

# Git Workshop





1

# Google Drive

git: Multiplayer Notepad





2

# Git

git-svn: now you have two problems.

# ¿Cómo habéis trabajado hasta ahora?


- Trabajo **secuencial**
- **No hay visibilidad** del trabajo del resto hasta el final (**asíncrono**).
- Ficheros replicados
  - Versión final
  - Versión definitiva
  - Versión final definitiva
  - ...
- **No hay vuelta atrás** (sencilla)



A line-art illustration of a space scene. At the top center is a planet with a ring and three small circles on its surface. Surrounding the planet are five stars of varying sizes. Below the planet, towards the left, is a small rocket ship with a flame trail. The entire scene is set against a background of concentric circles.

# ¿Solución?

Git (VCS)



“ **Git** is a version-control system for **tracking changes** in computer files and **coordinating work** on those files among **multiple people**.

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

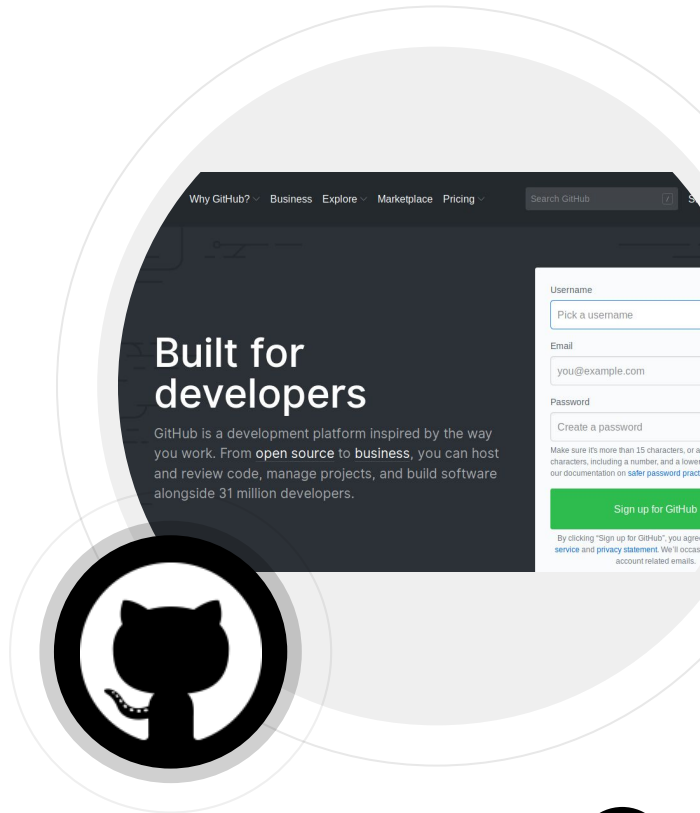
COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.



# Un ejemplo: GitHub

- Es la más conocida
- Permite su uso privado
- Ampliamente usado en el ámbito profesional
- <https://github.com/> (¡aunque hay muchos más!)







# Blue-Bash

**Repositories** 3 **People** 4 **Teams** 0 **Projects** 0 **Settings**

## UrlShortener

Forked from UNIZAR-30246-WebEngineering/UrlShortener

Web Engineering 2018-2019 / URL Shortener

Java 1 MIT Updated 8 hours ago

## doc

Contiene documentos de entrega periódica.

MIT Updated 24 days ago

**README.md**

## Web Engineering 2018-2019 / URL Shortener

### Badges

Branch	Build	Passed	Rating	Coverage	Bugs
Master	build passing	quality gate passed	reliability A	coverage 0 %	bugs 0
Develop	build passing	quality gate failed	reliability A	coverage 0 %	bugs 0

This is the shared repository for the project developed in this course. Go to the [wiki](#) to start your project.

### Projects

- [Common](#) is the project that provides a minimum set of shared features.
- [Demo](#) is an example application and the sandbox for solving blocking issues.
- [Team](#) is an empty application that can be used for development.



# OnePlus One

## THE BEST OF BOTH WORLDS

Born from a partnership between master software and hardware makers, the OnePlus One redefines modern smartphone design.

LEARN MORE

STARTING AT \$299

## Your Mobile Canvas

Batteries included by allowing users thousands of themes on the new Theme Store. Personalize your mobile life and create a home, from wallpaper, and more to create your perfect look.



Computer Hoy

# Uber Demonstrates its Dedication to Open Source With Linux Foundation Gold Membership

LEARN MORE ABOUT HOW UBER IS LEVERAGING OPEN SOURCE

## Largest Shared Technology Investment

The Linux Foundation supports the creation of sustainable open source ecosystems by providing financial and intellectual resources, infrastructure, services, events, and training. Working together, The Linux Foundation and its projects form the most ambitious and successful investment in the creation of shared technology.



**16B USD**

Estimated development of the 100+ world's leading projects hosted



**25,000**

Technologists attend our events annually, from more than 4,500



**1 Million**

Open source professionals have enrolled in our free



**10 / 10**

Largest cloud service providers are Linux Foundation projects



3

# Git: De 0 a 1

In Git we trust!

# Repositorios

- Es la **unidad básica** de organización
- Puede representar
  - Una práctica (o varias)
  - Un proyecto
  - **TODOS** los proyectos de tu empresa
  - ...



# ¡Vamos a trabajar en local!

*\$ git config --global user.name Nombre*

*\$ git config --global user.email mail@mail.com*

```
langel@lAngel:~/GitWorkshop$ git config --global user.name lAngel
langel@lAngel:~/GitWorkshop$ git config --global user.email angelcanal97@gmail.com
langel@lAngel:~/GitWorkshop$
```



# ¡Vamos a trabajar en local!

*\$ git init*

```
langel@lAngel:~$ mkdir GitWorkshop
langel@lAngel:~$ cd GitWorkshop/
langel@lAngel:~/GitWorkshop$ git init
Initialized empty Git repository in /home/langel/GitWorkshop/.git/
langel@lAngel:~/GitWorkshop$
```



# ¡Vamos a trabajar en local!

*\$ git status*

```
langel@lAngel:~$ mkdir GitWorkshop
langel@lAngel:~$ cd GitWorkshop/
langel@lAngel:~/GitWorkshop$ git init
Initialized empty Git repository in /home/langel/GitWorkshop/.git/
langel@lAngel:~/GitWorkshop$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
langel@lAngel:~/GitWorkshop$
```





# ¡Vamos a trabajar en local!

*\$ echo "....." > ficheroDeSalida*

*\$ git status*

```
langelelAngel:~/GitWorkshop$ echo "This is my first file" >README.md
langelelAngel:~/GitWorkshop$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README.md

nothing added to commit but untracked files present (use "git add" to track)
langelelAngel:~/GitWorkshop$
```



# ¡Vamos a trabajar en local!

*\$ git add ficheroA*

*\$ git status*

```
langel@lAngel:~/GitWorkshop$ git add README.md
langel@lAngel:~/GitWorkshop$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   README.md

langel@lAngel:~/GitWorkshop$
```



# ¡Vamos a trabajar en local!

*\$ git commit -m "Mensaje"*

*\$ git status*

```
langel@lAngel:~/GitWorkshop$ git commit -m "This is my first commit"
[master (root-commit) 8bb180d] This is my first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
langel@lAngel:~/GitWorkshop$ git status
On branch master
nothing to commit, working tree clean
langel@lAngel:~/GitWorkshop$
```



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFT	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

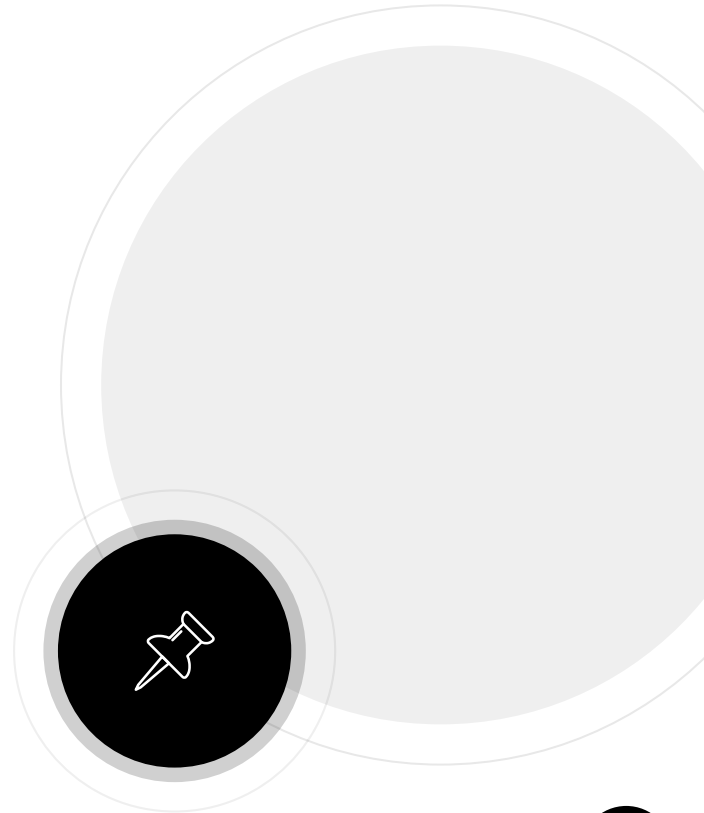
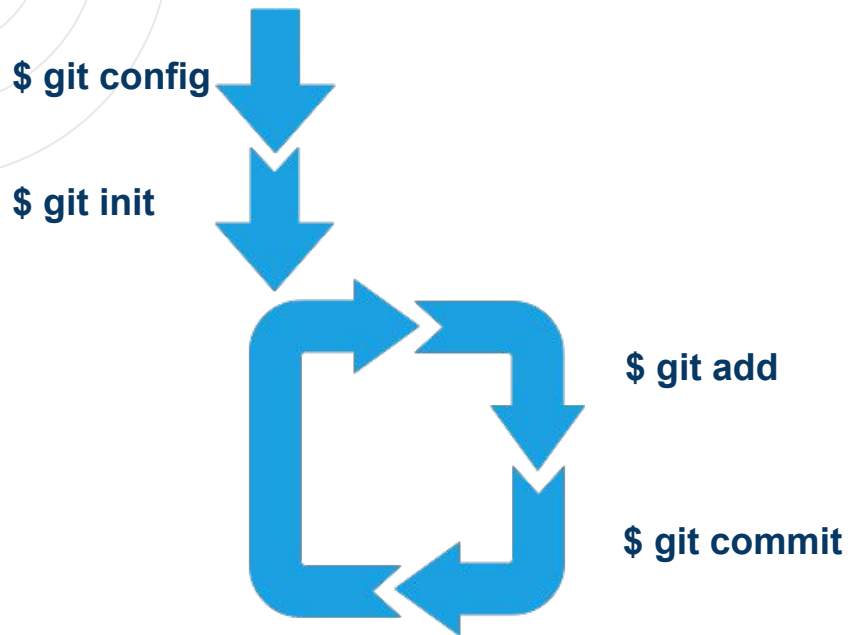
# ¡Vamos a trabajar en local!

*\$ git log*

```
langel@lAngel:~/GitWorkshop$ git log
commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c (HEAD -> master)
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100

    This is my first commit
langel@lAngel:~/GitWorkshop$
```

# Local workflow



# ¡Creemos un repositorio en Internet!

## Create a new repository

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

Owner

 CodeLabZGZ ▾

Repository name

/ GitWorkshop ✓

Great repository names are short and memorable. Need inspiration? How about **studious-rotary-phone**.

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**

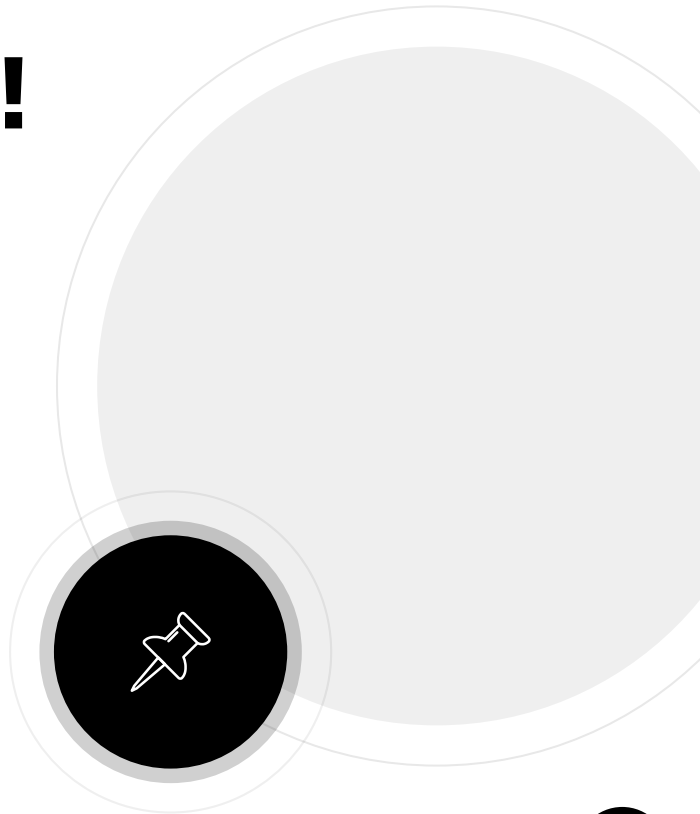
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository



# ¡Creemos un repositorio en Internet!

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

or **HTTPS** **SSH** `git@github.com:CodeLabZGZ/GitWorkshop.git`



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

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



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

```
git remote add origin git@github.com:CodeLabZGZ/GitWorkshop.git
git push -u origin master
```





# ¡Subámoslo a Internet!

*\$ git remote add origin URL*

```
langel@lAngel:~/GitWorkshop$ git remote add origin git@github.com:CodeLabZGZ/GitWorkshop.git  
langel@lAngel:~/GitWorkshop$
```

## ¡Ya estamos conectados con GitHub!





# ¡Subámoslo a Internet!


*\$ git push*


```
langel@lAngel:~/GitWorkshop$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 241 bytes | 241.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/CodeLabZGZ/GitWorkshop/pull/new/master
remote:
To github.com:CodeLabZGZ/GitWorkshop.git
 * [new branch]      master -> master
langel@lAngel:~/GitWorkshop$
```

# ¡Subámoslo a Internet!

 1 commit

 1 branch

 0 releases

 1 contributor

Branch: master ▾


New pull request


Create new file



Upload files

Find file

Clone or download ▾

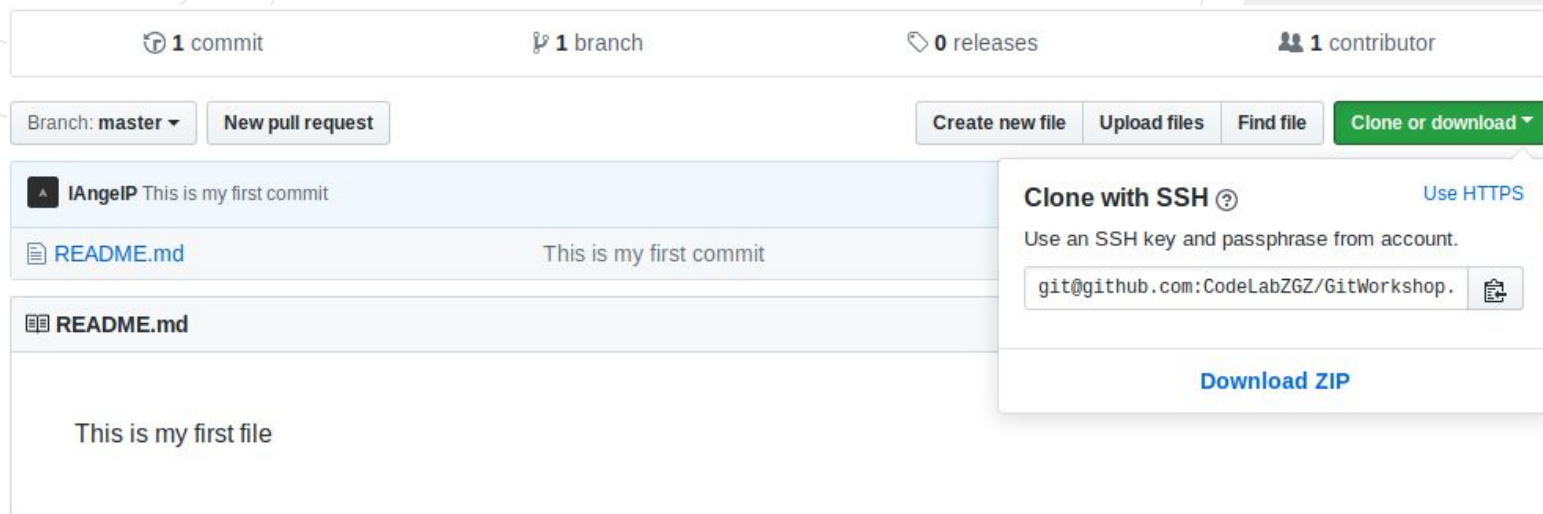
 **lAngelP** This is my first commit Latest commit 8bb180d 34 minutes ago

 [README.md](#) This is my first commit 34 minutes ago

 **README.md** 

This is my first file

# ¡Descarguémoslo de Internet!



The screenshot displays a GitHub repository interface. At the top, it shows repository statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. Below this, there are navigation buttons: 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The main content area shows a commit by 'lAngelP' with the message 'This is my first commit'. Below the commit, there is a file named 'README.md' with the content 'This is my first file'. A dropdown menu is open from the 'Clone or download' button, showing options to 'Clone with SSH' (with a help icon) and 'Use HTTPS'. The SSH URL is 'git@github.com:CodeLabZGZ/GitWorkshop.' with a copy icon. There is also a 'Download ZIP' button at the bottom of the dropdown.

1 commit    1 branch    0 releases    1 contributor

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

lAngelP This is my first commit


README.md This is my first commit

README.md

This is my first file

Clone with SSH ?    Use HTTPS

Use an SSH key and passphrase from account.

git@github.com:CodeLabZGZ/GitWorkshop.    

Download ZIP

# ¡Descarguémoslo de Internet!

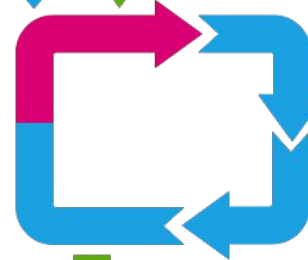
*\$ git clone url [nombreDestino]*

*\$ git log*

```
langel@lAngel:~$ git clone git@github.com:CodeLabZGZ/GitWorkshop.git GitHubWorkshop
Cloning into 'GitHubWorkshop'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
langel@lAngel:~$ cd GitHubWorkshop
langel@lAngel:~/GitHubWorkshop$ git log
commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c (HEAD -> master, origin/master, origin/HEAD)
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100

    This is my first commit
langel@lAngel:~/GitHubWorkshop$
```

# Work



# ¿Qué hemos solucionado?

- Trabajo **secuencial**
- **No hay visibilidad** del trabajo del resto hasta el final (**asíncrono**).
- ~~Ficheros replicados~~
  - ~~○ Versión final~~
  - ~~○ Versión definitiva~~
  - ~~○ Versión final definitiva~~
  - ~~○ ...~~
- ~~No hay vuelta atrás (seneilla)~~





4

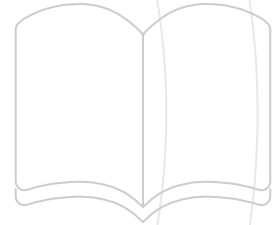
# Git: De l a N

Don't forget to bring a towel





# **Retomemos Google Drive**



# Fetch vs pull

- Ambos parece que hacen lo mismo...
- ... o no:
  - “*git pull*” descarga los cambios del repositorio remoto y los incorpora automáticamente al repositorio local.
  - “*git fetch*” descarga los cambios del repositorio remoto **SIN** incorporarlos al repositorio local.



# ¡Actualicemos el repositorio local!

*\$ git pull origin master*

*\$ git log*

```
langel@lAngel:~/GitHubWorkshop$ git pull origin master
```

```
langel@lAngel:~/GitHubWorkshop$ git log
```

```
commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c
```

```
Author: lAngel <angelcanal97@gmail.com>
```

```
Date: Sat Dec 8 23:10:46 2018 +0100
```

```
This is my first commit
```

```
langel@lAngel:~/GitHubWorkshop$ █
```

# ¡Modifiquemos el repositorio remoto!

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

IAngelP Create example.md

Latest commit 6330a85 7 minutes ago

README.md

Adding content

an hour ago

2 lines (1 sloc) | 21 Bytes

Raw

Blame

History



# ¡Modifiquemos el repositorio remoto!



## Commit changes

Update README.md

Add an optional extended description...

angelcanal97@gmail.com ⓘ

- ☒ Commit directly to the `master` branch.
- ☐ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel



# ¡Actualicemos el repositorio local!

*\$ git pull origin master*

*\$ git log*

```
langel@lAngel:~/GitHubWorkshop$ git pull origin master
langel@lAngel:~/GitHubWorkshop$ git log
commit c764d31b8e66c9389686302d02e1082f3f258764 (HEAD -> master, origin/master, origin/HEAD)
Author: lAngelP <angelcanal97@gmail.com>
Date: Mon Dec 10 21:31:47 2018 +0100

    Update README.md

commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100

    This is my first commit
langel@lAngel:~/GitHubWorkshop$
```

# ¡Actualicemos el repositorio local (II)!

*\$ git fetch*

*\$ git log*

```
langel@lAngel:~/GitHubWorkshop$ git fetch
langel@lAngel:~/GitHubWorkshop$ git log
commit c764d31b8e66c9389686302d02e1082f3f258764 (HEAD -> master)
Author: lAngelP <angelcanal97@gmail.com>
Date: Mon Dec 10 21:31:47 2018 +0100

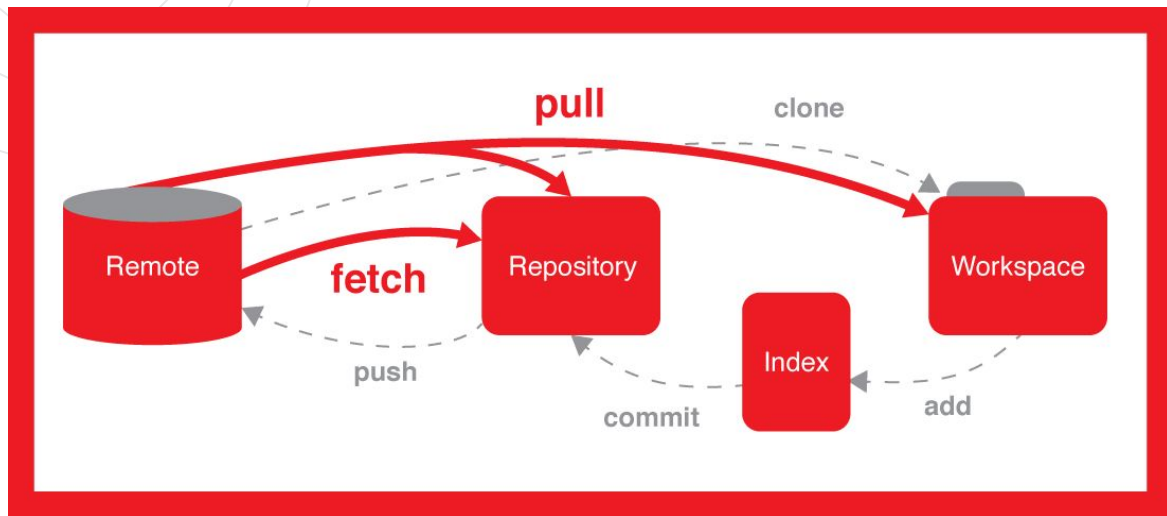
    Update README.md

commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100

    This is my first commit
langel@lAngel:~/GitHubWorkshop$
```



# Git Workflow



StackOverflow





```
langel@lAngel:~/GitHubWorkshop$ git log
commit c764d31b8e66c9389686302d02e1082f3f258764 (HEAD -> master, origin/master, origin/HEAD)
Author: lAngelP <angelcanal97@gmail.com>
Date: Mon Dec 10 21:31:47 2018 +0100

    Update README.md

commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100

    This is my first commit
langel@lAngel:~/GitHubWorkshop$
```

*\$ git pull*

```
langel@lAngel:~/GitHubWorkshop$ git log
commit c764d31b8e66c9389686302d02e1082f3f258764 (HEAD -> master)
Author: lAngelP <angelcanal97@gmail.com>
Date: Mon Dec 10 21:31:47 2018 +0100

    Update README.md

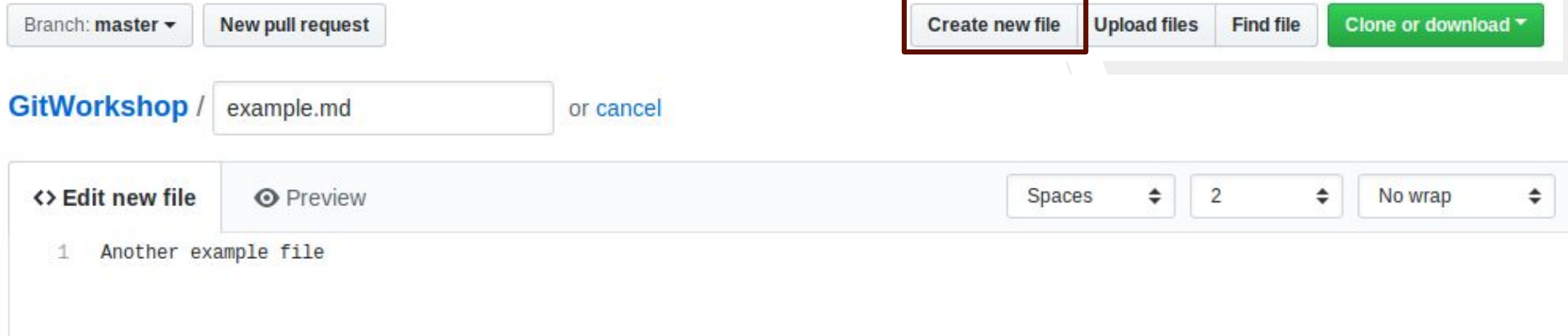
commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100

    This is my first commit
langel@lAngel:~/GitHubWorkshop$
```

*\$ git fetch*

# Vamos a hacer un pequeño ejercicio

1. Desde GitHub, vamos a crear un nuevo fichero en el repositorio.

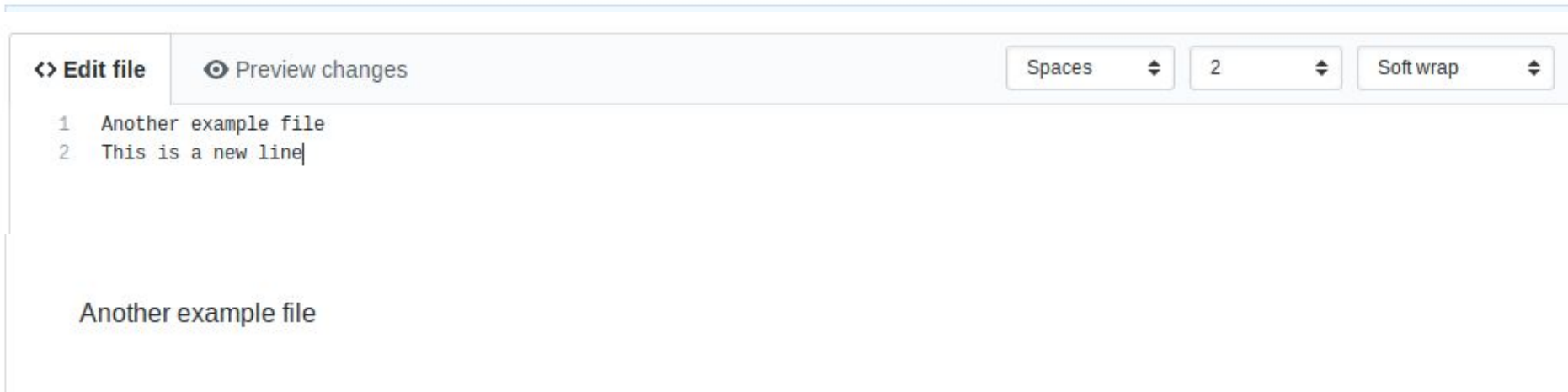


# Vamos a hacer un pequeño ejercicio

1. Desde GitHub, vamos a crear un nuevo fichero en el repositorio.
2. Desde el ordenador, vamos a traer los cambios que hemos hecho (git pull).

# Vamos a hacer un segundo ejercicio

1. Desde GitHub, vamos a editar el nuevo fichero en el repositorio añadiendo una línea al final.



# Vamos a hacer un segundo ejercicio

1. Desde GitHub, vamos a editar el nuevo fichero en el repositorio añadiendo una línea al final.
2. En el ordenador, también vamos a añadir una línea a ese fichero.
3. Por último vamos a enviar los cambios que habíamos hecho en el paso 2 a GitHub.

# Vamos a segundo

1. Desde GitHub  
fichero en el  
línea al final.
2. En el ordenador, tar  
una línea a ese fichero.
3. Por último vamos a enviar los cambios  
que habíamos hecho en el paso 2 a GitHub..

```
langel@lAngel:~/GitHubWorkshop$ git pull origin master
langel@lAngel:~/GitHubWorkshop$ git pull origin master
From github.com:CodeLabZGZ/GitWorkshop
To github.com:CodeLabZGZ/GitWorkshop
* branch                master      -> FETCH_HEAD
! [rejected] master: master -> FETCH_HEAD
Auto-merging example.md
error: failed to merge: CONFLICT (content): Merge conflict in example.md
hint: Update the branch with 'git pull --no-commit'
hint: Automatic merge failed; fix conflicts and then commit the result.
langel@lAngel:~/GitHubWorkshop$ cat example.md
Another example file
hint: to the HEAD
hint: (e.g., 'git commit -m "New line"')
hint: See the 'git commit' help for more details
langel@lAngel:~/GitHubWorkshop$ git commit -m "New line"
[main 5070ee2] New line
1 file changed, 1 insertion(+)
langel@lAngel:~/GitHubWorkshop$ git push origin master
langel@lAngel:~/GitHubWorkshop$ git pull origin master
langel@lAngel:~/GitHubWorkshop$ git push origin master
```

```
$ git push origin master
$ git pull origin master
$ git push origin master
```

# ¿Qué hemos solucionado?

- Trabajo **secuencial**
- ~~No hay visibilidad~~ del trabajo del resto hasta el final (~~asínerono~~).
- ~~Ficheros replicados~~
  - ~~Versión final~~
  - ~~Versión definitiva~~
  - ~~Versión final definitiva~~
  - ~~...~~
- ~~No hay vuelta atrás~~ (seneilla)





5

# Git: De1aN (advanced)

May the forks be with you



# Git y los commits

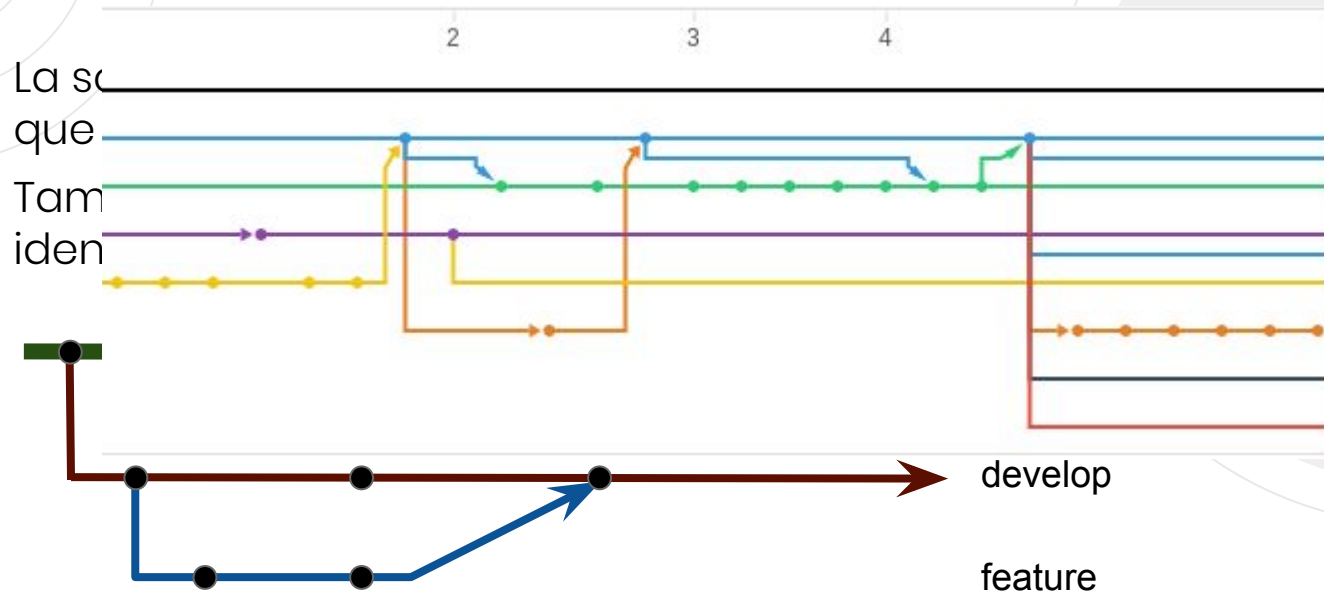
¿Cómo almacena Git los cambios en el código?

Cada commit guarda:

- Las diferencias (snapshots) que hay entre la versión previa y la versión con los cambios (ficheros completos).
- El commit (o commits) que le precede(n)
- Algunos metadatos (quién, cuándo...)

Entonces, ¿cómo dos personas pueden trabajar con la misma versión sin crear problemas?

# Branches (ramas)



Actualiza la rama desde la que creas la nueva  
antes (git pull)

# Crear una rama

*\$ git branch [name]*

*\$ git checkout [name]*

```
langel@lAngel:~/GitHubWorkshop$ git branch exampleBranch  
langel@lAngel:~/GitHubWorkshop$ git checkout exampleBranch  
Switched to branch 'exampleBranch'
```



# Crear una rama

*\$ git status*

*\$ git log*

```
lAngel@lAngel:~/GitHubWorkshop$ git status
On branch exampleBranch
nothing to commit, working tree clean
lAngel@lAngel:~/GitHubWorkshop$ git log
commit c764d31b8e66c9389686302d02e1082f3f258764 (HEAD -> exampleBranch, master)
Author: lAngelP <angelcanal97@gmail.com>
Date: Mon Dec 10 21:31:47 2018 +0100
```

Update README.md

```
commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100
```

This is my first commit

```
lAngel@lAngel:~/GitHubWorkshop$
```

# Añadiendo cambios a una rama

*\$ echo "Even more content" >> README.md*

*\$ git commit -am "Adding content"*

*\$ git push origin [name]*

```
langel@lAngel:~/GitHubWorkshop$ echo "Even more content" >> README.md
langel@lAngel:~/GitHubWorkshop$ git commit -am "Adding content"
[exampleBranch 5b915b0] Adding content
 1 file changed, 1 insertion(+)
langel@lAngel:~/GitHubWorkshop$ git push origin exampleBranch
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 304.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'exampleBranch' on GitHub by visiting:
remote:   https://github.com/CodeLabZGZ/GitWorkshop/pull/new/exampleBranch
remote:
To github.com:CodeLabZGZ/GitWorkshop.git
 * [new branch]      exampleBranch -> exampleBranch
```

Vas a necesitar actualizar las dos ramas antes  
de poder mezclar las dos ramas (git pull)

# Mezclar dos ramas

*\$ git checkout [destBranch]*

*\$ git merge [originBranch]*

```
langel@lAngel:~/GitHubWorkshop$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
langel@lAngel:~/GitHubWorkshop$ git merge exampleBranch
Updating 0fe5be0..7f17ba2
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

```
langel@lAngel:~/GitHubWorkshop$ git log
commit 7f17ba23bbe9546f7a5e6f935bed3fc9d1a1f42d (HEAD -> master, exampleBranch)
Author: lAngel <angelcanal97@gmail.com>
Date: Mon Dec 10 23:49:06 2018 +0100
```

Adding content

```
commit 0fe5be04111cec9fb13c1b6e77d8b6b73481a5e5 (origin/master, origin/HEAD)
Author: lAngelP <angelcanal97@gmail.com>
Date: Mon Dec 10 21:40:46 2018 +0100
```

Update README.md

```
commit c764d31b8e66c9389686302d02e1082f3f258764
Author: lAngelP <angelcanal97@gmail.com>
Date: Mon Dec 10 21:31:47 2018 +0100
```

Update README.md

```
commit 8bb180d2ea37bb2062ec4f6a32d88e257c3f2d5c
Author: lAngel <angelcanal97@gmail.com>
Date: Sat Dec 8 23:10:46 2018 +0100
```

This is my first commit

# Borrar una rama

*\$ git branch -d [name]*

*\$ git push origin --delete [name]*

```
langel@lAngel:~/GitHubWorkshop$ git branch -d exampleBranch
Deleted branch exampleBranch (was 7f17ba2).
langel@lAngel:~/GitHubWorkshop$ git push origin --delete exampleBranch
To github.com:CodeLabZGZ/GitWorkshop.git
- [deleted]          exampleBranch
langel@lAngel:~/GitHubWorkshop$
```



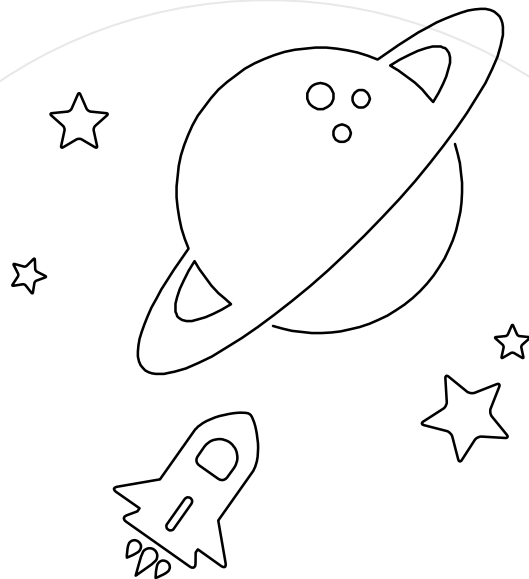


# ¿Qué hemos solucionado?

¿Se han solucionado todos los problemas?

- ~~Trabajo secuencial~~
- ~~No hay visibilidad del trabajo del resto hasta el final (asíncrono).~~
- ~~Ficheros replicados~~
  - ~~Versión final~~
  - ~~Versión definitiva~~
  - ~~Versión final definitiva~~
  - ~~...~~
- ~~No hay vuelta atrás (sencilla)~~





# ¡¡Conflictos!!

¿Qué ocurre si dos personas modifican el **mismo** fragmento de un **mismo** fichero en una **misma** rama?

# Aspectos más avanzados: Fork

En un proyecto grande, por ejemplo, el kernel de Linux:

- ¿Qué pasaría si cualquiera pudiera hacer push al repositorio?
- ¿Qué pasaría si sólo ciertas personas designadas por el dueño del repositorio pudieran hacer push?

La respuesta es tener una forma de controlar las escrituras al repositorio.

# Aspectos más avanzados: Fork

Para ello existe el fork, que se podría resumir como una copia del repositorio de la cuenta oficial a la cuenta personal de GitHub.  
(Se trata de un tema relacionado con GitHub, no con Git.)

# Aspectos más avanzados: Fork

Pasos a seguir para trabajar con fork:

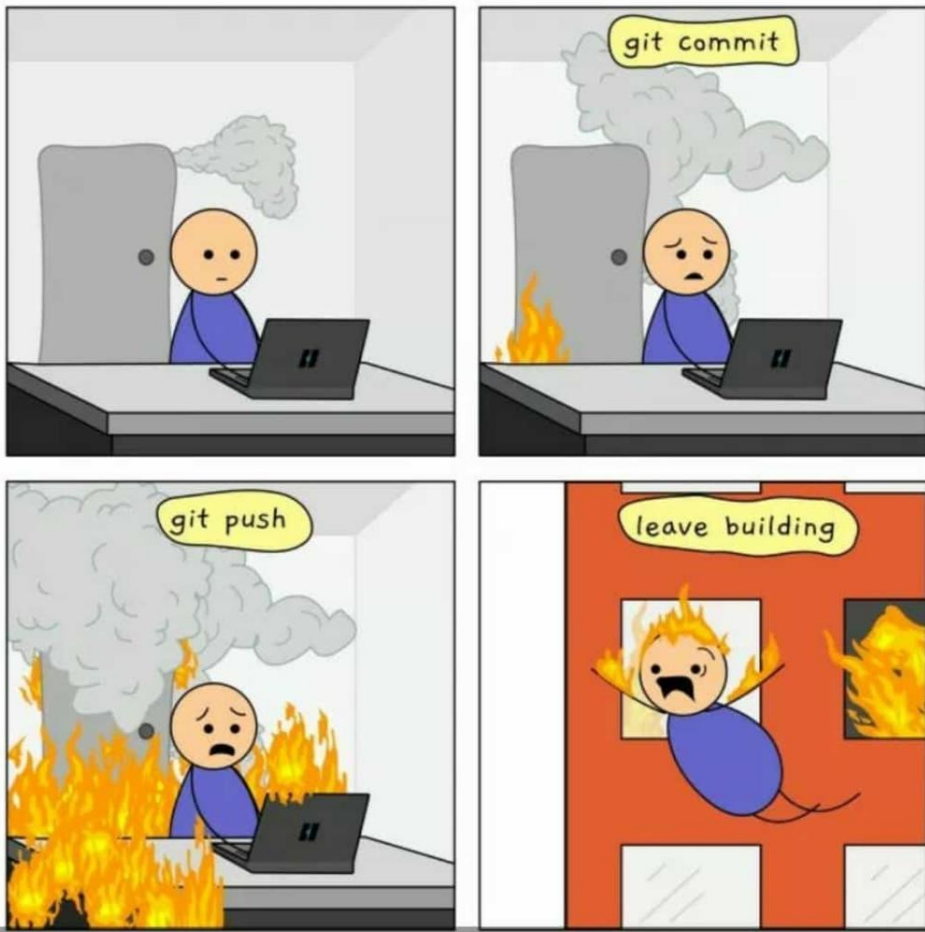
- Se hace fork del repositorio al que se quiere aportar algo.
- Se hace clone a local para poder trabajar en el repositorio (personal).
- Se crea una nueva rama en la que se trabajará, como si fuera un repositorio normal de una persona.
- Cuando ya has acabado la aportación, se sube a GitHub toda la aportación.
- Para juntarla con el repositorio original, es necesario hacer un Pull Request.

# ¿Qué se puede hacer en GitHub?

GitHub es una herramienta muy potente, que permite gestionar muchos de los aspectos que necesita un proyecto:

- Integración continua (test + deploy)
- Análisis de la calidad del código
- Tareas (scrum)
- Wikis
- Automatización
- Y mucho más...

## In case of fire





# Thanks!

**Ángel Cañal**

**Jorge García Pueyo**

- @CodeLabZGZ

