**Git**

**Temes**

* [Introducció](https://jonmircha.com/git#introducci%C3%B3n)
* [Configuració inicial](https://jonmircha.com/git#configuraci%C3%B3n-inicial)
* [Flux bàsic](https://jonmircha.com/git#flujo-b%C3%A1sico)
* [De *màster* a *main*](https://jonmircha.com/git#de-master-a-main)
* [Ajuda](https://jonmircha.com/git#ayuda)
* [Ignorar fitxers](https://jonmircha.com/git#ignorar-archivos)
* [Clonar repositoris](https://jonmircha.com/git#clonar-repositorios)
* [Branques](https://jonmircha.com/git#ramas)
* [Fusions](https://jonmircha.com/git#fusiones)
* [Canvis](https://jonmircha.com/git#cambios)
* [Registre de l'historial](https://jonmircha.com/git#registro-del-historial)
* [Reseteig de l'historial](https://jonmircha.com/git#reseteo-del-historial)
* [Resetejar un repositori](https://jonmircha.com/git#resetear-un-repositorio)
* [Remots](https://jonmircha.com/git#remotos)
* [Etiquetes](https://jonmircha.com/git#etiquetas)
* [*GitHub Pages*](https://jonmircha.com/git#github-pages)
* [Col·laboració a *GitHub*](https://jonmircha.com/git#colaboraci%C3%B3n-en-github)
* [Aprèn més](https://jonmircha.com/git#aprende-m%C3%A1s)

**Introducció**

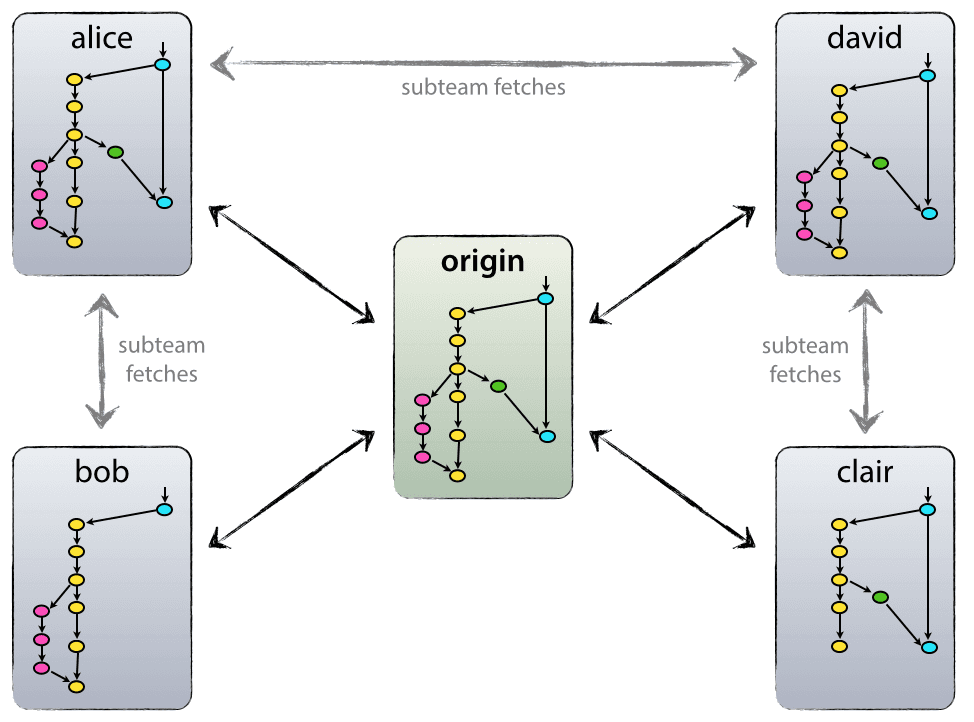
[*Git*](https://git-scm.com/) és un *programari* de control de versions distribuït i descentralitzat que permet a un equip de desenvolupadors treballar sobre el mateix codi .

S'anomena **" distribuït "** perquè cada membre de l'equip disposa d'una còpia completa del codi .

Els membres de l'equip poden enviar-se codi , rebre'l i desenvolupar funcionalitats de manera conjunta i separada del servidor central.

Algunes avantatges d' usar-lo :

* És l' estàndard actual.
* Codi col·laboratiu , versionat i distribuït .
* Recuperació d' arxius .
* Major control.
* *Shorcuts* i *plugins* .
* Millora la productivitat .

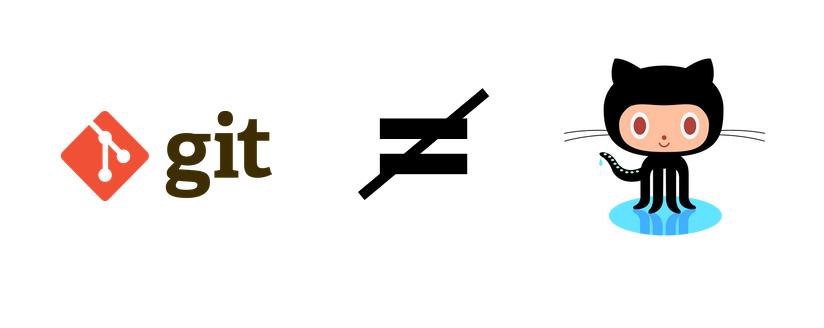


**Instal·lació :**

* [*Git*](https://git-scm.com/downloads) .
* Interfícies gràfiques :
  + [*Source Tree*](https://www.sourcetreeapp.com/) .
  + [*GitHub Desktop*](https://desktop.github.com/) .
  + [*GitKraken*](https://www.gitkraken.com/) .
  + [*Visual Studio Code*](https://code.visualstudio.com/#built-in-git) .
  + etc.

**Plataformes *web* que treballen amb *Git* :**

* [*GitHub*](https://github.com/) .
* [*GitLab*](https://gitlab.com/) .
* [*BitBucket*](https://bitbucket.org/) .
* etc.



**Configuració inicial**

**Configurant *Git* per primera vegada**

git -- version

git config --global user.name "Jonathan MirCha "

git config --global user.email jonmircha@gmail.com

git config --global user.ui true

git config --global init.defaultBranch main

git config -- list

# assignant visual studio code com a editor de configuració de git

git config --global core.editor " code -- wait "

git config --global -e

# per estandarditzar els salts de línia a windows

git config --global core.autocrlf true

# per estandarditzar els salts de línia en linux /mac

git config --global core.autocrlf input

# veure totes les opcions de la configuració a la terminal

git config -h

# veure totes les opcions de la configuració al navegador

git help config

**Inicialitzar *Git* en un directori local**

mkdir carpeta

cd carpeta

touch README.md

touch . gitignore

git init

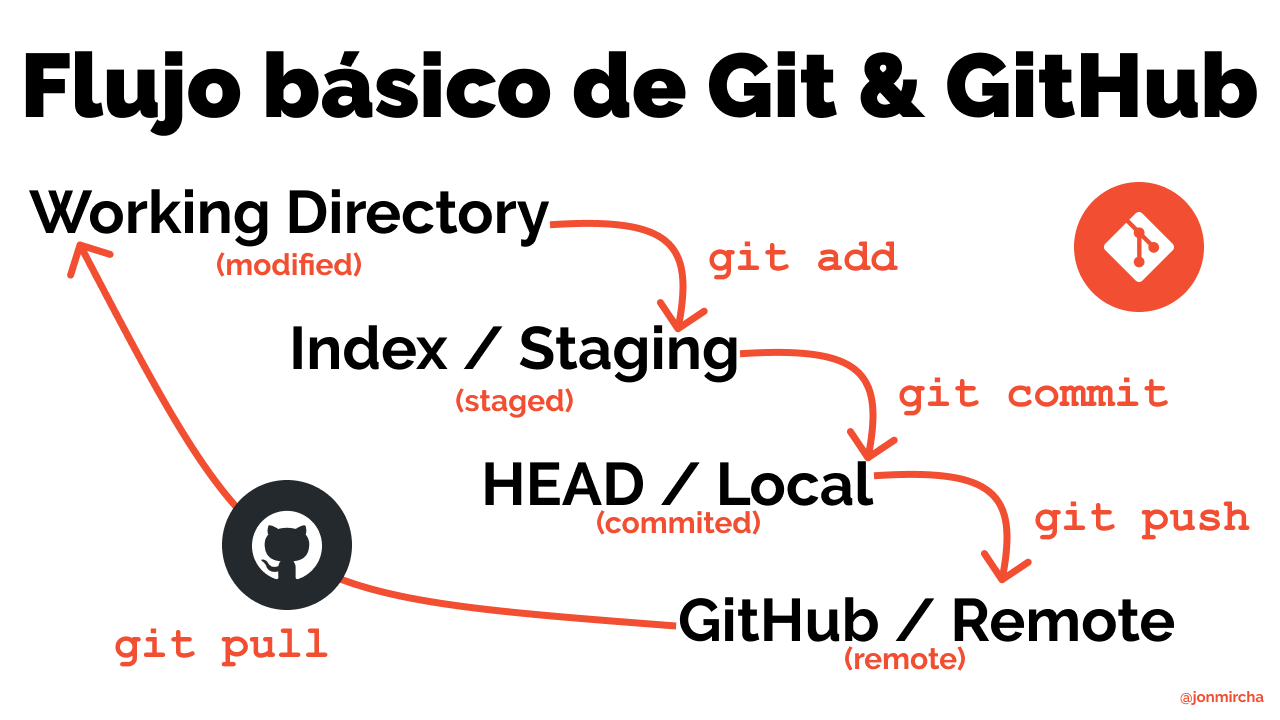
code .

**Flux bàsic**

El flux de *Git* , consta de tres estats locals , és a dir a l'ordinador on s'està treballant i un més de forma remota quan accedim al codi centralitzat en plataformes com *GitHub* , *Gitlab* , *Bitbucket* , etc.

Dites estats són ***modified*** , ***staged*** , ***committed*** i ***remote*** . A cadascun d' ells li correspon una àrea de treball :

1. ***Working Directory*** : És l' àrea corresponent a l'estat ***modified*** i és la carpeta local del teu ordinador on emmagatzemes els fitxers del teu projecte .
2. ***Staging Àrea*** : És l' àrea corresponent a l'estat ***staged*** també se'n diu index perquè és l' ***àrea*** on *git* indexa i afegeix els canvis realitzats als arxius previs a comprometre'ls en el registre.
3. ***Local Repository*** : És l' àrea corresponent a l'estat ***committed*** , on els canvis ja s'han registrat al repositori de *git* també se'n diu HEAD perquè indica en ***què*** canvi es troba el punter del repositori .
4. ***Remot Repository*** : És l' àrea corresponent a l'estat ***remot i*** és el directori remot on emmagatzemem els arxius del projecte en alguna plataforma *web* com *GitHub* , *GitLab* , *BitBucket* . *Git* denomina ***origin*** al repositori remot .



# afegir els canvis d'un fitxer a l'staged

git add arxiu / directori

# afegir tots els canvis de tots els fitxers a l'staged

git add .

# els canvis són compromesos al repositori

# has de escriure el missatge del canvi

# quan s'obri el fitxer de configuració \_

# en acabar guarda i tanca el fitxer

# perquè els canvis tinguin efecte

git commit

# és un shortcut de l'ordre anterior

# escrius i confirmes el missatge del canvi en un sol pas

git commit -m " missatge descriptiu del canvi "

# s'afegeix l'origen remot del vostre dipòsit de GitHub

git remote add origin https://github.com/usuari/repositorio.git

# la primera vegada que vinculem el repositori remot amb el local

git push -u origin màster

# per a les subseqüents actualitzacions , sinó canvies de branca

git push

#per descarregar els canvis del repositori remot al local

git pull

**De *màster* a *main***

Amb els desafortunats esdeveniments del 25 de maig de 2020 als Estats Units que van culminar amb l' assassinat de l' afroamericà [*George Floyd*](https://es.wikipedia.org/wiki/Muerte_de_George_Floyd) a mans de policies de la ciutat de *Mineàpolis* , es va intensificar de manera global el moviment [*#BlackLivesMatter*](https://es.wikipedia.org/wiki/Black_Lives_Matter) . [*\_*](https://es.wikipedia.org/wiki/Black_Lives_Matter)

Amb dit moviment moltes indústries i empreses van començar a prendre accions per eradicar el racisme .

A la indústria de la tecnologia per anys s'han emprat paraules com *master* , *slave* , *whitelist* , *blacklist* entre d'altres que actualment no són bé vistes pel context i la semàntica que impliquen .

Sobre això *Microsoft* empresa propietària de *GitHub* va decidir començar una campanya per reemplaçar el nom de la branca principal dels dipòsits de *màster* a *main* ; com ho han explicat en aquest [document](https://github.com/github/renaming) :

"L'1 d'octubre del 2020, qualsevol nou repositori que creïs utilitzarà ' *main* ' com la branca per defecte , en lloc de ' *master* '. Aquest canvi no afecta cap dels teus repositoris existents : els repositoris existents continuaran tenint la mateixa branca per defecte que tenen ara ”.

Aquest canvi implica afegir un parell de línies d'ordres addicionals per crear la branca ' *main* ' i fer-la principal al repositori .

Aleshores el flux bàsic quedaria de la següent manera:

**Per a repositoris nous**

git init

git add .

git commit -m "Primer commit "

git branch -M main

git remote add origin https://github.com/usuari/repositorio.git

git push -u origin main

**Per a repositoris existents**

git branch -M main

git remote add origin https://github.com/usuari/repositorio.git

git push -u origin main

**Per reemplaçar la branca *master* per *main* a *GitHub***

# Pas 1

# Crea la branca local main i passa-li l'historial de la branca master

git branch -m master main

# Pas 2

# Fes un push de la nova branca local main al repositori remot de GitHub

git push -u origin main

# Pas 3

# Canvia el HEAD actual a la branca main

git symbolic-ref refs /remotes/ origin /HEAD refs /remotes/ origin / main

**Pas 4**

Canvia la branca *default* de *master* a *main* al teu repositori de *GitHub* .

Per fer-ho , seguiu les instruccions d'aquest [enllaç](https://docs.github.com/en/github/administering-a-repository/setting-the-default-branch) .

# Pas 5

# Elimina la branca màster del repositori remot

git push origin -- delete màster

**Per reemplaçar la branca *master* per *main* a *Git***

git config --global init.defaultBranch main

**Ajuda**

# ajuda a la terminal

git ordre -h

# ajuda al navegador

git help comanda

**Ignorar fitxers**

A l' arxiu ***. gitignore*** incloem tot el que **NO** vulguem incloure al nostre repositori . El podem crear manualment o amb [*gitignore.io*](https://www.gitignore.io/) .

# això és un comentari

fitxer.ext

carpeta

/arxiu\_desde\_raiz.ext \_

# ignorar tots els fitxers que acabin a . log

\*. log

# excepte production.log

!production.log

# ignorar els fitxers acabats a . txt dins de la carpeta doc,

# però no en els seus subcarpetes

doc/\*. txt

# ignorar tots els fitxers acabats a . txt dins de la carpeta doc

# i també als seus subcarpetes

doc/\*\*/\*. txt

**Clonar repositoris**

git clone https://github.com/usuari/repositorio.git

**Branques**

Una branca ens permet aïllar una nova funcionalitat al nostre codi que després podrem afegir a la versió principal.

# crear branca

git branch nom-branca

# canviar de branca

git checkout nom-branca

# crear una branca i canviar- t'hi

git checkout -b branca

# eliminar branca

git branch -d nom-branca

# eliminar branques remotes

git push origin -- delete nom-branca

#eliminar branca ( forçat )

git branch -D nom-branca

# llistar totes les branques del repositori

git branch

# llista branques no fusionades a la branca actual

git branch --no- merged

# llista branques fusionades a la branca actual

git branch -- merged

# depassar branques

git checkout branca-secundària

git sobrepassa branca-principal

**Fusions**

Uneix dues branques . Per fer una fusió necessitem :

1. Situar-nos a la branca que es quedarà amb el contingut fusionat .
2. Fusionar.

Quan es fusionen branques es poden donar 2 resultats diferents :

* ***Fast- Forward*** : La fusió es fa automàtica , no hi ha conflictes per resoldre .
* ***Manual Merge*** : La fusió cal fer - la manual, per resoldre conflictes de duplicació de contingut .

# ens canviem a la branca principal que quedarà de la fusió

git checkout branca-principal

# executem l'ordre merge amb la branca secundària a fusionar

git merge branca-secundària

**Canvis**

Pots afegir modificacions al darrer canvi

# sense editar el missatge del darrer commit

git commit -- amend --no- edit

# editant el missatge del darrer commit

git commit -- amend -m " nou missatge per a l' últim commit "

# eliminar l' últim commit

git reset -- hard HEAD~1

Podem desplaçar-nos a l'historial del repositori cap enrere o endavant en canvis o branques , sense afectar el repositori com a tal.

# canviar a una branca

git checkout nom-branca

# canviar a un commit en particular

git checkout id-commit

**Registre de l'historial**

**git log** ens permet conèixer tot l'historial d'un projecte , amb la informació de la data , l'autor i aneu de cada canvi .

git log

# mostra en una sola línia per canvi

git log -- oneline

# desa el log a la ruta i fitxer que especifiquem

git log > commits.txt

# mostra l'historial amb el format que indiquem

git log -- pretty =format:"%h - % an , % ar : %s"

# canviem la n per qualsevol nombre sencer i mostrarà els n canvis recents

git log -n

# mostra els canvis realitzats després de la data especificada

git log -- after ="2019-07-07 00:00:00"

# mostra els canvis realitzats abans de la data especificada

git log -- before ="2019-07-08 00:00:00"

# mostra els canvis realitzats al rang de data especificat

git log -- after ="2019-07-07 00:00:00" -- before ="2019-07-08 00:00:00"

# mostra una gràfica de l'historial de canvis , branca i fusions

git log -- oneline -- graph --all

# mostra tot el registre d'accions del log

# incloent insercions , canvis , eliminacions , fusions, etc.

git reflog

# diferències entre el Working Directory i el Staging Àrea

git diff

**Reseteig de l'historial**

Podem eliminar l'historial de canvis del projecte cap endavant pel que fa a un punt de referència.

#ens mostra el llistat de fitxers nous ( untracked ), esborrats o editats

git status

# esborra HEAD

git reset -- soft

# esborra HEAD i Staging

git reset -- mixed

# ho esborra tot : HEAD, Staging i Working Directory

git reset -- hard

# desfà tots els canvis després del commit indicat , preservant els canvis localment

git reset id-commit

# rebutja tot l'historial i torna al commit especificat

git reset -- hard id-commit

**Resetejar un repositori**

Si en algun moment tens la necessitat de resetejar l'historial de canvis d'un repositori perquè quedi com si l' acabaràs de crear executa aquesta sèrie d'ordres:

cd carpeta- repositori

mv .git/ config ~/ saved\_git\_config

rm - rf .git

git init

git branch -M main

git add .

git commit -m " Commit inicial"

mv ~/ saved\_git\_config .git/ config

git push -- force origin main

**Remots**

# mostra els orígens remots del repositori

git remote

# mostra els orígens remots amb detall

git remote -v

# afegir un orígen remot

git remote add nom- origen https://github.com/usuari/repositorio.git

# reanomenar un orígen remot

git remote rename nom- vell nom- nou

# eliminar un orígen remot

git remote remove nom- origen

# descarregar una branca remota a local diferent de la principal

git checkout -- track -b branca-remota origin /branca-remota

**Etiquetes**

Amb aquesta opció *git* ens permet versionar nostre codi , llibreria o projecte .

# llistar etiquetes

git tag

# crea una etiqueta

git tag numero- versió

# eliminar una etiqueta

git tag -d numero- versió

# mostrar informació d'una etiqueta

git show numero- versió

# sincronitzant l'etiqueta del repositori local al remot

git add .

git tag v1.0.0

git commit -m "v1.0.0"

git push origin numero- versió

# generant una etiqueta anotada (amb missatge de commit )

git add .

git tag -a "v1.0.0" -m " Missatge de l'etiqueta"

git push -- tags

***GitHub Pages***

[**gh-pages**](https://pages.github.com/) és una branca especial per crear un lloc *web* al teu projecte allotjat directament al vostre repositori de *GitHub* .

* *URL* del repositori : [**https://github.com/usuari/repositori**](https://github.com/usuario/repositorio)
* *URL* del lloc : [**https://usuari.github.io/repositorio**](https://usuario.github.io/repositorio)

Per crear aquesta branca especial a *GitHub* executem les següents ordres:

git branch gh-pages

git checkout gh-pages

git remote add origin https://github.com/usuari/repositorio.git

git push origin gh-pages

# per descarregar els canvis del repositori remot al local

git pull origin gh-pages

**Col·laboració a *GitHub***

Per poder col·laborar en projectes allotjats a *GitHub* necessitem fer ús dels *forks* i *pull requests* , eines que ens ofereix la plataforma per a aquest objectiu .

A continuació descric el procés de col·laboració a *GitHub* .

1. *Forkea* el repositori on vulguis col·laborar , per fer-ho , segueix les instruccions d'aquest [enllaç](https://docs.github.com/en/get-started/quickstart/fork-a-repo) .
2. Un cop *forkejat* el repositori al teu compte de *GitHub* , clona'l al teu equip de còmput .
3. Al repositori local cal configurar els orígens remots de la teva nova copia per tenir tots dos remots , els originals ( *origin* ) i els de la teva còpia, per fer-ho , segueix les instruccions d'aquest [enllaç](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/configuring-a-remote-for-a-fork) .
4. Crea una branca nova al teu *fork* local per fer la teva col·laboració , i sincronitza-la amb el teu repositori remot , per fer-ho , seguiu les instruccions d'aquest [enllaç](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/syncing-a-fork) .
5. Configura el teu repositori perquè vaig acceptar canvis ( *pull requests* ), per fer-ho , segueix les instruccions d'aquest [enllaç](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/allowing-changes-to-a-pull-request-branch-created-from-a-fork) .
6. Crea una *pull request* , per fer-ho , segueix les instruccions d'aquest [enllaç](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request) .
7. Espereu que l' amo del dipòsit original accepti els vostres canvis .
8. Quan acceptin el teu *pull request* , és recomanable que esborris la branca en què vas treballar el canvi i actualitzis el teu repositori *forkeado* , amb els canvis del repositori original.

Annex un resum de les ordres a executar per col·laborar en un repositori de *GitHub* :

# forkear repositori

git clone https://github.com/usuari/repositorio.git

git remote -v

git remote rename origin fork

git remote add origin https://github.com/usuari/repositorio.git

git checkout -b branca- nova

git push fork rama nova

# sol·licitar el pull request

# acceptar el pull request

git checkout main

git pull origin main

git push fork main

git branch -d branca- nova   
git push fork -- delete branca- nova

**Aprèn més**

A continuació et deixo alguns enllaços on pots aprofundir els teus coneixements sobre *Git* i *GitHub* :

* [*Git* - la guia senzilla](http://rogerdudler.github.io/git-guide/index.es.html) .
* [Llibre *Pro Git*](https://git-scm.com/book/es) .
* [Guies oficials de *GitHub*](https://docs.github.com/es) .