

# Introducción al control de versiones y trabajo colaborativo con GitHub

Oscar Perpiñán Lamigueiro

# AIGORA

Lo expuesto en este documento se enmarca en el proyecto de innovación educativa AIGORA (Aprendizaje de Informática con Github Organizado en Repositorios Abiertos).



<https://innovacioneducativa.upm.es/proyectosIE/informacion?anyo=2018-2019&id=2840>

- 1 Conceptos básicos
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub

- 1 Conceptos básicos
  - ¿Qué es el control de versiones?
  - ¿Qué son Git y GitHub?
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub

# "FINAL".doc



FINAL.doc!



FINAL\_rev.2.doc

<http://phdcomics.com/comics/archive.php?comid=1531>



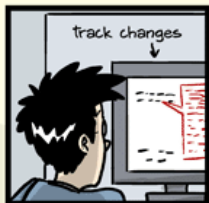
FINAL\_rev.6.COMMENTS.doc



FINAL\_rev.8.comments5.  
CORRECTIONS.doc



<http://phdcomics.com/comics/archive.php?comid=1531>



FINAL\_rev.18.comments7.  
corrections9.MORE.30.doc

FINAL\_rev.22.comments49.  
corrections.10.#@\$%WHYDID  
ICOMETOGRADSCHOOL?????.doc

<http://phdcomics.com/comics/archive.php?comiciid=1531>

# ¿Qué es el control de versiones y por qué debería importarte?

*El control de versiones es un sistema que **registra los cambios** realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se puedan **recuperar** versiones específicas más adelante.*<sup>1</sup>

## Viajar en el tiempo

- Nada que haya sido sometido a un control de versiones se pierde jamás (*salvo que realmente quieras eliminarlo...*)
- **Todas** las versiones antiguas de un fichero se almacenan: un fichero se puede revertir a un estado anterior sin límites.

---

<sup>1</sup>[https:](https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones)

[//git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones](https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones)



# ¿Qué es el control de versiones y por qué debería importarte?

*El control de versiones es el **cuaderno de laboratorio** en el mundo digital. Es lo que los profesionales usan para realizar un **seguimiento** de lo que han hecho y para **colaborar** con otras personas.<sup>2</sup>*

## ¿Qué? ¿Cuándo? ¿Quién?

Un sistema de control de versiones registra:

- El detalle de los cambios realizados.
- La fecha y hora en la que fueron realizados.
- La persona que los realizó.

---

<sup>2</sup><https://swcarpentry.github.io/git-novice/>

# ¿Qué es el control de versiones y por qué debería importarte?

*El control de versiones es el **cuaderno de laboratorio** en el mundo digital. Es lo que los profesionales usan para realizar un **seguimiento** de lo que han hecho y para **colaborar** con otras personas.<sup>2</sup>*

## Trabajo Colaborativo

- Cuando un equipo de personas trabaja conjuntamente en un proyecto, es posible que se produzcan cambios incompatibles en un mismo fichero.
- El sistema de control de versiones **impide** cambios simultáneos en un fichero. A cambio, permite la **resolución de conflictos** y los documenta.

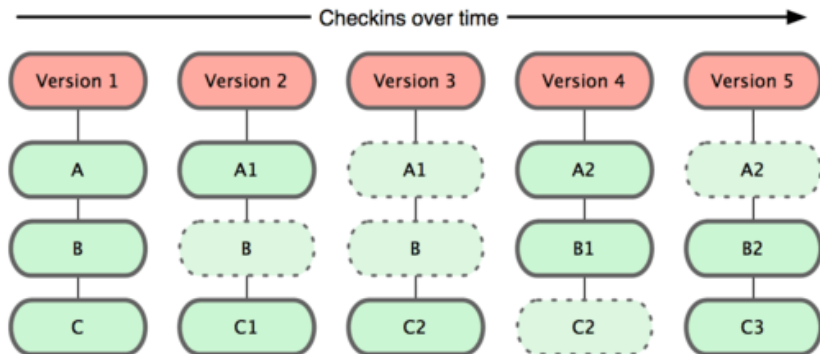
# ¿Qué es el control de versiones y por qué debería importarte?

*No sirve sólo para software: libros, documentos, pequeños conjuntos de datos y cualquier cosa que cambie con el tiempo o que deba compartirse puede y debe almacenarse en un sistema de control de versiones.<sup>2</sup>*

- 1 Conceptos básicos
  - ¿Qué es el control de versiones?
  - ¿Qué son Git y GitHub?
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub

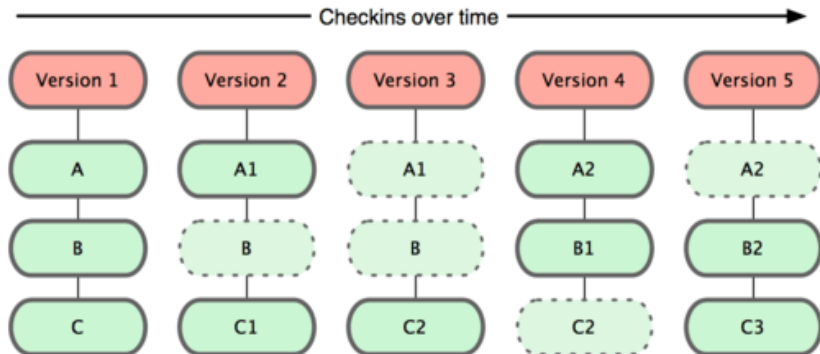
# Git es un Sistema de Control de Versiones

Git es una herramienta software (accesible mediante línea de comandos con git) que implementa un Sistema de Control de Versiones.



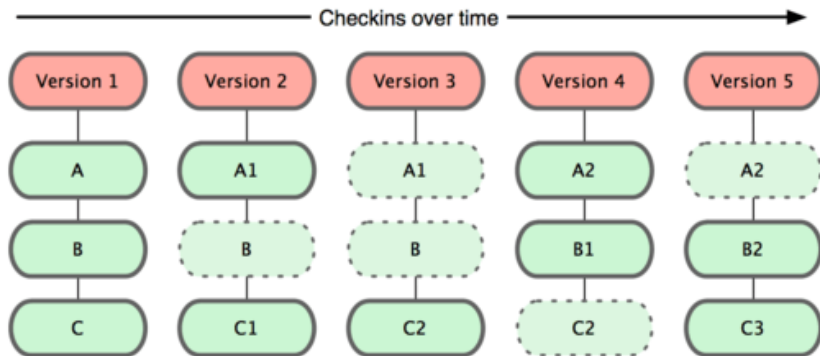
# Git es un Sistema de Control de Versiones

Cada vez que se ejecuta un cambio en una estructura de ficheros controlada con Git, realiza una «foto» del estado de los archivos en ese momento, y guarda una referencia a esa instantánea.



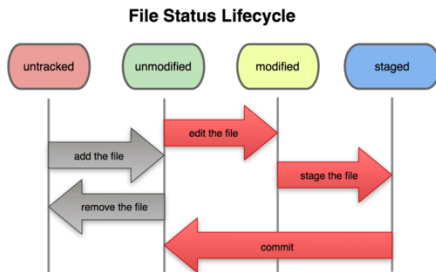
# Git es un Sistema de Control de Versiones

Por eficiencia, Git no almacena los archivos sin modificaciones sino un enlace al archivo anterior idéntico que ya está almacenado



# Los estados de Git

- El desarrollador incorpora uno o varios ficheros al control de versiones. (*tracked*)
- Realiza modificaciones en los ficheros (*modified*).
- Incorpora esos ficheros modificados al área de preparación (*staged*).
- Finalmente, confirma todos los cambios del área de preparación: se realiza la instantánea de los ficheros. (*committed*)





# ¿Qué es GitHub?

- GitHub es la plataforma de alojamiento de código más importante a nivel mundial.
- Emplea el sistema de control de versiones `git`
- Ofrece una amplia variedad de funcionalidades
  - ▶ Alojamiento de código
  - ▶ Revisión de código
  - ▶ Trabajo colaborativo
  - ▶ Publicación de páginas web

- 1 Conceptos básicos
- 2 **Uso de git y GitHub**
- 3 Trabajo en colaboración
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub


- 1 Conceptos básicos
- 2 Uso de git y GitHub
  - Primeros Pasos
  - Flujo de Trabajo
- 3 Trabajo en colaboración
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub


# Creación de una cuenta en GitHub


<https://github.com/join>

## Join GitHub

The best way to design, build, and ship software.

 **Step 1:**  
Set up your account

 **Step 2:**  
Choose your plan

 **Step 3:**  
Tailor your experience

### Create your personal account

**Username \***

This will be your username. You can add the name of your organization later.

**Email address \***

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

**Password \***

Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

[Verify account](#)

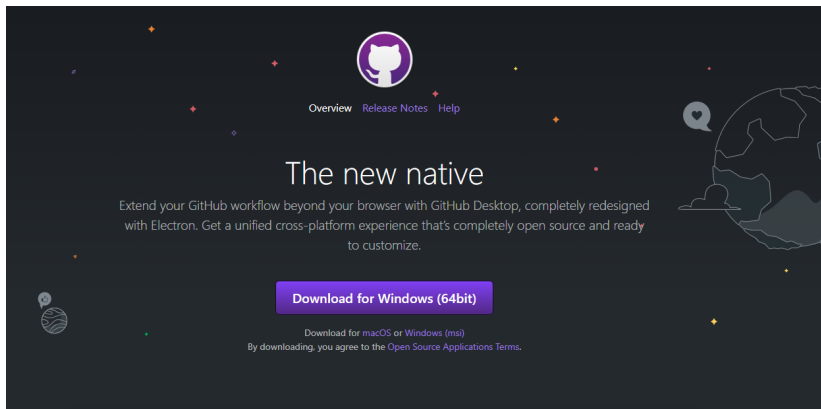
#### You'll love GitHub

- Unlimited** public repositories
- Unlimited** private repositories
- ✓ Limitless collaboration
- ✓ Frictionless development
- ✓ Open source community

Más información en [New GitHub account](#)

# Instalación de GitHub Desktop

<https://desktop.github.com/>



# Conectamos Git, GitHub y GitHub Desktop

- Una vez instalado comienza el proceso de autenticación, usando las credenciales del paso anterior<sup>3</sup>.

*File > Options > Accounts > Sign In*

- A continuación, conectamos la información de usuario con Git<sup>4</sup>.

*File > Options > Git*

---

<sup>3</sup>Más información en [Authenticating to GitHub](#).

<sup>4</sup>Más información en [Configuring Git](#).

## Ejercicio

Abre una cuenta en GitHub, empleando el correo electrónico de la UPM (será útil más tarde, al usar GitHub Classroom), y configura GitHub Desktop para usar esta cuenta.

# Remoto y Local

- **Github.com** aloja los **repositorios remotos** (nube).
- En tu ordenador trabajas con una **copia local** del repositorio. Otros desarrolladores tendrán sus propias copias locales.
- La(s) copia(s) local(es) y el repositorio remoto deben estar **sincronizados** mediante diferentes comandos de git.



# Nuevo repositorio *remoto* desde github.com

<https://github.com/new>

## Create a new repository

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

Owner



oscarperpinan ▾

/

Repository name \*

Great repository names are short and memorable. Need inspiration? How about **refactored-spoon**.

Description (optional)



Public

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



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

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

Add .gitignore: None ▾

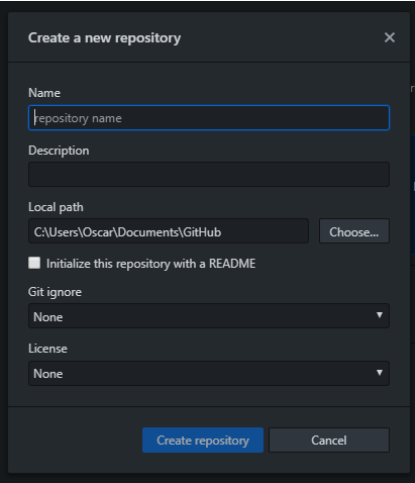
Add a license: None ▾



Create repository

# Nuevo repositorio *local* desde GitHub Desktop

*File > New Repository*



The screenshot shows the 'Create a new repository' dialog box in GitHub Desktop. The dialog has a dark theme and a title bar with a close button. It contains several input fields and checkboxes. The 'Name' field is highlighted with a blue border and contains the placeholder text 'repository name'. The 'Description' field is empty. The 'Local path' field contains the path 'C:\Users\Oscar\Documents\GitHub' and has a 'Choose...' button next to it. There is a checkbox labeled 'Initialize this repository with a README' which is currently unchecked. Below this are two dropdown menus for 'Git ignore' and 'License', both set to 'None'. At the bottom, there are two buttons: 'Create repository' (highlighted in blue) and 'Cancel'.

Create a new repository

Name  
repository name

Description

Local path  
C:\Users\Oscar\Documents\GitHub Choose...

☐ Initialize this repository with a README

Git ignore  
None

License  
None

Create repository Cancel

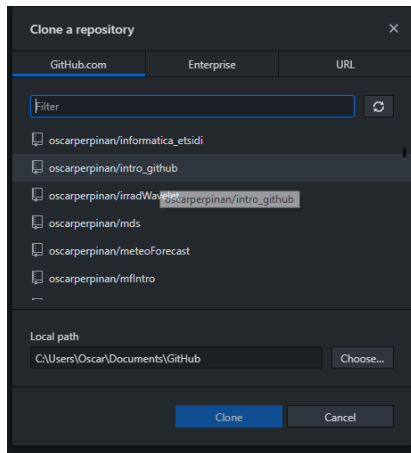
# Decisiones al crear un repositorio

- Elige un `.gitignore` adecuado al proyecto: Véase <https://github.com/github/gitignore>.
- No olvides inicializar y complimentar el `README.md`. Para el formato véase [Formatting syntax](#).
- Elige una **licencia** adecuada a tu proyecto y a tus intereses actuales y futuros. Véase <https://choosealicense.com>.

# Clonar un repositorio remoto

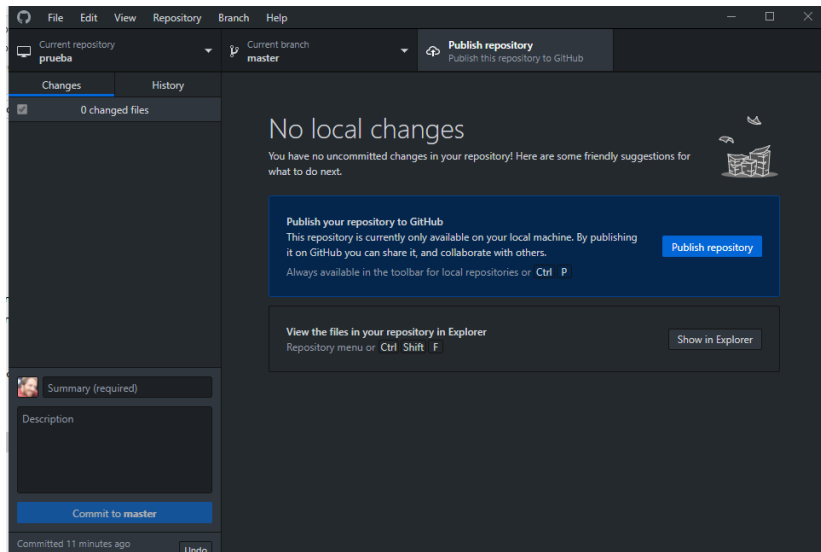
Si hemos creado el repositorio desde github.com (*repositorio remoto*), hay que clonarlo para poder trabajar con él (*copia local*).

*File > Clone Repository*



# Publicar un repositorio local

Si hemos creado el repositorio desde GitHub Desktop (*repositorio local*), hay que publicarlo en github.com (*repositorio remoto*)

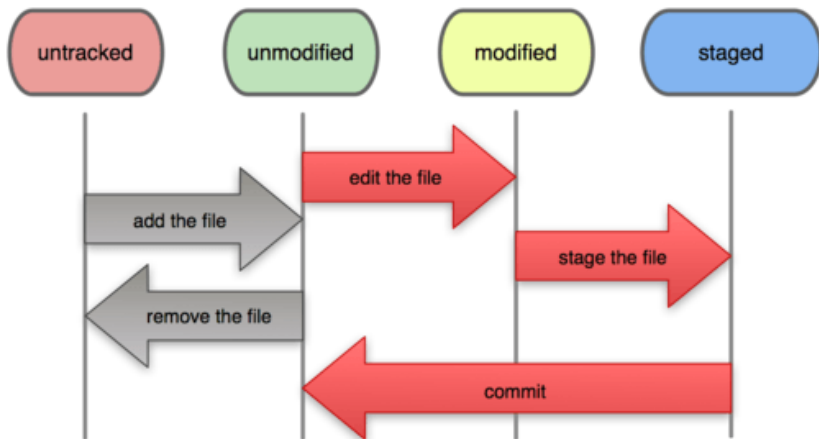


## Ejercicio

Crea un nuevo repositorio remoto y haz la clonación del mismo para tener una copia local. No olvides elegir la licencia, generar una README y un `.gitignore`.

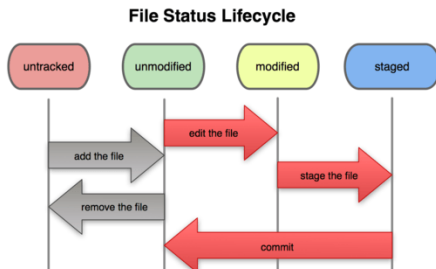
- 1 Conceptos básicos
- 2 Uso de git y GitHub
  - Primeros Pasos
  - Flujo de Trabajo
- 3 Trabajo en colaboración
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub

# File Status Lifecycle



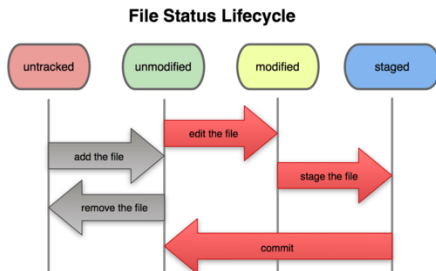


# Cambios en la copia local



En la carpeta que contiene la copia local, haz **modificaciones** en los ficheros (empleando tu editor de código/texto preferido).

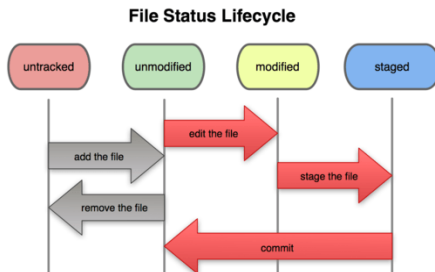
# Cambios en la copia local



**Añade los cambios** realizados a la siguiente «instantánea» del repositorio (`git add`)

Changes ●		History	README.md	
<input checked="" type="checkbox"/>	1 changed file			@@ -1 +1,3 @@
<input checked="" type="checkbox"/>	README.md		1	-# prueba1819
			2	+ # Esta es una prueba
			3	+Aquí debe ir la descripción del proyecto.

# Cambios en la copia local



**Confirma** los cambios, escribiendo un resumen de lo realizado (`git commit`)

A screenshot of a commit message form. At the top, there's a header with a small profile picture and the text "Añade texto a README". Below this is a text area containing the commit message: "Modifico el título de README. Añado algo de contenido en el cuerpo." At the bottom of the form is a blue button labeled "Commit to master". Below the form, it says "Committed 24 minutes ago" and "Initial commit", with an "Undo" button to the right.

# Consejos para commit

- *Commit early and often*: cada commit debe incluir cambios pequeños y coherentes.
- Escribir un mensaje de calidad al ejecutar cambios facilita tanto el trabajo personal como la colaboración en equipo con Git.<sup>5</sup>
- El **título** debe ser **conciso** (50 caracteres) y escrito en imperativo (*Do something...*)
- El **cuerpo** debe explicar **el qué y el por qué del cambio**, no el cómo, comparando con el comportamiento anterior al cambio.
- Se pueden incluir referencias a *issues* (ver a continuación) con #XX siendo XX el número de la *issue*.

---

<sup>5</sup><https://chris.beams.io/posts/git-commit/>

# Histórico de cambios

Los cambios confirmados con commit se anotan en la historia (`git log`)

*View > History*

Changes	History
No branches to compare	
<b>Añade texto a README</b>	
Oscar Perpiñán Lamigueiro committed 7 ...	
Modifico el título de README. Añado algo de contenido en el cuerpo.	
<b>Initial commit</b>	
Oscar Perpiñán Lamigueiro committed 3...	
Añade texto a README	
Oscar Perpiñán Lamigueiro committed eb5a029 1 changed file	
Modifico el título de README. Añado algo de contenido en el cuerpo.	
README.md	
@@ -1 +1,3 @@	
1 -# prueba1819	
2 +	
3 +Aquí debe ir la descripción del proyecto.	

# Publicar cambios al repositorio remoto

- Para sincronizar los cambios realizados **desde la copia local hasta el repositorio remoto** hay que publicar mediante `git push`.

*Repository > Push*

- A partir de este punto, la copia local y el repositorio remoto están sincronizados.

## Importante

En el caso de repositorios compartidos, antes de un `git push` es imprescindible actualizar la copia local incorporando los cambios del repositorio con `git pull`.

# Recibir cambios de un repositorio remoto

Para obtener en la copia local los cambios recientes que existan en el repositorio hay que emplear `git pull`, que es la combinación de la secuencia:

- 1 `git fetch`, para obtener los cambios recientes del repositorio remoto.
- 2 `git merge`, para combinarlos con la copia local.

*Repository > Pull*

# Resumen

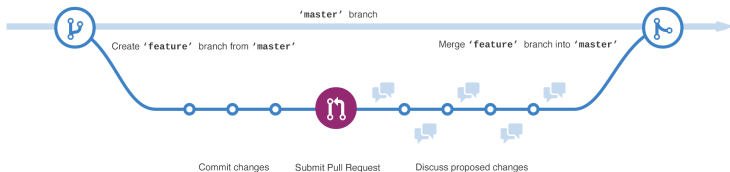
- ❶ Realiza modificaciones en los ficheros de la copia local.
- ❷ **COMMIT** :: Confirma los cambios con un mensaje informativo.
- ❸ **PULL** :: Incorpora los cambios del repositorio remoto a la copia local.
- ❹ **PUSH** :: Publica los cambios de la copia local al repositorio remoto.



- 1 Conceptos básicos
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración**
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub

- 1 Conceptos básicos
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
  - Ramas
  - Persiguiendo a los bichos
  - Herramientas gráficas para el análisis de un repositorio
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub

# Rama master

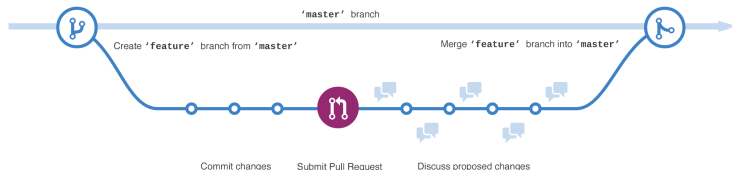


En un repositorio de GitHub existe una rama (*branch*) que se usa por defecto: **master**<sup>6</sup>.

---

<sup>6</sup>Understanding the GitHub Flow

# Ramas para facilitar la colaboración

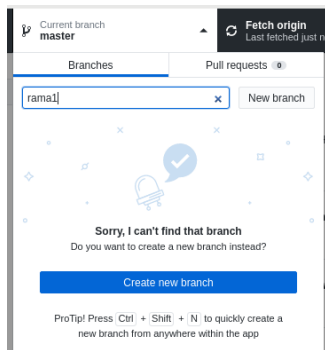


Cuando hay varias personas trabajando sobre un mismo repositorio, es necesario crear nuevas ramas para evitar conflictos.

De esta forma, cada persona implementa **cambios** en una **rama determinada** de forma paralela al resto del equipo.

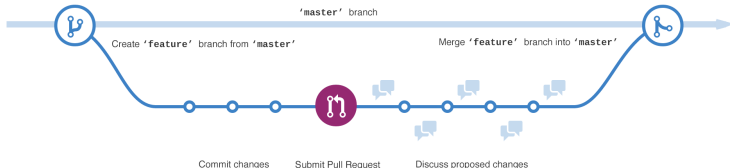
# Crear una nueva rama

- En menú: *Branch > New branch...*
- O en pantalla principal: *Current branch > Branches > New branch*

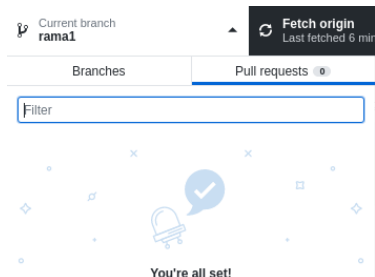


La nueva rama puede tener como origen la rama `master` u otra rama existente.

# Combinación de código

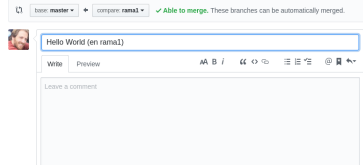


Cuando los cambios están listos y confirmados (*commit + push* en la rama específica), se realiza una petición (*pull request*) para combinar estos cambios en la rama **master**.

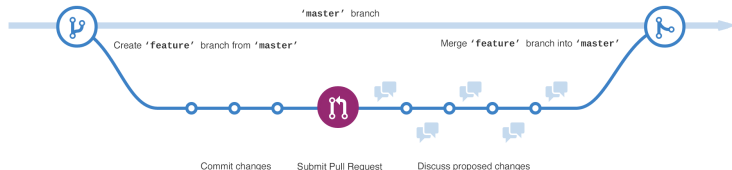


## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



# Combinación de código



El coordinador del proyecto es el encargado de revisar cada petición y, si todo está correcto, incluir los cambios (*merge*) en la rama **master**.



**Continuous integration has not been set up**

[Several apps are available](#) to automatically catch bugs and enforce style.



**This branch has no conflicts with the base branch**

Merging can be performed automatically.

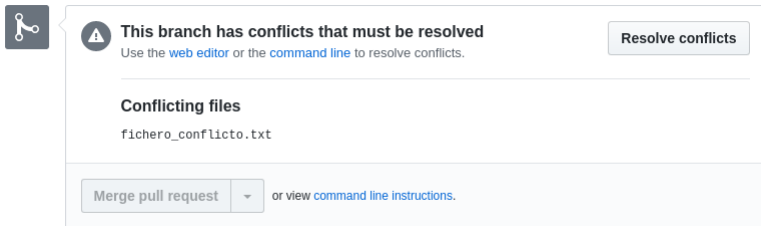
**Merge pull request**



or view [command line instructions](#).

# Resolución de conflictos

Si no se pueden combinar los cambios automáticamente se produce un conflicto (ejemplo: dos usuarios modifican un mismo fichero).



A screenshot of a GitHub pull request interface. At the top left is a GitHub logo. To its right is a warning icon (triangle with an exclamation mark) followed by the text "This branch has conflicts that must be resolved". Below this, it says "Use the [web editor](#) or the [command line](#) to resolve conflicts." In the top right corner of the box is a button labeled "Resolve conflicts". Below this is a section titled "Conflicting files" with the filename "fichero\_conflicto.txt" listed underneath. At the bottom of the box is a button labeled "Merge pull request" with a dropdown arrow, followed by the text "or view [command line instructions](#)."

Un conflicto se debe resolver manualmente.



A screenshot of a file conflict resolution interface. At the top, it says "Resolving conflicts between `rama1` and `master` and committing changes ➔ `rama1`". Below this, on the left, is a sidebar showing "1 conflicting file" and a file icon for "fichero\_conflicto.txt". The main area shows a diff for "fichero\_conflicto.txt". The diff highlights a conflict between two versions. The left version (rama1) has the text "Esto está escrito desde rama1". The right version (master) has the text "Esto está escrito en master.". The conflict is indicated by red and green brackets. At the top right of the diff area, it says "1 conflict" and has buttons for "Prev", "Next", and "Mark as resolved".

Lectura recomendada: [Merge Conflicts in the Classroom](#)



# Consejos

- No olvides hacer *pull* antes de iniciar una nueva interacción con el repositorio.
- Recuerda las recomendaciones sobre un buen mensaje de `git commit`.
- **Organización previa:** las **tareas** asignadas a un rama deben ser **independientes** de las otras ramas para evitar conflictos.
- Cuando el trabajo en una rama ha concluido, hay que **combinar cambios con master lo antes posible** para reducir la posibilidad de conflicto. Las ramas accesorias utilizadas se pueden eliminar una vez finalizado el proceso.
- Este proceso se debe repetir tantas veces como sea necesario para realizar cambios de forma colaborativa.

## Ejercicio 1

Crea una nueva rama en tu repositorio. En esta rama crea un nuevo fichero de texto y añade contenido en él. Sincroniza con el repositorio. Vuelve a la rama master y comprueba que este fichero nuevo no está presente. Combina ambas ramas.

## Ejercicio 2

Vuelve a la rama nueva y modifica un fichero. Vuelve a la rama master y modifica el mismo fichero. Combina ambas ramas y resuelve los conflictos.

- 1 Conceptos básicos
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
  - Ramas
  - Persiguiendo a los bichos
  - Herramientas gráficas para el análisis de un repositorio
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub

# Issues

Todos los repositorios de GitHub tienen una sección denominada «Issues»<sup>7</sup> a modo de *bug tracker*.

Pueden usarse para seguimiento de fallos, mejoras, tareas, etc.

Filters ▾

is:open label:cluster

Labels

Milestones

✕ Clear current search query, filters, and sorts

13 Open ✓ 105 Closed

Author ▾ Projects ▾ Labels ▾ Milestones ▾ Assignee ▾ Sort ▾

1 net::Server.unref() failed on cluster mode S-confirmed-bug cluster iojs-backport net 2

#25782 opened on Jul 30, 2015 by kyrioslili

1 ENOTSUP ErrorException in NodeJS using mssql in cluster worker cluster 1

#14382 opened on Apr 1, 2015 by ghost

1 listen() doesn't work with cluster on "node -e" bug S-confirmed-bug cluster v0.10 v0.12 4

#14168 opened on Mar 26, 2015 by FCO

1 cluster round robin could know about ipc send back pressure cluster defer-to-converged feature-request 0

#8746 opened on Nov 19, 2014 by tiffontaine

1 cluster: v0.11.13, cluster crashed in windows 2008r2 running on VMWare. asynclistener cluster windows 0

#7667 opened on May 23, 2014 by missing1984

1 Cluster worker.kill(signal) does not pass the signal parameter cluster 10

#6042 opened on Aug 12, 2013 by BorePlusPlus

1 cluster: implement API for pluggable distribute() for round-robin scheduler cluster defer-to-converged feature-request 6

#6001 opened on Aug 6, 2013 by kaero v0.13

<sup>7</sup><https://guides.github.com/features/issues/>

# Estructura de una issue

Una issue es un tablero de discusión en el que pueden participar los responsables del repositorio y cualquier usuario de GitHub.

**Debe** contener un título y una descripción.

**Puede** contener etiquetas, metas, y responsables.

Running cluster "hello world" on Win7 x64 returns -1073741819  
#4707

 **Open** turanuk opened this issue on Feb 3, 2013 · 3 comments



turanuk commented on Feb 3, 2013

...

Running code listed here: [http://nodejs.org/api/cluster.html#cluster\\_how\\_it\\_works](http://nodejs.org/api/cluster.html#cluster_how_it_works) seems to cause the worker to die with an error code of -1073741819 referring to an access violation. I assume this is a port conflict because if I change the code to only spawn a single worker the problem goes away and I'm able to access the web server. I tried port 3000 and ran into the same thing. Thoughts/debugging tips?

Assignees

No one assigned

Labels

cluster

windows

Projects

None yet

Milestone

No milestone

Notifications

 **Subscribe**

You're not receiving notifications from this thread.



bnoordhuis commented on Feb 10, 2013

Member

...

Ah, good old 0xc0000005. If you're using the official binary with no native add-ons, that obviously shouldn't happen. Try hooking up the processes to a debugger to see where the access violation happens.



bnoordhuis commented on Feb 10, 2013

Member

...

Also, what happens when you run the test suite? You invoke it with `python tools/test.py simple`.

# Contenido de una issue

- En la descripción de una issue se debe suministrar toda la información posible para el responsable del repositorio, **incluyendo un ejemplo mínimo, completo y verificable**<sup>8</sup>.
- El contenido será formateado como Markdown (incluye un *preview*)<sup>9</sup>.
- Se pueden incluir referencias al código y a otras issues<sup>10</sup>.

---

<sup>8</sup><https://stackoverflow.com/help/mcve>

<sup>9</sup>Veáse la guía [Basic Writing and Formatting syntax](#).

<sup>10</sup>Veáse la guía [Autolinked references and urls](#).

## Ejercicio

- Abre nuevas issues en tu propio repositorio, o en repositorios ajenos (por ejemplo, [https://github.com/oscarperpinan/prueba\\_github](https://github.com/oscarperpinan/prueba_github)).
- Responde y cierra issues de otros usuarios en tu propio repositorio.

- 1 Conceptos básicos
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
  - Ramas
  - Persiguiendo a los bichos
  - Herramientas gráficas para el análisis de un repositorio
- 4 GitHub Classroom
- 5 Publicación de páginas web en GitHub



Toda la actividad realizada en un repositorio puede verse de manera gráfica a través del botón *Insights* en la web del repositorio en GitHub<sup>11</sup>. Por ejemplo,

- Contribución de los integrantes del equipo
- Estructuras de ramas de un repositorio
- Histórico de cambios en un repositorio

---

<sup>11</sup>Más detalles en [Ver información del repositorio de forma gráfica](#).

# Contribución de los integrantes del equipo

Aug 14, 2011 - Jul 23, 2014

Contributions: **Commits** ▾

Contributions to master, excluding merge commits



**kevinsawicki**

5,913 commits / 122,808 ++ / 220,877 --

#1



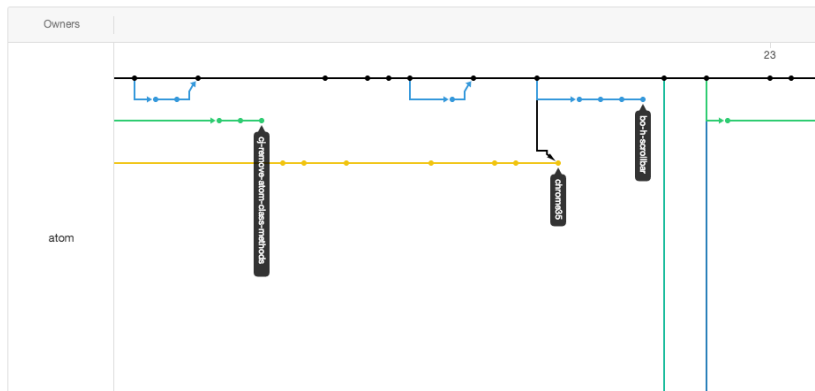
**nathansobo**

2,850 commits / 441,964 ++ / 340,415 --

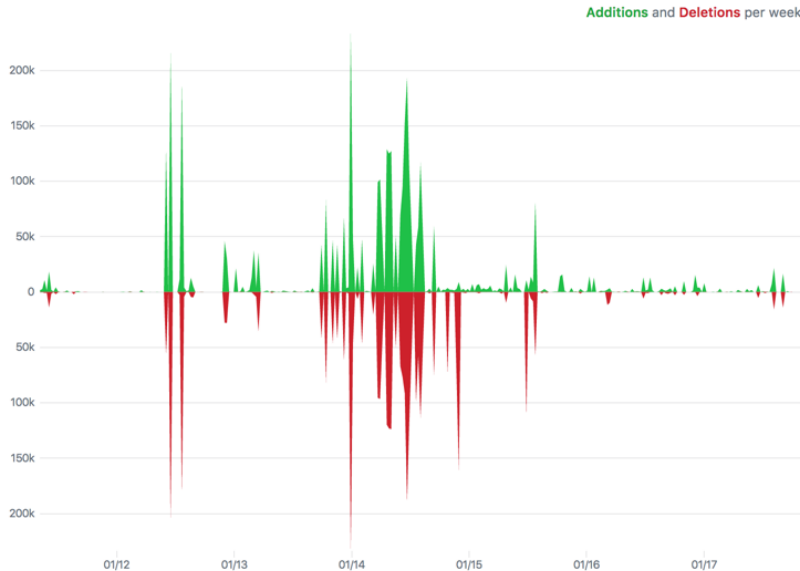
#2



# Estructura de ramas de un repositorio



# Cambios en un repositorio



## Ejercicio

Visita la sección Insights de los repositorios del proyecto AIGORA (<https://github.com/aigora>).

Por ejemplo: <https://github.com/aigora/twE105-cafranpa>

- 1 Conceptos básicos
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
- 4 GitHub Classroom**
- 5 Publicación de páginas web en GitHub

# ¿Qué es GitHub Classroom?

- Es un asistente para configurar grupos y asignar tareas en GitHub. Accesible en: <https://classroom.github.com/>
- Es recomendable [solicitar descuento por cuenta académica](#) (repositorios públicos y privados ilimitados, colaboradores ilimitados).

# GitHub Classroom necesita una Organización

- Para poder trabajar con GitHub Classroom es necesario crear una Organización en Github (grupo de cuentas que tienen acceso compartido a un grupo de repositorios):  
<https://github.com/organizations/new>
- Ejemplo: <https://github.com/swcarpentry>
- Dentro de una organization pueden coexistir diferentes equipos (Team), cada uno de ellos con diferentes permisos de acceso y escritura a los repositorios.
- Las cuentas tipo organization también pueden solicitar el [descuento por cuenta académica](#).
- Más información en el [blog de GitHub](#).



# Configuración de aulas y tareas

- Una única cuenta organization puede dar cabida a múltiples aulas (classrooms):  
<https://classroom.github.com/classrooms/>
- Dentro de cada classroom se pueden incluir múltiples tareas (assignments).
- Un assignment puede ser individual o grupal (los grupos se pueden definir antes por el profesor, o por los estudiantes en el momento de aceptar la tarea.)
- Cada classroom contiene estudiantes (identificados por su usuario de GitHub y su correo electrónico) y grupos (teams).
- Esta correspondencia se puede facilitar mediante el classroom roster.

# Cómo hemos usado GitHub Classroom en AIGORA

## Configuración General

- Usamos una `organization` común para todos los grupos de matriculación de una misma asignatura.
- Definimos un `classroom` para cada grupo de matriculación.
- En cada `classroom` subimos la lista de correos UPM de los estudiantes de ese grupo de matriculación mediante el `roster management`.
- Dentro de cada `classroom` definimos diferentes equipos de trabajo (Teams) para trabajos grupales.

# Cómo hemos usado GitHub Classroom en AIGORA

## Registro de estudiantes

- 1 En cada classroom creamos una Group Assignment denominada «registro estudiantes». Como identificador del «Group» utilizamos el grupo de matriculación. Este identificador queda almacenado en GitHub Classroom para futuros usos, pero **no** crea un Team en GitHub.
- 2 Abrimos el enlace del assignment con nuestro usuario (podemos hacer skip cuando nos pide que enlacemos a algún miembro del roster), y **creamos un nuevo Team**, que se debe llamar igual que el grupo de matriculación. Este paso **sí** crea un nuevo Team en la organization de Github (útil para asignar permisos grupales).
- 3 A continuación enviamos el enlace del assignment a los alumnos para que enlacen su usuario de GitHub a su correo incluido en el roster. **Deben elegir el Team que hemos definido nosotros, sin crear uno nuevo**. A partir de este punto forman parte del Team con los permisos y repositorios definidos para el mismo.

# Cómo hemos usado GitHub Classroom en AIGORA

## Grupos de Trabajo

- ❶ Para el trabajo en equipo creamos igualmente una Group Assignment, pero dejamos libertad a la hora de definir los Teams.
- ❷ El primer estudiante de un grupo de trabajo que acepta este assignment define el nombre del Team. Los siguientes estudiantes de ese mismo grupo deben seleccionar ese Team.
- ❸ Se pueden definir repositorios plantilla para incluir contenido inicial en todos los repositorios creados para cada Team. Ejemplo:  
<https://github.com/aigora/starter-code>

- 1 Conceptos básicos
- 2 Uso de git y GitHub
- 3 Trabajo en colaboración
- 4 GitHub Classroom
- 5 **Publicación de páginas web en GitHub**

# Página web de proyecto

## Si no sabes HTML

- En la página del repositorio:

*Settings > GitHub Pages > Source > master branch*

*Settings > GitHub Pages > Theme Chooser*

- Modifica el fichero README.md<sup>a</sup> (commit + push).
- Con un navegador ve a la dirección  
<https://<username>.github.io/<repository>>

---

<sup>a</sup>Más información sobre formato Markdown

<https://guides.github.com/features/mastering-markdown/>.

# Página web de proyecto

## Si sabes HTML

- Crea una carpeta docs en la rama master del repositorio.
- En esta carpeta docs crea/modifica un fichero index.html (commit + push).
- En la página del repositorio:

*Settings > GitHub Pages > Source > docs folder*

- Con un navegador ve a la dirección  
<https://<username>.github.io/<repository>>

# Ejemplo de web de **proyecto**

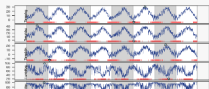
- Página web: <https://oscarperpinan.github.io/bookvis/>
- Repositorio: <https://github.com/oscarperpinan/bookvis/tree/master/docs>

## Displaying time series, spatial and space-time data with R

This is the accompanying website of the **second edition** of the book “*Displaying time series, spatial and space-time data with R*”, published with [Chapman&Hall/CRC](#).

Code, data, and figures are available at this [GitHub repository](#).

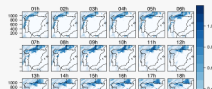
**Time Series**



**Spatial Data**



**Space-time Data**





# Página web de **usuario u organización**

- 1 Crea un repositorio nuevo con el nombre `<username>.github.io`<sup>12</sup>.
- 2 Sube (commit + push) un fichero `index.html` a la rama master con código HTML.
- 3 Con un navegador ve a la dirección `https://<username>.github.io`

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

---

<sup>12</sup>Siendo `<username>` tu nombre de usuario en GitHub.

# Ejemplo de web de **organización**

- Página web: <https://aigora.github.io/>
- Repositorio: <https://github.com/aigora/aigora.github.io>

