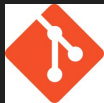


**git**

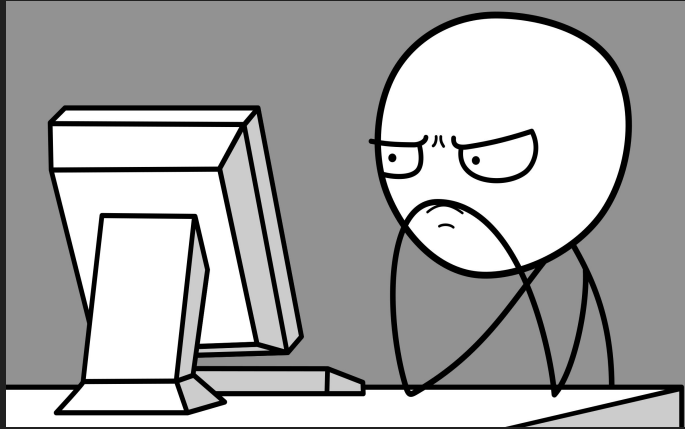


# Qué es git?

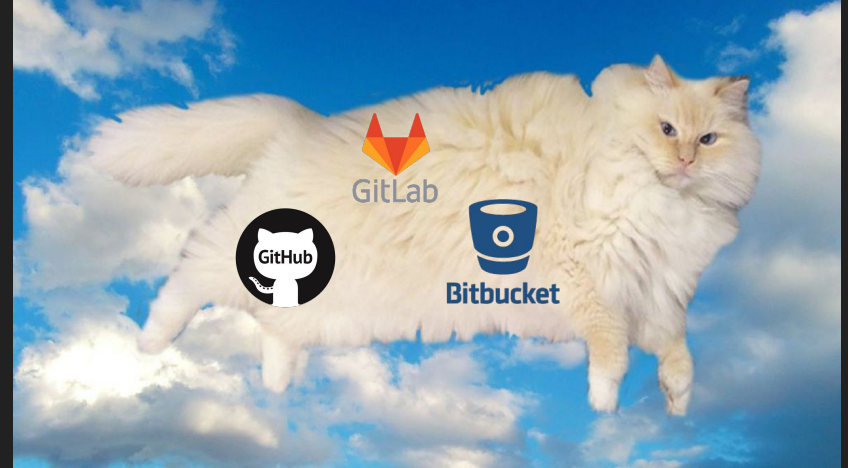
Es un software de control de versiones diseñado por Linus Torvalds.

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 puedas recuperar **versiones** específicas más adelante.

# Repo Local



# Repo Remoto (La Nube)



# GIT Workflow

- Remote Repo
- Local Repo (head)
- Index (stage)
- Workspace (working dir)



# GIT - Lo Básico

# GIT Workflow



workspace  
(working dir)



index  
(stage)



local repository  
(HEAD)



remote repository

# Clone



Add (-u)

Commit





# Commit -a



Commit -a

Push



workspace  
(working dir)



index  
(stage)



local repository  
(HEAD)



remote repository

Fetch



workspace  
(working dir)



index  
(stage)



local repository  
(HEAD)



remote repository

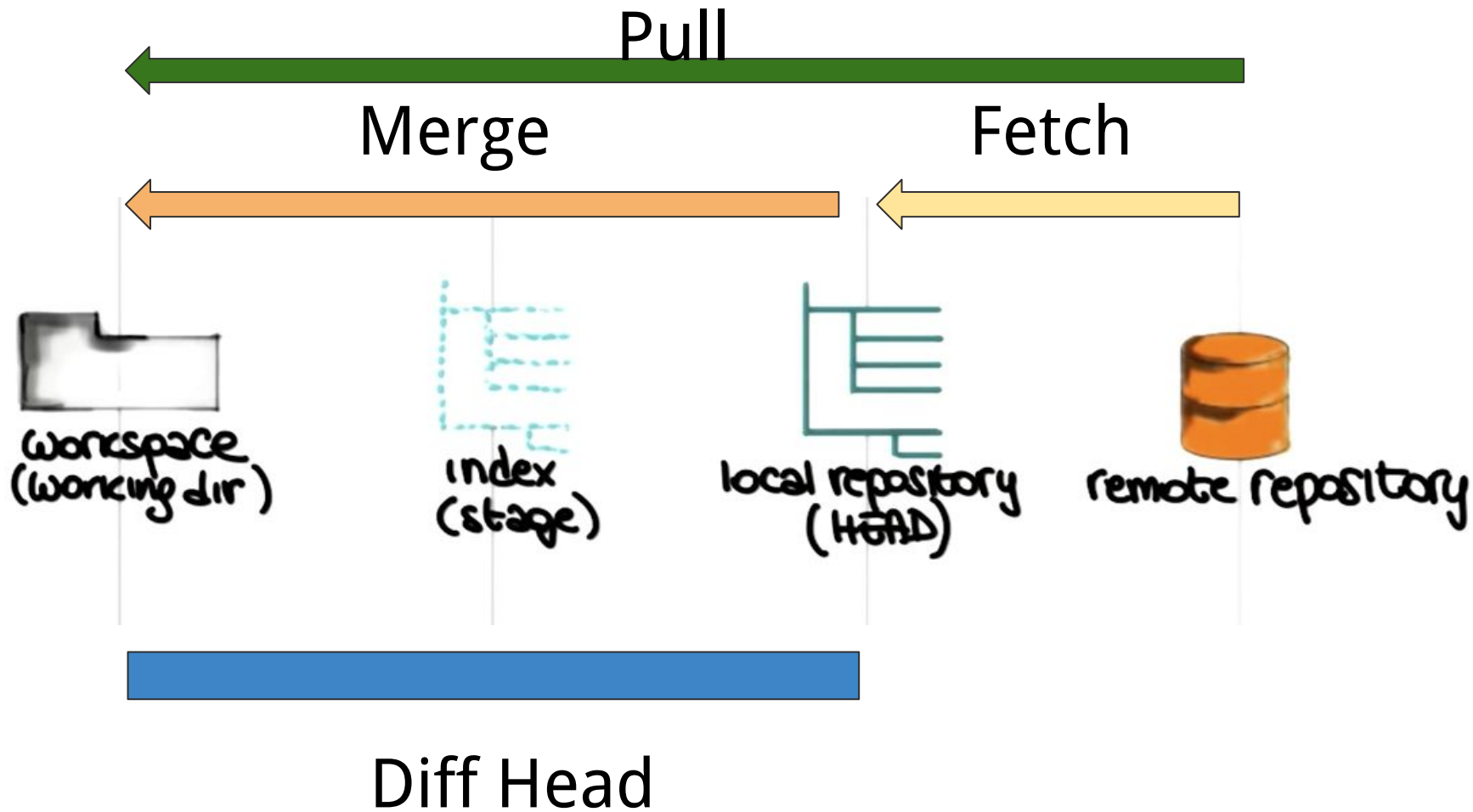
Merge

Fetch



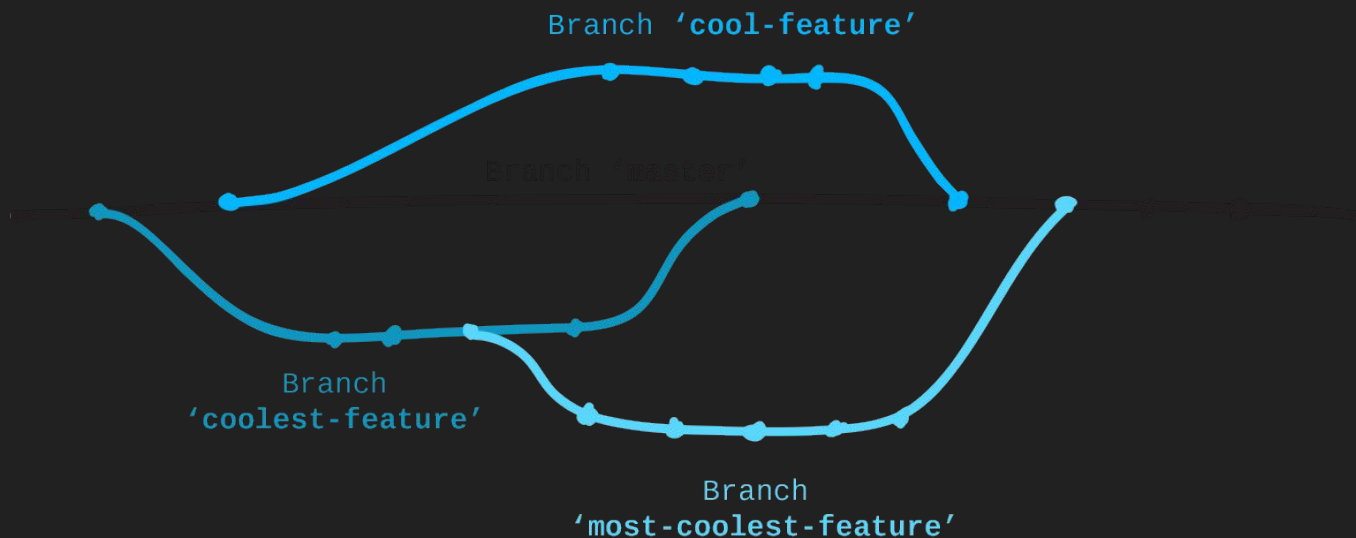
# Pull





# Que es un branch?

Dentro de nuestro sistema de control de versiones podemos ver el histórico de cambios como si de un árbol se tratase. De esta forma podemos ir abriendo ramas que parten bien de la rama principal (master) o de otra rama (branch).



La principal utilidad que tienen los branch es la de organizar nuestro trabajo, por ejemplo:

- para desarrollar una nueva funcionalidad sin afectar al máster mientras lo hacemos.

- para hacer un hotfix en una versión que ya ha salido a producción.

- para hacer un branch de producción, otro de pre, otro de testing, ... y así ir promoviendo los cambios de uno a otro.

Uno de los usos más comunes, es el de desarrollar las nuevas funcionalidades dentro de un branch, en lugar de hacerlo directamente en el master. La principal ventaja que tiene esto es que mantengo el master “limpio”.

Además esto me permite desarrollar la funcionalidad sin “estorbar” a mis compañeros, y una vez esté estable pasarlos a master para compartir los cambios con todo el equipo.

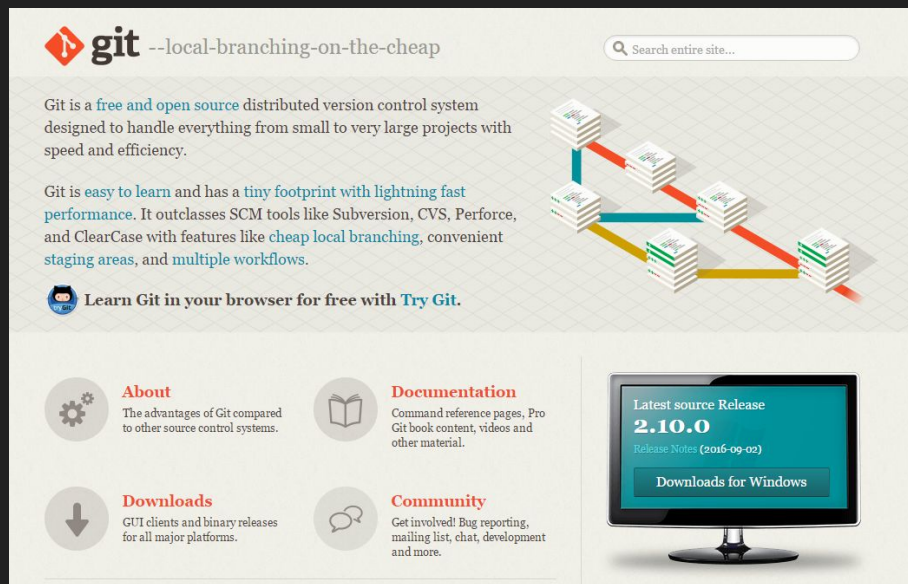


# Instalación

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

SourceTree: <https://www.sourcetreeapp.com/>

TortoiseGit: <https://tortoisegit.org/>




The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--local-branching-on-the-cheap". A search bar is located in the top right corner. The main content area features two paragraphs of text describing Git as a free and open source distributed version control system, highlighting its speed, efficiency, ease of learning, and performance. To the right of the text is a diagram illustrating branching and merging with stacks of boxes and colored lines. Below the text is a section titled "Learn Git in your browser for free with Try Git." with a GitHub logo. The bottom of the page has four circular icons with text: "About" (gear icon), "Documentation" (book icon), "Downloads" (downward arrow icon), and "Community" (speech bubble icon). On the right side, there is a monitor displaying the latest source release "2.10.0" and a button for "Downloads for Windows".

**git** --local-branching-on-the-cheap

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

 Learn Git in your browser for free with **Try Git**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.10.0**  
Release Notes (2016-09-02)  
Downloads for Windows

# Configuración en Unity

1. Cambiar a **Visible Meta Files** en:  
Edit → Project Settings → Editor → Version Control Mode.
2. Cambiar a **Force Text** en  
Edit → Project Settings → Editor → Asset Serialization Mode.

# CLONE

`git clone /path/to/repository`

- Crea una copia local del repositorio

`git clone username@host: /path/to/repository`

- Crea una copia local si estás usando un repositorio remoto



# STATUS

## git status

Muestra cual es la situación actual con nuestro repositorio local.

master



```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   public/index.html
#       deleted:    public/sitemap.xml
#       new file:   public/stylesheets/mobile.css
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    app.rb
#       deleted:    test/add_test_crash_report.sh
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       public/javascripts/
```

Working  
Directory

Index

HEAD

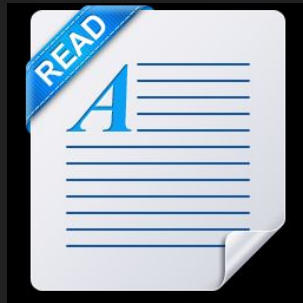
NADA

NADA

NADA

# Agregamos algo de trabajo

Tenemos nuestro proyecto vacío. Le queremos agregar un **Readme.txt** con una pequeña descripción inicial. Lo escribimos y lo guardamos en la carpeta de nuestro proyecto.



Working  
Directory

Index

HEAD

**Readme.txt**

NADA

NADA

# ADD

`git add`

Y ahora le tenemos que decir a git, “git este archivo tiene que ser parte de mi repositorio”.

`git add readme.txt`

o

`git add .` (para agregar TODO lo que haya en la carpeta)



Working  
Directory

Index

HEAD

**Readme.txt**

**Readme.txt**

**NADA**

# COMMIT

`git commit -m "Descripción de este commit"`

- Confirma los cambios realizados. El “mensaje” generalmente se usa para asociar al *commit* una breve descripción de los cambios realizados. Ahora el archivo está incluido en el HEAD, pero aún no en tu repositorio remoto.

Working  
Directory

Index

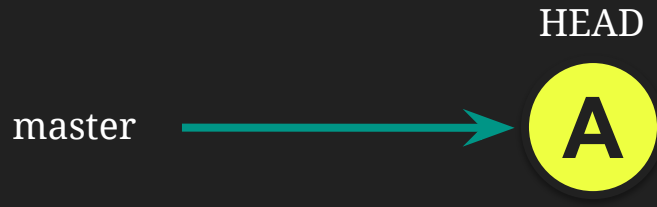
HEAD

**Readme.txt**

**Readme.txt**

**Readme.txt**

# Nuestro primer Commit



El Readme.txt que agregamos

# PUSH

`git push`

Ahora el archivo está en tu repositorio remoto.



# GIT Basics

## git fetch

- Descarga los cambios realizados en el repositorio remoto.

## git merge <nombre\_rama>:

- Impacta en la rama en la que te encuentras parado, los cambios realizados en la rama “nombre\_rama”.

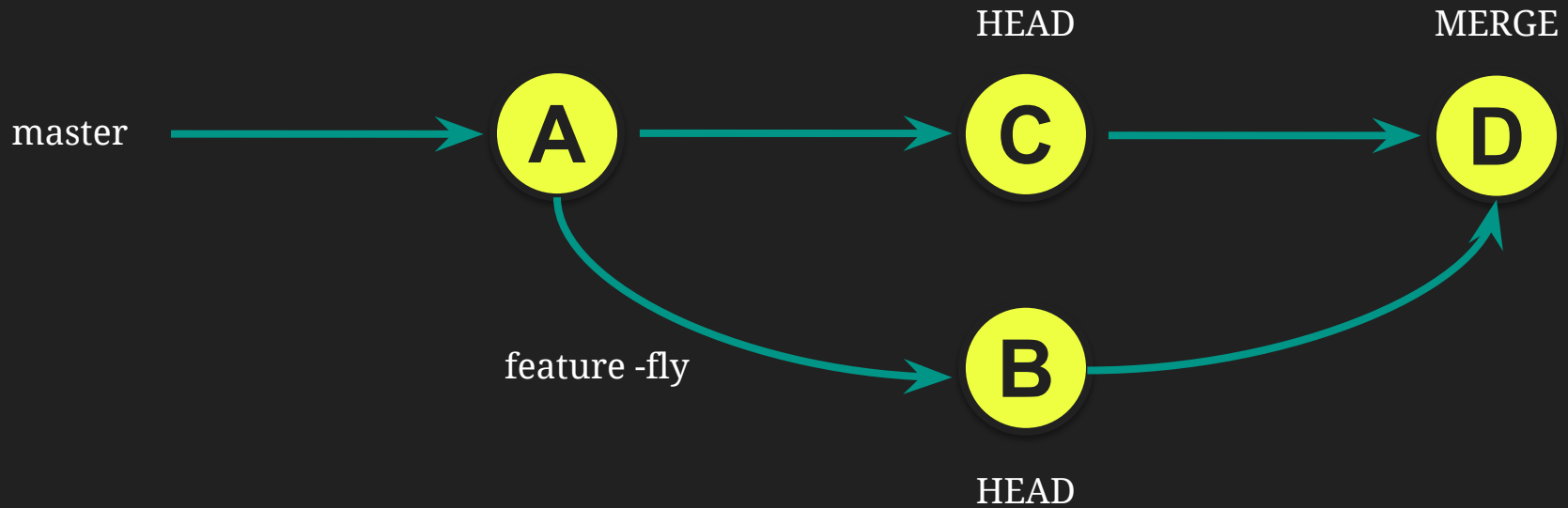
## git pull:

- Unifica los comandos *fetch* y *merge* en un único comando.

# Fetch



# Merge





# GIT Basics

`git checkout -b <nombre_rama_nueva>`

- Crea una rama a partir de la que te encuentres parado con el nombre “nombre\_rama\_nueva”, y luego salta sobre la rama nueva, por lo que quedas parado en ésta última.

`git checkout -t origin/<nombre_rama>`

- Si existe una rama remota de nombre “nombre\_rama”, al ejecutar este comando se crea una rama local con el nombre “nombre\_rama” para hacer un seguimiento de la rama remota con el mismo nombre.

`git branch`

- Lista todas las ramas locales.

`git branch -a`

- Lista todas las ramas locales y remotas.

# GIT Basics

## git stash

- Es un guardado rápido y provisional, lo utilizamos cuando queremos cambiar de rama durante un breve tiempo para ponerse a trabajar en algún otro tema urgente

## git reset --hard origin/master

- Reseteamos el HEAD a un estado específico.

## git rebase

- Re-aplicamos los commits en la parte superior de otra base.

# Stash

