head>  
<body>  
Number:  
<input type="text" id="n1"><p>  
<button onclick="square()"> x2 </button>  
</body>  
</html>
```

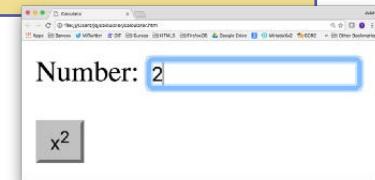
Ejemplo de diff

```
venus:cal jq$ git diff inverse master
diff --git a/calculator.html b/calculator.html
index fc74320..839b37e 100644
--- a/calculator.html
+++ b/calculator.html
@@ -2,9 +2,9 @@
 <title>Calculator</title><meta charset="utf-8">
 <script type="text/javascript">

-function inverse() {
+function square() {
    var num = document.getElementById("n1");
-    num.value = 1/num.value;
+    num.value = num.value * num.value;
}
</script>
</head>
@@ -12,6 +12,6 @@ function inverse() {
    Number:
    <input type="text" id="n1"><p>
-    <button onclick="inverse()"> 1/x </button>
+    <button onclick="square()"> x2 </button>
</body>
</html>
venus:cal jq$
```

Muestra las diferencias entre los últimos commits de las ramas inverse y master.

b/calculator.html es del commit de master



a/calculator.html es del commit de inverse



Negro: código que delimita lo añadido o eliminado

Rojo: código eliminado del primer commit.

Verde: código añadido en el segundo commit.

Ejercicio opcional

- ◆ Clonar el repositorio <su_cuenta>/cal_2com copiado con Fork en el tema anterior
 - `git clone https://github.com/<su_cuenta>/cal_2com`
 - `git clone ..` copia el repositorio remoto en un repositorio local, con directorio de trabajo, de nombre **cal_2com**
 - Mostrar con **ls** el directorio que se acaba de crear y con **ls -a cal_2com** su contenido, incluyendo .git
- ◆ Entrar en el directorio de trabajo cal_2com del repositorio clonado
 - `cd cal_2com` para que los directorios de trabajo sean el mismo
 - Mostrar estado del directorio con **git status ..**, historia y grafo con **git log ..**, contenido con **ls ..**, etc.
- ◆ Crear la rama **mi_rama** en el commit "**Readme & License**"
 - Crearla con **git branch mi_rama <commit>** identificando el commit por id o por master~1
 - Restaurar la rama con **git checkout mi_rama** para poder empezar a hacer cambios en ella
- ◆ Copiar **calculadora.html** del commit "**x^2 button**" al directorio de trabajo
 - Copiarlo del commit x^2 button, descargarlo de GitHub, editararlo desde cero, etc.
- ◆ Cambiar HTML y JavaScript de x^2 para que calcule el cubo y muestre x^3
 - Línea 5: **function square() ...** por **function cube() ...**
 - Línea 7: **... num.value * num.value;** por **... Math.pow(num.value, 3);**
 - Línea 14: **...onclick="square()">x²...** por **...onclick="cube()">x³...**
- ◆ Registrar con **git add .** y crear commit con **git commit -m "x^3 button"**
 - Mostrar la historia de la rama master **git log --oneline**
- ◆ Subir **mi_rama** al repositorio copiado con Fork con
 - `git push https://github.com/<su_cuenta>/cal_2com mi_rama`
 - Inspeccionar con el navegador el repositorio actualizado y la nueva rama creada



Git y GitHub

Integración de ramas: merge, commit, checkout y fast-forward

Juan Quemada, DIT - UPM

Integración de commits con merge

◆ **git merge <commit>** intenta automatizar la integración de commit

- Por defecto utiliza la estrategia recursiva que integra el commit indicado, fichero a fichero, en el commit actual, aproximadamente así:
 - Ficheros iguales en ambos commits: incluye el fichero común (**sin conflicto**)
 - Fichero está solo en uno de los commits: incluye el fichero (**sin conflicto**)
 - Ficheros con diferencias disjuntas: une ambos ficheros con auto-merge (**sin conflicto**)
 - Ficheros con ancestro común en sus historias: incluye el último fichero (**sin conflicto**)
 - Ficheros con diferencias en el mismo contexto: une ambos, marca diferencias y **genera conflicto**
- Si al acabar no hay conflictos, genera un commit de integración de tipo auto-merge
 - Aunque la integración tenga éxito, el commit resultante debe probarse para comprobar que funciona bien
- Si hay conflictos, no se genera commit
 - Los conflictos se marcan en los ficheros donde ocurren

◆ **git merge**

- Intenta integrar un commit con el activo (HEAD)
 - **git merge master** integra el último commit de master en el commit activo (HEAD)
 - Si hay conflictos, estos deben ser resueltos, cerrando el commit de integración con **git commit ...**
 - **git merge g699g8** integra el commit g699g8 con el commit activo (HEAD)
 - **git merge -m “msg” master** integra master en HEAD y asigna el mensaje “**msg**” al commit de integr.
 - **git merge --abort** aborta la integración sin generar commit de integración cuando ha habido conflictos

Conflictos de integración

◆ Si hay conflictos no se genera commit de integración

- Los **conflictos** quedan marcados en el fichero en el directorio de trabajo
 - en un estado especial denominado **unmerged**
- Los conflictos se resuelven editando manualmente
 - Una vez resueltos los conflictos se genera el commit de integración con **git commit ...**
- Los **ficheros sin conflictos** quedan también en el directorio de trabajo
 - en estado **staged**

```
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">
++<<<<< HEAD
+function inverse() {
+  var num = document.getElementById("n1");
+  num.value = 1/num.value;
+=====
+ function square() {
+  var num = document.getElementById("n1");
+  num.value = num.value * num.value;
+>>>>> master
}
</script>
</head>
```

Conflictos quedan marcado así en los ficheros fuente.

El contexto (en negro) son las líneas idénticas de ambos ficheros que delimitan las líneas con conflicto.

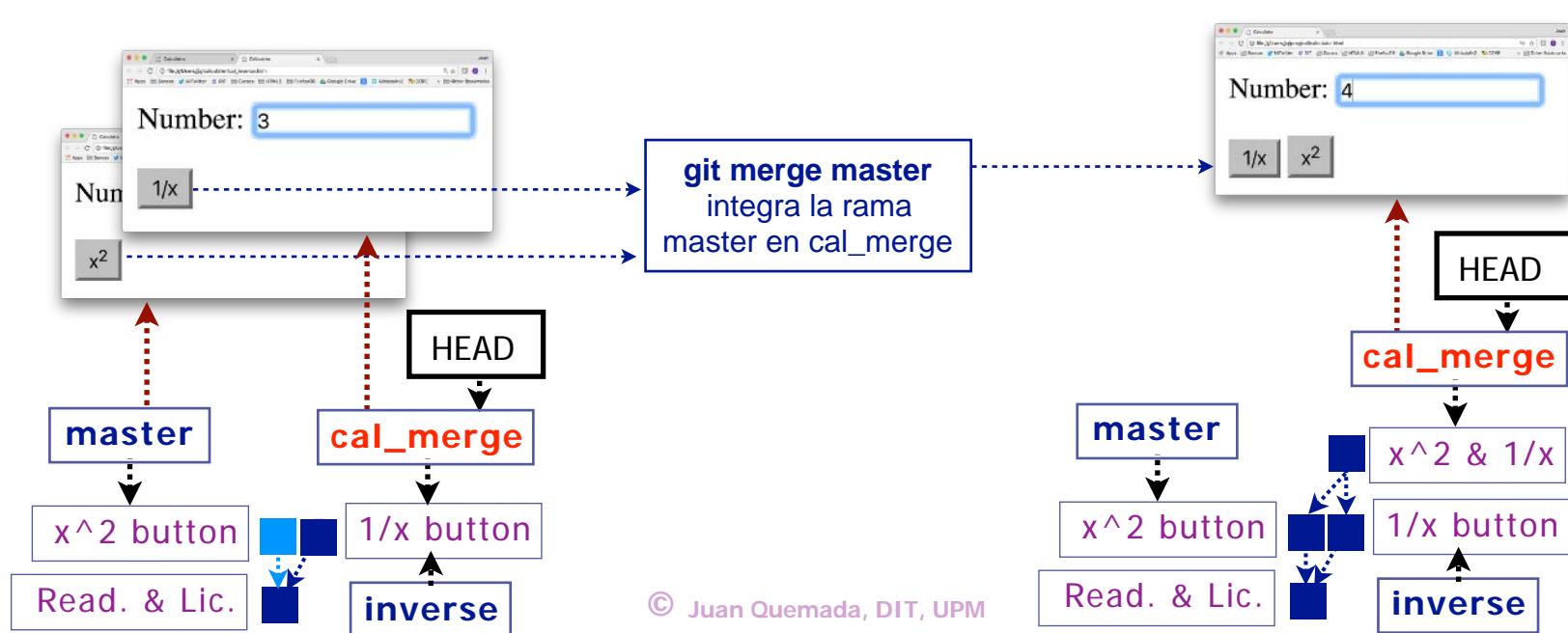
La primera parte (HEAD) son las líneas de código conflictivas de la rama activa.

La segunda parte (master) son las líneas de código conflictivas de la rama a integrar (master).

El contexto (en negro) son las líneas idénticas de ambos ficheros que delimitan las líneas con conflicto.

Integración de master en cal_merge

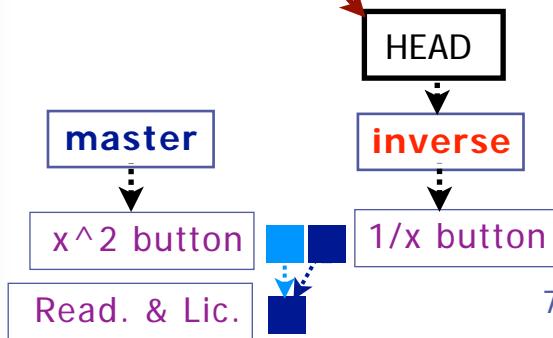
- ◆ Para integrar los 2 botones desarrollados en ramas diferentes
 - Se deben integrar las 2 ramas con **git merge** ...
- ◆ Se va a integrar master en cal_merge (HEAD) con **git merge master**
 - Los conflictos generados se resolverán manualmente
 - Una vez resueltos se genera el commit de integración con **git commit** ...



Mostrar ramas en modo verbose

```
venus:cal jq$ git branch -v
* inverse e868dc4 1/x button
  master b0e63ad x^2 button
venus:cal jq$ git checkout -b cal_merge
Switched to a new branch 'cal_merge'
venus:cal jq$ git branch -v
* cal_merge e868dc4 1/x button
  inverse e868dc4 1/x button
  master b0e63ad x^2 button
venus:cal jq$ git merge master
Auto-merging calculator.html
CONFLICT (add/add): Merge conflict in calculator.html
Automatic merge failed; fix conflicts and then commit the result.
venus:cal jq$ git status -s
AA calculator.html
venus:cal jq$ # Fix conflicts with editor
venus:cal jq$ # -> in calculator.html
venus:cal jq$ git commit -a -m "Integrate x^2 & 1/x"
[cal_merge 1898ac7] Integrate x^2 & 1/x
venus:cal jq$ git log --oneline --graph
* 1898ac7 Integrate x^2 & 1/x
|\ 
| * b0e63ad x^2 button
* | e868dc4 1/x button
|/
* 1096247 Readme & License
venus:cal jq$
```

Muestra ramas.



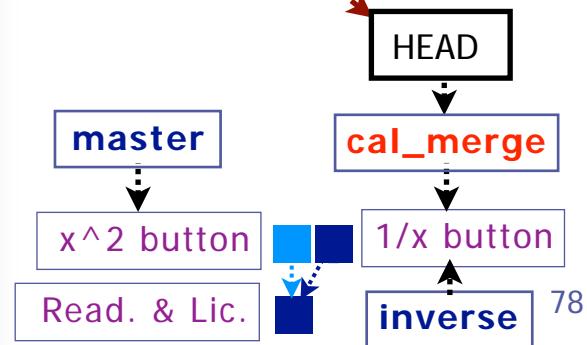
Integrar master en inverse y mostar ramas

```
[venus:cal jq$] git branch -v  
* inverse e868dc4 1/x button  
[ master b0e63ad x^2 button  
[venus:cal jq$] git checkout -b cal_merge  
Switched to a new branch 'cal_merge'  
[venus:cal jq$] git branch -v  
* cal_merge e868dc4 1/x button  
[ inverse e868dc4 1/x button  
[ master b0e63ad x^2 button  
[venus:cal jq$] git merge master  
Auto-merging calculator.html  
CONFLICT (add/add): Merge conflict in calculator.html  
Automatic merge failed; fix conflicts and then commit the result.  
[venus:cal jq$] git status -s  
AA calculator.html  
[venus:cal jq$] # Fix conflicts with editor  
[venus:cal jq$] # -> in calculator.html  
[venus:cal jq$] git commit -a -m "Integrate x^2 & 1/x"  
[cal_merge 1898ac7] Integrate x^2 & 1/x  
[venus:cal jq$] git log --oneline --graph  
* 1898ac7 Integrate x^2 & 1/x  
|  
| * b0e63ad x^2 button  
* | e868dc4 1/x button  
|/  
* 1096247 Readme & License  
[venus:cal jq$]
```

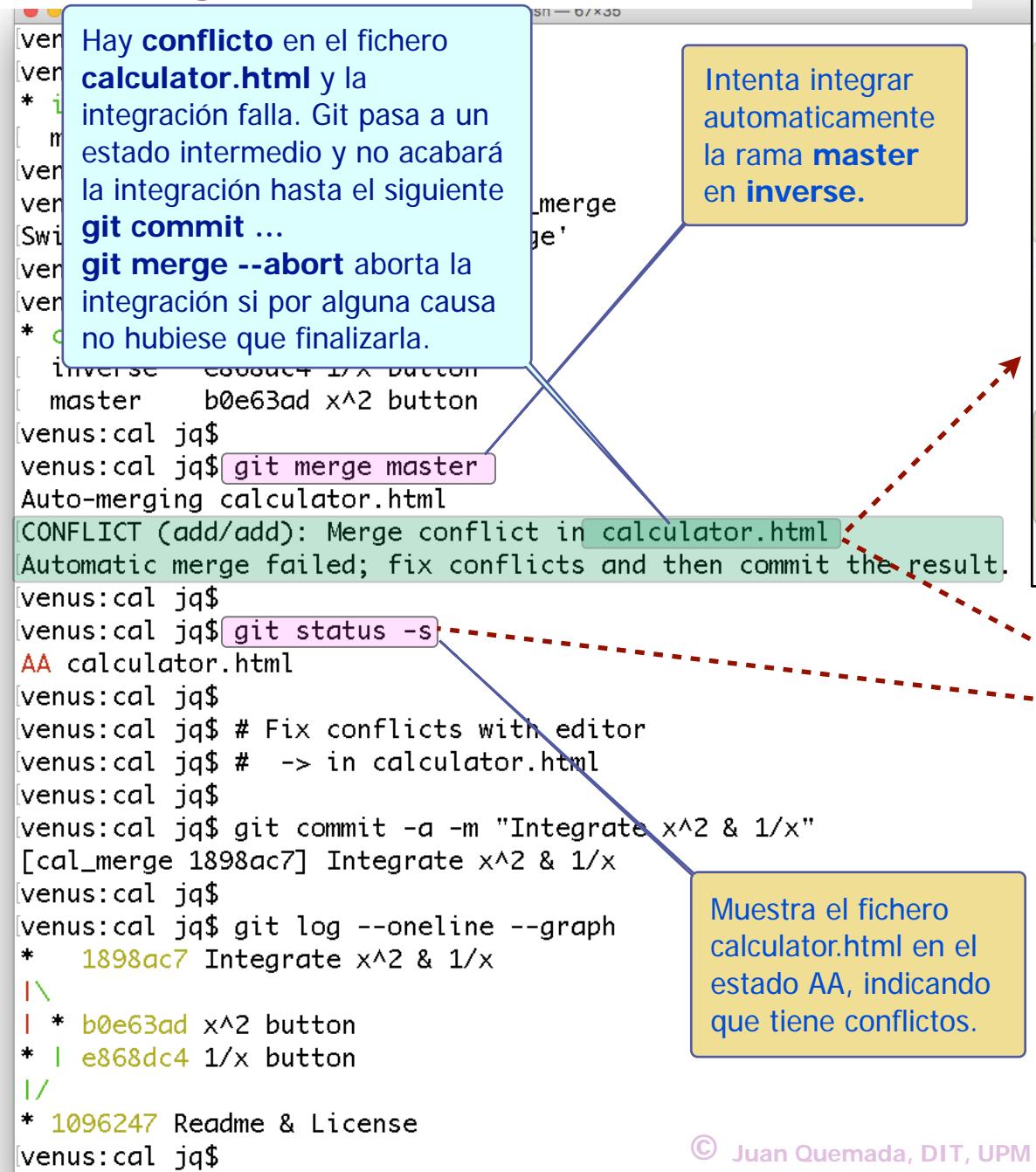
git checkout -b cal_merge crea la nueva rama **cal_merge** en HEAD y realiza un checkout a ella. Equivale a 2 comandos:

- git branch cal_merge
- git checkout cal_merge

Muestra la nueva rama **cal_merge** (activa). Al arrancar un nuevo desarrollo es conveniente crear una rama nueva, así dejamos el anterior disponible en la **rama inverse**.



Integrar master en inverse



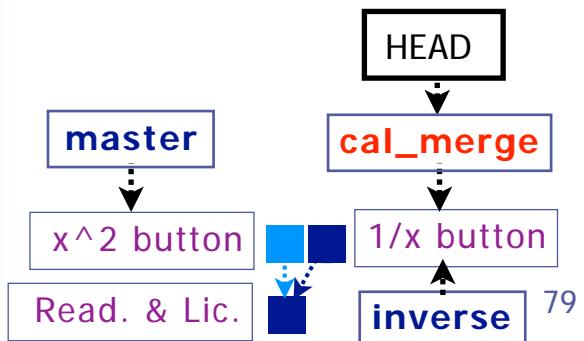
```
<!DOCTYPE html><html><head>
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">
```

```
<<<<< HEAD
function inverse() {
    var num = document.getElementById("n1");
    num.value = 1/num.value;
}
=====
```

```
function square() {
    var num = document.getElementById("n1");
    num.value = num.value * num.value;
}
>>>>> master
}
</script>
</head>
<body>
Number:
<input type="text" id="n1"><p>
```

```
<<<<< HEAD
<button onclick="inverse()"> 1/x </button>
=====
<button onclick="square()"> x2 </button>
>>>>> master
</body>
</html>
```

Conflictos en
calculator.html
en dir. de trab.



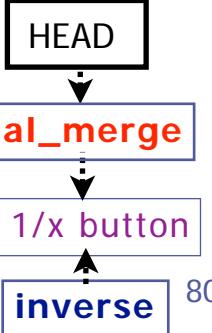
Resolver conflictos

```
venus:cal jq$  
venus:cal jq$ git branch -v  
* inverse e868dc4 1/x button  
| master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout -b cal_merge  
Switched to a new branch 'cal_merge'  
venus:cal jq$  
venus:cal jq$ git branch -v  
* cal_merge e868dc4 1/x button  
| inverse e868dc4 1/x button  
| master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git merge master  
Auto-merging calculator.html  
CONFLICT (add/add): Merge conflict in calculator.html  
Automatic merge failed; fix conflicts and then commit the result.  
venus:cal jq$  
venus:cal jq$ git status -s  
AA calculator.html  
venus:cal jq$  
venus:cal jq$ # Fix conflicts with editor  
venus:cal jq$ # -> in calculator.html  
venus:cal jq$  
venus:cal jq$ git commit -a -m "Integrate x^2 & 1/x"  
[cal_merge 1898ac7] Integrate x^2 & 1/x  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph  
* 1898ac7 Integrate x^2 & 1/x  
|\  
| * b0e63ad x^2 button  
| * e868dc4 1/x button  
|/  
* 1096247 Readme & License  
venus:cal jq$
```

```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
function inverse() {  
    var num = document.getElementById("n1");  
    num.value = 1/num.value;  
}  
  
function square() {  
    var num = document.getElementById("n1");  
    num.value = num.value * num.value;  
}  
</script>  
</head>  
<body>  
    Number:  
    <input type="text" id="n1"><p>  
        <button onclick="inverse()"> 1/x </button>  
        <button onclick="square()"> x2 </button>  
</body>  
</html>
```

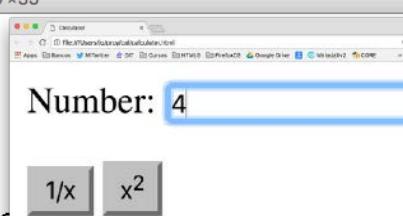
Los conflictos se resuelven con el editor, integrando **x²** y **1/x**.

Conflictos resueltos en calculator.html



Crear commit de integración

```
venus:cal jq$  
venus:cal jq$ git branch -v  
* inverse e868dc4 1/x button  
| master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout -b cal_merge  
Switched to a new branch 'cal_merge'  
venus:cal jq$  
venus:cal jq$ git branch -v  
* cal_merge e868dc4 1/x button  
| inverse e868dc4 1/x button  
| master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git merge master  
Auto-merging calculator.html  
CONFLICT (add/add): Merge conflict in calculator.html  
Automatic merge failed; fix conflicts and then commit the result.  
venus:cal jq$  
venus:cal jq$ git status -s  
AA calculator.html  
venus:cal jq$  
venus:cal jq$ # Fix conflicts with editor  
venus:cal jq$ # -> in calculator.html  
venus:cal jq$  
venus:cal jq$ git commit -a -m "Integrate x^2 & 1/x"  
[cal_merge 1898ac7] Integrate x^2 & 1/x  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph  
* 1898ac7 Integrate x^2 & 1/x  
|\_ | * b0e63ad x^2 button  
* | e868dc4 1/x button  
|/_ | * 1096247 Readme & License  
venus:cal jq$
```

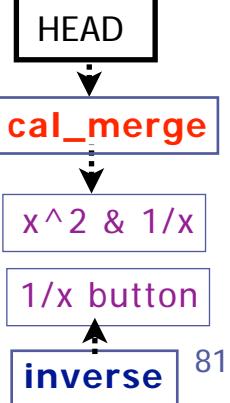


```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
function inverse() {  
    var num = document.getElementById("n1");  
    num.value = 1/num.value;  
}  
  
function square() {  
    var num = document.getElementById("n1");  
    num.value = num.value * num.value;  
}  
</script>  
</head>  
<body>  
Number:  
<input type="text" id="n1"><p>  
  
<button onclick="inverse()"> 1/x </button>  
<button onclick="square()"> x2 </button>  
</body>  
</html>
```

git commit -a -m "... finaliza la integración. opción -a añade ficheros modificados al índice y cierra commit.

```
git commit -a -m "Integrate x^2 & 1/x"
```

git log --oneline --graph muestra el grafo de integración de la rama cal_merge. El commit de integración junta las 2 líneas del desarrollo. Une las 2 ramas en cal_merge.



Conflictos vistos con diff

```
venus:cal jq$ git diff
diff --cc calculator.html
index fc74320,839b37e..0000000
--- a/calculator.html
+++ b/calculator.html
@@@ -2,9 -2,9 +2,15 @@@
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">

++<<<<< HEAD
+function inverse() {
+  var num = document.getElementById("n1");
+  num.value = 1/num.value;
+=====
+ function square() {
+  var num = document.getElementById("n1");
+  num.value = num.value * num.value;
+>>>>> master
}
</script>
</head>
@@@ -12,6 -12,6 +18,10 @@@
Number:
<input type="text" id="n1"><p>

++<<<<< HEAD
+ <button onclick="inverse()> 1/x </button>
+=====
+ <button onclick="square()> x2 </button>
+>>>>> master
</body>
</html>
venus:cal jq$
```

git diff muestra así los conflictos **antes** de resolverlos.

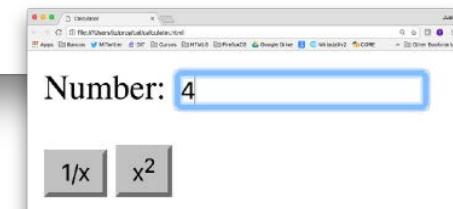
Resolver conflictos con un editor.

```
venus:cal jq$ git diff
diff --cc calculator.html
index fc74320,839b37e..0000000
--- a/calculator.html
+++ b/calculator.html
@@@ -2,16 -2,16 +2,22 @@@
<title>Calculator</title><meta charset='utf-8'>
<script type="text/javascript">

+function inverse() {
+  var num = document.getElementById("n1");
+  num.value = 1/num.value;
+}
+=====
+ function square() {
+  var num = document.getElementById("n1");
+  num.value = num.value * num.value;
+ }
</script>
</head>
<body>
Number:
<input type="text" id="n1"><p>

+ <button onclick="inverse()> 1/x </button>
+ <button onclick="square()> x2 </button>
</body>
</html>
venus:cal jq$
```

git diff muestra así los conflictos resueltos, **después** de integrarlos con un editor y antes de cerrar el commit.





Integración ff (fast-forward)

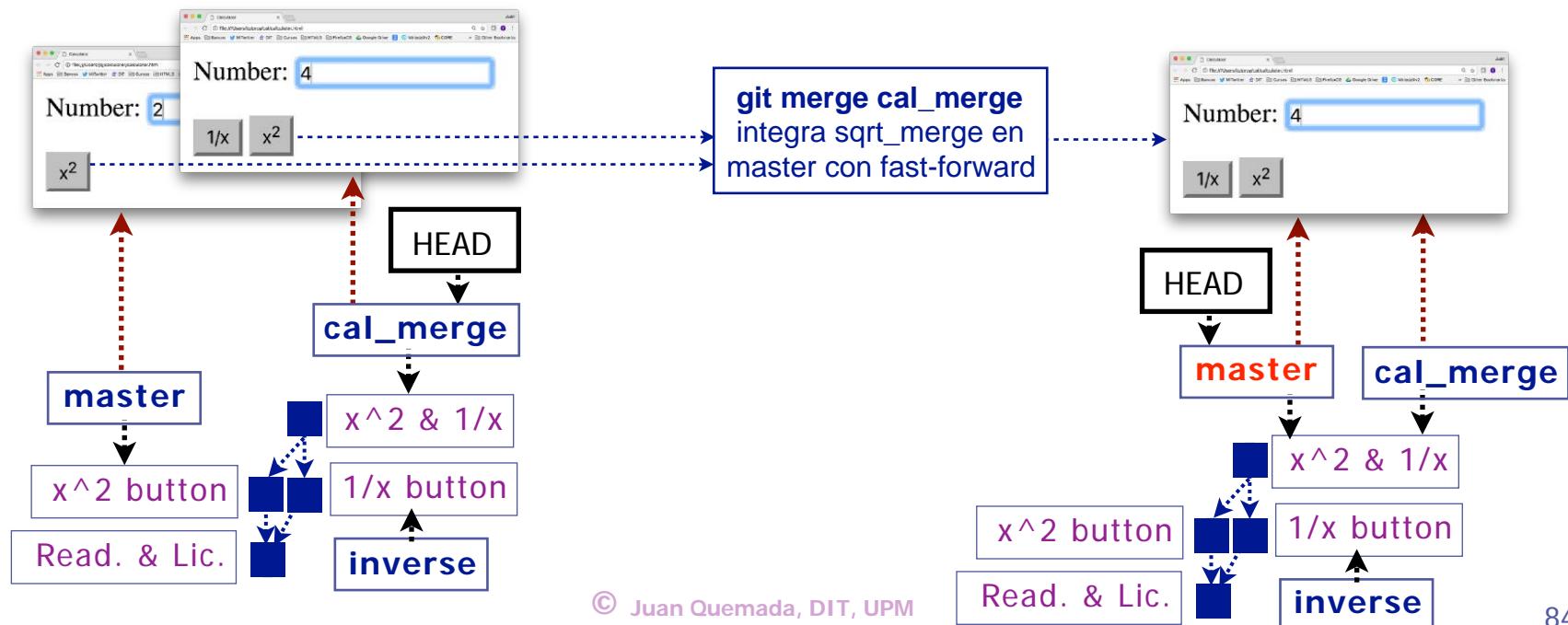
Integración ff (fast-forward)

◆ **git merge ...** detecta si la integración pedida se ha realizado ya en otra rama

- Si es así, Git está configurado para realizar una integración ff (Fast-Forward)
 - Simplemente avanzara el puntero de rama hasta el commit que contiene la integración solicitada
 - No se generará un nuevo commit, se reutilizará el existente

◆ Al integrar la rama `cal_merge` en la rama `master`

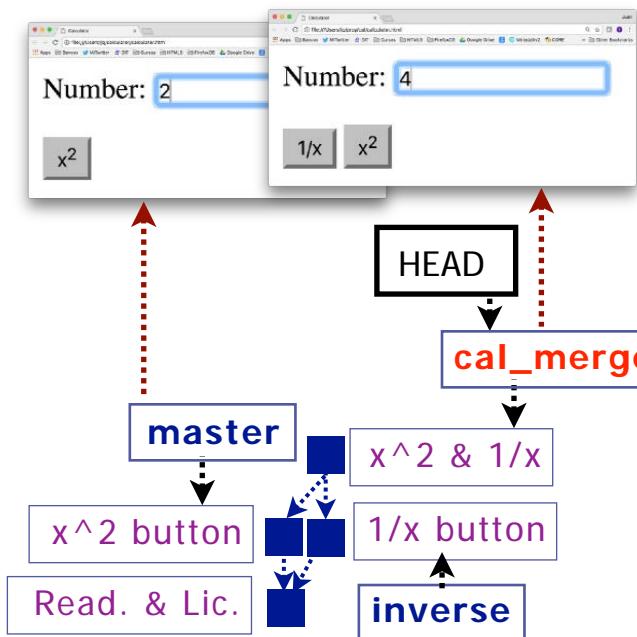
- Se reutiliza con **fast-forward** la integración realizada en la rama `cal_merge`
 - Porque master se ha integrado ya en `cal_merge`, tal y como muestra el grafo de commits



Mostrar ramas

```
venus:cal jq$ git branch -v
* cal_merge 1898ac7 Integrate x^2 & 1/x
  inverse   e868dc4 1/x button
  master    b0e63ad x^2 button
venus:cal jq$ git checkout master
Switched to branch 'master'
venus:cal jq$ git branch square
venus:cal jq$ git branch -v
  cal_merge 1898ac7 Integrate x^2 & 1/x
  inverse   e868dc4 1/x button
* master    b0e63ad x^2 button
  square    b0e63ad x^2 button
venus:cal jq$ git merge cal_merge
Updating b0e63ad..1898ac7
Fast-forward
  calculator.html | 6 ++++++
  1 file changed, 6 insertions(+)
venus:cal jq$ git log --oneline --graph
* 1898ac7 Integrate x^2 & 1/x
|\_
| * b0e63ad x^2 button
* | e868dc4 1/x button
|/
* 1096247 Readme & License
venus:cal jq$
```

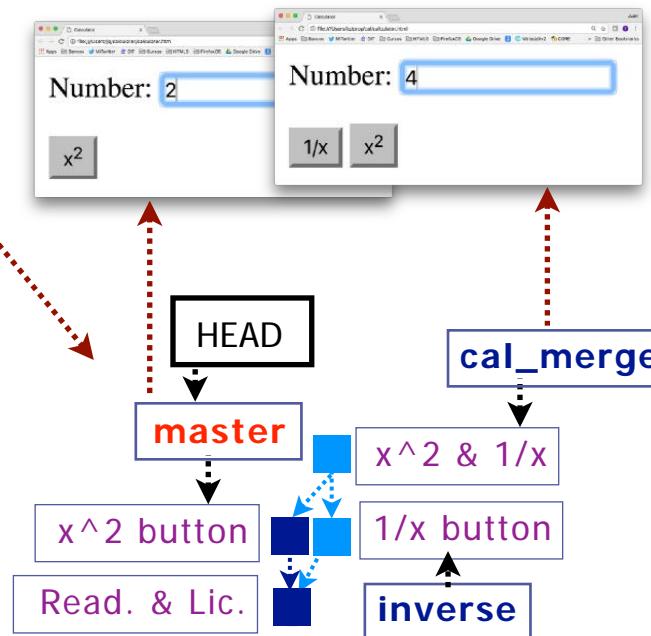
git branch muestra las 3 ramas existentes, marcando la rama cal_merge como activa.



Restaurar rama master

```
venus:cal jq$  
venus:cal jq$ git branch -v  
* cal_merge 1898ac7 Integrate x^2 & 1/x  
  inverse   e868dc4 1/x button  
  master    b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout master  
Switched to branch 'master'  
venus:cal jq$  
venus:cal jq$ git branch square  
venus:cal jq$  
venus:cal jq$ git branch -v  
  cal_merge 1898ac7 Integrate x^2 & 1/x  
  inverse   e868dc4 1/x button  
* master    b0e63ad x^2 button  
  square    b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git merge cal_merge  
Updating b0e63ad..1898ac7  
Fast-forward  
  calculator.html | 6 ++++++  
   1 file changed, 6 insertions(+)  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph  
* 1898ac7 Integrate x^2 & 1/x  
|  
| * b0e63ad x^2 button  
* | e868dc4 1/x button  
|/  
* 1096247 Readme & License  
venus:cal jq$
```

git checkout master restaura la rama master para trabajar en ella.



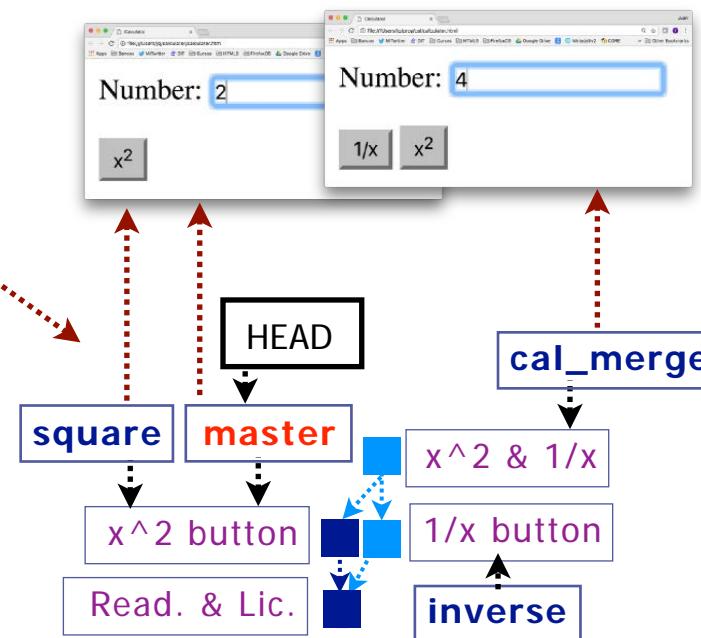
cal — bash — 42x3

```
venus:cal jq$  
venus:cal jq$ git branch -  
* cal_merge 1898ac7 Integr  
    inverse e868dc4 1/x button  
    master b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout master  
Switched to branch 'master'  
venus:cal jq$  
venus:cal jq$ git branch square  
venus:cal jq$  
venus:cal jq$ git branch -v  
    cal_merge 1898ac7 Integrate x^2 & 1/x  
    inverse e868dc4 1/x button  
* master b0e63ad x^2 button  
    square b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git merge cal_merge  
Updating b0e63ad..1898ac7  
Fast-forward  
  calculator.html | 6 ++++++  
   1 file changed, 6 insertions(+)  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph  
* 1898ac7 Integrate x^2 & 1/x  
|  
| * b0e63ad x^2 button  
* | e868dc4 1/x button  
|/  
* 1096247 Readme & License  
venus:cal jq$
```

Crear rama square y mostrarla

git branch square crea una **rama square** con base en master (b0e63ad), que permitirá acceder a este commit cuando master avance.

git branch muestra ahora las 4 ramas y **master** como activa.



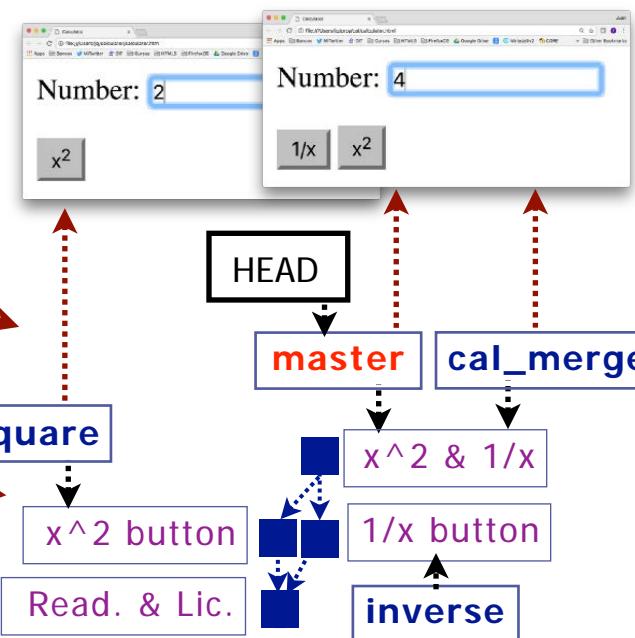
Integrar en rama master

```
venus:cal jq$ git branch -v
* cal_merge 1898ac7 Integrate x^2 & 1/x
  inverse   e868dc4 1/x button
  master    b0e63ad x^2 button
venus:cal jq$ git checkout master
Switched to branch 'master'
venus:cal jq$ git branch square
venus:cal jq$ git branch -v
  cal_merge 1898ac7 Integrate x^2 & 1/x
  inverse   e868dc4 1/x button
* master    b0e63ad x^2 button
  square    b0e63ad x^2 button
venus:cal jq$ git merge cal_merge
Updating b0e63ad..1898ac7
Fast-forward
  calculator.html | 6 ++++++
  1 file changed, 6 insertions(+)
venus:cal jq$ git log --oneline --graph
* 1898ac7 Integrate x^2 & 1/x
| \
| * b0e63ad x^2 button
| * | e868dc4 1/x button
| /
* 1096247 Readme & License
venus:cal jq$
```

git merge cal_merge integra la rama cal_merge en la rama activa (master).

La integración es **ff** o **fast-forward**, porque el commit a integrar está ya integrado con el último de master, tal y como indica el grafo.

Ambas ramas comparten ahora el mismo commit con el mismo identificador.



git log --oneline --graph muestra ahora el mismo grafo de commits que cal_merge. Ambas ramas están en el mismo commit. cal_merge se podría borrar ahora sin perder información.



Final del tema

Ejercicio opcional

- ◆ Restaurar **mi_rama** para integrar en ella la rama master
 - restaurarla con: **git checkout mi_rama**
- ◆ Integrar la rama master en la rama actual con: **git merge master**
 - Para integrar los botones x^2 y x^3 en una única calculadora
 - La integración dará conflictos en el fichero calculator.html en las líneas modificadas
 - Mostrar los conflictos tanto con **git diff**, como con **git status**
- ◆ Resolver los conflictos (editor) para que los 2 botones funcionen bien
 - Dejar el código de ambas calculadoras, tanto en HTML, como en JavaScript
 - Una vez que la calculadora funcione correctamente, registrar el cambio en el índice con **git add** .
- ◆ Crear commit con **git commit -m "Integrar x^2 & x^3 "**
 - Mostrar la historia de integración de **mi_rama** con **git log --online --graph**
- ◆ Restaurar master con: **git checkout master**
 - Integrar **mi_rama** en **master** con: **git merge mi_rama**
 - Al haberse integrado ambas ramas anteriormente, la integración se realiza ahora con fast-forward
- ◆ Subir ambas ramas, **master** y **mi_rama**, al repositorio
 - **git push --all https://github.com/<su_cuenta>/cal_2com**
 - Inspeccionar con el navegador el repositorio actualizado y sus ramas



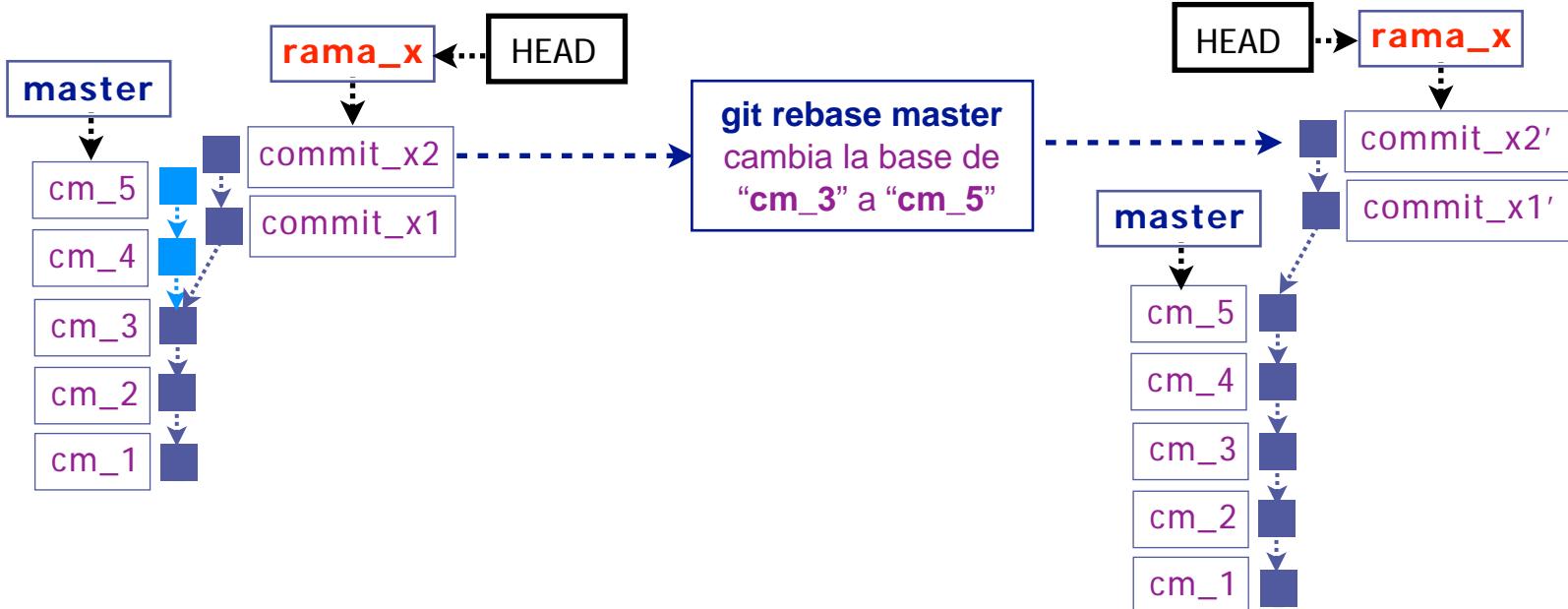
Git y GitHub

Integración de ramas con rebase

Juan Quemada, DIT - UPM

Cambiar la base de una rama

- ◆ Una rama se usa para hacer desarrollos separados de la rama master
 - Al finalizar el desarrollo de la rama, se puede integrar con master u otra rama
- ◆ Cambiar la base de una rama permite también integrar desarrollos
 - Cambiar de base (**rebase**) integra los desarrollos linealmente (muy limpio)
 - Pero **elimina la historia** de ramas utilizadas e integradas para el desarrollo



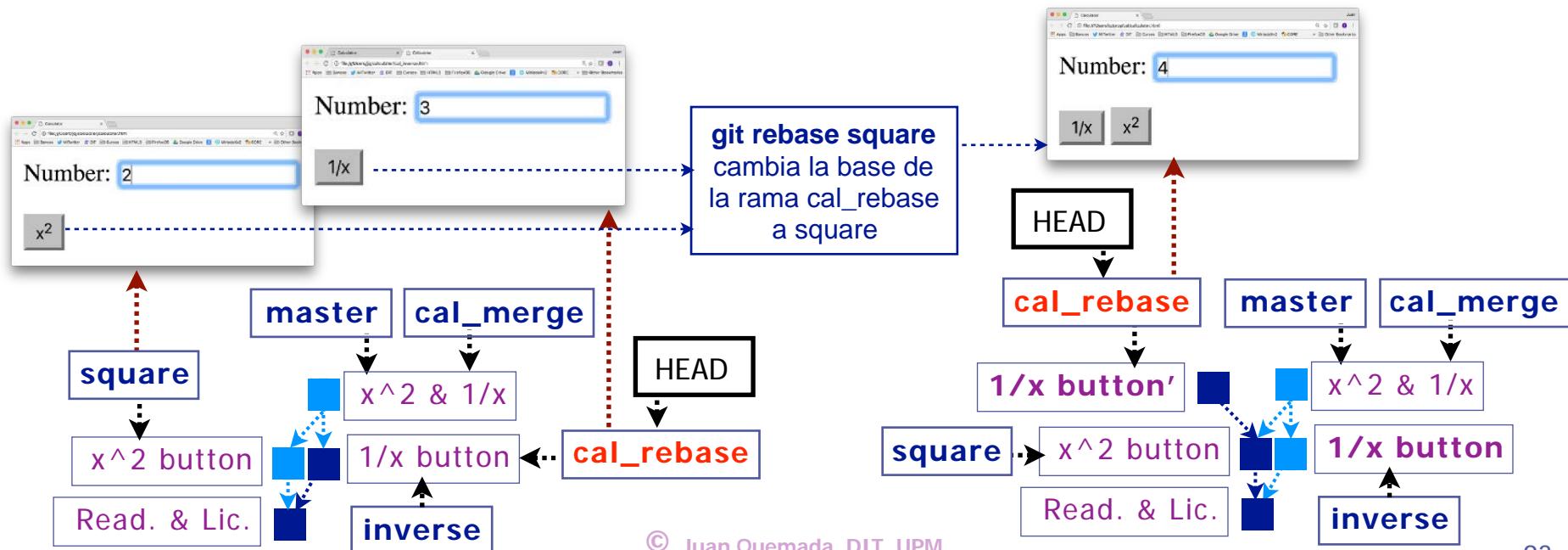
Cambiar la base de la rama cal_rebase

◆ Una rama puede cambiar de base con **git rebase**

- **git rebase ..** arranca un bucle que integra un commit cada vuelta
 - La primera iteración integra la base con el primer commit de la rama
 - La segunda iteración integra el primer commit ya integrado con el segundo commit de la rama
 - y así hasta finalizar la rama
- En cada iteración del bucle la integración es similar a la realizada con merge

◆ La rama **cal_rebase** cambia la base “**Read. & Lic.**” por “**x^2 button**”

- El cambio de base integra los botones de forma similar a merge



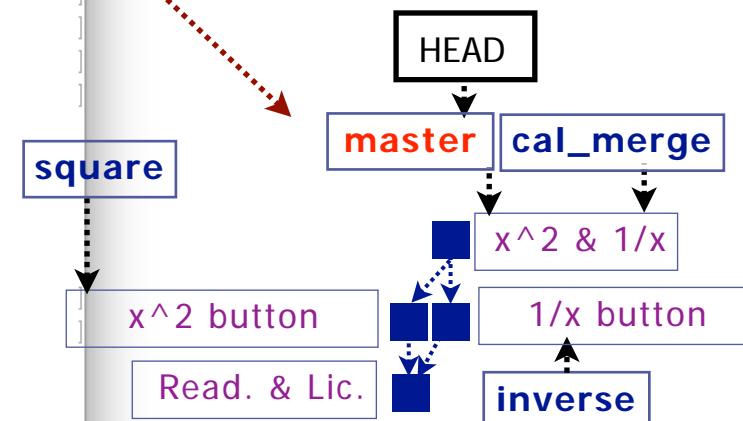
Mostrar ramas

```
venus:cal jq$ git branch -v
cal_merge 1898ac7 Integrate x^2 & 1/x
inverse e868dc4 1/x button
* master 1898ac7 Integrate x^2 & 1/x
square b0e63ad x^2 button
venus:cal jq$ git checkout -b cal_rebase inverse
Switched to a new branch 'cal_rebase'
venus:cal jq$ git rebase -q square
Failed to merge in the changes.
Patch failed at 0001 1/x button
The copy of the patch that failed is found in:
 /Users/jq/proy/cal/.git/rebase-apply/patch
```

When you have resolved this problem, run "git rebase --continue".
 If you prefer to skip this patch, run "git rebase --skip" instead.
 To check out the original branch and stop rebasing, run "git rebase --abort".

```
venus:cal jq$ git status -s
AA calculator.html
venus:cal jq$ # Fix conflicts with editor
venus:cal jq$ # -> in calculator.html
venus:cal jq$ git add .
venus:cal jq$ git rebase --continue
venus:cal jq$ git branch -v
cal_merge 1898ac7 Integrate x^2 & 1/x
* cal_rebase 795c2da 1/x button
inverse e868dc4 1/x button
master 1898ac7 Integrate x^2 & 1/x
square b0e63ad x^2 button
venus:cal jq$ git log --oneline --graph
* 795c2da 1/x button
* b0e63ad x^2 button
* 1096247 Readme & License
venus:cal jq$
```

git branch muestra 4 ramas y marca master como activa.



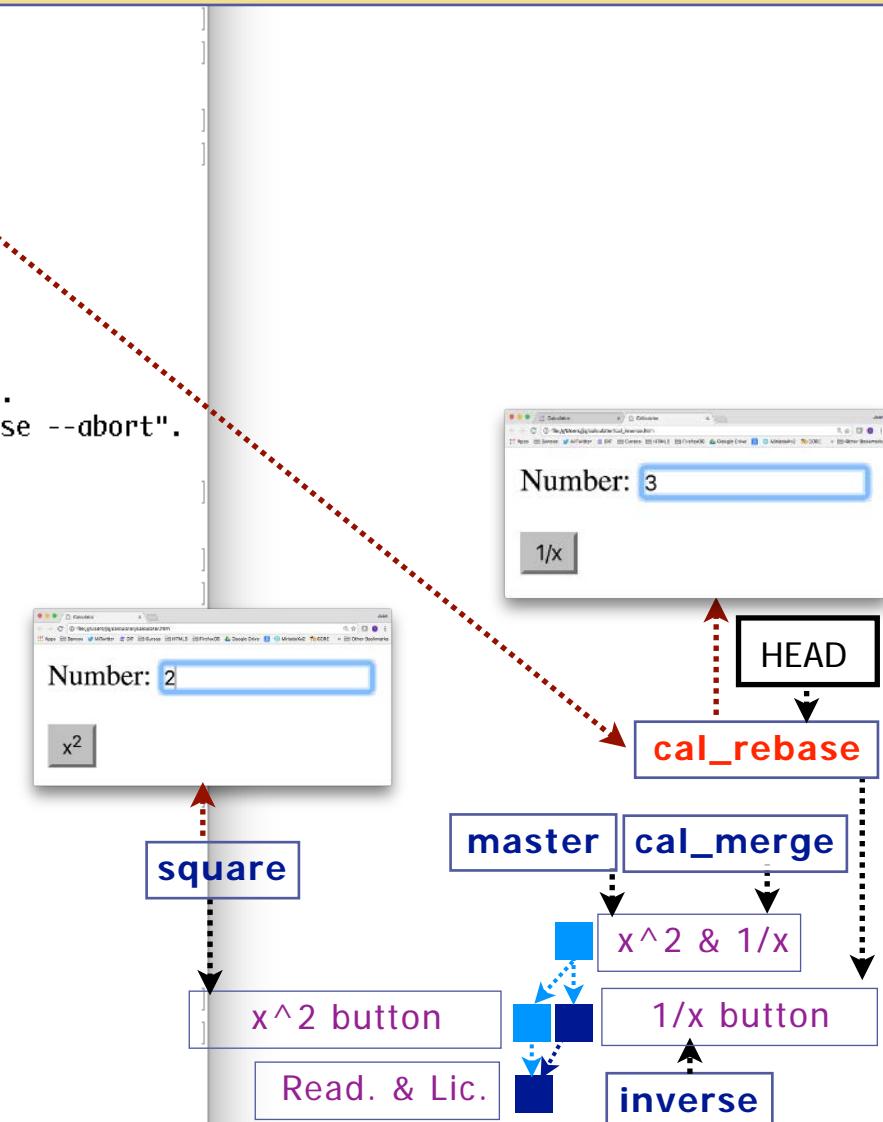
```
[venus:cal jq$ git branch -v
  cal_merge 1898ac7 Integrate x^2 & 1/x
  inverse   e868dc4 1/x button
* master    1898ac7 Integrate x^2 & 1/x
  square    b0e63ad x^2 button
[venus:cal jq$ git checkout -b cal_rebase inverse
Switched to a new branch 'cal_rebase'
[venus:cal jq$ git rebase -q square
Failed to merge in the changes.
Patch failed at 0001 1/x button
The copy of the patch that failed is found in:
 /Users/jq/proj/cal/.git/rebase-apply/patch

When you have resolved this problem, run "git
If you prefer to skip this patch, run "git rebase --skip"
To check out the original branch and stop rebasing, run "git rebase --abort"

[venus:cal jq$ git status -s
AA calculator.html
[venus:cal jq$ git add .
[venus:cal jq$ git rebase --continue
[venus:cal jq$ git branch -v
  cal_merge 1898ac7 Integrate x^2 & 1/x
* cal_rebase 795c2da 1/x button
  inverse   e868dc4 1/x button
  master    1898ac7 Integrate x^2 & 1/x
  square    b0e63ad x^2 button
[venus:cal jq$ git log --oneline --graph
* 795c2da 1/x button
* b0e63ad x^2 button
* 1096247 Readme & License
[venus:cal jq$ ]
```

Crear y restaurar cal_rebase

Crea la rama **cal_rebase** en inverse y realiza un checkout a **cal_rebase**.



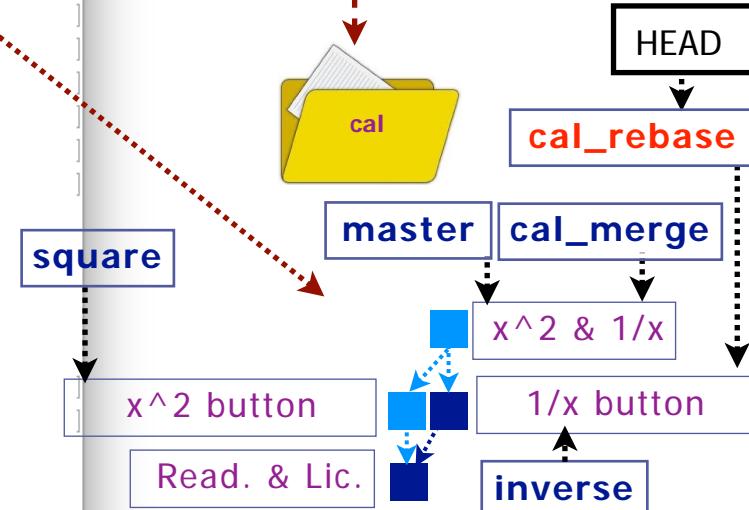
Comenzar cambio de base

```
venus:cal jq: cal_merge  
inverse e868dc4 1/x button  
* master 1898ac7 Integrate x^2 & 1/x  
square b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git checkout -b cal_rebase inverse  
Switched to a new branch 'cal_rebase'  
venus:cal jq$  
venus:cal jq$ git rebase -q square  
Failed to merge in the changes.  
Patch failed at 0001 1/x button  
The copy of the patch that failed is found in:  
/Users/jq/proy/cal/.git/rebase-apply/patch  
  
When you have resolved this problem, run "git rebase --continue".  
If you prefer to skip this patch, run "git rebase --skip" instead.  
To check out the original branch and stop rebasing, run "git rebase --abort".  
  
venus:cal jq$ git status -s  
AA calculator.html  
venus:cal jq$  
venus:cal jq$ # Fix conflicts with editor  
venus:cal jq$ # -> in calculator.html  
venus:cal jq$  
venus:cal jq$ git add .  
venus:cal jq$ git rebase --continue  
venus:cal jq$  
  
venus:cal jq$ git branch -v  
  cal_merge 1898ac7 Integrate x^2  
* cal_rebase 795c2da 1/x button  
  inverse e868dc4 1/x button  
  master 1898ac7 Integrate x^2  
  square b0e63ad x^2 button  
venus:cal jq$  
venus:cal jq$ git log --oneline --abbrev-commit  
* 795c2da 1/x button  
* b0e63ad x^2 button  
* 1096247 Readme & License  
venus:cal jq$
```

git rebase -q square inicia el cambio de base de la rama **cal_rebase**, del commit **"Readme & License"** al commit **"x^2 button"**.
-q (quiet) limita estadísticas.

```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
===== HEAD  
function inverse() {  
    var num = document.getElementById("n1");  
    num.value = 1/num.value;  
=====  
function square() {  
    var num = document.getElementById("n1");  
    num.value = num.value * num.value;  
===== master  
}  
</script>  
</head>  
<body>  
    Number:  
        <input type="text" id="n1"><p>  
  
===== HEAD  
<button onclick="inverse()"> 1/x </button>  
=====  
        <button onclick="square()"> x<sup>2</sup> </button>  
===== master  
</body>  
</html>
```

OJO! Hay **conflicto** en el fichero **calculator.htm** y el cambio de base se interrumpe para integración manual con el editor. Git da instrucciones sobre como continuar.



venus:cal jq\$

venus:cal jq\$ git branch -v

```
cal_merge 1898ac7 Integrate x^2 & 1/x
inverse e868dc4 1/x button
* master 1898ac7 Integrate x^2 & 1/x
square b0e63ad x^2 button
```

venus:cal jq\$

venus:cal jq\$ git checkout -b cal_rebase inverse

Switched to a new branch 'cal_rebase'

venus:cal jq\$

venus:cal jq\$ git rebase -q square

Failed to merge in the changes.

Patch failed at 0001 1/x button

The copy of the patch that failed is found in:
 /Users/jq/proy/cal/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
 If you prefer to skip this patch, run "git rebase --skip" instead.
 To check out the original branch and stop rebasing, run "git rebase --ab

venus:cal jq\$ git status -s

AA calculator.html

venus:cal jq\$

venus:cal jq\$ # Fix conflicts with editor

venus:cal jq\$ # -> in calculator.html

venus:cal jq\$

venus:cal jq\$ git add .

venus:cal jq\$ git rebase --continue

venus:cal jq\$

venus:cal jq\$ git branch -v

```
cal_merge 1898ac7 Integrate x^2 & 1/x
* cal_rebase 795c2da 1/x button
inverse e868dc4 1/x button
master 1898ac7 Integrate x^2 & 1/x
square b0e63ad x^2 button
```

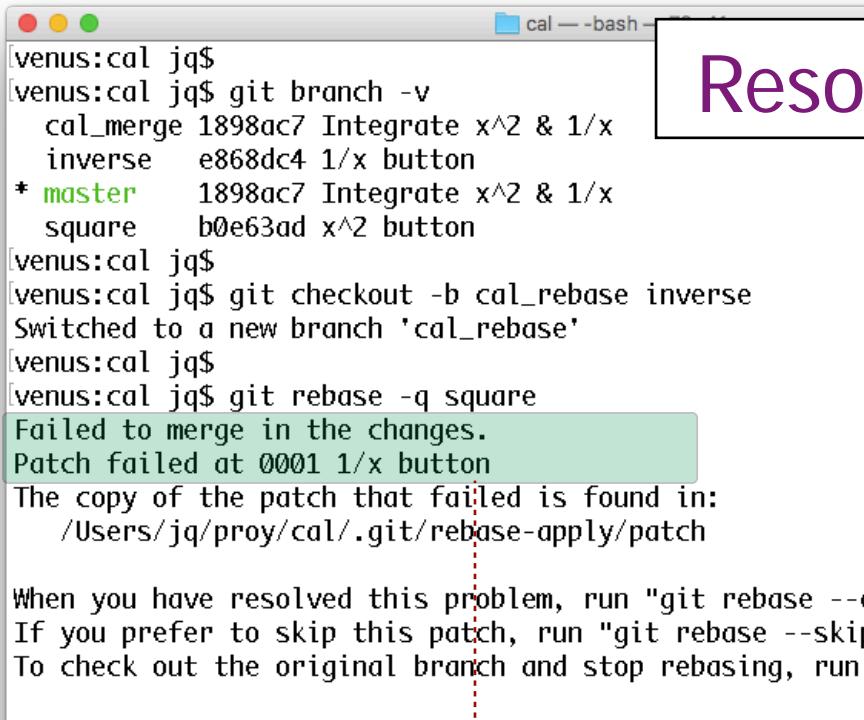
venus:cal jq\$

venus:cal jq\$ git log --oneline --graph

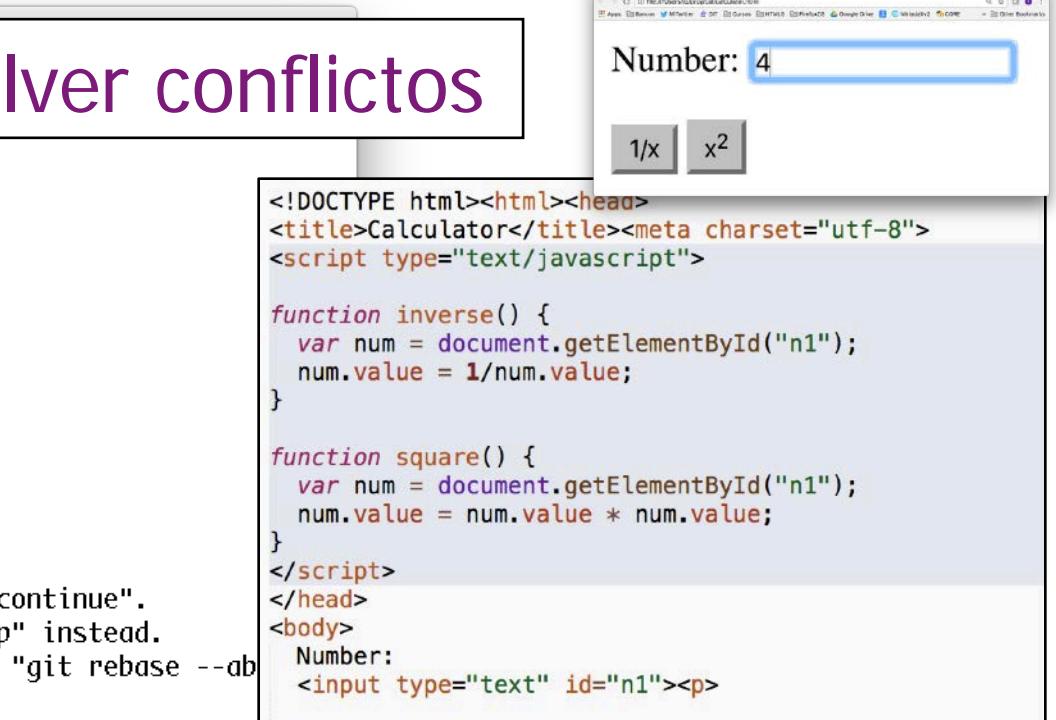
```
* 795c2da 1/x button
* b0e63ad x^2 button
* 1096247 Readme & License
```

venus:cal jq\$

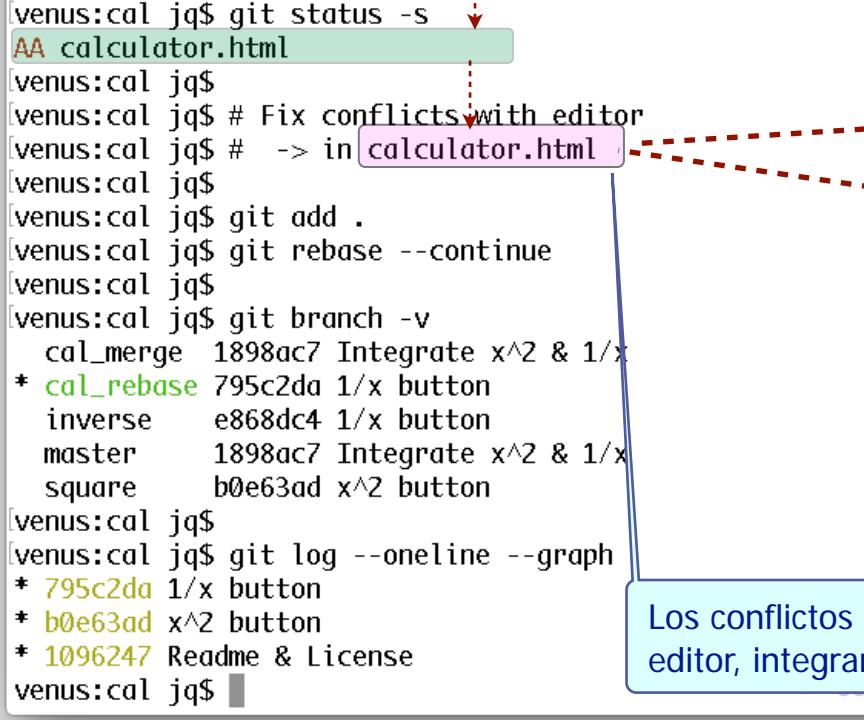
Resolver conflictos



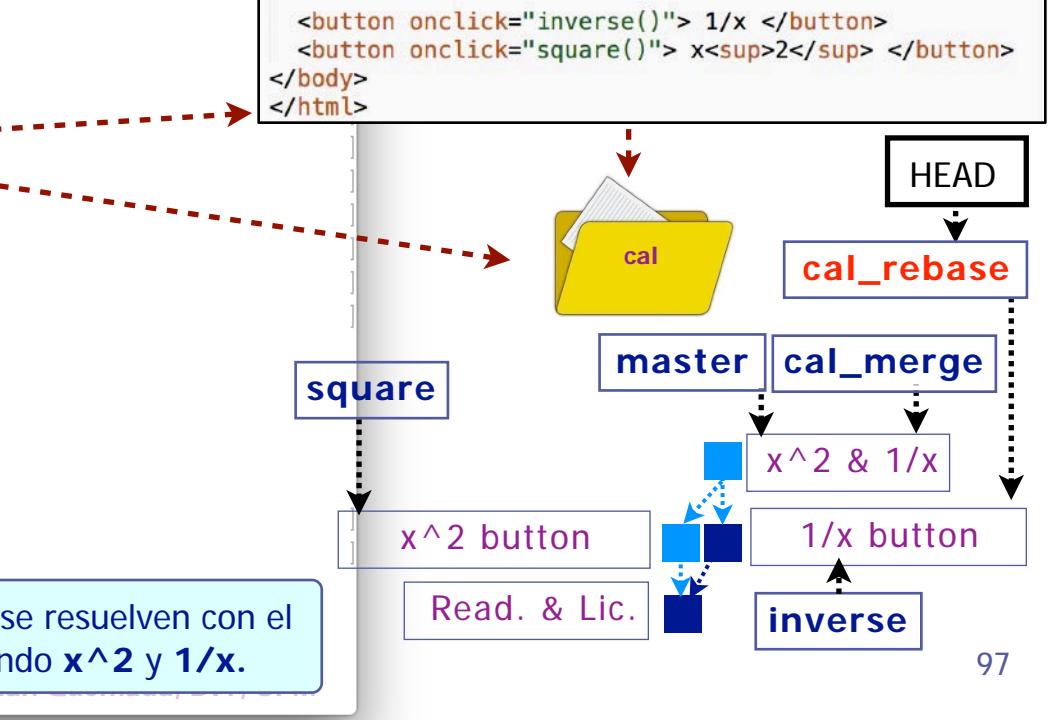
The terminal shows a git rebase conflict for the file 'calculator.html'. A specific commit in the 'cal_rebase' branch is failing to merge because it contains a conflict with the 'square' branch. The conflict is located in the 'calculator.html' file, specifically in the 'inverse()' function where it tries to set 'num.value' to 1/num.value.



A screenshot of a web browser displaying a simple calculator application. It has two buttons: '1/x' and 'x²'. The URL bar shows 'Number: 4'. The page content includes a script that handles these operations by reading values from input fields and setting them back to the document.



The terminal shows the user fixing the conflict in the 'calculator.html' file. After committing the fix, they run 'git rebase --continue' to resolve the conflict. The final state of the branches is shown with the 'cal_rebase' branch now containing the fix and the 'inverse' commit.



A diagram illustrating the Git repository structure. At the top is 'HEAD' pointing to a yellow folder labeled 'cal'. Below it is a blue box labeled 'cal_rebase'. A dashed arrow points from the 'cal_rebase' box to a blue box labeled 'square'. From 'square', dashed arrows point to two blue boxes: 'x^2 & 1/x' and '1/x button'. From 'x^2 & 1/x', dashed arrows point to two more blue boxes: 'Read. & Lic.' and 'inverse'. The 'inverse' box has an upward-pointing arrow pointing to the '1/x button' box.

Los conflictos se resuelven con el editor, integrando x^2 y $1/x$.

Finalizar rebase, mostrar ramas y grafo

```
venus:cal jq$ git branch -v
cal_merge 1898ac7 Integrate
inverse e868dc4 1/x button
* master 1898ac7 Integrate x^2 & 1/x
square b0e63ad x^2 button
venus:cal jq$ git checkout -b cal_rebase
Switched to a new branch 'cal_rebase'
venus:cal jq$
venus:cal jq$ git rebase -q square
Failed to merge in the changes.
Patch failed at 0001 1/x button
The copy of the patch that failed is found in:
 /Users/jq/proy/cal/.git/rebase-apply/patch
```

git add . añade las modificaciones al índice.
git rebase --continue finaliza la adaptación del primer commit a la nueva base y finaliza el bucle porque solo hay 1 commit.

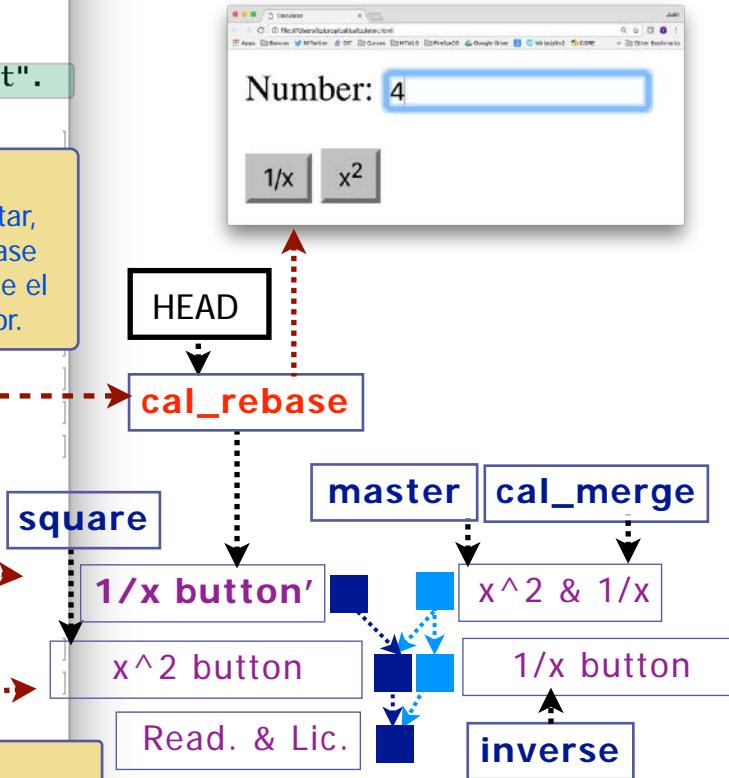
When you have resolved this problem, run "git rebase --continue".

If you prefer to skip this patch, run "git rebase --skip" instead.

To check out the original branch and stop rebasing, run "git rebase --abort".

```
venus:cal jq$ git status -s
AA calculator.html
venus:cal jq$ git add .
venus:cal jq$ git rebase --continue
venus:cal jq$ git branch -v
cal_merge 1898ac7 Integrate x^2 & 1/x
* cal_rebase 795c2da 1/x button
inverse e868dc4 1/x button
master 1898ac7 Integrate x^2 & 1/x
square b0e63ad x^2 button
venus:cal jq$ git log --oneline --graph
* 795c2da 1/x button
* b0e63ad x^2 button
* 1096247 Readme & License
venus:cal jq$
```

git branch muestra 5 ramas y **cal_rebase** activa. Hacemos notar, que el commit generado por rebase tiene el mismo mensaje/título que el original, pero distinto identificador.



Conflictos vistos con diff

git diff muestra así los conflictos resueltos, después de integrarlos con un editor y antes de cerrar el commit.

```
venus:cal jq$ git diff
diff --cc calculator.html
index fc74320,839b37e..0000000
--- a/calculator.html
+++ b/calculator.html
@@@ -2,9 -2,9 +2,15 @@@
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">
```

git diff muestra así los conflictos **antes** de resolverlos.

```
+<<<<< HEAD
+function inverse() {
+  var num = document.getElementById("n1");
+  num.value = 1/num.value;
+=====
+ function square() {
+  var num = document.getElementById("n1");
+  num.value = num.value * num.value;
+>>>>> master
}
</script>
</head>
@@@ -12,6 -12,6 +18,10 @@@
Number:
<input type="text" id="n1"><p>
```

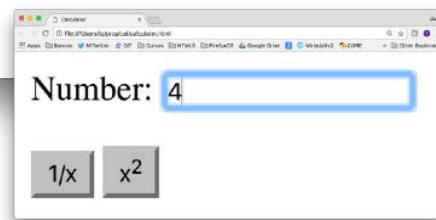
Resolver conflictos con un editor.

```
+<<<<< HEAD
+ <button onclick="inverse()> 1/x </button>
+=====
+ <button onclick="square()> x2 </button>
+>>>>> master
</body>
</html>
venus:cal jq$
```

```
venus:cal jq$ git diff
diff --cc calculator.html
index fc74320,839b37e..0000000
--- a/calculator.html
+++ b/calculator.html
@@@ -2,16 -2,16 +2,22 @@@
<title>Calculator</title><meta charset='utf-8'>
<script type="text/javascript">

+function inverse() {
+  var num = document.getElementById("n1");
+  num.value = 1/num.value;
+}
+=====
+ function square() {
+  var num = document.getElementById("n1");
+  num.value = num.value * num.value;
+ }
</script>
</head>
<body>
Number:
<input type="text" id="n1"><p>
```

```
+ <button onclick="inverse()> 1/x </button>
+ <button onclick="square()> x2 </button>
</body>
</html>
venus:cal jq$
```



Actualizar repositorio jquemada/cal en GitHub

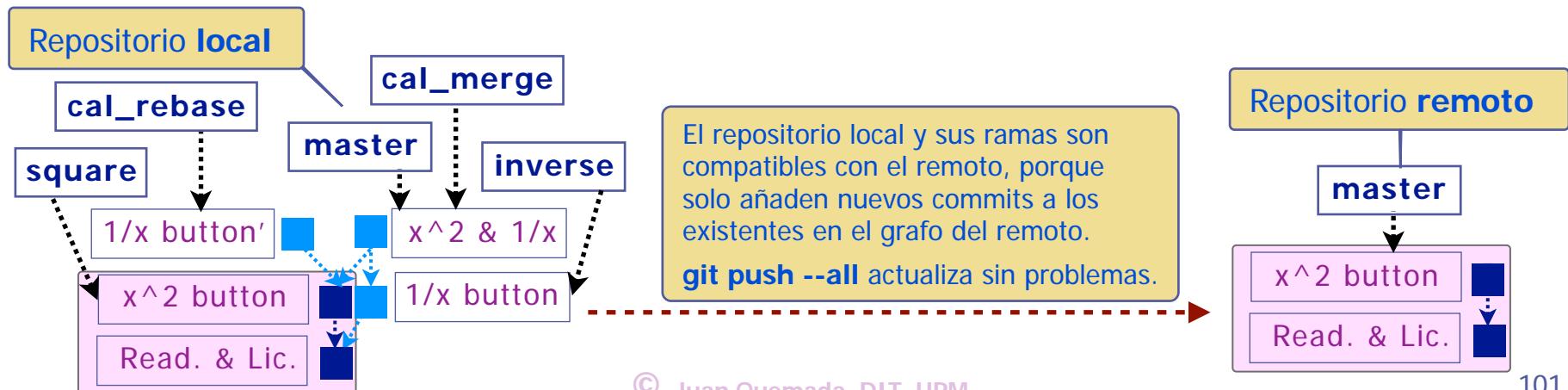
Actualizar un repositorio en GitHub con push

◆ git push ...

- Actualiza el repositorio remoto con los nuevos commits de una rama local
 - `git push https://github.com/jquemada/cal master`
 - actualiza los nuevos commits de la rama master en el repositorio GitHub `jquemada/cal`
 - `git push https://github.com/jquemada/cal_2com master`
 - actualiza los nuevos commits de la rama master en el repositorio GitHub `jquemada/cal_2com`

◆ git push ... necesita 2 condiciones para finalizar con éxito

- Se debe tener credenciales de acceso al repositorio remoto
 - Por ejemplo, un repositorio en una cuenta u organización del usuario que lo actualiza
- La actualización de commits debe ser compatible con la rama actualizada en el remoto
 - Solo debe **añadir nuevos commits al final de la rama remota** o actualizar un **repositorio vacío**
 - **Peligroso!** La opción `-f` permite actualizar una rama incompatible, pero se pierden commits

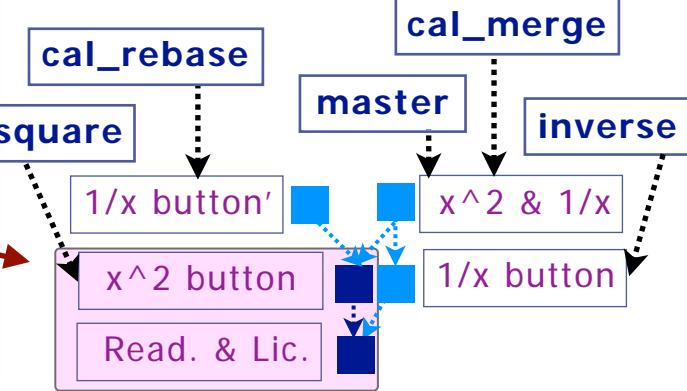


Historia del repositorio local

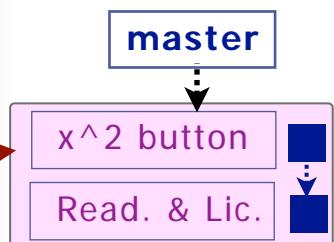


```
venus:cal jq$  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph --all  
* 795c2da 1/x button  
| * 1898ac7 Integrate x^2 & 1/x  
| |\_ ←  
| |\_ /  
|\_ /  
* | b0e63ad x^2 button  
| * e868dc4 1/x button  
|\_ /  
* 1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git push -q --all https://github.com/jquemada/cal  
venus:cal jq$
```

git log --oneline --graph --all muestra el grafo de commits del repositorio local.



GitHub muestra que el repositorio remoto jquemada/cal tiene solo los 2 commits iniciales de la rama master antes de sincronizarlo con el local.

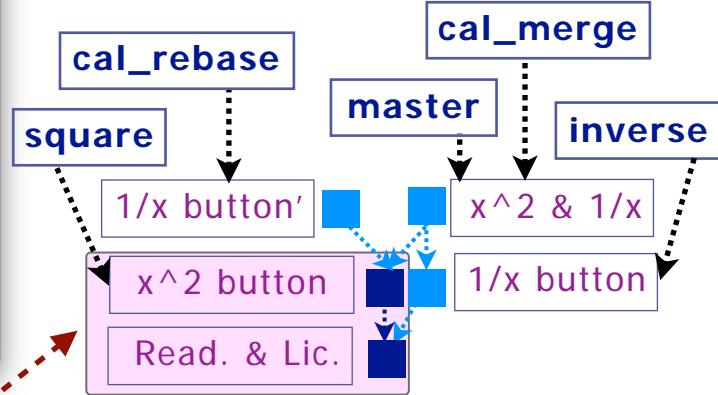


Historia del repositorio local



```
venus:cal jq$  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph --all  
* 795c2da 1/x button  
| * 1898ac7 Integrate x^2 & 1/x  
| |\n| |\n|\n* b0e63ad x^2 button  
| * e868dc4 1/x button  
|\n* 1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git push -q --all https://github.com/jquemada/cal  
venus:cal jq$
```

All sincronizar todas las ramas con push (opcion `--all`) el repositorio remoto `jquemada/cal` actualiza todas las ramas del local.



This repository Juan

<https://github.com/jquemada/cal>

Pull requests Issues Gist

jquemada / cal

No description, website, or topics provided.

New Add topics

4 commits 5 branches 0 releases 1 contributor MIT

Your recently pushed branches:

cal_rebase (2 minutes ago)

Compare & pull request

Crear repositorio jquemada/cal_branches

Crear jquemada/cal_branches

The screenshot illustrates the steps to import a repository from GitHub:

- Left Panel (Toolbar):** Shows a sidebar with options: "New repository", "Import repository" (highlighted in pink), "New gist", and "New organization".
- Middle Panel (Import Step):** A modal window titled "Importar un repositorio a GitHub" at <https://github.com/new/import>. It contains:
 - Import your project to GitHub:** "Import all the files, including the revision history, from another version control system."
 - URL de identificación del repositorio a importar:** "Your old repository's clone URL" field containing <https://github.com/jquemada/cal>.
 - Your new repository details:**
 - Owner:** jquemada
 - Name:** cal_branches (highlighted in pink)
 - Note:** "Your new repository will be public. In order to make this repository private, you'll need to upgrade your account."
- Right Panel (Repository Overview):** The imported repository "jquemada / cal_branches" is shown with the following details:
 - Code:** 4 commits
 - Issues:** 0
 - Pull requests:** 0
 - Description:** No description, website, or topics provided.
 - Topics:** New Add topics
 - Branch:** master
 - Files:**
 - jquemada Integrate x^2 & 1/x
 - LICENSE
 - README.md
 - calculator.html
 - README.md Content:** cal
 - Description:** Educational Git project. Creates a simple calc

Nuevo repositorio creado con URL:
https://github.com/jquemada/cal_branches

Ejercicio opcional

- ◆ Crear una nueva rama **square** en el commit "x^2 button"
 - Crearla con: **git branch square <id_del_commit>**
 - Mostrar el grafo de commits y sus identificadores con **git log --oneline --graph --all**
- ◆ Crear la rama **cube** en el commit "x^3 button" y restaurarla con
 - **git checkout -b cube <id_del_commit>**
 - Mostrar las ramas existentes con **git branch -v**
- ◆ Cambiar la base de la rama **cube** a la rama **square** (último commit)
 - **git rebase square** integra los botones x^2 y x^3 en una única calculadora
 - La integración dará conflictos en el fichero calculator.html en las líneas con diferencias
 - Mostrar los conflictos tanto con **git diff**, como con **git status**
- ◆ Resolver los conflictos (editor) para que los 2 botones funcionen bien
 - Dejar el código de ambas calculadoras, tanto en HTML, como en JavaScript
 - Una vez que la calculadora funcione correctamente, registrar el cambio en el índice con **git add .**
- ◆ Finalizar el cambio de base con **git rebase --continue**
 - Mostrar la historia de cube y el grafo con **git log --online --graph [--all]**
- ◆ Subir todas las ramas al repositorio en GitHub con
 - **git push --all https://github.com/<su_cuenta>/cal_2com**
 - Inspeccionar con el navegador el repositorio actualizado y sus ramas



Git y GitHub

Crear commit inicial en GitHub, clonar y actualizar:
new_repository, .gitignore, clone, remote y push

Juan Quemada, DIT - UPM



Funciones principales de GitHub

- ◆ La función principal de **GitHub** es **compartir** repositorios con terceros
- ◆ Las operaciones principales de un **usuario registrado** son
 - **Crear repositorio remoto** inicial nuevo para albergar un proyecto
 - Utilizando el botón: **New repository**
 - Puede añadir los ficheros **README.md**, **LICENSE** o **.gitignore** al repositorio creado
 - **Copia** un repositorio albergado en GitHub a otra cuenta (para contribuir)
 - Utilizando el botón: **Fork**
 - **Importa** un repositorio identificado por su URL a GitHub, incluso en otro formato
 - Utilizando el botón: **Import repository**
 - Equivale a crear repositorio vacío (**New_repository**) e importar en él otro repositorio con un URL
 - **Crear una organización** para albergar múltiples proyectos relacionados
 - Utilizando el botón: **New organisation**
 - Organización de asignatura CORE: <https://github.com/CORE-UPM>
 - Y otras operaciones de compartición, gestión y mantenimiento
- ◆ Permite operaciones Git de **sincronización de repositorios**
 - **push** (subir rama), **clone** (clonar repositorio), **fetch** (traer rama), **pull** ..

Commit inicial en GitHub



◆ New repository permite crear el repositorio inicial de un proyecto

- Con un commit inicial típico que puede incluir hasta 3 ficheros
 - **README.md** fichero con descripción del proyecto o software
 - **LICENSE** fichero con la licencia de distribución del software
 - **.gitignore** fichero donde se indica que ficheros debe ignorar Git al generar versiones

◆ Es la forma habitual de arrancar un proyecto alojado en GitHub

- Se crea el repositorio inicial con 1 commit en: https://github.com/jquemada/cal_1
 - Este repositorio se clona a continuación en uno local con el comando: `git clone ...`
 - Se desarrolla un segundo commit en el repositorio local y se sube a GitHub con: `git push ...`
- Esto ilustra el desarrollo 2 commits iniciales similares a los de jquemada/cal
 - Siguiendo el procedimiento habitual de creación de un proyecto

Crear nuevos objetos en GitHub

The screenshot shows a web browser window with several tabs open. The active tab is GitHub, Inc. [US] at https://github.com/jquemada. The sidebar on the left has a 'New repository' button highlighted with a yellow box. Other options in the sidebar include 'Import repository', 'New gist', and 'New organization'. The main content area shows an 'Overview' section with statistics: Repositories 27, Stars 6, and Followers 111. Below this is a 'Popular repositories' section featuring 'quiz-2015' and 'planet_1'.

Crear commit inicial en GitHub

The screenshot shows the GitHub 'Create a New Repository' page. A context menu is open over the top navigation bar, showing options: 'New repository', 'Import repository', 'New gist', and 'New organization'. A dashed red arrow points from the 'New repository' option to the main form area. The form has fields for 'Owner' (set to 'jquemada'), 'Repository name' ('cal_I'), 'Description (optional)' ('Educational Git project. Creates a simple calculator in HTML and JavaScript in short time.'), 'Visibility' (radio button selected for 'Public'), 'Initialize this repository with a README' (checkbox checked), 'Add .gitignore' (dropdown set to 'None'), and 'Add a license' (dropdown set to 'MIT License'). A large green 'Create repository' button is at the bottom. Several callout boxes with blue borders and yellow backgrounds highlight specific steps: 'Crear nuevo repositorio GitHub' (over the context menu), 'Nombre del repositorio' (over the repository name field), 'Description del repositorio' (over the description field), 'Incluir README.md' (over the 'Initialize this repository with a README' checkbox), 'Incluir .gitignore (no incluido)' (over the '.gitignore' dropdown), 'Seleccionar licencia' (over the 'license' dropdown), and 'Crear el repositorio' (over the 'Create repository' button). A timeline at the bottom shows contributions from February to September of the previous year.

Crear nuevo repositorio GitHub

Nombre del repositorio

Description del repositorio

Incluir README.md

Incluir .gitignore (no incluido)

Seleccionar licencia

Crear el repositorio

Create a new repository

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

Owner

Repository name

jquemada / cal_I

Great repository names are short and descriptive

Description (optional)

Educational Git project. Creates a simple calculator in HTML and JavaScript in short time.

Public

Anyone can see this repository. You choose who can fork it.

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.

Add .gitignore: None

Add a license: MIT License

Create repository

137 contributions in the last year

Contribution settings

Feb Mar Apr May Jun Jul Aug Sep

Mon

110

© Juan Quemada, DIT, UPM

Repositorio inicial en GitHub



The screenshot shows a GitHub repository named 'jquemada/cal_I'. The repository description is 'Educational Git project. Creates a simple calculator in HTML and JavaS...'. It has 0 issues, 0 pull requests, and 0 projects. A 'Code' tab is selected. A 'New' button is visible. The commit history shows one commit from 'jquemada' on Nov 18, 2016, titled 'Initial commit'. Below the commit, there are three files listed: 'LICENSE', 'README.md', and 'README.md' (repeated). A yellow callout box points to the first 'README.md' entry with the text 'Este commit incluye 2 ficheros: - LICENSE - README.md'.

jquemada / cal_I

Educational Git project. Creates a simple calculator in HTML and JavaS...

New Add topics

Branch: master New

1 commit

Commits on Nov 18, 2016

Initial commit
jquemada committed 17 seconds ago

LICENSE

README.md

README.md

cal_I

Educational Git project. Creates a simple calculator in HTML and JavaS...

El primer paso al arrancar un proyecto suele ser crear un repositorio inicial en GitHub para el proyecto.

El URL del repositorio inicial creado es:

https://github.com/jquemada/cal_I

A este repositorio se subirán (con push) los desarrollos realizados en el repositorio local clonado.

The screenshot shows the 'LICENSE' file content in the GitHub repository. The file contains the MIT License text. A yellow callout box highlights the first two lines: '1 MIT License' and '2'. The rest of the file content is shown below.

jquemada / cal_I

Branch: master cal_I / LICENSE

jquemada Initial commit

1 contributor

22 lines (17 sloc) | 1.04 KB

1 MIT License

2

3 Copyright (c) 2017 Juan Quemada

4

5 Permission is hereby granted, free of charge, to any person obtainin...

6 of this software and associated documentation files (the "Software").

7 in the Software without restriction, including without limitation

8 to use, copy, modify, merge, publish, distribute, sublicense,

9 copies of the Software, and to permit persons to whom the Software

10 furnished to do so, subject to the following conditions:

11

12 The above copyright notice and this permission notice shall be

13 included in all copies or substantial portions of the Software.

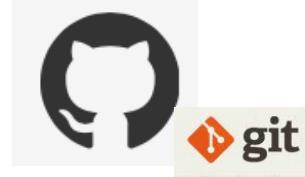
.gitignore

◆ **git ls-file --other --ignored --exclude-standard**

- lista todos los ficheros de este proyecto ignorados por Git

```
# .gitignore es un fichero que informa a Git de los ficheros que no debe gestionar.  
# - git status no los presentará como ficheros untracked.  
# - git add . no los añadira al índice (staging area).  
  
# .gitignore se crea en el directorio o subdirs de trabajo y afecta todo el arbol asociado.  
  
# Su contenido: líneas con patrones de nombres.  
# - Puede usarse los comodines * y ?  
# - Patrones terminados en / indican directorios  
# - Un patron que empiece con ! indica negación  
# - Se ignoran líneas en blanco y que comiencen con #  
# - [abc] indica cualquiera de los caracteres entre corchetes  
# - [a-z] indica cualquier carácter ASCII (rango desde a hasta z)  
  
# Ejemplo  
private.txt      # excluir los ficheros con nombre "private.txt"  
*.class          # excluir los ficheros acabados en ".class"  
*[oa]             # excluir ficheros acabados en ".o" y ".a"  
!lib.a           # no excluir el fichero "lib.a"  
*~               # excluir ficheros acabados en "~"  
testing/          # excluir directorio "testing"
```

Clonar un repositorio remoto



Referenciar un repositorio remoto

◆ Un repositorio remoto se identifica en Internet con un URL

- Por ejemplo
 - <https://github.com/jquemada/cal>
 - https://github.com/jquemada/cal_I
- **git remote ...** permite asociar un nombre, denominado **remote**, a un URL
 - El nuevo nombre puede utilizarse, en vez del URL, en los comandos para identificar el repositorio

◆ **git remote [-v]**

- Muestra los repositorios remotos definidos en un repositorio (-v modo verboso)

◆ **git remote add ...**

- Define un nuevo remote en el repositorio asociado a un URL
 - **git remote add cal_2com https://github.com/jquemada/cal_2com**
 - asocia el nombre **cal_2com** con el URL https://github.com/jquemada/cal_2com

◆ **git remote remove ...**

- Borra la definición de un remote en el repositorio
 - **git remote remove cal_2com**
 - Borra el nombre **cal_2com** del repositorio y ya no podrá ser utilizado en comandos

Clonar un repositorio remoto

◆ Un proyecto publicado en un servidor en Internet

- Puede copiarse al ordenador local con **git clone ...** para desarrollar en sus ramas

◆ **git clone ...**

- Crea un nuevo repositorio local, donde copia la rama master del rep. remoto clonado
 - Además asocia el nombre de remote **origin** al repositorio remoto origen de la clonación
- Ejemplos de uso
 - **git clone https://github.com/jquemada/cal_I**
 - el repositorio remoto en el directorio local **cal_I** (mismo nombre que el repositorio)
 - **origin** referencia el repositorio clonado: https://github.com/jquemada/cal_I
 - **git clone https://github.com/jquemada/cal_I mi_cal**
 - Copia el repositorio remoto en el directorio local **mi_cal**
 - **origin** referencia el repositorio clonado: https://github.com/jquemada/cal_I

Clonar repositorio remoto

git clone https://github.com/jquemada/cal_I clona el repositorio identificado por el URL <https://github.com/jquemada/cal>, en el directorio (repositorio) local **cal_I**.

```
venus:proy jq$ git clone https://github.com/jquemada/cal_I
Cloning into 'cal_I'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
venus:proy jq$ ls
cal    cal_I
venus:proy jq$ cd cal_I
venus:cal_I jq$ git log --oneline
5410ac7 Initial commit
venus:cal_I jq$ ls
LICENSE      README.md
venus:cal_I jq$
```

Muestra el repositorio clonado **cal_I**, además del ya existente **cal**.

cal_I es un repositorio local diferente de **cal**.

Inspeccionar repositorio clonado

```
venus:proy jq$  
venus:proy jq$ git clone https://github.com/jquemada/cal_I  
Cloning into 'cal_I'...  
remote: Counting objects: 4, done.  
remote: Compressing objects: 100% (4/4), done.  
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (4/4), done.  
Checking connectivity... done.  
venus:proy jq$  
venus:proy jq$ ls  
cal      cal_I  
venus:proy jq$ cd cal_I  
venus:cal_I jq$  
venus:cal_I jq$ git log --oneline  
5410ac7 Initial commit ←  
venus:cal_I jq$  
venus:cal_I jq$ ls  
LICENSE      README.md  
venus:cal_I jq$
```

`cd cal_I` entra en el directorio del nuevo repositorio clonado `cal_I`, para que los comandos git se asocien a este repositorio.

`git log --oneline` muestra que el repositorio contiene el commit generado automáticamente en GitHub con el mensaje/título "Initial commit".

master

Initial commit

`ls` muestra que el directorio de trabajo del repositorio contiene los ficheros `README.md` y `LICENSE`.

Generar nuevo commit en el repositorio clonado

Directorio de trabajo limpio

```
venus:cal_I jq$ git status -s
venus:cal_I jq$ # Edit/create calculator.html
venus:cal_I jq$ ls
LICENSE          README.md      calculator.html
venus:cal_I jq$ git status -s
?? calculator.html
venus:cal_I jq$ git add calculator.html
venus:cal_I jq$ git commit -q -m "x^2 button"
venus:cal_I jq$ git log --oneline
9cba3e6 x^2 button
5410ac7 Initial commit
venus:cal_I jq$
```

git status -s muestra un directorio de trabajo limpio, sin cambios respecto al último commit.



Añadir fichero calculadora.html

```
venus:cal_I jq$ git status -s
venus:cal_I jq$ # Edit/create calculator.html
venus:cal_I jq$ ls
LICENSE          README.md      calculator.html
venus:cal_I jq$ git status -s
?? calculator.html
venus:cal_I jq$ git add calculator.html
venus:cal_I jq$ git commit -q -m "x^2 button"
venus:cal_I jq$ git log --oneline
9cba3e6 x^2 button
5410ac7 Initial commit
venus:cal_I jq$
```

Se añade el fichero **calculator.html** al directorio de trabajo con un editor, copiando (cp), ...

calculator.html
calculator.html no
está registrada.

```
<!DOCTYPE html><html>
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">

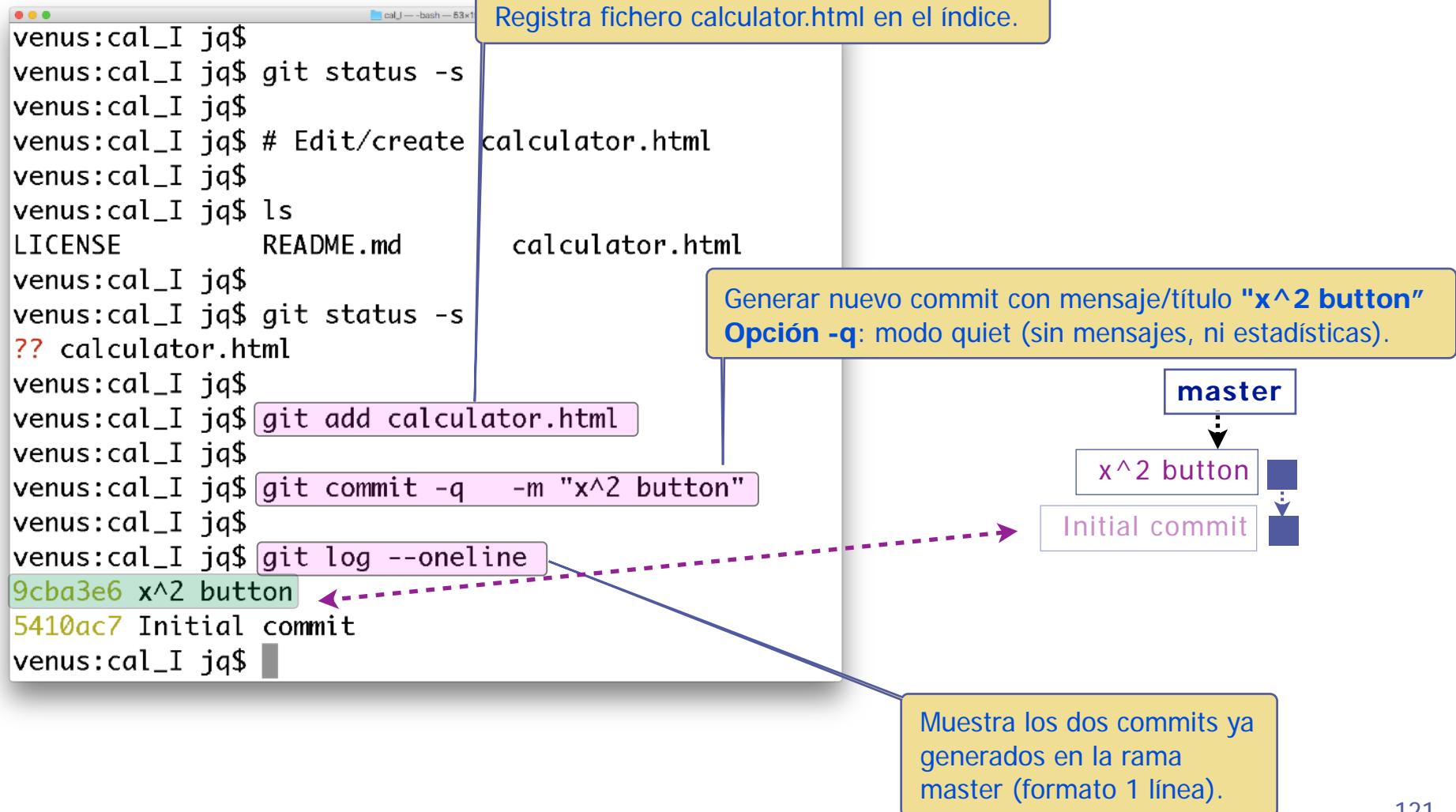
function square() {
    var num = document.getElementById("n1");
    num.value = num.value * num.value;
}
</script>
</head>
<body>
Number:
<input type="text" id="n1"><p>

<button onclick="square()"> x<sup>2</sup> </button>
</body>
</html>
```

calculator.html tiene la misma calculadora con el botón x^2 , que se generó en el repositorio cal.



Crear nuevo commit



Actualizar el repositorio origin

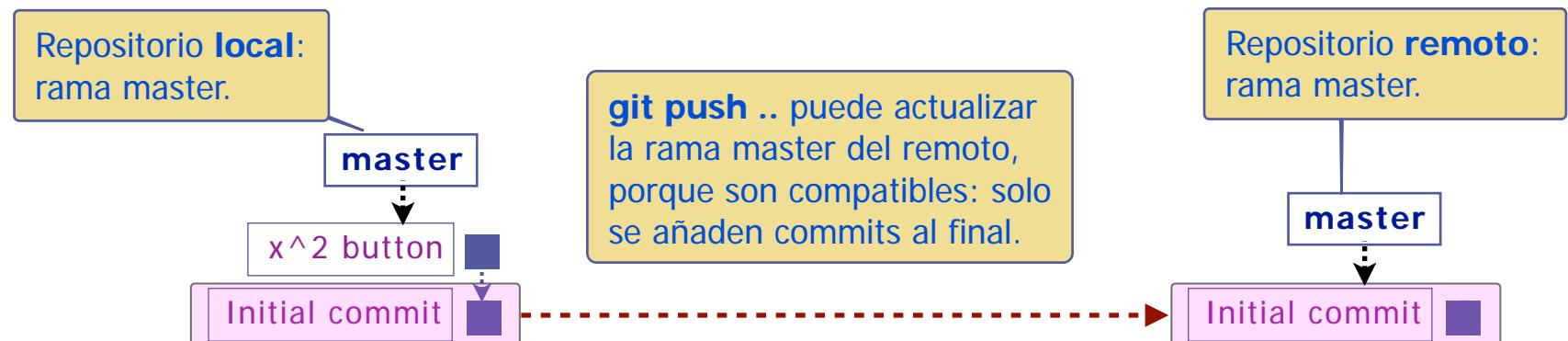
Actualizar un repositorio en GitHub con push

◆ git push ...

- Actualiza el repositorio remoto con los nuevos commits de una rama local
 - `git push origin master`
 - actualiza master en el repositorio remoto **origin** (<https://github.com/jquemada/cal>)
 - `git push https://github.com/jquemada/cal master`
 - Es equivalente al comando: `git push origin master`

◆ git push ... necesita 2 condiciones para finalizar con éxito

- Se debe tener credenciales de acceso al repositorio remoto
 - Por ejemplo, un repositorio en una cuenta u organización del usuario que lo actualiza
- La actualización de commits debe ser compatible con la rama actualizada en el remoto
 - Solo debe **añadir nuevos commits al final de la rama remota** o actualizar un **repositorio vacío**
 - **Peligroso!** La opción `-f` permite actualizar una rama incompatible, pero se pierden commits



Mostrar historia y remote origin

```
venus:cal_I jq$ git log --oneline
9cba3e6 x^2 button
5410ac7 Initial commit
venus:cal_I jq$ git remote
origin
venus:cal_I jq$ git remote -v
origin https://github.com/jquemada/cal_I (fetch)
origin https://github.com/jquemada/cal_I (push)
venus:cal_I jq$ git push -q origin master
venus:cal_I jq$
```

Muestra los dos commits del repositorio local.

Rama master del repositorio **local**.

master

x^2 button

Initial commit

git remote muestra el remote **origin** definido por **git clone ..**

Mostrar URL asociado a origin

```
venus:cal_I jq$  
venus:cal_I jq$ git log --oneline  
9cba3e6 x^2 button  
5410ac7 Initial commit  
venus:cal_I jq$  
venus:cal_I jq$ git remote  
origin  
venus:cal_I jq$ git remote -v  
origin https://github.com/jquemada/cal_I (fetch)  
origin https://github.com/jquemada/cal_I (push)  
venus:cal_I jq$  
venus:cal_I jq$ git push -q origin master  
venus:cal_I jq$
```

git remote -v (verboso) muestra también el **URL** asociado al remote **origin** definido por git clone ...:
https://github.com/jquemada/cal_I

Rama master del repositorio **remoto**.

master

Initial commit

Actualizar rama master remota

```
venus:cal_I jq$  
venus:cal_I jq$ git log --oneline  
9cba3e6 x^2 button  
5410ac7 Initial commit  
venus:cal_I jq$  
venus:cal_I jq$ git remote  
origin  
venus:cal_I jq$ git remote -v  
origin https://github.com/jquemada/cal_I (fetch)  
origin https://github.com/jquemada/cal_I (push)  
venus:cal_I jq$  
venus:cal_I jq$ git push -q origin master  
venus:cal_I jq$
```

Rama master del repositorio local.

master

x^2 button

Initial commit

Sube la rama **master** del repositorio local a **origin**:

https://github.com/CORE-UPM/cal_I

Rama master del repositorio remoto.

master

Initial commit

La actualización solo añade nuevos commits al final de la rama master ya guardada en el repositorio remoto **origin**: es **compatible!**

This screenshot shows a GitHub repository page for 'jquemada/cal_I'. The repository has 2 commits, 1 branch, 0 releases, 1 contributor, and is licensed under MIT. The URL is https://github.com/jquemada/cal_I.

Ejercicio opcional

◆ Crear en su cuenta en GitHub un nuevo repositorio **cal_I**

- Utilizar el botón **New_repository** para crearlo
 - Crear el repositorio con los 3 ficheros:
 - **.gitignore** seleccionando ExtJS
 - **LICENSE** con la licencia MIT
 - **README.md** con un breve texto metido en la descripción

◆ Clonar el repositorio **<su_cuenta>/cal_I** en un repositorio local con

- **git clone https://github.com/<su_cuenta>/cal_I**
 - **git clone ..** copia el repositorio remoto en uno local, con directorio de trabajo, de nombre **cal_I**
 - Mostrar con **ls** el directorio creado y con **ls -a cal_2com** su contenido, incluyendo **.git**

◆ Entrar en el directorio de trabajo **cal_I** del repositorio clonado

- **cd cal_I** para que los directorios de trabajo sean el mismo
 - Mostrar estado del directorio con **git status ..**, historia y grafo con **git log ..**, contenido con **ls ..**, etc.

◆ Modificar el título/mensaje del commit con

- **git commit --amend -m ".gitignore, Readme, License"**
 - Mostrar con **git log ..** como sigue habiendo un commit, pero con id y mensaje diferente

◆ Subir **master** al repositorio **<su_cuenta>/cal_I** en GitHub con

- **git push -f origin master**
 - La opción **-f** actualiza aunque hay incompatibilidades de ramas, como es el caso aquí



Git y GitHub

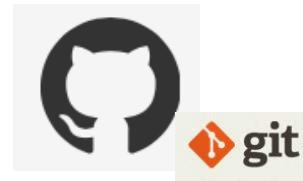
Clonar con Fork en GitHub: fork_repository, clone y push

Juan Quemada, DIT - UPM

Funciones principales de GitHub



- ◆ La función principal de **GitHub** es **compartir** repositorios con terceros
- ◆ Las operaciones principales de un **usuario registrado** son
 - **Crear repositorio remoto** inicial nuevo para albergar un proyecto
 - Utilizando el botón: **New repository**
 - **Copia** un repositorio albergado en GitHub a otra cuenta (para contribuir)
 - Utilizando el botón: **Fork**
 - **Importa** un repositorio identificado por su URL a GitHub, incluso en otro formato
 - Utilizando el botón: **Import repository**
 - Equivale a crear repositorio vacío (New_repository) e importar en él otro repositorio con un URL
 - **Crear una organización** para albergar múltiples proyectos relacionados
 - Utilizando el botón: **New organisation**
 - Organización de asignatura CORE: <https://github.com/CORE-UPM>
 - Y otras operaciones de compartición, gestión y mantenimiento
- ◆ Permite operaciones Git de **sincronización de repositorios**
 - **push** (subir rama), **clone** (clonar repositorio), **fetch** (traer rama), **pull** ..



Fork de un repositorio en GitHub

◆ Un equipo suele tener un **repositorio remoto de referencia**

- Donde se consolidan las últimas versiones de un proyecto
 - Es la referencia que utilizan todos los miembros del equipo y sus usuarios
- El repositorio remoto suele administrarse por unas pocas personas

◆ Otros desarrolladores lo copian con **Fork** a su cuenta en GitHub

- Los desarrollos se realizan en la copia en GitHub clonada con Fork
 - GitHub permite contribuir los desarrollos al repositorio de referencia con **pull request**

◆ El proyecto **jquemada/cal** se clono con **Fork** a **CORE-UPM/cal**

- Cuando el repositorio tenía solo los 2 primeros commits de la calculadora
 - **CORE-UPM/cal** no ha evolucionado y sigue teniendo solo los **2 commits iniciales**
 - En cambio, **jquemada/cal** ha evolucionando desde que se clonó con nuevas ramas y commits

◆ **CORE-UPM/cal** debe ser clonado en local para realizar desarrollos

- Los desarrollos se añaden como nuevos commits en el repositorio local
 - Una vez finalizados se pueden subir a **CORE-UPM/cal** y contribuir a **jquemada/cal** con **pull_request**



Fork: clonar un proyecto de GitHub

Repository cal del usuario jquemada en:
<https://github.com/jquemada/cal>

Copia el repositorio a otra cuenta u organización del usuario en GitHub pulsando el botón Fork.

Copia de jquemada/cal en CORE-UPM:
<https://github.com/CORE-UPM/cal>

En el momento del Fork el repositorio jquemada/cal tiene estos 2 commits. A partir de este momento cada repositorio evolucionara por separado a partir de estos 2 commits.

This repository Search Pull requests Issues Gist

Where should we fork this repository?

@ging @CORE-UPM...

Can't find what you're looking for?
You already have a fork of this repository:
https://github.com/jquemada/cal_2com

ts 0 Wiki Pulse Graphs Settings

Unwatch 1 Star 0 Fork 0

No description, website, or topics

New Add topics

2 commits

Branch: master New pull request

jquemada x^2 button

Commits on Feb 12, 2017

x^2 button jquemada committed 3 hours ago

Commits on Feb 9, 2017

Readme & License jquemada committed 3 days ago

This repository Search Pull requests Issues Gist

CORE-UPM / cal forked from jquemada/cal

Code Pull requests 0 No description, website, or topics New Add topics

2 commits

1 branch 0 releases

Unwatch 6 Star 1 Settings

131

Clonar un repositorio remoto

Pasar a directorio de proyectos

Estando en el directorio **cal** (repositorio) pasamos al directorio **proy**, donde están los repositorios de proyectos para que el nuevo repositorio se clone en el directorio adecuado.

```
venus:cal jq$  
venus:cal jq$ cd ..  
venus:proy jq$  
venus:proy jq$ git clone https://github.com/CORE-UPM/cal cal_2  
Cloning into 'cal_2'...  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
Checking connectivity... done.  
venus:proy jq$  
venus:proy jq$ cd cal_2  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$
```

Clonar repositorio CORE-UPM/cal de GitHub

```
venus:cal jq$  
venus:cal jq$ cd ..  
venus:proy jq$  
venus:proy jq$ git clone https://github.com/CORE-UPM/cal cal_2  
Cloning into 'cal_2'...  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
Checking connectivity... done.  
venus:proy jq$  
venus:proy jq$ cd cal_2  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE           README.md      calculator.html  
venus:cal_2 jq$
```

git clone https://github.com/jquemada/cal cal_2
copia el repositorio remoto en GitHub, identificado por el URL
<https://github.com/jquemada/cal>, en el directorio
(repositorio) local **cal_2**.

Entrar en el nuevo repositorio clonado

```
venus:cal jq$  
venus:cal jq$ cd ..  
venus:proy jq$  
venus:proy jq$ git clone https://github.com/CORE-UPM/cal cal_2  
Cloning into 'cal_2'...  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
Checking connectivity... done.  
venus:proy jq$  
venus:proy jq$ cd cal_2  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$
```

Se cambia el directorio de trabajo del terminal de comandos para que coincida con el directorio de trabajo Git.

Ver historia del repositorio clonado

```
venus:cal jq$  
venus:cal jq$ cd ..  
venus:proy jq$  
venus:proy jq$ git clone https://github.com/CORE-UPM/cal cal_2  
Cloning into 'cal_2'...  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
Checking connectivity... done.  
venus:proy jq$  
venus:proy jq$ cd cal_2  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$
```

Muestra los 2 commits del repositorio remoto clonado.

```
graph TD; master["master"] --> x2button["x^2 button"]; master --> readme["Readme & License"]; x2button --> x2buttonFile["x^2 button"]; readme --> readmeFile["Readme & License"];
```

Mostrar estado del directorio de trabajo

```
venus:cal jq$  
venus:cal jq$ cd ..  
venus:proy jq$  
venus:proy jq$ git clone https://github.com/CORE-UPM/cal cal_2  
Cloning into 'cal_2'...  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
Checking connectivity... done.  
venus:proy jq$  
venus:proy jq$ cd cal_2  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$
```

Muestra un directorio de trabajo limpio,
sin cambios respecto al último commit.

master

x^2 button

Readme & License

Mostrar ficheros del directorio de trabajo

```
venus:cal jq$  
venus:cal jq$ cd ..  
venus:proy jq$  
venus:proy jq$ git clone https://github.com/CORE-UPM/cal cal_2  
Cloning into 'cal_2'...  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
Checking connectivity... done.  
venus:proy jq$  
venus:proy jq$ cd cal_2  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$
```

git clone .. ha restaurado los 3 ficheros del último commit en el directorio de trabajo.

master

x^2 button

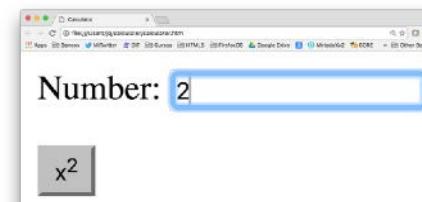
Readme & License

Generar nuevo commit en el repositorio clonado

Ficheros del directorio de trabajo

```
venus:cal_2 jq$ ls
venus:cal_2 jq$ LICENSE README.md calculator.html
venus:cal_2 jq$ # Edit calculator.html
venus:cal_2 jq$ git status -s
M calculator.html
venus:cal_2 jq$ git diff
diff --git a/calculator.html b/calculator.html
index 839b37e..db62028 100644
--- a/calculator.html
+++ b/calculator.html
@@ -8,6 +8,9 @@ function square() {
}
</script>
</head>
+
+<h3>My Calculator</h3>
+
<body>
Number:
<input type="text" id="n1"><p>
venus:cal_2 jq$ git commit -a -q -m "Añadir título"
venus:cal_2 jq$ git log --oneline
5c70f9c Añadir título
b0e63ad x^2 button
1096247 Readme & License
venus:cal_2 jq$
```

ls muestra los ficheros del directorio de trabajo.



```
<!DOCTYPE html><html><head>
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">

function square() {
    var num = document.getElementById("n1");
    num.value = num.value * num.value;
}
</script>
</head>
<body>
Number:
<input type="text" id="n1"><p>

<button onclick="square()"> x<sup>2</sup> </button>
</body>
</html>
```

calculator.html tiene la calculadora generada en el commit anterior.

Añadir título a calculadora.html

```
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ # Edit calculator.html.  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
 M calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git diff  
diff --git a/calculator.html b/calculator.html  
index 839b37e..db62028 100644  
--- a/calculator.html  
+++ b/calculator.html  
@@ -8,6 +8,9 @@ function square() {  
}  
 </script>  
 </head>  
+  
+<h3>My Calculator</h3>  
+  
<body>  
 Number:  
 <input type="text" id="n1"><p>  
venus:cal_2 jq$  
venus:cal_2 jq$ git commit -a -q -m "Añadir título"  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$
```

calculator.html

Se edita el fichero y se añade el título HTML (**<h3>My Calculator</h3>**) a la calculadora (resaltado en verde).

My Calculator

Number: 45

x²

```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
function square() {  
    var num = document.getElementById("n1");  
    num.value = num.value * num.value;  
}  
</script>  
</head>  
  
<h3>My Calculator</h3>  
  
<body>  
 Number:  
 <input type="text" id="n1"><p>  
  
 <button onclick="square()"> x<sup>2</sup> </button>  
</body>  
</html>
```

Inspeccionar estado

```
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md      calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ # Edit calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
M calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git diff  
diff --git a/calculator.html b/calculator.html  
index 839b37e..db62028 100644  
--- a/calculator.html  
+++ b/calculator.html  
@@ -8,6 +8,9 @@ function square() {  
}  
</script>  
</head>  
+  
+<h3>My Calculator</h3>  
+  
<body>  
  Number:  
    <input type="text" id="n1"><p>  
venus:cal_2 jq$  
venus:cal_2 jq$ git commit -a -q -m "Añadir título"  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$
```

calculator.html está en estado modified, no se ha registrado todavía en el índice.

My Calculator

Number: 45

x²

```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
function square() {  
  var num = document.getElementById("n1");  
  num.value = num.value * num.value;  
}  
</script>  
</head>  
  
<h3>My Calculator</h3>  
  
<body>  
  Number:  
    <input type="text" id="n1"><p>  
  
    <button onclick="square()"> x<sup>2</sup> </button>  
</body>  
</html>
```

Inspeccionar diferencias

```
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ # Edit calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
 M calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git diff  
diff --git a/calculator.html b/calculator.html  
index 839b37e..db62028 100644  
--- a/calculator.html  
+++ b/calculator.html  
@@ -8,6 +8,9 @@ function square() {  
}  
 </script>  
 </head>  
+  
+<h3>My Calculator</h3>  
+  
<body>  
 Number:  
 <input type="text" id="n1"><p>  
venus:cal_2 jq$  
venus:cal_2 jq$ git commit -a -q -m "Añadir título"  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$
```

git diff muestra las diferencias en los ficheros **modified** del directorio de trabajo.
Solo se ha modificado **calculator.html**.

My Calculator

Number: 45

x²

```
<!DOCTYPE html><html><head>  
<title>Calculator</title><meta charset="utf-8">  
<script type="text/javascript">  
  
function square() {  
    var num = document.getElementById("n1");  
    num.value = num.value * num.value;  
}  
</script>  
</head>  
  
<h3>My Calculator</h3>  
  
<body>  
 Number:  
 <input type="text" id="n1"><p>  
  
 <button onclick="square()"> x<sup>2</sup> </button>  
</body>  
</html>
```

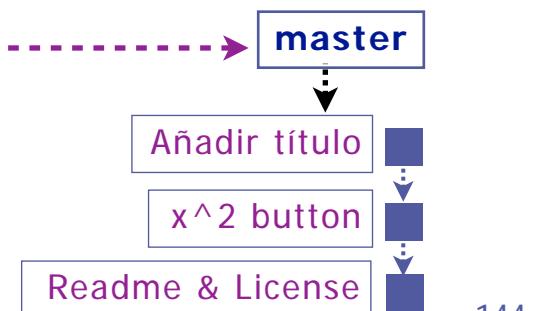
Cerrar commit

```
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ # Edit calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
 M calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git diff  
diff --git a/calculator.html b/calculator.html  
index 839b37e..db62028 100644  
--- a/calculator.html  
+++ b/calculator.html  
@@ -8,6 +8,9 @@ function square() {  
}  
</script>  
</head>  
+  
+<h3>My Calculator</h3>  
+  
<body>  
 Number:  
 <input type="text" id="n1"><p>  
venus:cal_2 jq$  
venus:cal_2 jq$ git commit -a -q -m "Añadir título"  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$
```

Generar nuevo commit con mensaje/título “**Añadir título**”.

Opción -a: añadir al índice todos los cambios antes del commit.

Opción -q: modo quiet (sin mensajes, ni estadísticas).



Mostrar historia de master

```
venus:cal_2 jq$  
venus:cal_2 jq$ ls  
LICENSE README.md calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ # Edit calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git status -s  
 M calculator.html  
venus:cal_2 jq$  
venus:cal_2 jq$ git diff  
diff --git a/calculator.html b/calculator.html  
index 839b37e..db62028 100644  
--- a/calculator.html  
+++ b/calculator.html  
@@ -8,6 +8,9 @@ function square() {  
}  
</script>  
</head>  
+  
+<h3>My Calculator</h3>  
+  
<body>  
 Number:  
 <input type="text" id="n1"><p>  
venus:cal_2 jq$  
venus:cal_2 jq$ git commit -a -q -m "Añadir título"  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$
```

Muestra los tres commits (formato 1 línea).

master

Añadir título

x^2 button

Readme & License

Actualizar el repositorio origin

Mostrar historia

Rama master del repositorio local.

```
venus:cal_2 jq$ git log --oneline
5c70f9c Añadir título
b0e63ad x^2 button
1096247 Readme & License
venus:cal_2 jq$ git remote
origin
venus:cal_2 jq$ git remote -v
origin https://github.com/CORE-UPM/cal (fetch)
origin https://github.com/CORE-UPM/cal (push)
venus:cal_2 jq$ git push -q origin master
venus:cal_2 jq$
```

Muestra los tres commits (formato 1 línea).

master

Añadir título

x^2 button

Readme & License

Mostrar remote origin

```
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git remote  
origin  
venus:cal_2 jq$ git remote -v  
origin https://github.com/CORE-UPM/cal (fetch)  
origin https://github.com/CORE-UPM/cal (push)  
venus:cal_2 jq$  
venus:cal_2 jq$ git push -q origin master  
venus:cal_2 jq$ █
```

git remote muestra el remote origin definido por git clone ..

Mostrar URL asociado a origin

```
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git remote  
origin  
venus:cal_2 jq$ git remote -v  
origin https://github.com/CORE-UPM/cal (fetch)  
origin https://github.com/CORE-UPM/cal (push)  
venus:cal_2 jq$  
venus:cal_2 jq$ git push -q origin master  
venus:cal_2 jq$
```

git remote -v (verboso) muestra también el **URL** asociado al remote **origin** definido por git clone ...
<https://github.com/jquemada/cal>

Rama master del repositorio **remoto**.

master

x^2 button

Readme & License

Sincronizar origin con push

```
venus:cal_2 jq$ git log --oneline
5c70f9c Añadir título
b0e63ad x^2 button
1096247 Readme & License
venus:cal_2 jq$ git remote
origin
venus:cal_2 jq$ git remote -v
origin https://github.com/CORE-UPM/cal (fetch)
origin https://github.com/CORE-UPM/cal (push)
venus:cal_2 jq$ git push -q origin master
venus:cal_2 jq$
```

El remote **origin** es equivalente a utilizar el URL:
<https://github.com/jquemada/cal>

Sube la rama **master** del repositorio local al repositorio **origin**:
<https://github.com/CORE-UPM/cal>

The screenshot shows a GitHub repository page for 'CORE-UPM / cal'. The repository has 3 commits, 1 branch, 0 releases, and 1 contributor. The commits listed are 'Añadir título', 'x^2 button', and 'Readme & License'. The repository description is 'No description, website, or topics provided.'

Rama master del repositorio **local**.

master

Añadir título

x^2 button

Readme & License

Rama master del repositorio **remoto**.

master

x^2 button

Readme & License

La actualización solo añade nuevos commits al final de la rama master ya guardada en el repositorio remoto **origin**: es compatible!

Crear repositorio CORE-UPM/cal_3com

Crear CORE-UPM/cal_3com

The screenshot illustrates the steps to import a repository from GitHub:

- Left Panel (Toolbar):** Shows a sidebar with options: New repository, Import repository, New gist, and New organization. The "Import repository" option is highlighted.
- Middle Panel (Import Step):** A modal window titled "Importar un repositorio a GitHub" is open. It displays the URL <https://github.com/new/import>. Below it, the text "Import your project to GitHub" and "Import all the files, including the revision history, from another version control system" is visible. A yellow callout box highlights the "URL de identificación del repositorio a importar." field, which contains the URL <https://github.com/CORE-UPM/cal>.
- Bottom Panel (New Repository Step):** The "Nombre del nuevo repositorio" field is highlighted with a yellow callout box, containing the text "cal_3com". Below it, a note states: "be public. In order to make this repository private, you'll need to upgrade your account." At the bottom right of this panel are "Cancel" and "Begin import" buttons.
- Right Panel (Repository Details):** A separate GitHub repository page for "CORE-UPM / cal_3com" is shown. The repository has 3 commits and 1 branch. The README.md file contains the text "cal". A note at the top says "No description, website, or topics provided." and there are "New" and "Add topics" buttons.

Ejercicio opcional

- ◆ Copiar con Fork el repositorio **jquemada/cal_branches** de GitHub a su cuenta
 - Inspeccionar ambos repositorios en GitHub (original y copiado) con el navegador Web
 - Comprobar que ambos tienen las mismas ramas con los mismos commits y los mismos identificadores
- ◆ Crear una nueva organización o cuenta que denominamos **<cuenta_2>**
 - Crean una de las dos siguiendo las instrucciones que les da GitHub
- ◆ Copiar con Fork el repositorio **<su_cuenta>/cal_branches** a **<cuenta_2>**
 - Inspeccionar ambos repositorios en GitHub (original y copiado) con el navegador Web
 - Comprobar que también tiene las mismas ramas con los mismos commits y los mismos identificadores



Git y GitHub

Ramas locales, remotas, tracking y refspecs:
branch, checkout, clone, fetch y pull

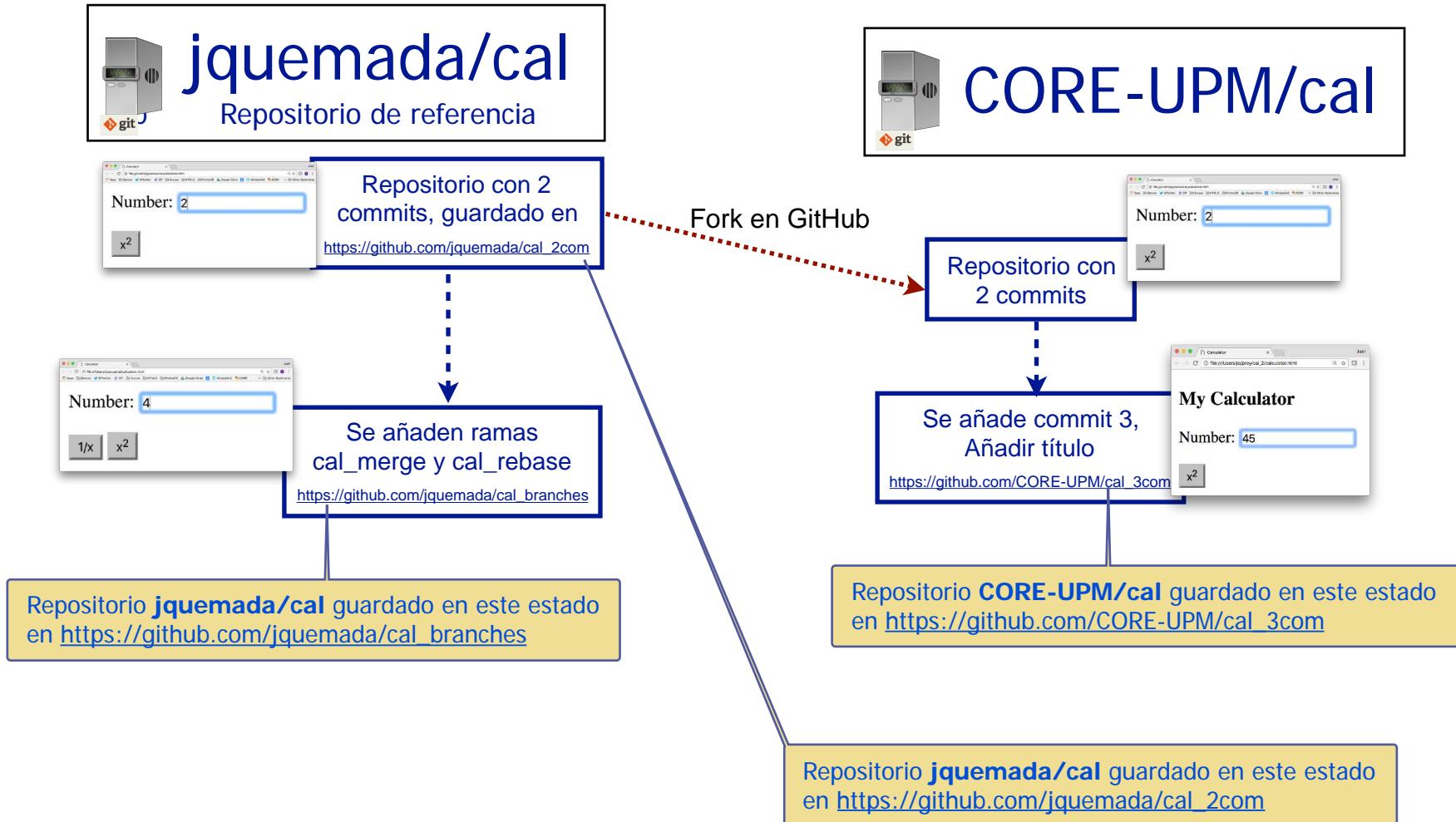
Juan Quemada, DIT - UPM

Rama local, remota y tracking

- ◆ **Rama local:** rama donde se realiza el desarrollo de nuevos commits
 - Las ramas locales almacenan sus commits en el repositorio de commits en el directorio .git
- ◆ **Rama remota:** rama de un **remote** definido en el repositorio local
 - Son copias de las ramas del remote mostradas en el grafo de commits del repositorio local
 - Su estado se actualiza en las operaciones de sincronización: **clone**, **fetch**, **pull** o **push**
 - **remote:** nombre simbólico asociado al URL de un repositorio definido con **git remote ...**
- ◆ **Nombre de rama remota:** va precedido por <remote> o por remotes/<remote>
 - **origin/master** o **remotes/origin/master** identifica la rama **master** del **remote origin**
 - **cal/square** o **remotes/cal/square** identifica la rama **square** del **remote cal**
- ◆ **Rama tracking:** rama local asociada a una remota
 - La rama tracking simplifica las operaciones de sincronización con la remota
 - Una rama de desarrollo local suele ser tracking de la remota equivalente, por ej. **master** de **origin/master**



Repositorios remotos utilizados



Crear, enlazar o listar ramas remotas

◆ git clone ...

- Copiar un repositorio identificado por <URL> a un directorio local:
 - Define el repositorio remoto como repositorio remote origin, copia solo la rama master definiéndola como rama tracking de origin/master y actualiza todas las ramas remotas de origin
 - `git clone https://github.com/jquemada/cal` copia el repositorio remoto al directorio cal
 - `git clone https://github.com/jquemada/cal cal_2` copia el repositorio remoto al directorio cal_2

◆ git branch ...

- Muestra ramas locales, remotas o tracking con opciones **-a** (--all), **-r** (--remote) o **-vv** (--verbose)
 - `git branch -r` muestra solo las **ramas remotas** de un repositorio local
 - `git branch -a` muestra las **ramas locales y remotas** de un repositorio local
 - `git branch -vv` muestra en las ramas **tracking** la rama remota asociada y su estado

◆ git branch ...

- Crea una rama tracking asociada con una rama remota
 - `git branch -b square r1/square` crea rama tracking local **square** y la asocia como tracking a **r1/square**
 - `git branch --track r1/square` transforma la rama local **square**, ya existente, en tracking de **r1/square**
 - `git branch --track s2 r1/s1` transforma la rama local **s2**, ya existente, en tracking de **r1/s1**

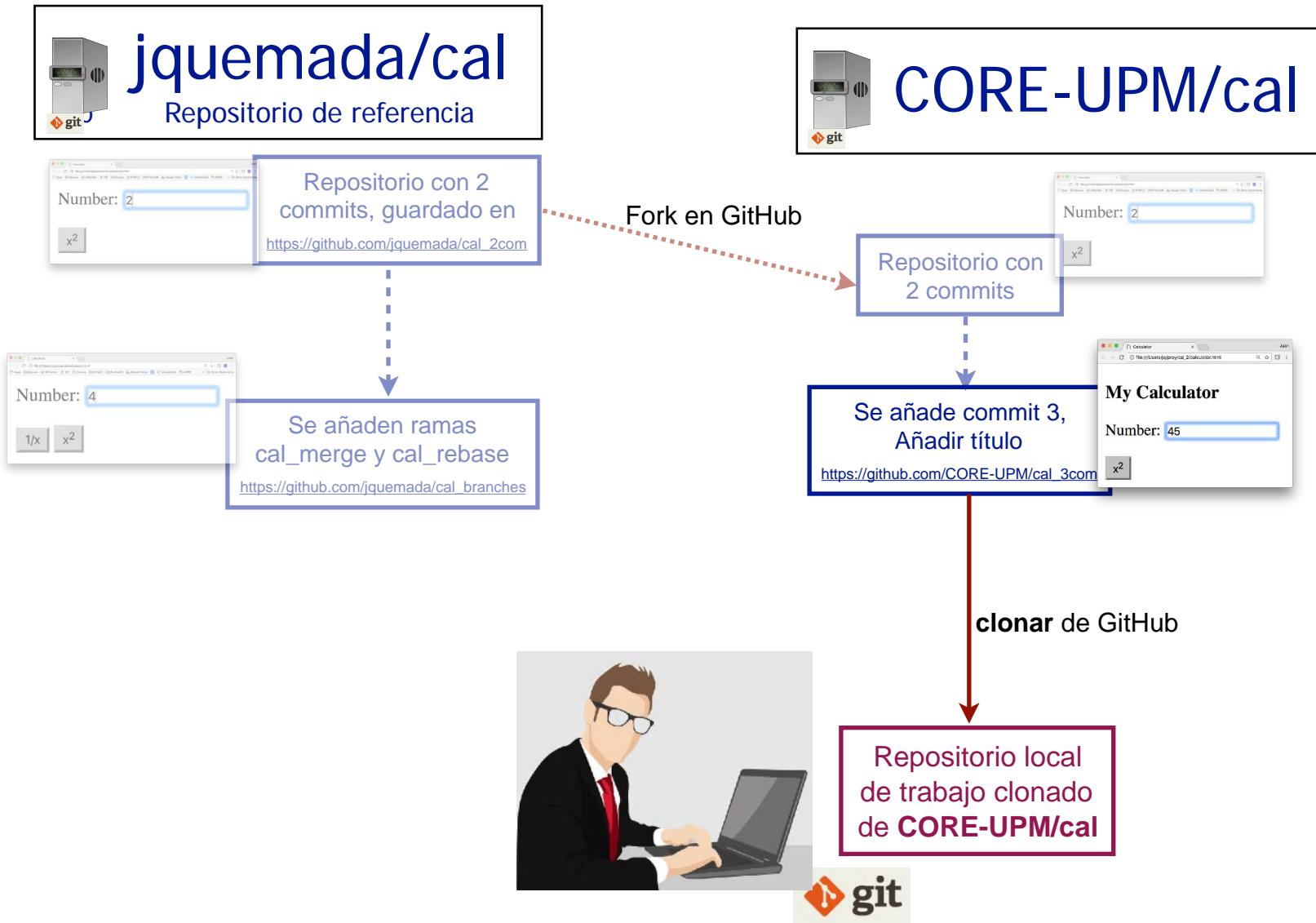
◆ git fetch ...

- Crea si no existen o actualiza las ramas remotas de un remote creado con `git remote`
 - `git fetch cal` crea las **ramas remotas** del remote **cal** o actualiza su estado si existen
 - `git fetch` actualiza el estado de todas las ramas tracking
 - `git fetch --all` crea o actualiza el estado de todas las ramas de todos los remotes definidos
 - `git fetch -p origin` la opción `-p` (--prune) actualiza las ramas de origin eliminando las que ya no existen



Clonar repositorio

Clonar CORE-UPM/cal



Clonar un repositorio remoto

Clonar el repositorio **CORE-UPM/cal** de GitHub. Es un fork de **jquemada/cal** que se realizó cuando tenía los 2 primeros commits y al que se añadió un tercer commit.

```
venus:proy jq$ git clone -q https://github.com/CORE-UPM/cal cal_2
venus:proy jq$ cd cal_2
venus:cal_2 jq$ git branch
* master
venus:cal_2 jq$ git log --oneline
5c70f9c Añadir título
b0e63ad x^2 button
1096247 Readme & License
venus:cal_2 jq$ git remote -v
origin https://github.com/CORE-UPM/cal (fetch)
origin https://github.com/CORE-UPM/cal (push)
venus:cal_2 jq$ git branch -a -vv
* master 5c70f9c [origin/master] Añadir título
  remotes/origin/HEAD -> origin/master
  remotes/origin/master 5c70f9c Añadir título
venus:cal_2 jq$
```

El repositorio local clonado

Clonar el repositorio **CORE-UPM/cal** de GitHub. Es un fork de **jquemada/cal** que se realizó cuando tenía los 2 primeros commits y al que se añadió un tercer commit.

```
venus:proy jq$ git clone -q https://github.com/CORE-UPM/cal cal_2
venus:proy jq$ cd cal_2
venus:cal_2 jq$ git branch
* master
venus:cal_2 jq$ git log --oneline
5c70f9c Añadir título
b0e63ad x^2 button
1096247 Readme & License
venus:cal_2 jq$ git remote -v
origin https://github.com/CORE-UPM/cal (fetch)
origin https://github.com/CORE-UPM/cal (push)
venus:cal_2 jq$ git branch -a -vv
* master      5c70f9c [origin/master] Añadir título
  remotes/origin/HEAD  -> origin/master
  remotes/origin/master 5c70f9c Añadir título
venus:cal_2 jq$
```

El repositorio local clonado solo contiene la rama master

La rama master contiene estos 3 commits

```
graph TD; master[master] --> Añadir_título[Añadir título]; Añadir_título --> x2_button[x^2 button]; x2_button --> Readme_License[Readme & Lic.]
```

El repositorio remoto

```
venus:proy jq$  
[venus:proy jq$ git clone -q      https://github.com/CORE-UPM/cal    cal_2  
[venus:proy jq$ cd cal_2  
venus:cal_2 jq$  
venus:cal_2 jq$ git branch  
* master  
venus:cal_2 jq$  
venus:cal_2 jq$ git log --oneline  
5c70f9c Añadir título  
b0e63ad x^2 button  
1096247 Readme & License  
venus:cal_2 jq$  
venus:cal_2 jq$ git remote -v  
origin https://github.com/CORE-UPM/cal (fetch)  
origin https://github.com/CORE-UPM/cal (push)  
venus:cal_2 jq$  
venus:cal_2 jq$ git branch -a -vv  
* master          5c70f9c [origin/master] Añadir título  
  remotes/origin/HEAD  -> origin/master  
  remotes/origin/master 5c70f9c Añadir título  
venus:cal_2 jq$
```

git remote -v muestra como también se ha definido el remote **origin** (para operaciones fetch y push) asociado al repositorio en GitHub: <https://github.com/CORE-UPM/cal>

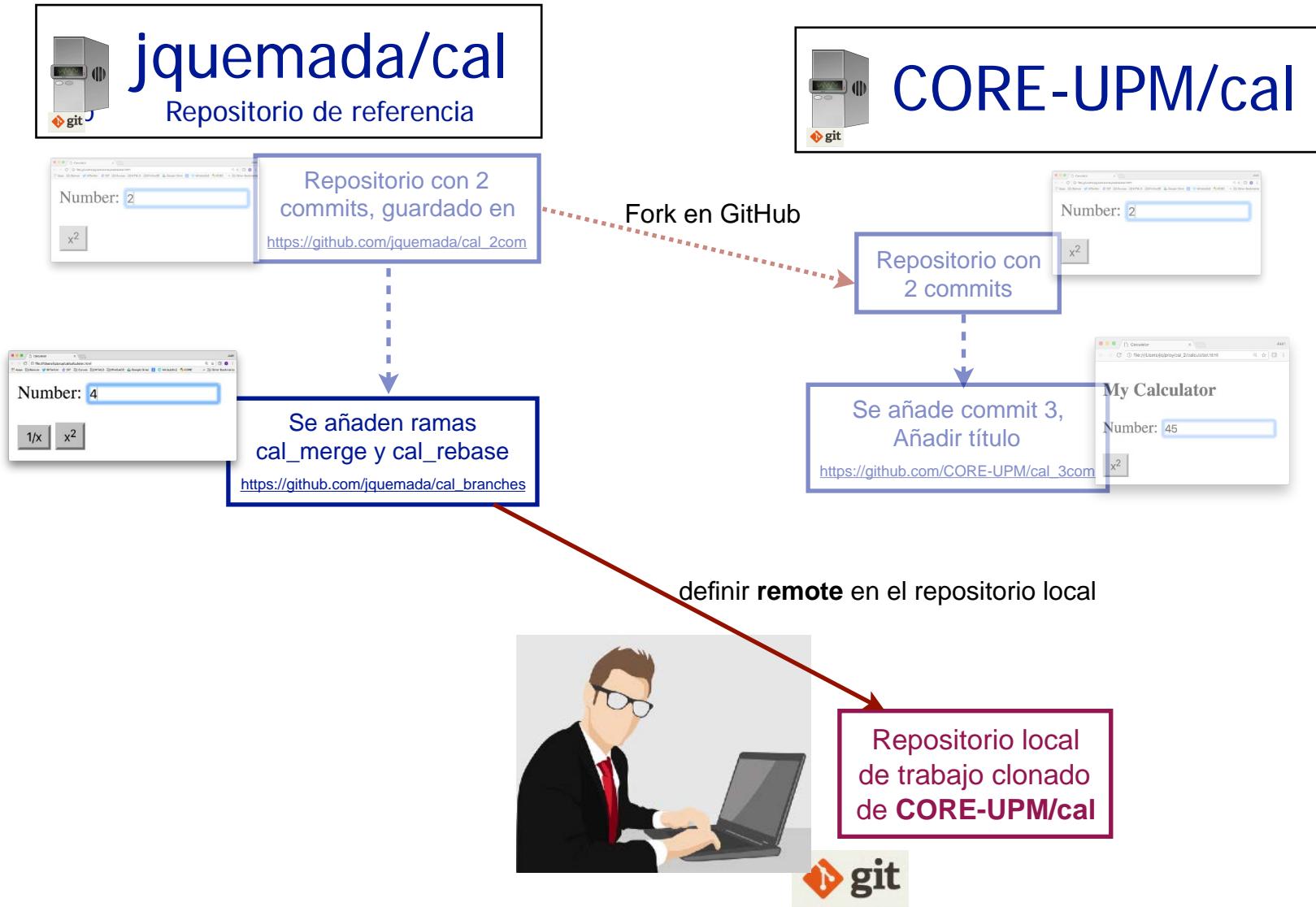
git branch -a -vv muestra la rama **master** definida como **tracking** de **origin/master**.

git branch -a -vv muestra las ramas remotas de **origin** definidas con fetch: **HEAD** y **master**



Definir un remote y sus ramas

jquemada/cal como remote



Crear un remote

Se crea el remote **cal** asociado <https://github.com/jquemada/cal> del que se creo el repositorio remoto origin (<https://github.com/CORE-UPM/cal>) con **Fork** cuando tenía 2 commits.

```
venus:cal_2 jq$ git remote add cal https://github.com/jquemada/cal
venus:cal_2 jq$ git remote -v
cal    https://github.com/jquemada/cal (fetch)
cal    https://github.com/jquemada/cal (push)
origin https://github.com/CORE-UPM/cal (fetch)
origin https://github.com/CORE-UPM/cal (push)
venus:cal_2 jq$ git branch -a -vv
* master          5c70f9c [origin/master] Añadir título
  remotes/origin/HEAD -> origin/master
  remotes/origin/master 5c70f9c Añadir título
venus:cal_2 jq$ git log --oneline --graph --all
* 5c70f9c Añadir título
* b0e63ad x^2 button
* 1096247 Readme & License
venus:cal_2 jq$
```

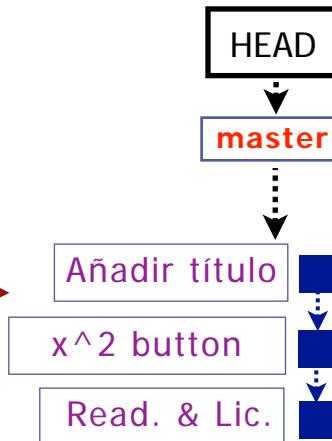
Muestra el nuevo remote cal.

Crear un remote

```
venus:cal_2 jq$ git remote add cal https://github.com/jquemada/cal
venus:cal_2 jq$ git remote -v
cal    https://github.com/jquemada/cal (fetch)
cal    https://github.com/jquemada/cal (push)
origin https://github.com/CORE-UPM/cal (fetch)
origin https://github.com/CORE-UPM/cal (push)
venus:cal_2 jq$ git branch -a -vv
* master          5c70f9c [origin/master] Añadir título
  remotes/origin/HEAD -> origin/master
  remotes/origin/master 5c70f9c Añadir título
venus:cal_2 jq$ git log --oneline --graph --all
* 5c70f9c Añadir título
* b0e63ad x^2 button
* 1096247 Readme & License
venus:cal_2 jq$
```

git branch -a -vv muestra que al crear el remote cal no se han creado nuevas ramas remotas, esto ocurrirá cuando se actualicen con fetch.

git log --oneline --graph --all muestra que el grafo de commits tampoco se ha modificado. Solo tiene los 3 commits que se clonaron.



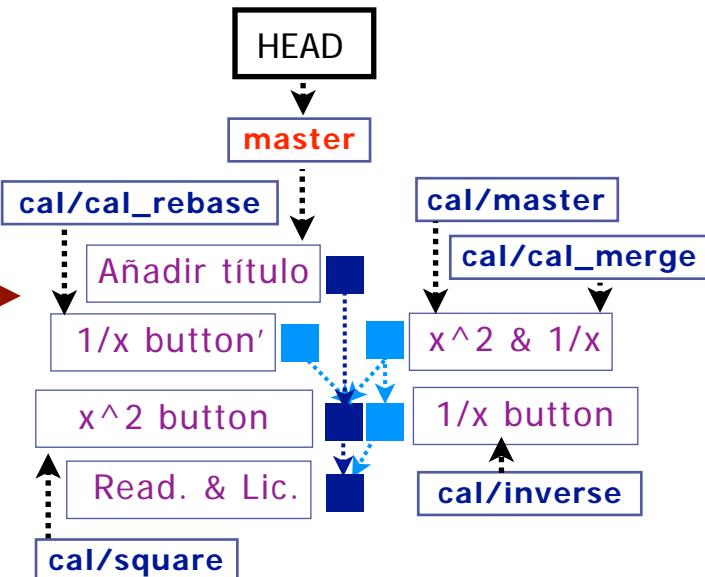
Crear ramas remotas

git fetch -q cal crea **ramas remotas** en el repositorio local asociadas a las ramas que existen en el repositorio remoto asociado a cal.

```
venus:cal_2 jq$ git fetch -q cal
venus:cal_2 jq$ git branch -a -vv
* master 5c70f9c [origin/master] Añadir título
  remotes/cal/cal_merge 1898ac7 Integrate x^2 & 1/x
  remotes/cal/cal_rebase 795c2da 1/x button
  remotes/cal/inverse e868dc4 1/x button
  remotes/cal/master 1898ac7 Integrate x^2 & 1/x
  remotes/cal/square b0e63ad x^2 button
  remotes/origin/HEAD -> origin/master
  remotes/origin/master 5c70f9c Añadir título
venus:cal_2 jq$ git log --oneline --graph --all
* 5c70f9c Añadir título
| * 795c2da 1/x button
|/
| * 1898ac7 Integrate x^2 & 1/x
| \
| / ←
|/
* | b0e63ad x^2 button
| * e868dc4 1/x button
|/
* 1096247 Readme & License
venus:cal_2 jq$
```

git branch -a -vv muestra las ramas del remote **cal** creadas con fetch.

git log --oneline --graph --all muestra que el grafo de commits del repositorio local incluye ahora también los commits de las nuevas ramas remotas creadas.





Copiar ramas remotas en locales

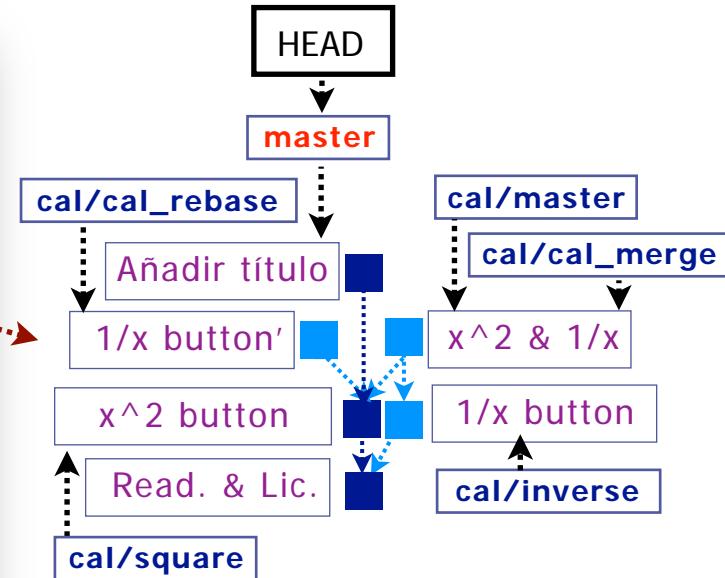
Copiar una rama remota en una local

- ◆ Para añadir desarrollos a una **rama remota** debemos copiarla en una local
 - Los desarrollos se realizan sobre la rama local añadiendo nuevos commits
- ◆ **git checkout ...**
 - Copia una rama remota en una local tracking y restaura la rama local (solo cuando no existe)
 - **git checkout square** crea y restaura la rama tracking square asociada a `<remote>/square`
 - Restaura una rama remota en modo detached HEAD, es decir no asociada a ninguna rama
 - **git checkout origin/square** restaura `origin/square` en modo detached HEAD
- ◆ **git fetch**
 - Copia una rama remota en una local utilizando refsspecs: `[+]<local_branch>:<remote_branch>`
 - **git fetch origin square:sqrt**
 - Crea o actualiza la rama local `sqrt` con los commits de la remota `origin/square`
 - **git fetch origin pull/1/head:s1**
 - Crea o actualiza la rama local `s1` con el pull_request 1 del repositorio remoto origin en GitHub
 - **git fetch cal_branches +s1:s1**
 - Crea o actualiza la rama local `s1` con la remota `cal_branches/s1` aunque sean incompatibles (+)
 - **git fetch https://github.com/jquemada/cal square:square**
 - Crea o actualiza la rama local `square` con la rem. `square` de `https://github.com/jquemada/cal`
- ◆ **git pull**
 - traer la rama remota indicada e integrarla con una rama del repositorio local
 - **git pull cal_branches square** integra la rama `square` de `cal_branches` en la rama activa
 - **git pull origin pull/1/head** Integra el `pull_request #1` en la rama activa

Ramas remotas y locales

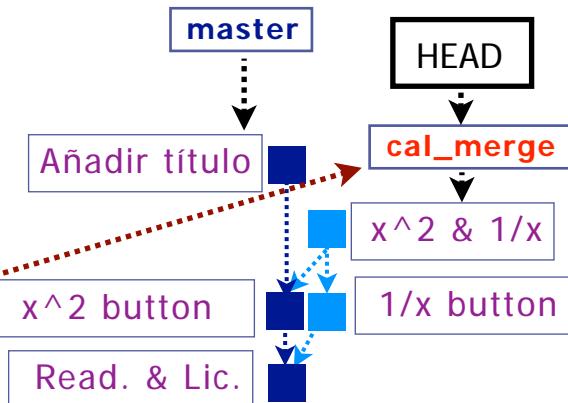
git branch -a -vv muestra que master es la única rama local y todas las remotas de los remotes origin y cal.

```
venus:cal_2 jq$ git branch -a -vv
* master          5c70f9c [origin/master] Añadir título
  remotes/cal/cal_merge 1898ac7 Integrate x^2 & 1/x
  remotes/cal/cal_rebase 795c2da 1/x button
  remotes/cal/inverse    e868dc4 1/x button
  remotes/cal/master     1898ac7 Integrate x^2 & 1/x
  remotes/cal/square     b0e63ad x^2 button
  remotes/origin/HEAD    -> origin/master
  remotes/origin/master  5c70f9c Añadir título
venus:cal_2 jq$ git checkout -q cal_merge
venus:cal_2 jq$ git branch -vv
* cal_merge 1898ac7 [cal/cal_merge] Integrate x^2 & 1/x
  master      5c70f9c [origin/master] Añadir título
venus:cal_2 jq$ git fetch -q cal cal_rebase:cal_rebase
venus:cal_2 jq$ git branch -vv
* cal_merge 1898ac7 [cal/cal_merge] Integrate x^2 & 1/x
  cal_rebase 795c2da 1/x button
  master      5c70f9c [origin/master] Añadir título
venus:cal_2 jq$
```



checkout: crea rama local tracking

```
venus:cal_2 jq$  
venus:cal_2 jq$ git branch -a -vv  
* master 5c70f9c [origin/master] Añadir título  
  remotes/cal/cal_merge 1898ac7 Integrate x^2 & 1/x  
  remotes/cal/cal_rebase 795c2da 1/x button  
  remotes/cal/inverse e868dc4 1/x button  
  remotes/cal/master 1898ac7 Integrate x^2 & 1/x  
  remotes/cal/square b0e63ad x^2 button  
  remotes/origin/HEAD -> origin/master  
  remotes/origin/master 5c70f9c Añadir título  
venus:cal_2 jq$  
venus:cal_2 jq$ git checkout -q cal_merge  
venus:cal_2 jq$  
venus:cal_2 jq$ git branch -vv  
* cal_merge 1898ac7 [cal/cal_merge] Integrate x^2 & 1/x  
  master 5c70f9c [origin/master] Añadir título  
venus:cal_2 jq$  
venus:cal_2 jq$  
venus:cal_2 jq$ git fetch -q cal cal_rebase:cal_rebase  
venus:cal_2 jq$  
venus:cal_2 jq$ git branch -vv  
* cal_merge 1898ac7 [cal/cal_merge] Integrate x^2 & 1/x  
  cal_rebase 795c2da 1/x button  
  master 5c70f9c [origin/master] Añadir título  
venus:cal_2 jq$
```



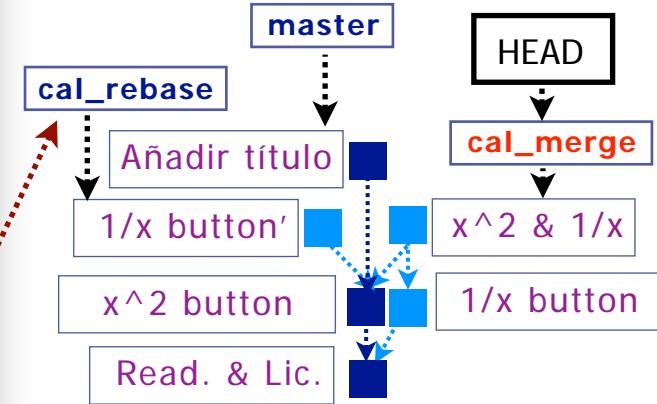
git checkout cal_merge copia la rama remota **cal/cal_merge** en una local del mismo nombre y la restaura en el directorio de trabajo.

Es una forma muy cómoda de **restaurar ramas locales** para trabajar en ellas.

git branch -vv muestra como la nueva rama creada y restaurada es tracking.

fetch con refspec: crea rama local

```
venus:cal_2 jq$  
venus:cal_2 jq$ git branch -a -vv  
* master 5c70f9c [origin/master] Añadir título  
  remotes/cal/cal_merge 1898ac7 Integrate x^2 & 1/x  
  remotes/cal/cal_rebase 795c2da 1/x button  
  remotes/cal/inverse e868dc4 1/x button  
  remotes/cal/master 1898ac7 Integrate x^2 & 1/x  
  remotes/cal/square b0e63ad x^2 button  
  remotes/origin/HEAD -> origin/master  
  remotes/origin/master 5c70f9c Añadir título  
  
venus:cal_2 jq$  
venus:cal_2 jq$ git checkout -q cal_merge  
venus:cal_2 jq$  
venus:cal_2 jq$ git branch -vv  
* cal_merge 1898ac7 [cal/cal_merge] Integrate x^2 & 1/x  
  master 5c70f9c [origin/master] Añadir título  
  
venus:cal_2 jq$  
venus:cal_2 jq$  
venus:cal_2 jq$ git fetch -q cal cal_rebase:cal_rebase  
venus:cal_2 jq$  
venus:cal_2 jq$ git branch -vv  
* cal_merge 1898ac7 [cal/cal_merge] Integrate x^2 & 1/x  
  cal_rebase 795c2da 1/x button  
  master 5c70f9c [origin/master] Añadir título  
venus:cal_2 jq$
```

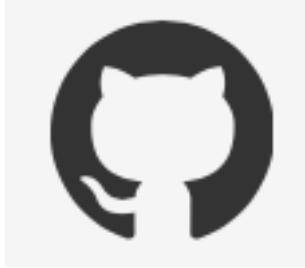


git fetch cal cal_rebase:cal_rebase
es otra forma de crear la rama local
(cal_rebase) utilizando fetch y refspecs a
partir de la rama remota (cal/cal_rebase)

git branch -vv muestra
como la nueva rama creada
ahora **no es tracking**.

Ejercicio opcional

- ◆ Clonar el repositorio <cuenta_2>/cal_branches a un repo. local mi_cal, con
 - `git clone https://github.com/<cuenta_2>/cal_branches mi_cal`
 - Entrar en el directorio mi_cal e inspeccionar con `git branch -a -vv` las ramas locales, remotas y tracking
- ◆ Copiar la rama master de https://github.com/<su_cuenta>/cal_2com
 - a la rama local master_2com del repositorio local mi_cal con
 - `git fetch https://github.com/<su_cuenta>/cal_2com master:master_2com`
 - Copia la rama master de https://github.com/<su_cuenta>/cal_2com en la rama local master_2com
- ◆ Restaurar master con `git checkout master`
 - Integrar master_2com en master con `git merge master_2com`
 - Para integrar los 3 botones en una única calculadora en la rama master
- ◆ Resolver conflictos de integración: los 3 botones de la calc. funcionen bien
 - Registrar los cambios en el índice con `git add .`
 - Cerrar un nuevo commit con `git commit -m "Integrar 3 botones"`
- ◆ Actualizar master en origin para guardar la nueva calculadora
 - `git push`
 - Recordar que master es tracking de origin/master y `git push` actualiza las ramas tracking
 - Inspeccionar con el navegador el repositorio actualizado
 - que contiene en master tanto el commit de integración como los commits heredados de <su_cuenta>/cal_2com



Git y GitHub

Contribuir a un repositorio central con pull request:
auto-merge, branch, clone, fetch, merge, pull y push

Juan Quemada, DIT - UPM



El repositorio central

◆ **Repositorio central o de referencia** de un proyecto

- Publica las versiones maduras y probadas de un proyecto, permitiendo descargarlas

◆ Este repositorio se clonar con **Fork** en otra cuenta para contribuir

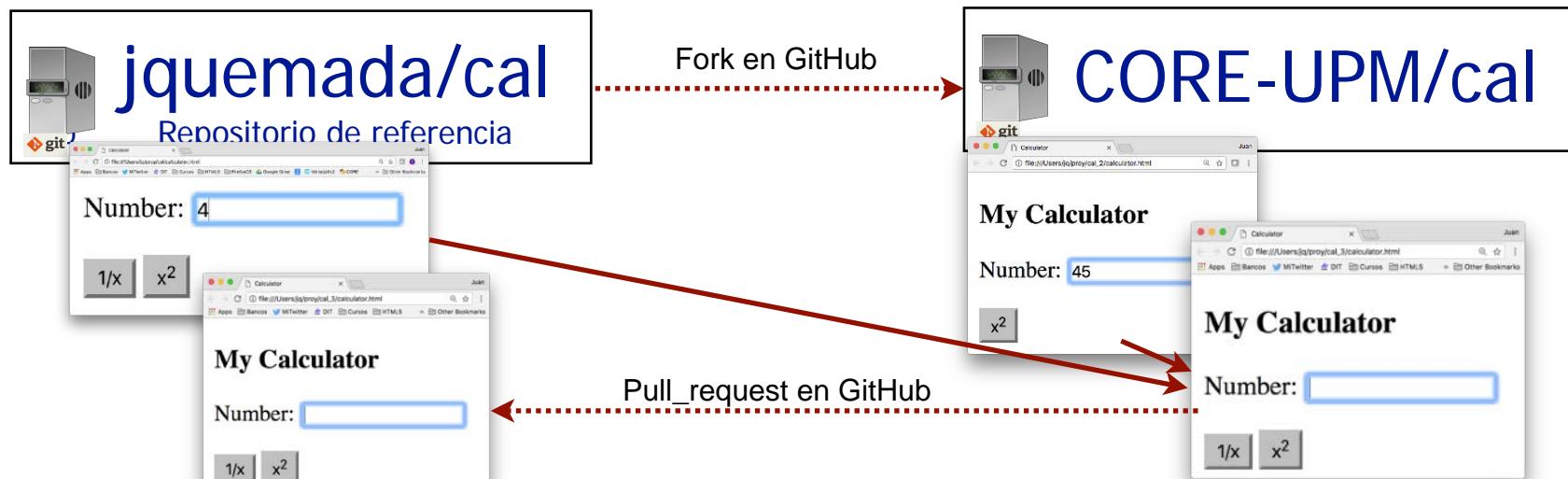
- Las **contribuciones** se envían al repositorio de referencia como **pull requests**
 - Fork y pull request sistematizan la forma de contribuir a proyectos albergados en GitHub

◆ Los administradores del proyecto analizan las contribuciones (**pull_requests**)

- Y las integran si consideran que enriquecen el proyecto

◆ La **calculadora con título** se integra con la de **2 botones** en **CORE-UPM/cal**

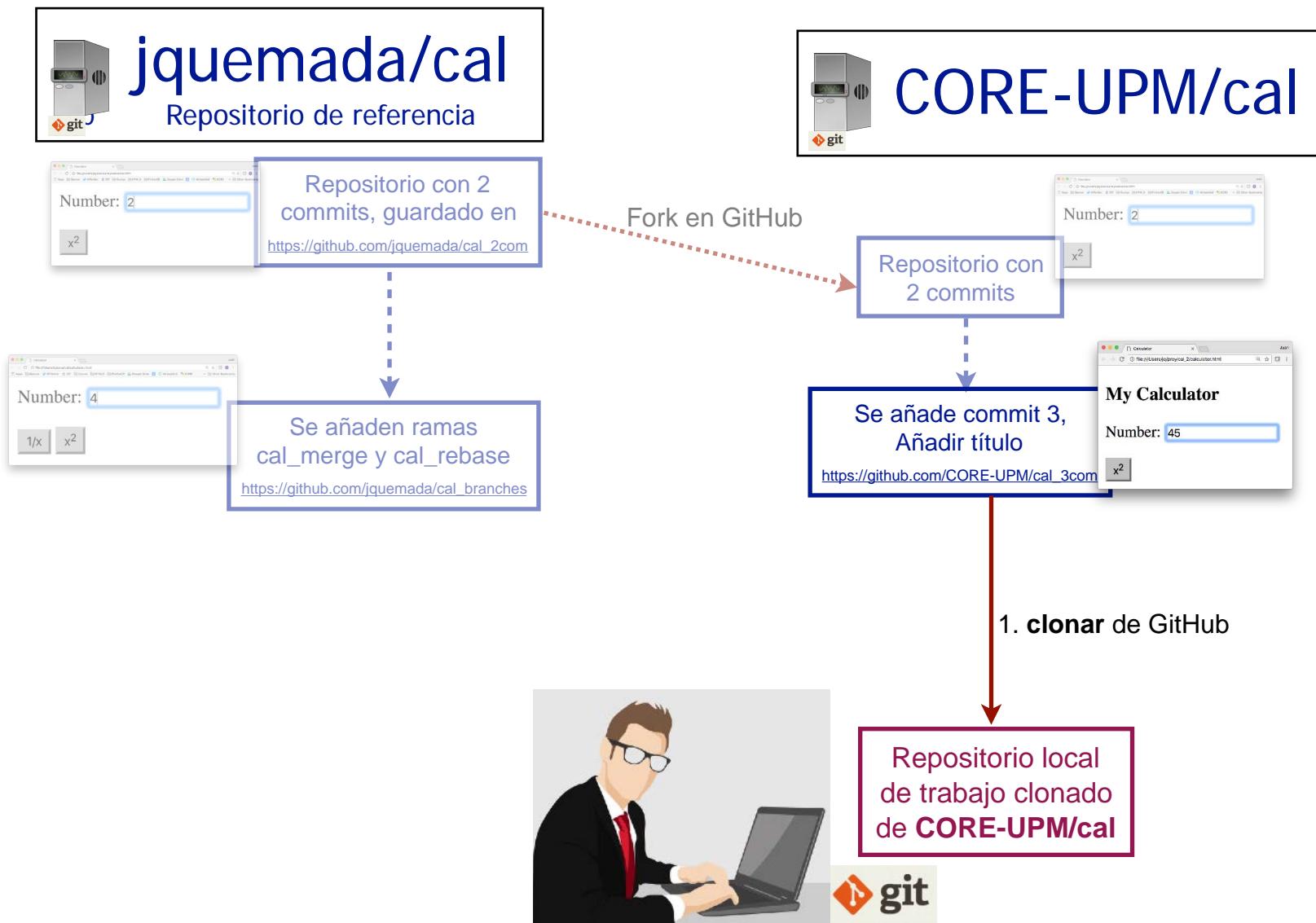
- La integración se contribuye con **pull_request** a **jquemada/cal**, donde se integra en master





Clonar rep. CORE-UPM/cal e integrar
rama cal_merge de jquemada/cal

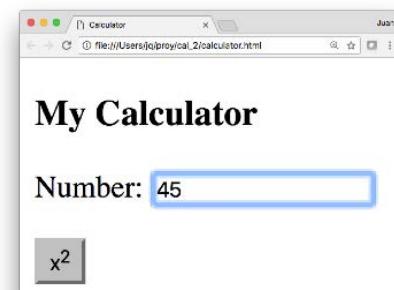
Clonar CORE-UPM/cal



Clonar CORE-UPM/cal

```
venus:proy jq$ git clone -q https://github.com/CORE-UPM/cal cal_2
venus:proy jq$ cd cal_2
venus:cal_2 jq$ git branch -vv
* master 5c70f9c [origin/master] Añadir título
venus:cal_2 jq$ git log --oneline --graph
* 5c70f9c Añadir título
* b0e63ad x^2 button
* 1096247 Readme & License
venus:cal_2 jq$ git fetch -q https://github.com/jquemada/cal
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master 5c70f9c [origin/master] Añadir título
venus:cal_2 jq$ git merge -q -m "Integr, title & cal" cal_merge
Auto-merging calculator.html
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master 760145c [origin/master: ahead 3] Integr, title & cal
venus:cal_2 jq$ git log --oneline --graph
* 760145c Integr, title & cal
| \
| * 1898ac7 Integrate x^2 & 1/x
| |
| * | e868dc4 1/x button
* | | 5c70f9c Añadir título
| |
| * | b0e63ad x^2 button
|
* 1096247 Readme & License
venus:cal_2 jq$
```

Clonar **CORE-UPM/cal** de GitHub en **cal_2**. Es un fork de **jquemada/cal** para permitir pull requests.



Number: 45

HEAD

master

Añadir título

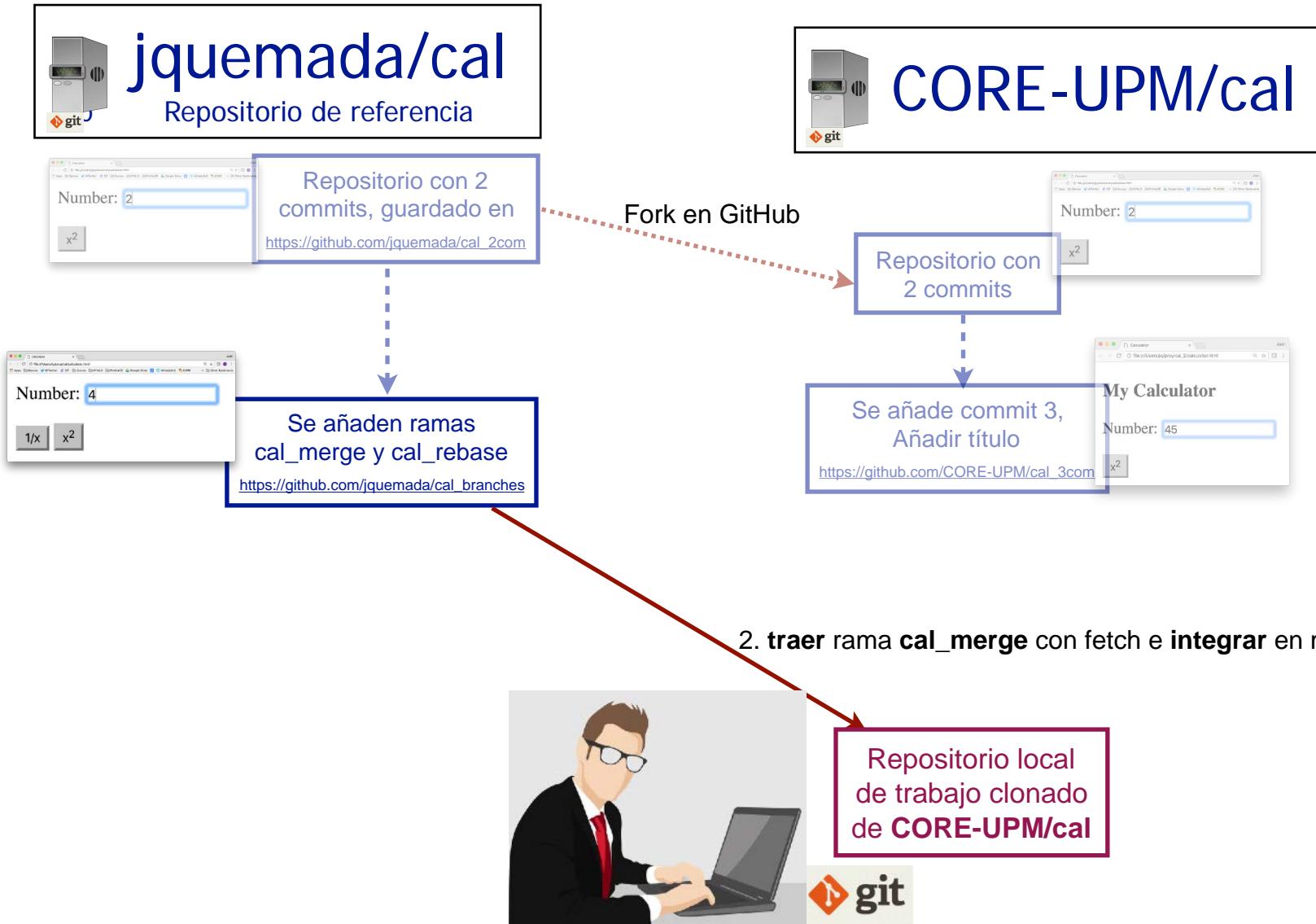
x^2 button

Read. & Lic.

El repositorio contiene la rama **master tracking** de **origin/master**.

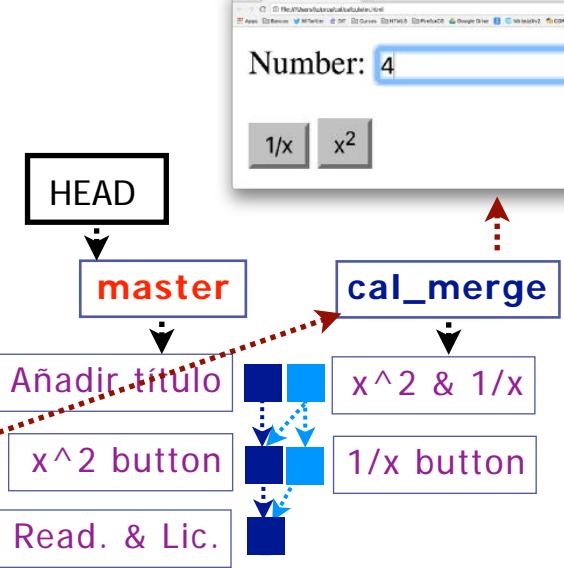
Muestra el grafo de la rama master con 3 commits lineales.

Traer cal_merge de jquemada/cal e integrar



fetch cal_merge de GitHub

```
venus:proy jq$ git clone -q https://github.com/CORE-UPM/cal cal_2
venus:proy jq$ cd cal_2
venus:cal_2 jq$ git branch -vv
* master 5c70f9c [origin/master] Añadir título
venus:cal_2 jq$ git log --oneline --graph
* 5c70f9c Añadir título
* b0e63ad x^2 button
* 1096247 Readme & License
venus:cal_2 jq$ git fetch -q https://github.com/jquemada/cal cal_merge:cal_merge
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master 5c70f9c [origin/master] Añadir título
venus:cal_2 jq$ git merge -q -m "Integr, title & cal" cal_merge
Auto-merging calculator.html
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master 760145c [origin/master: ahead 3] Integr, title & cal
venus:cal_2 jq$ git log --oneline --graph
* 760145c Integr, title & cal
| \
| * 1898ac7 Integrate x^2 & 1/x
| |
| * | e868dc4 1/x button
* | | 5c70f9c Añadir título
| |
| * | b0e63ad x^2 button
| |
* 1096247 Readme & License
venus:cal_2 jq$
```



Copiar **cal_merge** en el repositorio local. El repositorio remoto se referencia con su URL y la rama con refspec, sin definir ningún remote.

El repositorio incluye ahora la nueva rama local **cal_merge**.

Integrar cal_merge

```

venus:proy jq$ git clone -q https://github.com/CORE-UPM/cal cal_2
venus:proy jq$ cd cal_2
venus:cal_2 jq$ git branch -vv
* master 5c70f9c [origin/master] Añadir título
venus:cal_2 jq$ git log --oneline --graph
* 5c70f9c Añadir título
* b0e63ad x^2 button
* 1096247 Readme & License
venus:cal_2 jq$ git fetch -q https://github.com/jquemada/cal
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master 5c70f9c [origin/master] Añadir título
venus:cal_2 jq$ git merge -q -m "Integr, title & cal" cal_merge
Auto-merging calculator.html
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master 760145c [origin/master: ahead 3] Integr, title & cal
venus:cal_2 jq$ git log --oneline --graph
* 760145c Integr, title & cal
| \
| * 1898ac7 Integrate x^2 & 1/x
| |
| * | e868dc4 1/x button
* | | 5c70f9c Añadir título
| |
| / 
| / 
* | b0e63ad x^2 button
| /
* 1096247 Readme & License
venus:cal_2 jq$ 

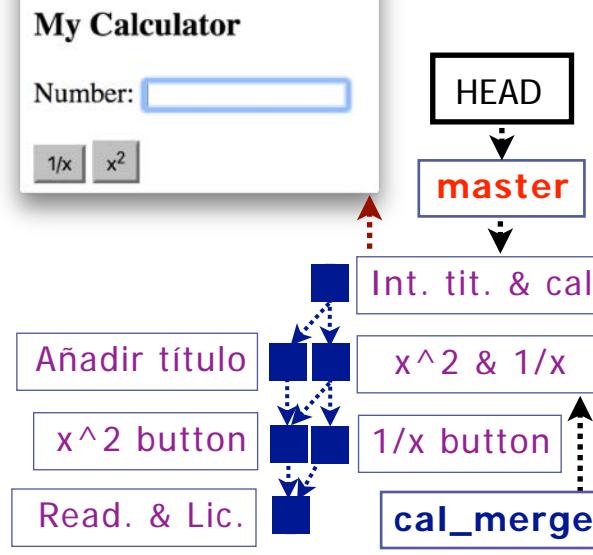
```

git merge .. integra la rama local `cal_merge` en master.

`cal_merge:cal_merge`

master está 3 commits por delante de origin.

El commit integrado añade solo nuevas líneas de código a `calculator.html`, se realiza auto-merging.



```

<!DOCTYPE html><html><head>
<title>Calculator</title><meta charset="utf-8">
<script type="text/javascript">

function inverse() {
  var num = document.getElementById("n1");
  num.value = 1/num.value;
}

function square() {
  var num = document.getElementById("n1");
  num.value = num.value * num.value;
}
</script>
</head>

<h3>My Calculator</h3>

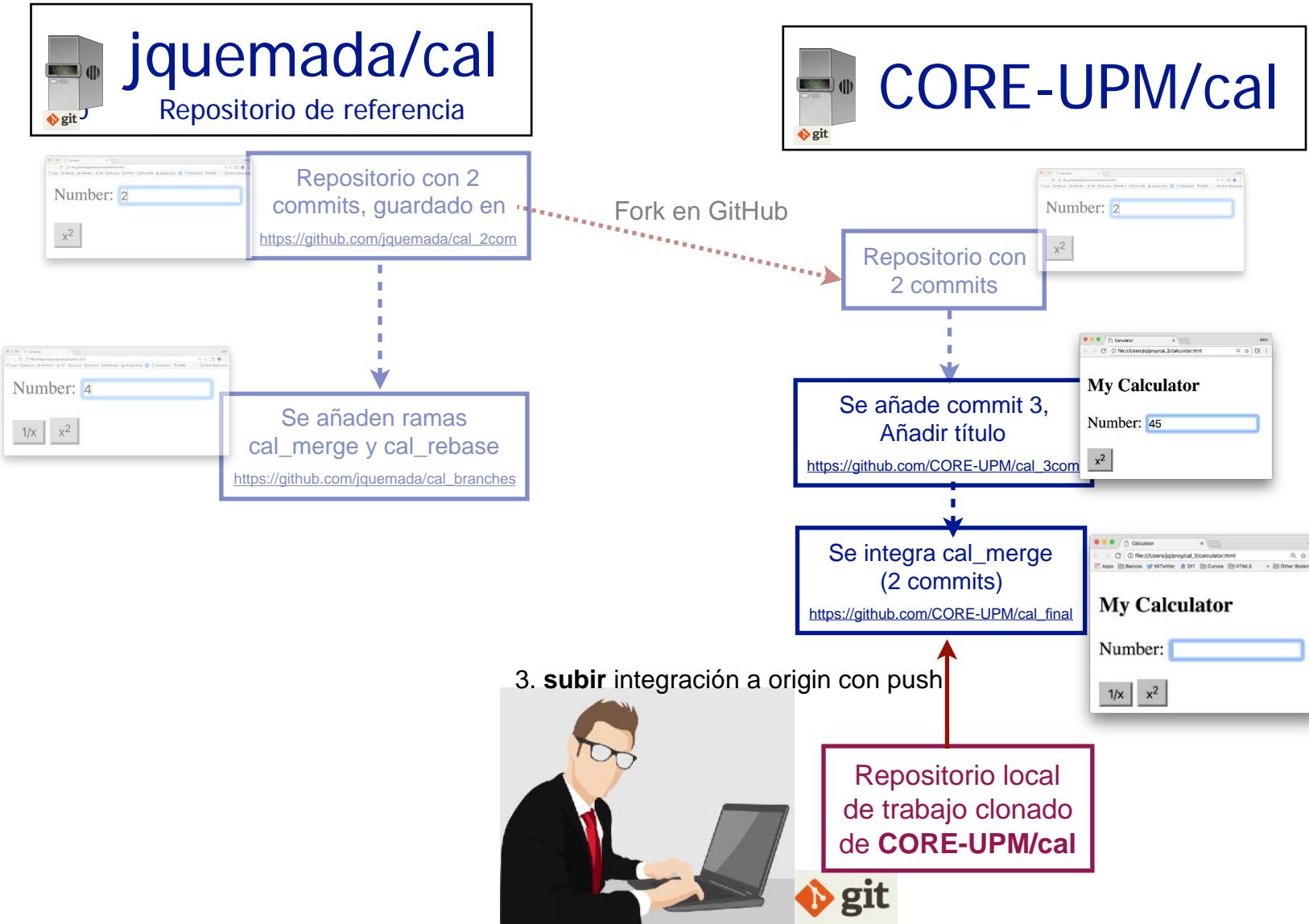
<body>
Number:
<input type="text" id="n1"><p>
<button onclick="inverse()"> 1/x </button>
<button onclick="square()"> x2 </button>
</body>
</html>

```



Actualizar rama master (tracking) de CORE-UPM/cal

Actualizar origin



3. Rama master local no sincronizada

```
cal_2 — bash — 69x11
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master    760145c [origin/master: ahead 3] Integr, title & cal
venus:cal_2 jq$ git push -q
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master    760145c [origin/master] Integr, title & cal
venus:cal_2 jq$
```

git branch -vv muestra la rama master desincronizada respecto a la remota asociada como tracking, está 3 commits por delante de la remota.

The screenshot shows a Mac OS X desktop with two windows open. The top window is a terminal session titled 'cal_2 — bash — 69x11' containing command-line output about Git branches. The bottom window is a web browser displaying a GitHub repository page for 'CORE-UPM/cal'. A dashed purple arrow points from the terminal's output to the GitHub repository's commit history, indicating the state of the local 'master' branch relative to its remote counterpart.

GitHub Repository Page Details:

- Repository:** CORE-UPM/cal
- Forked From:** jquemada/cal
- Commits:** 3 (highlighted with a pink box)
- Branches:** 1 (highlighted with a pink box)
- Releases:** 0
- Contributors:** 1

Terminal Output Summary:

- Initial command: `git branch -vv`
- Output:
 - Local 'cal_merge' branch at commit 1898ac7, tracking remote 'cal_merge'.
 - Local 'master' branch at commit 760145c, which is ahead of remote 'origin/master' by 3 commits.
- After pushing:
 - Local 'cal_merge' branch remains at 1898ac7.
 - Local 'master' branch remains at 760145c.

La rama master contiene solo los 3 commits iniciales

3. Sincronizar rama master remota

```
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master    760145c [origin/master]: ahead 3] Integr, title & cal
venus:cal_2 jq$ git push -q
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master    760145c [origin/master] Integr, title & cal
venus:cal_2 jq$
```

git push sincroniza las ramas tracking subiendo sus nuevos commits a las remotas asociadas.

git branch -vv muestra que la rama master local se ha sincronizado con la remota.

The screenshot shows a Mac OS X desktop with a terminal window and a web browser window.

Terminal Window:

```
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master    760145c [origin/master]: ahead 3] Integr, title & cal
venus:cal_2 jq$ git push -q
venus:cal_2 jq$ git branch -vv
  cal_merge 1898ac7 Integrate x^2 & 1/x
* master    760145c [origin/master] Integr, title & cal
venus:cal_2 jq$
```

GitHub Repository Page:

The GitHub page for the repository `CORE-UPM/cal` shows the following details:

- Code:** 6 commits
- Pull requests:** 0
- Projects:** 0
- Wiki:** Pulse
- Graphs:** Settings
- Unwatch:** 6
- Star:** 0
- Fork:** 1

A callout box points to the "6 commits" link with the text: **Se han añadido los 3 commits de la integración**.

At the bottom of the GitHub page, there are buttons for **Branch: master**, **New pull request**, **Create new file**, **Upload files**, **Find file**, and **Clone or download**.

Actualizar ramas remotas

◆ **git push ... sube** los cambios en las ramas locales a los repositorios remotos

- Las ramas locales sobre las que se trabaja son normalmente ramas tracking de las remotas

◆ **git push**

- Copia una rama local en una remota y puede utilizar refsspecs: `[+]<local_branch>:<remote_branch>`
 - **git push**
 - Actualiza las ramas remotas de las ramas locales **tracking** definidas en el repositorio local
 - **git push -f cal_branches master**
 - Actualiza la rama **cal_branches/master** con la local **master**, aunque sean incompatibles
 - **git push cal_branches sqrt:square**
 - Actualiza la rama remota **cal_branches/square** con los nuevos commits de la local **sqrt**
 - **git push https://github.com/jquemada/cal sqrt:square**
 - Actualiza la rama remota **cal/square** con los nuevos commits de la local **sqrt**

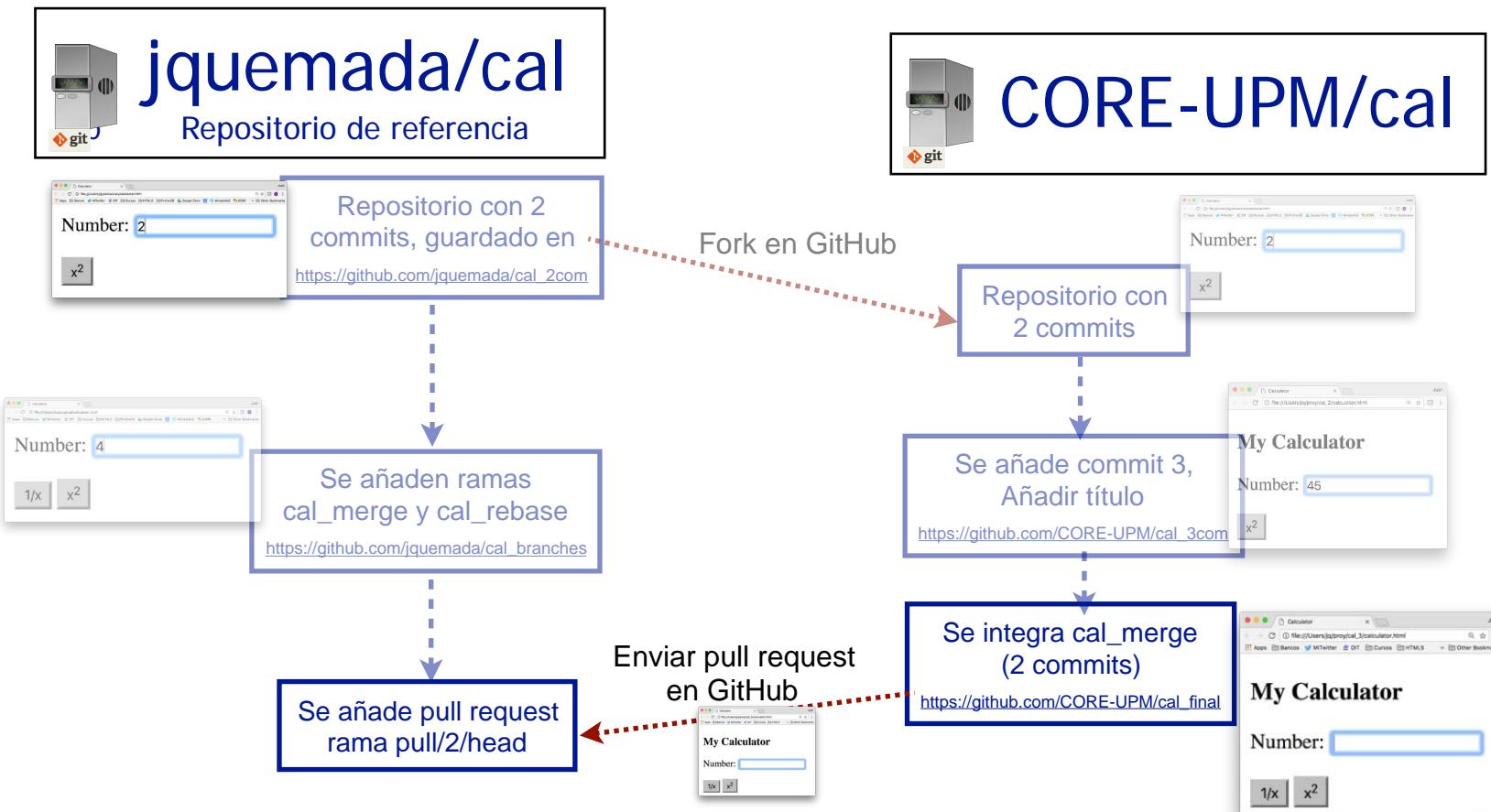
◆ **git push**

- Borrar ramas en un repositorio remoto (**OJO!** son las ramas reales del repositorio remoto)
 - **git push origin :sqrt** Borra la rama **sqrt** en el repositorio remoto **origin**
 - **git push origin --delete sqrt** Similar a anterior, opción posible en versiones recientes de git



Enviar pull request desde CORE-UPM/cal a jquemada/cal en GitHub

Enviar un pull request



Preparar pull_request

The screenshot shows a GitHub interface comparing two branches. On the left, the repository `jquemada / cal` is selected. A yellow box highlights the title "Comparing changes". Another yellow box highlights the status message "Selectores de rama destino y rama origen ya configurados". Below this, a green checkmark indicates "Able to merge". A pink button labeled "Create pull request" is visible. The commit history shows two commits from "jquemada" on Feb 23, 2017, and one commit from "jquemada" on Mar 03, 2017. The file `calculator.html` has been changed. A code diff is shown at the bottom.

The screenshot shows the repository `CORE-UPM / cal`, which is a fork of `jquemada / cal`. A yellow box highlights the title "CORE-UPM / cal". Another yellow box highlights the "New pull request" button. The repository has 6 commits and 1 branch. The commit history shows the same three commits from "jquemada". A pink box highlights the "cal" folder. A yellow box highlights the "Botón para enviar pull_request" (Send pull request button).

Editar pull_request

The screenshot shows the GitHub interface for comparing branches. At the top, it says "Comparing jquemada:master...CORE-UPM:master". Below that, there are buttons for "Code", "Issues 0", "Pull requests 0", "Projects 0", and "Wiki". The main title is "Comparing changes". It says "Choose two branches to see what's changed or to start a new pull request. If you ne..." followed by a dropdown menu: "base fork: jquemada/cal", "base: master", "...", "head fork: CORE-UPM/cal". A green checkmark says "Able to merge. These branches can be automatically merged." A pink button labeled "Create pull request" is highlighted with a yellow box. Below it, a message says "Discuss and review the changes in this comparison with your team." It shows "2 commits" and "1 file changed". The commit history includes "Commits on Feb 23, 2017" by "jquemada" and "Commits on Mar 03, 2017" by "jquemada". A yellow box highlights the "Descripción del pull request" (Description of the pull request) section at the bottom, which contains the file content "calculator.html".

The screenshot shows the GitHub interface for opening a pull request. At the top, it says "Comparing jquemada:master...CORE-UPM:master". Below that, there are buttons for "Code", "Issues 0", "Pull requests 0", "Projects 0", "Wiki", "Pulse", and "Graphs". The main title is "Open a pull request". It says "Create a new pull request by comparing changes across two branches. If you need to, you can also compare acros..." followed by a dropdown menu: "base fork: jquemada/cal", "base: master", "...", "head fork: CORE-UPM/cal", "compare: master". A green checkmark says "Able to merge. These branches can be automatically merged." A pink button labeled "Integrate HTML Title & merge branch" is highlighted with a yellow box. Below it, a message says "Please integrate this version of calculator.html". There is a "Write" tab and a "Preview" tab. A checkbox says "Allow edits from maintainers. Learn more". A green button labeled "Create pull request" is highlighted with a yellow box. Below it, it shows "2 commits" and "Commits on Feb 23, 2017" by "jquemada". A yellow box highlights the "Titulo del pull request" (Title of the pull request) section at the top right.

Enviar pull_request a origen del Fork

The screenshot shows the GitHub interface for a forked repository. A yellow box labeled "Indicador de pull requests" points to the "Pull requests" tab in the top navigation bar. Another yellow box labeled "Tituo del pull request" points to the title area of a pull request card. A third yellow box labeled "Open a pull request" points to the "Create pull request" button at the bottom right. A fourth yellow box labeled "Identificador de pull request: #2" points to the pull request number in the title. A fifth yellow box labeled "GitHub numera los pull request en el orden de llegada." points to the numerical ordering of the pull request. A sixth yellow box labeled "Comentario del pull request" points to a comment from the owner. A seventh yellow box labeled "Nuevos commits a integrar" points to a commit message. A eighth yellow box labeled "Como ha detectado auto-merge podría integrarse directamente en GitHub." points to a message indicating mergeability. A ninth yellow box labeled "Botón de envio de pull request" points to the "Create pull request" button.

Indicador de pull requests

Tituo del pull request

Open a pull request

Identificador de pull request: #2

GitHub numera los pull request en el orden de llegada.

Comentario del pull request

Nuevos commits a integrar

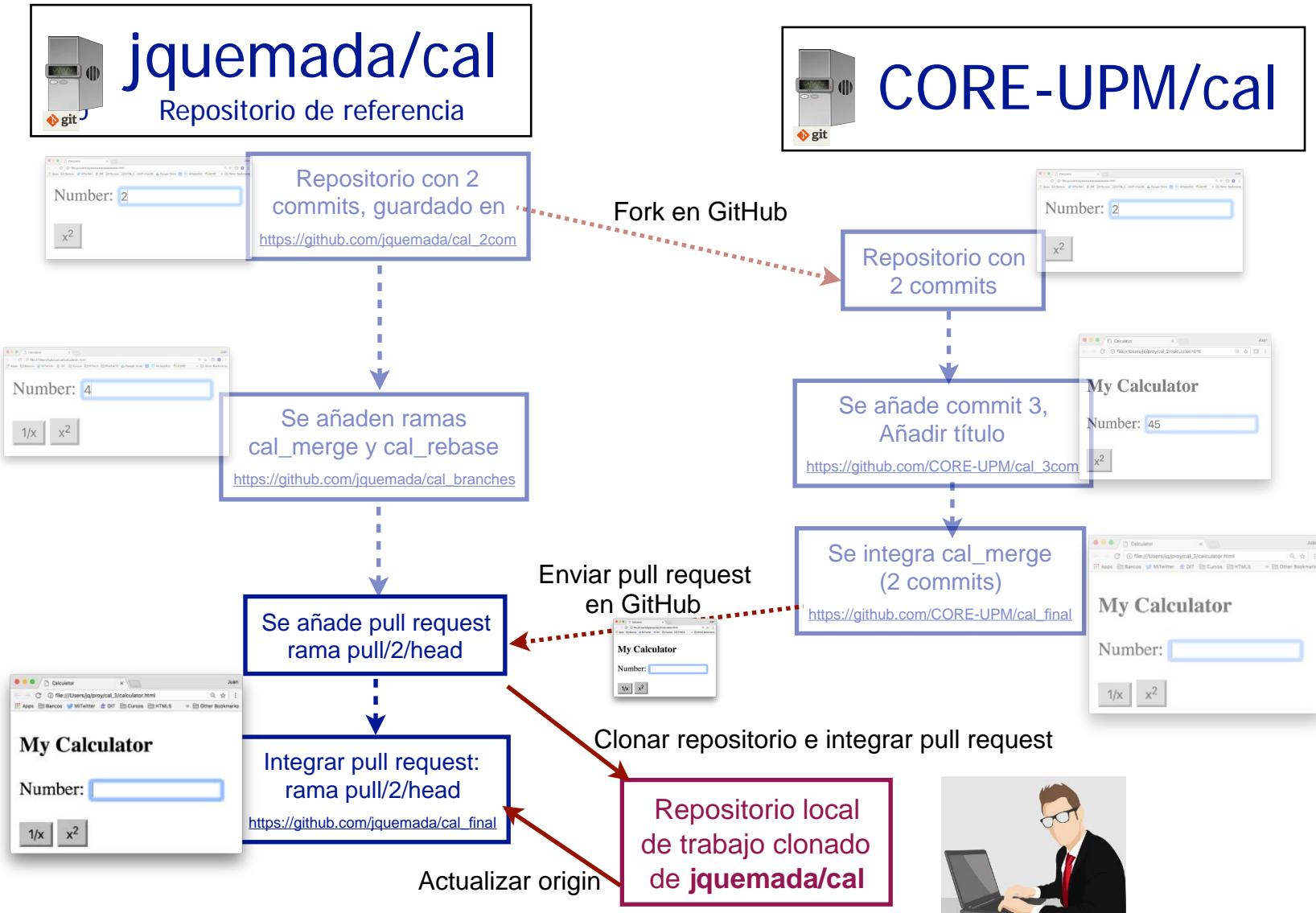
Como ha detectado auto-merge podría integrarse directamente en GitHub.

Botón de envio de pull request



Integrar pull request en jquemada/cal

Integrar pull request en master



```

venus:proy jq$ git clone -q https://github.com/jquemada/cal
venus:proy jq$ cd cal
venus:cal jq$ git branch -vv
* master 1898ac7 [origin/master] Integrate x^2 & 1/x
rama master tracking local.

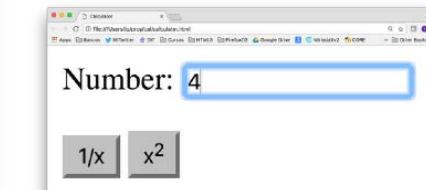
venus:cal jq$ git log --oneline --graph
* 1898ac7 Integrate x^2 & 1/x
|\ 
| * b0e63ad x^2 button
| * e868dc4 1/x button
|/
* 1096247 Readme & License
venus:cal jq$ git pull -q origin pull/2/head
venus:cal jq$ git branch -vv
* master 760145c [origin/master: ahead 2] Integr, title & cal
venus:cal jq$ git log --oneline --graph
* 760145c Integr, title & cal
|\ 
| * 1898ac7 Integrate x^2 & 1/x
| |\ 
| | * e868dc4 1/x button
| * 5c70f9c Añadir título
| |
|/
* | b0e63ad x^2 button
|/
* 1096247 Readme & License
venus:cal jq$ git push -q
venus:cal jq$ git branch -vv
* master 760145c [origin/master] Integr, title & cal
venus:cal jq$ 

```

clonar repositorio jquemada/cal

Clonar el repositorio **jquemada/cal** de GitHub, para integrar el pull request enviado desde **CORE-UPM/cal**.

Grafo de integración de la rama master local.



Number: 4

1/x

x²

HEAD

master

I. x² & 1/x

x² button

Read. & Lic.

1/x button

cal_merge

Integrar pull request #2

```

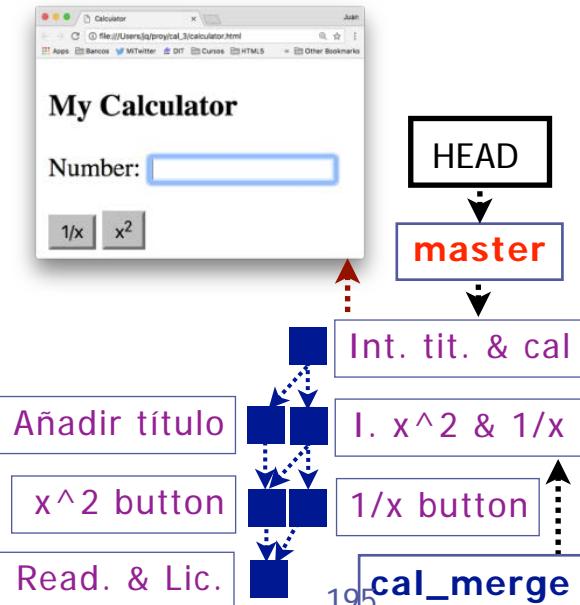
venus:proy jq$ git clone -q https://github.com/jquemada/cal
venus:proy jq$ cd cal
venus:cal jq$ git branch -vv
* master 1898ac7 [origin/master] Integrate x^2 & 1/x
venus:cal jq$ git log --oneline --graph
* 1898ac7 Integrate x^2 & 1/x
|\ 
| * b0e63ad x^2 button
| * e868dc4 1/x button
|/
* 1096247 Readme & License
venus:cal jq$ git pull -q origin pull/2/head
venus:cal jq$ git branch -vv
* master 760145c [origin/master: ahead 2] Integr, title & cal
venus:cal jq$ git log --oneline --graph
* 760145c Integr, title & cal
|\ 
| * 1898ac7 Integrate x^2 & 1/x
| | 
| | * e868dc4 1/x button
* | | 5c70f9c Añadir título
| | / 
| | 
| * b0e63ad x^2 button
|/
* 1096247 Readme & License
venus:cal jq$ git push -q
venus:cal jq$ git branch -vv
* master 760145c [origin/master] Integr, title & cal
venus:cal jq$ 

```

La rama asociada a un pull request se identifican en GitHub por **pull/x/head**, x es el número de pull request.

git pull .. integra una rama remota en la rama activa. La integración se realiza con ff (Fast-Forward), porque la rama enviada en el pull request lleva la integración realizada.

Muestra la integración realizada. se han integrado los 2 commits resaltados.

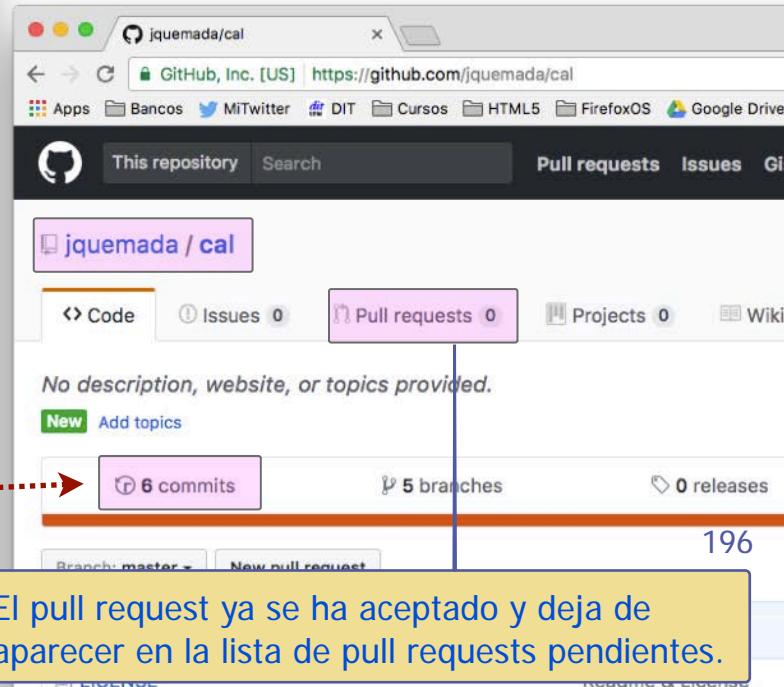


Actualizar en remoto

```
venus:proy jq$  
venus:proy jq$ git clone -q https://github.com/jquemada/cal  
venus:proy jq$ cd cal  
venus:cal jq$  
venus:cal jq$ git branch -vv  
* master 1898ac7 [origin/master] Integrate x^2 & 1/x  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph  
* 1898ac7 Integrate x^2 & 1/x  
|\  
| * b0e63ad x^2 button  
* | e868dc4 1/x button  
|/  
* 1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git pull -q origin pull/2/head  
venus:cal jq$  
venus:cal jq$ git branch -vv  
* master 760145c [origin/master: ahead 2] Integr, title & cal  
venus:cal jq$  
venus:cal jq$ git log --oneline --graph  
* 760145c Integr, title & cal  
|\  
| * 1898ac7 Integrate x^2 & 1/x  
| |\  
| | * e868dc4 1/x button  
* | | 5c70f9c Añadir título  
| |/  
|/  
* | b0e63ad x^2 button  
|/  
* 1096247 Readme & License  
venus:cal jq$  
venus:cal jq$ git push -q  
venus:cal jq$  
venus:cal jq$ git branch -vv  
* master 760145c [origin/master] Integr, title & cal  
venus:cal jq$
```

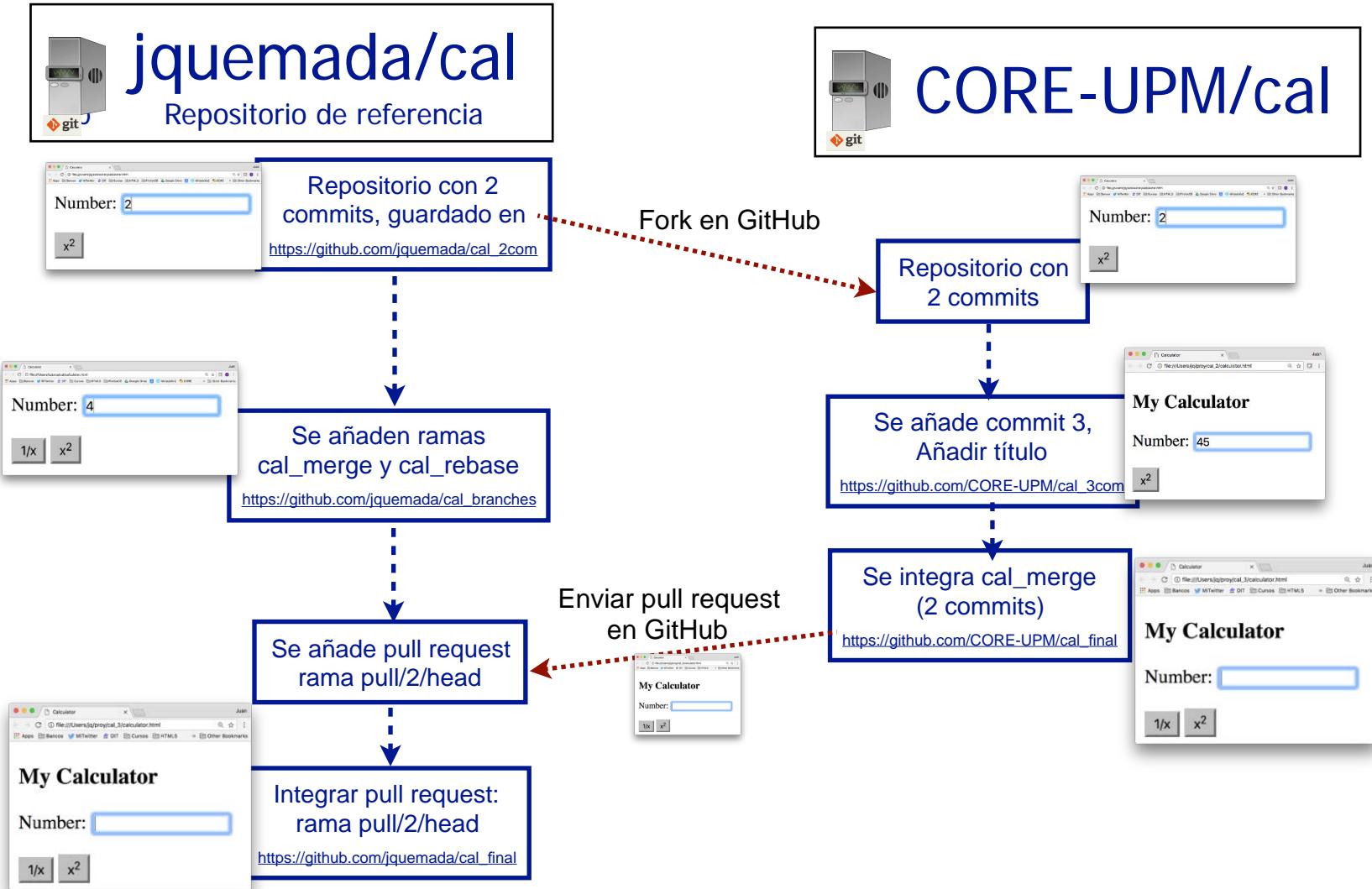
git push sincroniza la rama master (es tracking) subiendo los 2 nuevos a master en jquemada/cal

git branch -vv muestra que la rama master local se ha sincronizado con la remota.



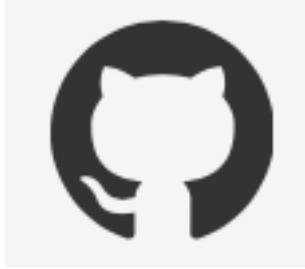
El pull request ya se ha aceptado y deja de aparecer en la lista de pull requests pendientes.

Resumen de la evolución del proyecto



Ejercicio

- ◆ Enviar la rama **master** de su repositorio **<cuenta_2>/cal_branches**
 - Como un **pull_request** a la rama **master** de **<su_cuenta>/cal_branches**
 - Identificando el pull_request con el título “Calculadora con 3 botones”
- ◆ Clonar el repositorio **<mi_cuenta>/cal_branches** a un rep. local **mi_cal_2**, con
 - **git clone https://github.com/<cuenta_2>/cal_branches mi_cal_2**
 - Entrar en el directorio **mi_cal_2** e inspeccionar con **git branch -a -vv** las ramas locales, remotas y tracking
- ◆ Integrar el pull request en master con **git pull origin pull/x/head**
 - La integración deberá ser fast-forward, porque ya a sido realizada en us otra cuenta
 - Si por alguna razón no es ff resolver los conflictos y generar el commit de integración
- ◆ Actualizar **master** en **origin** para guardar la nueva calculadora
 - **git push**
 - Recordar que **master** es **tracking** de **origin/master** y **git push** actualiza las ramas tracking
 - Inspeccionar con el navegador el repositorio actualizado
 - que contiene en master tanto el commit de integración como los commits heredados de **<su_cuenta>/cal_2com**



Git y GitHub

Guía de los principales comandos de Git

Juan Quemada, DIT - UPM

Reglas de sintaxis de comandos

Literal

◆ literal: git, clone, checkout, remove, --,....

- palabra o símbolo separado por blanco o nueva línea que debe incluirse literalmente

Elemento genérico

◆ < elemento >

- Representa un elemento (singular) o varios (plural) del tipo indicado por el nombre
 - <options> representa una o varias opciones, por ejemplo -q, -v, --all, --oneline, ...
 - <commit> representa un commit, por ejemplo d154fc4, master, master~2, ..

Elemento opcional

◆ [elemento]

- Los corchetes delimitan uno o varios elementos opcionales que podrán incluirse o no
 - [<options>] [<commit>] o [-q] indica que estos elementos son opcionales
 - [[--] <files>] los corchetes pueden anidarse indicando que todo ([[--] <files>]) o la primera parte ([--]) son opcionales

Elementos alternativos

◆ elemento1 | elemento2

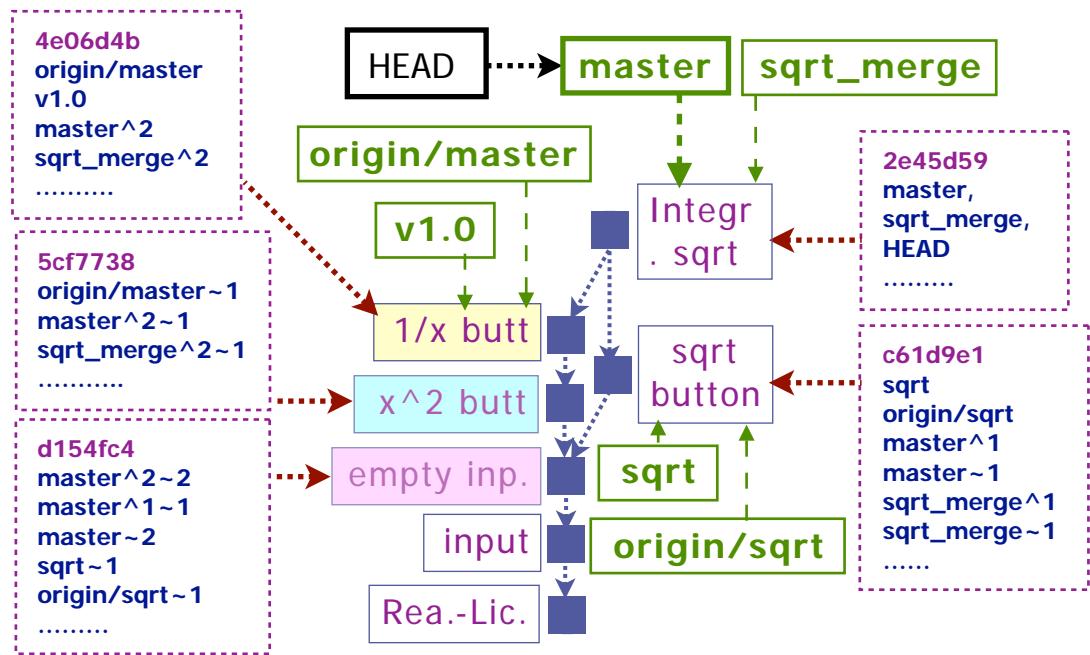
- La barra vertical separa alternativas, que irán delimitadas por corchetes o paréntesis
 - [add | rename | remove | add-url] indica opcionalidad, es decir puede incluirse uno de estos literales o ninguno
 - (-b | -B) paréntesis solo delimita, es decir se usan cuando debe incluirse necesariamente una de las opciones

Ejemplo de sintaxis de comando

◆**git diff (--cached | --staged) [--color] [<files>]**

- Muestra diferencias de código en los ficheros registrados en el índice (staged) respecto a la última versión guardada (commit)
- Algunas opciones de interés
 - **--staged, --cached** mostrar diferencias con ficheros registrados (staged)
 - **--color** mostrar diferencias en colores diferentes (opción por defecto)
- Ejemplos: mostrar cambios entre commits
 - **git diff --staged** mostrar diferencias en todos los ficheros registrados (staged)
 - **git diff --cached file.js** mostrar diferencias en el fichero file.js respecto al último commit

Grafo de commits de un repositorio y sus referencias



◆ Grafo de commits

- Relaciona cada **commit** de un repositorio con los **commits utilizados para generarla**
 - Un **commit** con una flecha saliente, solo tiene un parent porque se generó **modificando el parent**
 - Un **commit** con dos flechas salientes tiene dos padres porque se generó **integrando 2 ramas**

◆ Los commits del grafo se identifican con la siguiente notación

- **Padre o ancestro** de un commit: **commit^n** o **commit~n**
 - El primer **padre** (c^1) es el primer ancestro ($c~1$)
- **Identificador** (corto o largo) único de un commit: **d154fc4, 973751d2, ,...**
- **Rama local o remota**: **master, sqrt, .., origin/master, origin/sqrt, remote/origin/sqrt, ..**
 - El nombre de rama es un puntero al último commit de la rama local o remota
- **Tag o versión**: **v1.0, v0.1,**
- **HEAD**: puntero o referencia al commit restaurado en el directorio de trabajo

.gitignore

◆ **git ls-file --other --ignored --exclude-standard**

- lista todos los ficheros de este proyecto ignorados por Git

```
# .gitignore es un fichero que informa a Git de los ficheros que no debe gestionar.  
# - git status no los presentará como ficheros untracked.  
# - git add . no los añadira al índice (staging area).  
  
# .gitignore se crea en el directorio o subdirs de trabajo y afecta todo el arbol asociado.  
  
# Su contenido: líneas con patrones de nombres.  
# - Puede usarse los comodines * y ?  
# - Patrones terminados en / indican directorios  
# - Un patron que empiece con ! indica negación  
# - Se ignoran líneas en blanco y que comiencen con #  
# - [abc] indica cualquiera de los caracteres entre corchetes  
# - [a-z] indica cualquier carácter ASCII (rango desde a hasta z)  
  
# Ejemplo  
private.txt      # excluir los ficheros con nombre "private.txt"  
*.class          # excluir los ficheros acabados en ".class"  
*[oa]             # excluir ficheros acabados en ".o" y ".a"  
!lib.a           # no excluir el fichero "lib.a"  
*~               # excluir ficheros acabados en "~"  
testing/          # excluir directorio "testing"
```



Recetario de comandos

Clonar repositorios

◆ **git clone [<options>] <URL> [<directory>]**

- clonar un repositorio remoto en un directorio local

Mostrar la historia, commits o diferencias

◆ **git log [<options>] [<commit>] [--follow <files>]**

- Muestra la historia de commits que lleva al <commit> indicado

◆ **git log --oneline --graph --all [<options>]**

- Muestra el grafo con toda la historia de commits de un repositorio

◆ **git show [<options>] [<commit>] [[--] <files>]**

- Muestra metadatos de un commit y diferencias con el anterior

◆ **git diff [<opciones>] [<commit>] [<commit>] [[--] <files>]**

- Muestra diferencias de código en commits, ramas, directorio de trabajo, cambios

Restaurar un commit

◆ **git checkout [<options>] [<commit>]**

- Restaurar ficheros, ramas o commits en el directorio de trabajo

Crear un repositorio vacío

◆ **git init [<options>] [<repository>]**

- Crea <repository> y lo inicia como repositorio Git (crea directorio .git)

Crear commits

◆ **git add [<options>] [<files> | .]**

- Añade <files> al índice (para inclusión en nuevo commit)

◆ **git status [-s]**

- muestra el estado de los ficheros del directorio de trabajo: untracked, modified, staged, unmerged, ...

◆ **git commit [<options>] [-m <msg>]**

- Guarda lo registrado en el índice como un nuevo commit en la rama actual

Gestionar definiciones de repositorios remotos y actualizarlos

◆ **git remote [add | rename | remove | add-url | ..] [<options>]**

- gestionar definiciones de repositorios remotos (remote) preconfiguradas

◆ **git push [<options>] [<remote> [<branch>]]**

- actualizar la rama <branch> en el repositorio remoto <remote>

Rehacer la historia de commits

◆ git reset [<commit>]

- Cambia el puntero de rama y HEAD a <commit> y deja diferencias en ficheros **modified**

◆ git reset --hard [<commit>]

- Cambia el puntero de rama y HEAD a <commit> (**OJO!** commits eliminados se pierden)

Diferencias en el directorio de trabajo

◆ git diff [[--] <files>]

- Muestra diferencias en <files> **modified** respecto a la versión del commit anterior

◆ git diff (--cached | --staged) [[--] <files>]

- Muestra diferencias en <files> **staged** respecto a la versión del commit anterior

Cambios en el directorio de trabajo

◆ git mv <old> <new>

- Cambia nombre de un fichero en el directorio de trabajo (y en el índice)

◆ git rm [<files>]

- Borra <files> del directorio de trabajo y registra el borrado en el índice

◆ git rm (--cached | --staged) [<files>]

- Borra <files> del índice, los ficheros pasan de **staged** a **untracked**

◆ git reset [<ficheros>] (Deshace git add ..)

- Elimina <ficheros> del índice dejando los cambios. Los ficheros pasan de **staged** a **modified**.

◆ git checkout [<files> | .]

- Elimina cambios en ficheros **modified** que pasan a **unmodified** (**OJO!** Cambios se pierden)

Listar ramas

◆ **git branch [-v] [-vv] [-a] [-r]**

- Lista las ramas locales y remotas de un repositorio

Crear ramas locales

◆ **git branch [-t | --track] <branch> <commit>**

- Crear una rama de nombre <branch> que comienza en <commit>

◆ **git checkout [-f | ..] (-b | -B) <branch> [<commit>]**

- crea (o reinicia) <branch> en <commit>, que se restaura en el directorio de trabajo

Borrar ramas

◆ **git branch (-d | -D) <branch>**

- Borra la rama <branch> si se ha integrado ya (-d) o aunque no se haya integrado (-D)

Inspeccionar grafo del repositorio

◆ **git log --oneline --graph --all**

- Muestra el grafo con toda la historia de commits de un repositorio

Rehacer la historia de commits

◆ **git reset [<commit>]**

- Deshace commits posteriores a <commit> dejando cambios en ficheros que pasan a **modified**

◆ **git reset --hard [<commit>]**

- Elimina commits posteriores a <commit> (**OJO!** commits eliminados y sus cambios se pierden)

Integrar ramas locales

◆ **git merge [<options>] [-m <msg>] <commit>**

- Integra rama <branch> definida en repositorio local en rama actual

Cambiar la base de una rama

◆ **git rebase [<options>] [-m <msg>] <commit>**

- Cambiar el commit de comienzo (base) de una rama (rehaciendo commits)

Sincronización con repositorios remotos

◆ **git fetch [-f] [--all] [<options>] [<remote> [[+]<rbranch>[:<lbranch>]]]**

- Actualizar o copiar nuevas ramas de un repositorio remoto en el repositorio local

◆ **git pull [<options>] [<remote> [[+]<rbranch>[:<lbranch>]]]**

- Similar a realizar **git fetch ...** seguido de **git merge ...** con los mismos parámetros

◆ **git pull --rebase [<options>] [<remote> [[+]<rbranch>[:<lbranch>]]]**

- Similar a realizar **git fetch ...** seguido de **git rebase ...** con los mismos parámetros

◆ **git push [-f] [--all] [<options>] [<remote> [[+]<rbranch>[:<lbranch>]]]**

- Actualizar repositorio remoto y objetos asociados

Índice de comandos Git más importantes

◆ Índice de los principales comandos Git

- add 223
- branch 224
- checkout 225
- clone 227
- commit 228
- diff 229
- fetch 231
- init 232
- log 233
- merge 235
- mv 236
- pull 237
- push 238
- rebase 239
- remote 240
- reset 241
- rm 242
- show 243
- stash (new) 244
- status 245
- tag (new) 247

add

◆git add [<options>] [<files> | .]

- Añade <files> al índice (para inclusión en nuevo commit)
 - Ojo! Ficheros **modificados** pero no **añadidos** al índice con **add** no irán en la versión
- Algunas opciones de interés
 - **-u** añade todos los ficheros **modified**, pero no los **untracked**
 - **-v** modo verbose (--verbose)
 - **-n** no añade ficheros, solo muestra si existen o serán ignorados (--dry-run)
 - **-i** preguntar en cada caso (--interactive)
 - **-e** abrir editor (vi) con diffs respecto a índice para re-editar cambios (--edit)
- Ejemplos:
 - **git add -u** añade todos los ficheros modificados al índice
 - **git add .** añade todos los ficheros del directorio de trabajo(.) al índice
 - **git add -i .** intenta añadir todos los ficheros al índice, preguntando antes
 - **git add file_1 file_2** añade file_1 y file_2 al índice

branch

◆ **git branch [<options>] [<branch>] [<commit>]**

- Listar, crear o borrar ramas
- Algunas opciones de interés
 - **-d, -D** borrar la rama, si está integrada (-d) o aunque no se haya integrado (-D)
 - **-f** forzar operación indicada aunque se pierdan cambios (—force)
 - **--track** crear rama tracking cuando va acompañado de -b
 - **-a, -r** listar ramas: todas (-a) o solo tracking remotas (-r)
 - **-v, -vv** modo verboso, muestra último commit (-v) y ramas tracking (-vv)
 - **--merged, --no-merged** muestra solo ramas integradas o no integradas
- Ejemplos: listar ramas
 - **git branch** listar las ramas locales del repositorio
 - **git branch -a -v** listar las ramas locales y remotas del repositorio en formato extendido
 - **git branch --merged** listar las ramas locales del repositorio ya integradas en otras
 - **git branch -vv** listar las ramas **tracking** con la rama remota asociada y su estado
- Ejemplos: crear, borrar o reconfigurar ramas locales
 - **git branch sqrt** crear rama sqrt en commit actual (HEAD)
 - **git branch sqrt d154fc4** crear rama sqrt en commit d154fc4
 - **git branch -d sqrt** borra la rama sqrt del repositorio
 - **git branch -f sqrt** reinicia rama sqrt borrando cambios y commits
 - **git branch --track r1/square** transforma la rama local **square**, ya existente, en tracking de **r1/square**
 - **git branch --track s2 r1/s1** transforma la rama local s2, ya existente, en tracking de r1/s1

checkout | (restaurar commit o fichero)

◆git checkout [<options>] [<commit>]

- Restaurar ficheros, ramas o commits en el directorio de trabajo
 - Actualiza puntero **HEAD** al commit restaurado en el directorio de trabajo
- Algunas opciones de interés
 - **-q** no genera estadísticas de salida (—quiet)
 - **-f** forzar checkout aunque se pierdan cambios (--force)
- Ejemplos: cambio de rama
 - **git checkout master** restaura en el directorio de trabajo el último commit de master
 - **git checkout 4e06d4b** restaura en el directorio de trabajo el commit **4e06d4b**
 - **OJO! Peligro** modo detached HEAD, el commit activo no es una rama y los cambios a este commit no podrán guardarse a no ser que se cree una rama nueva

◆git checkout [<files> | .]

- Restaurar ficheros **modified** del directorio de trabajo al commit anterior
 - **!OJO** los cambios se perderán y no podrán volver a recuperarse
- Ejemplos: restaurar ficheros modified
 - **git checkout file1.js** elimina cambios en el fichero **modified file.js**
 - **git checkout .** elimina cambios en todos los ficheros **modified** del directorio de trabajo

checkout II (restaurar y crear rama)

◆ **git checkout [<options>] (-b | -B) <branch> [<commit>]**

- Crear rama y restaurar ficheros, ramas o commits de la rama
 - Actualiza puntero **HEAD** al commit que restaura en el directorio de trabajo
- Algunas opciones de interés
 - **-q** no genera estadísticas de salida (--quiet)
 - **-f** forzar checkout aunque se pierdan cambios (--force)
 - **-m** fuerza un merge con <commit> y con cambios en el área de trabajo
 - **-b <new_branch>** crea rama de nombre <new_branch> en el commit restaurado (HEAD)
 - **-B <new_branch>** crea rama de nombre <new_branch> en el commit restaurado (HEAD) y si existe la reinicia (borra los commits de la rama) hasta <commit>
- Ejemplos: creación de una rama en un commit determinado
 - **git checkout -b sqrt 4e06d4b** crea rama **sqrt** en 4e06d4b y la restaura en directorio de trabajo
 - **git checkout -B sqrt 4e06d4b** restaura rama **sqrt** en **4e06d4b** y borra sus commits si existen

◆ **git checkout [<branch>]**

- Copia una rama remota en una local tracking y restaura la rama local (solo si no existe)
 - Actualiza puntero **HEAD** al commit restaurado en el directorio de trabajo
- Ejemplos: de copia de rama remota
 - **git checkout square** crea y restaura la rama tracking square asociada a <remote>/square

clone

◆git clone [<options>] <URL> [<directory>]

- Copiar un repositorio identificado por <URL> a un directorio local:
 - Define el repositorio remoto como repositorio remote origin
 - Copia la rama master como rama tracking local de origin/master
 - Define además todas las ramas de origin como ramas remotas
- Algunas opciones de interés
 - -v modo verbose (--verbose)
 - -q no genera estadísticas de salida (--quiet)
 - --no-hardlinks fuerza una copia física de todos los ficheros cuando se copia repositorio local
- Ejemplos: clonar ficheros
 - `git clone https://github.com/CORE-UPM/cal_5com` clona en directorio local `cal_5com`
 - `git clone https://github.com/CORE-UPM/cal_5com cal` clona en directorio local `cal`

commit

◆**git commit [<options>] [-m <msg>]**

- Guarda lo registrado en el índice como un nuevo commit en la rama actual
- Algunas opciones de interés
 - **-m <msg>** incluir el mensaje <msg> que identifica la versión
 - **-q** no genera estadísticas de salida (quiet)
 - **-a** añade al índice todos los ficheros modificados antes de generar commit
 - **OJO!** No añade ficheros untracked
 - **--amend** rehace el commit anterior con lo indicado en el índice
 - **OJO!** Corrige errores. Cambia identificador de versión y será incompatible con anterior
- Ejemplos: crear nuevo commit
 - **git commit -m "hola"** crea commit en rama actual con cambios registrados y mensaje "hola"
 - **git commit -a -m "hola"** añade todos los ficheros modificados al índice y crea commit
- Ejemplos: modificar commits existentes (**OJO!** Cambia el id del commit)
 - **git commit --amend** rehace último commit en rama actual con cambios registrados en índice
 - **git commit --amend -m "hola que tal"** rehace último commit con cambios registrados y en la rama activa con los cambios registrados en el índice y con mensaje "hola"

diff | (diferencias entre commits)

◆**git diff [<opciones>] [<commit>] [<commit>] [[--] <files>]**

- Muestra diferencias de código entre commits, ramas, ficheros, etc
- Algunas opciones de interés
 - **--color** mostrar diferencias en colores diferentes (opción por defecto)
- Ejemplos: mostrar cambios entre commits
 - **git diff HEAD d154fc4** diferencias entre commits HEAD y d154fc4
 - **git diff master master~2** diferencias entre último commit de master y dos anteriores
 - **git diff sqrt master** diferencias entre las ramas master y sqrt
 - **git diff sqrt master file.js** diferencias en file.js entre las ramas master y sqrt

diff II (diferencias en el directorio de trabajo)

◆**git diff [[--] <files>]**

- Diferencias en ficheros (<files>) **modified** respecto al commit anterior
- Ejemplos: mostrar cambios de ficheros **modified**
 - **git diff** mostrar diffs de todos ficheros modified respecto al commit anterior
 - **git diff file.js** mostrar diffs de fichero modified file.js respecto al commit anterior

◆**git diff (--cached | --staged) [[--] <files>]**

- Diferencias en ficheros (<files>) **staged** respecto al commit anterior
- Ejemplos: mostrar cambios de ficheros staged
 - **git diff --cached** mostrar diffs de todos ficheros staged respecto al commit anterior
 - **git diff --cached file.js** mostrar diffs de fichero staged file.js respecto al commit anterior

fetch

◆ **git fetch [<options>] [<repository> [[+<local_branch>[:<remote_branch>]]]**

- actualizar en el repositorio local las ramas indicadas de un repositorio remoto
 - [<remote> [[+<rbranch>[:<lbranch>]]] actualiza la rama local (lbranch) con la remota (rbranch)
 - + fuerza actualización aunque haya commits incompatibles (no-ff) en la rama local
- Algunas opciones de interés
 - **--all** actualizar todas las ramas de todos los repositorios remotos definidos
 - **-p** eliminar ramas que no ya existan en el remoto (--prune)
 - **-v, -q** modos verbose (--verbose) y sin realimentación (--quiet)
- Ejemplos: actualización de ramas de repositorios remotos definidos
 - **git fetch cal** crea las ramas remotas del remote cal_branches o actualiza su estado si existen
 - **git fetch** actualiza el estado de todas las ramas tracking
 - **git fetch --all** crea o actualiza el estado de todas las ramas de todos los remotes definidos
 - **git fetch -p origin** la opción -p (--prune) actualiza las ramas de origin eliminando las que ya no existen
- Ejemplos: actualización de ramas con refsspecs: [[+<rbranch>[:<lbranch>]]]
 - **git fetch origin square:sqrt**
 - Crea o actualiza la rama local sqrt con los commits de la remota origin/square
 - **git fetch origin pull/1/head:s1**
 - Crea o actualiza la rama local s1 con el pull_request 1 del repositorio remoto origin en GitHub
 - **git fetch cal_branches +s1:s1**
 - Crea o actualiza la rama local s1 con la remota cal_branches/s1 aunque sean incompatibles
 - **git fetch https://github.com/jquemada/cal square:square**
 - Crea o actualiza la rama local square con la rem. square de <https://github.com/jquemada/cal>

init

◆**git init [<options>] [<repository>]**

- Crea <repository> y lo inicia como repositorio Git (crea directorio `.git`)
 - A partir de este momento ya se pueden invocar todos comandos Git en el directorio
- Algunas opciones de interés
 - `-q` no genera estadísticas de salida (`--quiet`)
 - `--bare` Crea repositorio bare, sin directorio de trabajo
- Ejemplos:
 - **git init** Inicia el directorio donde se invoca como repositorio Git
 - El repositorio debe ser el directorio de trabajo del terminal de comandos
 - **git init proy1** Crea o inicializa el repositorio en el directorio **proy1**
 - Crea el directorio proy1 (`mkdir proy1`), si no existe, e invoca en él el comando **git init**

log | (historia de un commit)

◆**git log [<options>] [<commit>] [--follow <file>]**

- Muestra la historia de commits que lleva al <commit> indicado
- Algunas opciones de interés
 - **--oneline**: muestra metadatos resumidos en una linea solo con id_corto y mensaje
 - **-n**: muestra n últimos commits
 - **-p**: muestra commits con diferencias (diff) con versión anterior
 - **--stat**: muestra también estadísticas del commit
 - **--since=2.weeks o --until=2.weeks**: muestra commits después o hasta hace 2 semanas
 - **--follow <file>** muestra la historia de un fichero particular, incluyendo los cambios de nombre
- Ejemplos: historia de commits
 - **git log** muestra todos los commits hasta HEAD (HEAD es la opción por defecto y se omite)
 - **git log -3 HEAD** muestra los 3 últimos commits hasta HEAD (HEAD se deja aquí).
 - **git log --since=2.weeks --until=1.week** muestra commits de las penúltima semana.
 - **git log --oneline** muestra commits resumidos en una linea (incluye id_corto y mensaje).
 - **git log master** historia de commits de la rama master (desde cualquier posición).
 - **git log --oneline -2 d154fc4**: muestra 2 últimos commit resumidos hasta d154fc4.
 - **git log -3 master~2:** historia de 3 commits desde segundo commit anterior a master.
- Ejemplos: historia de ficheros
 - **git log --follow calculator.htm**: historia de versiones del fichero calculator.htm desde HEAD.
 - **git log sqrt --follow .gitignore**: historia del fichero .gitignore en la rama sqrt.

log II (grafo de commits)

◆ **git log [<options>] [<commit>] [--follow <files>]**

- Muestra la historia de commits que lleva al <commit> indicado
- Algunas opciones de interés
 - **--oneline**: muestra metadatos resumidos en una linea solo con id_corto y mensaje
 - **-n:** muestra n últimos commits
 - **--all** muestra la historia de todas las ramas (debe omitirse <commit>)
 - **--graph** muestra grafo de ramas integradas en la rama actual
- Ejemplos: historia desde commits
 - **git log --oneline --graph** muestra el grafo de integración de la rama actual (formato corto)
 - **git log --oneline --graph sqrt** muestra el grafo de integración de la rama sqrt (formato corto)
 - **git log --oneline --graph --all** muestra el grafo completo de commits (formato corto)
 - **git log --oneline --all -2** muestra los 2 últimos commits del repositorio (formato corto)
 - **git log --oneline --all --graph -3** muestra grafo de commits con profundidad 3 (f. c.)

merge

◆**git merge [<options>] [-m <msg>] <commit>**

- Integra el commit indicado en HEAD y genera un commit de integración con “msg”
 - Si la integración no tiene conflictos, genera automáticamente un commit por auto-merge
 - Si HEAD es un ancestro del commit a integrar pasa al commit a integrar con ff (fast-forward)
 - Si hay conflictos, git los incluye como modificaciones de los ficheros afectados
 - Los conflictos se resuelven y después se genera un commit de integración con **git commit ...**
- Algunas opciones de interés
 - **-m <msg>** fijar el mensaje del commit de integración
 - **--abort** aborta la operación de merge y trata de reconstruir el estado anterior
 - **-v, -q** modos verbose (--verbose) y sin realimentación (--quiet)
- Ejemplos:
 - **git merge -m “msg” master** integra master en HEAD y genera commit de integr. con “msg”
 - **git merge master** integrar la rama master con el desarrollo o rama actual (HEAD)
 - Pedira editar el mensaje con el editor por defecto para el commit de integración
 - **git merge 4e06d4b** integrar el commit 4e06d4b con el desarrollo o rama actual (HEAD)
 - **git merge --abort** aborta la operación de merge en curso, trata de reconstruir estado anterior

Documentación completa: <https://git-scm.com/docs>.

Merging vs. Rebasing: <https://www.atlassian.com/git/tutorials/merging-vs-rebasing>

Merge: <https://git-scm.com/docs/git-merge>

Rebase: <https://git-scm.com/docs/git-rebase> © Juan Quemada, DIT, UPM

mv

◆git mv [<options>] <origin> <destination>

- Mover o renombrar un fichero, directorio o enlace eliminando del índice
 - OJO! El comando **mv** de **UNIX** crea un fichero untracked, pero el original sigue en el índice
- Algunas opciones de interés
 - **-f** forzar mover o renombrar aunque fichero destino exista (**--force**)
 - **-v** modo verboso (**--verbose**)
- Ejemplos: cambios de ficheros (eliminando el original del índice)
 - **git mv file1.js file2.js** cambia el nombre de file1.js a file2.js
 - **git mv file1.js dir1/file2.js** mueve file1.js al directorio dir1 con nombre file2.js
 - **git mv -f file1.js dir1** mueve file1.js al directorio dir1, aunque ya exista dir1/file1.js

pull

◆ **git pull [<options>] [<repository> [[+]<local_branch>[:<remote_branch>]]]**

- traer la rama remota indicada e integrarla con una rama del repositorio local
 - Equivale normalmente a hacer primero **git fetch ..** y a continuación **git merge ..**
- Algunas opciones de interés
 - **-p** eliminar ramas tracking que no existan ya en el remoto (**--prune**)
 - **-r** realizar un rebase en vez de merge (**--rebase**)
 - **-v, -q** modos verbose (**--verbose**) y sin realimentación (**--quiet**)
- Ejemplos: integración de ramas remotas con ramas locales
 - **git pull cal_branches square** integra la rama square de cal_branches en la rama activa
 - **git pull https://github.com/jquemada/cal_branches square**
 - integra la rama square de jquemada/cal_branches en la rama activa
 - **git pull origin pull/1/head** Integra el pull_reques #1 en la rama activa

push

◆ git push [<options>] [<repository> [[+<local_branch>[:<remote_branch>]]]]

- actualizar repositorio remoto y objetos asociados con las ramas locales indicadas
 - [[+<local_branch>[:<remote_branch>]]] actualiza la rama remota indicada con los commits de la local
 - + fuerza actualización aunque haya commits incompatibles (no-ff) en la rama local (equivale a -f)
- Algunas opciones de interés
 - **--all** actualizar todas las ramas del repositorio remoto
 - **-f** forzar actualización si hay commits incompatibles (--force) **OJO!** Los commits incompatibles se pierden
 - **-v, -q** modos verbose (--verbose) y sin realimentación (--quiet)
- Ejemplos: actualización de ramas de un repositorio remoto
 - **git push**
 - Actualiza las ramas remotas de las ramas locales **tracking** definidas en el repositorio local
 - **git push cal_branches sqrt:square**
 - Actualiza la rama remota **cal_branches/square** con los nuevos commits de la local **sqrt**
 - **git push https://github.com/jquemada/cal sqrt:square**
 - Actualiza la rama remota **cal/square** con los nuevos commits de la local **sqrt**
- Ejemplos: sustitución de ramas de un repositorio remoto
 - **git push -f origin sqrt** sustituye rama **sqrt** en repositorio **origin** por la rama local, aunque haya commits incompatibles (**OJO! Peligroso:** Los commits borrados no podrán ser recuperados)
- Ejemplos: eliminación de ramas de un repositorio remoto
 - **git push origin :sqrt** Borra la rama **sqrt** en el repositorio remoto **origin**
 - **git push origin --delete sqrt** Similar a anterior, opción posible en versiones recientes de git

rebase

◆git rebase [<options>] [-m <msg>] <commit>

- cambiar el commit de comienzo (base) de una rama (rehaciendo commits)
 - Rebase rehace todos los commits de la rama utilizando un bucle, donde en cada iteración se integra un commit de la rama con el anterior ya integrado
 - La integración es similar a la realizada con **git merge ..**, pudiendo haber auto-merge, fast-forward, ..
 - Si hay conflictos, Git los marca y para el proceso (bucle). Los conflictos deben integrarse con un editor y el proceso (bucle) debe continuarse con **git rebase --continue | --skip | ..** para procesar el siguiente commit. Y así hasta integrar todos los commits.
- Algunas opciones de interés
 - **--continue** continuar el bucle de rebase después de editar conflictos de integración de commit
 - **--skip** continuar el bucle de rebase integrando commit actual con siguiente
 - **--abort** abortar operación de rebase y volver a estado inicial
 - **-i** rebase interactivo para reorganizar los commits de la rama
- Ejemplos:
 - **git rebase master** cambiar comienzo de rama actual (HEAD) a master
 - **git rebase 4e06d4b** cambiar comienzo de rama actual (HEAD) a commit 4e06d4b
 - **git rebase --continue** continuar rebase después de editar conflictos de integración
 - **git rebase --skip** continuar rebase integrando commit actual con siguiente

Documentación completa: <https://git-scm.com/docs>.

Merging vs. Rebasing: <https://www.atlassian.com/git/tutorials/merging-vs-rebasing>

Merge: <https://git-scm.com/docs/git-merge>

Rebase: <https://git-scm.com/docs/git-rebase> © Juan Quemada, DIT, UPM

remote

◆git remote [-v]

- gestionar repositorios remotos (remote) relacionados, por ejemplo
 - **git remote** muestra repositorios remotos (remotes) definidos
 - **git status -v** muestra enlace de repositorios remotos (remotes) definidos (--verbose)

◆git remote add <name> <URL>

- Crea la referencia <remote_name> a repositorio remoto identificado por un **URL**, p.e.
 - **git remote add cal_inic https://github.com/jquemada/cal_inic**

◆git remote remove <name>

- Elimina la definición del repositorio <remote_name>, por ejemplo
 - **git remote remove cal_inic**

◆git remote rename <old_name> <new_name>

- cambia el nombre de <remote_name1> por <remote_name2>, por ejemplo
 - **git remote rename cal_inic cal_initialize**

◆git remote set-url <name> <URL>

- Cambia el URL del repositorio <remote_name>, por ejemplo
 - **git remote rename cal_initialize https://github.com/jquemada/cal_initialize**

◆git remote set-branches [--add] <name> <branches>

- Sustituye o añade (opción --add) tracking branches de <remote>, por ejemplo
 - **git remote set-branches origin master** sustituye tracking branches actuales por master
 - **git remote set-branches --add origin sqrt** añade sqrt a tracking branches

reset

◆ git reset [-q] [<files>]

- Operación inversa a git add .., ficheros (<files>) dejan de estar **staged**
- Ejemplo: borrar ficheros del índice (de **staged** a **modified** o **untracked**)
 - `git reset file1.js` extrae file1.js del índice, preserva los cambios (inverso `git add <files>`)
 - `git reset` extrae todos los ficheros del índice, preserva cambios (inverso `git add .`)

◆ git reset [-q] <commit>

- Cambia el puntero de rama y HEAD a <commit> y deja diferencias en ficheros
- Ejemplos:
 - `git reset 4e06d4b` resetea la rama actual a **4e06d4b** eliminando los commits posteriores y preservando cambios acumulados en el directorio de trabajo
 - `git reset --mixed 4e06d4b` equivalente a ejemplo anterior, --mixed es la opción por defecto

◆ git reset --hard [-q] <commit>

- Cambia el puntero de rama y HEAD a <commit>
- Ejemplo: (**OJO!** commits se pierden y no se pueden recuperar)
 - `git reset --hard 4e06d4b` resetea la rama actual a **4e06d4b** eliminando los commits posteriores sin preservar cambios (**OJO!** Todos los commits eliminados se pierden)

◆git rm <files>

- borra ficheros del directorio de trabajo y del índice
 - **OJO!** El comando `rm` de **UNIX** no debe utilizarse porque borra un fichero del directorio, pero el fichero sigue en el índice y en los commits siguientes
- Algunas opciones de interés
 - `-f` forzar borrado si fichero es modified (`--force`)
 - `-n` simula ejecución sin hacer cambios (`--dry-run`)
- Ejemplos de borrado de ficheros del directorio de trabajo
 - `git rm file1.js file2.js` borra file1.js y file2.js del directorio de trabajo y del índice
 - `git rm -f file1.js` borra file1.js del directorio de trabajo y del índice, aunque sea modified
 - `git rm -n f.*` muestra que ficheros borraría el comando, sin borrar dichos ficheros

◆git rm (--cached | --staged) <files>

- borra los ficheros (<files>) del índice, pasándolos de **staged** a **untracked**
- Ejemplos:
 - `git rm --cached file1.js` borra file1.js del índice, pasando de **staged** a **untracked**.

show

◆ **git show [<options>] [<commit>] [[--] <file>]**

- Muestra metadatos de un commit y diferencias con el anterior
- Algunas opciones de interés
 - **--oneline**: muestra metadatos resumidos en una linea solo con id_corto y mensaje
- Ejemplos:
 - **git show** muestra metadatos de HEAD y diferencias con el commit anterior
 - **git show master** muestra metadatos del último commit de master y diferencias con anterior
 - **git show HEAD~1** muestra metadatos de commit anterior a HEAD y diferencias
 - **git show d154fc4** muestra metadatos de **d154fc4** y diferencias con el commit anterior
 - **git show HEAD file1.js** muestra metadatos HEAD y diferencias de file.js con commit anterior

stash

◆git stash

- Guarda en una pila las modificaciones del directorio de trabajo y en el índice,
- deja restaurados el directorio de trabajo y el índice, por ejemplo
 - `git stash` guarda las modificaciones un la pila

◆git stash list

- Lista el contenido de la pila de stashed p.e.
 - `git stash`

◆git stash apply [<name>] [<options>]

- Aplica los cambios del último stash guardado, o los del stash llamado **name**,
- a los ficheros del area de trabajo,
- y no actualiza el índice, excepto si se usa la opción `—index`,
- y no elimina el stash aplicado de la pila. Por ejemplo:
 - `git stash apply`

◆git stash drop [<name>]

- Elimina el último stash de la pila (o el indicado por **name**), por ejemplo
 - `git stash drop`

◆git stash pop [<name>]

- Aplica el último stash (o el indicado por **name**) y lo elimina de la pila, por ejemplo
 - `git stash pop`

status

◆**git status [-s]**

- muestra el estado del directorio de trabajo
 - los ficheros pueden estar **untracked**, **modified**, **staged**, **unmerged**, ...
- Algunas opciones de interés
 - **-s**: muestra estado en formato corto (1 linea por fichero)
- Ejemplos de uso
 - **git status** muestra estado de cambios del directorio de trabajo
 - **git status -s** muestra estado de cambios del directorio de trabajo en formato corto

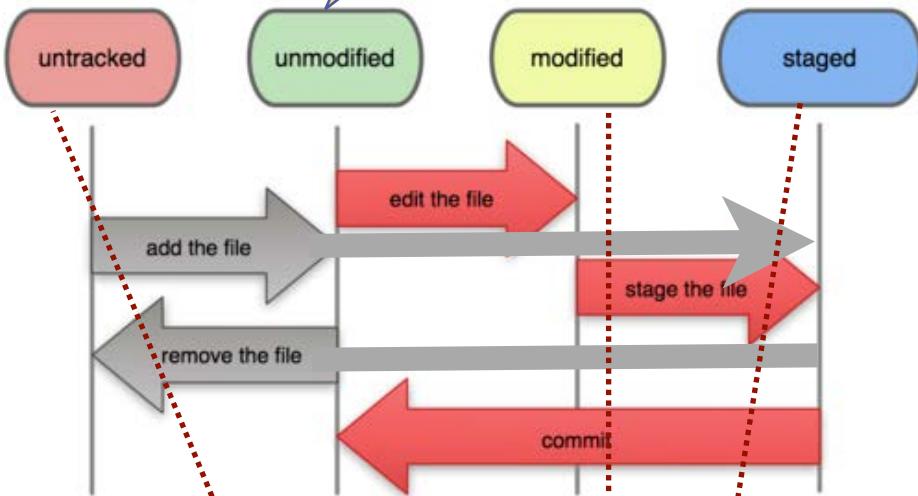
“git status”

Ficheros de la versión anterior no modificados para la siguiente

git status muestra el estado del índice o área de cambios:

- ficheros staged (en la versión)
- ficheros modified (fuera de versión)
- ficheros untracked (fuera de versión)

opción -s: salida corta (short)



```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified: README
#       new file: CharIO.java
#
```

Ficheros modificados incluidos en próxima versión

```
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#
#       modified: benchmarks.rb
#
```

Ficheros modificados no incluidos en próxima versión

```
# Untracked files:
#   (use "git add <file>..." to include what will be committed)
#
#       merge.java
#       library/lib.js
```

Ficheros excluidos de versión

tag

◆ **git tag [-a] [<options>] <name> [<commit>]**

- Crea un tag con el nombre <name> en el commit <commit>, o en el commit actual,
- de tipo ligero si no se usa la opción **-a** (para uso temporal),
- o de tipo anotado si se usa la opción **-a** (se crea un commit nuevo).
- Por ejemplo, crear un tag anotado llamado v1.4 con un mensaje para el commit:
 - `git tag -a v1.4 -m "Version 1.4"`

◆ **git tag [-l <pattern>]**

- Lista todos los tags existentes,
- o los que encajan con el patrón dado con la opción **-l**, por ejemplo
 - `git tag` lista todos los tags existentes.
 - `git tag -l v1.*` lista los tags que encajan con el patrón v1.*



Final del tema