# Introducció a Git

## Què és Git?

Git és un sistema de control de versions que ens ajuda a seguir els canvis al codi, col·laborar amb altres desenvolupadors i gestionar diferents versions d'un projecte.

És un protocol que opera en un entorn distribuït, és a dir cada usuari té una còpia completa del codi sobre la qual pot treballar i després tots els membres del equip, poden sincronitzar els canvis en un servidor central.

## Per què usar Git?

### Històric de canvis:

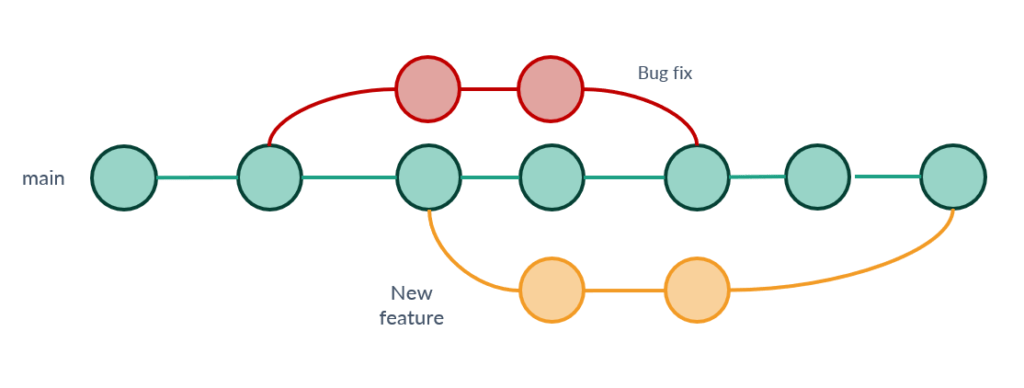
Imagina que afegeixes una funcionalitat que no funciona i vols tornar a la versió anterior. Amb Git, pots fer-ho fàcilment.

### Col·laboració:

Treballar en equip en el mateix fitxer sense sobrescriure el treball dels altres.

### Gestió de versions:

Permet tenir diverses versions del projecte actives (producció, desenvolupament, noves funcionalitats).



## Instal·lació de Git

Descarregar aplicació GIT ([git-scm.com/downloads](https://git-scm.com/downloads))

## Instal·lació VSCode

Descarregar l’aplicació VSCode ([code.visualstudio.com/download](https://code.visualstudio.com/download))

## Configuració Git

git config –global user.name “Jordà Garcia Ametller”

git config –global user.email jorda.garcia-ametller@t-systems.com

git config --global core.editor "code --wait" (configurem el editor VSCode)

git config --global -e (visalitzem el fitxer de configuració .gitconfig)

# Primer repte: Inicialitzar i treballar en local

## Què significa inicialitzar un repositoris?

En Git, inicialitzar un repositori crea una estructura interna per a gestionar les versions del projecte, anomenada .git. Aquest directori és on Git emmagatzema tota la informació sobre els canvis realitzats.

Cada membre del equip tindrà aquesta carpeta en local, en el seu directori vinculat al repositori git. **Aquesta carpeta és única de cada membre de l’equip, per tant, no s’ha de sincronitzar al repositoris remot**.

## Pas a pas per inicialitzar el repositoris local

1. Crear un directori nou:

*mkdir myProject*

*cd myProject*

1. Inicialitzar el repositoris

*git init*

1. Creem un fitxer nou, des del VSCode o des de comandes

*echo “Hello World” > file1.txt*

*code file1.txt (per obrir el fitxer amb l’editor preconfigurat)*

1. Comprovem l’estat del nostre repositoris

*git status*

## Stage Area

Existeix una àrea de pas “Stage Àrea” que és on posarem els fitxers que vulguem comprometre (commit).

1. Afegir fitxer a l’àrea “Stage”

*git add file1.txt*

Cada cop que fem algun canvi d’algun arxiu en la fase “stage” (on els arxius encara no s’han compromès “commit”), els haurem de tornar a afegir a la fase “stage” amb la comanda “git add <nom del fitxer>”.

1. Comprovem ara l’estat del fitxer

*git status*

1. Comprometem els canvis

*git cimmit -m “Primer commit – afegit file1.txt”*

Un commit és un punt de control que conté tots els canvis que has afegit a l'índex (staging area). Git guarda aquest commit juntament amb un missatge explicant què s'ha canviat.

1. Comprovem l’estat del nostre repositoris

*git status*

1. Visualitzem l’historial de commits

*git log*

Mostra una llista de tots els commits realitzats en el repositori, amb detalls com l’identificador (hash), el nom de l’autor i el missatge del commit.

1. Per eliminar un fitxer del nostre repositoris utilitzem la comanda

*git rm <filename>*

Amb aquesta acció estem eliminant el fitxer de la nostra àrea “stage”. Si comprometem els canvis, el fitxer s’eliminarà i no podrem revertir els canvis

1. En cas de que no comprometem els canvis, podem recuperar el fitxer en la nostra àrea “stage”, utilitzant la comanda

*git restore file1.txt*

Aquesta comanda el que ens ha fet ha estat “desfer” l’acció d’esborrar el fitxer en el nostre estat “stage” però si executem la comanda “ls” veurem que el fitxer NO existeix en el nostre directori perquè l’hem eliminat. Per tant, per recuperar el fitxer eliminat, tornem a executar la comanda *git restore <filename>*

## .gitignore

Com obviar fitxers que no volem sincronitzar amb el git:

1. Crear un fitxer .gitignore
2. Especificar els fitxers que volem ometre o directoris
3. Afegir els canvis a la fase “stage” git add .gitignore
4. Comprometre els canvis amb git commi -m <comentari>

# Treball col·laboratiu amb branques

Anem a modificar un fitxer existent en el nostre repositoris i a comprovar les diferències amb la darrera versió compromesa (commited), per observar com git gestiona les diferències de codi en un mateix fitxer.

1. Afegim nou contingut al fitxer existent file1.txt

*echo "Afegint més contingut al fitxer" >> fitxer1.txt*

1. Comprovem l’estat del repositoris

*git status -s*

1. Anem a comprovar les diferències entre el fitxer modificat i el seu darrer commit

*git diff*

Com podem observar, git el que fa és mantenir la primera línia que ja existia en el fitxer i seguidament crea la segona línia del fitxer.

1. Afegim els canvis a la nostra àrea “stage” i comprometem els canvis

*git add file1.txt*

*git commit -m “Afegit contingut addicional file1.txt”*

1. Comprovem l’historial de commits

*git log*

## Què són les branques?

Les branques permeten treballar en funcionalitats o canvis de manera aïllada sense afectar el codi principal. Així es poden fer canvis de manera segura, fer proves, i fusionar-los més tard.

1. Crear una nova branca a partir de la branca master

*git branch feature/task1*

Per crear una nova branca, hem de situar el HEAD del repositoris sobre la branca que colem “clonar”.

1. Per canviar de branca i situar el HEAD del repositoris sobre la nova branca creada, utilitzem la comanda

*git switch feature/task1*

1. Comprovem les branques existents

git branch

1. Fer un canvi en el fitxer file1.txti comprometre el fitxer (commit)

*git add file1.txt*

*git commit -m “Afegir nous canvis a fitxer file1.txt”*

1. Comprovem el log (opció resumida)

*git log --oneline*

1. Per comprovar el contingut d’un commit, utilitzem el hash que identifica el commit

*git show <hash>*

# Treballar amb un repositoris remot

## Repositori remot

El repositori remot és una còpia del repositori que s'emmagatzema en un servidor (com GitHub, GitLab o Bitbucket). Treballar amb un repositori remot permet col·laborar amb altres desenvolupadors.

## Clonar repositoris

Tenim aquest repositori remot públic <https://github.com/balistrieri/myweb.git>

1. Creem una nova carpeta local i clonem el repositoris

*mkdir myweb\_remote*

1. Clonem el repositoris remot al nostre directori local

*git clone* [*https://github.com/balistrieri/myweb.git*](https://github.com/balistrieri/myweb.git)

## Pujar canvis

La comanda *git push* envia els canvis a la teva branca local al repositoris remot.

1. Afegeix un canvi a un dels fitxers del repositoris clònic (per exemple afegir una línia al fitxer info.txt)

*code info.txt (opció 1)*

*echo “I’m here, working” >> info.txt (opció 2)*

*git add info.txt*

*git commit -m “Afegim una línia nova a info.txt”*

1. Pugem els canvis al repositoris remot

*git push origin master (indiquem en quina branca volem pujar els canvis, recomanable)*

*git push (pugem els canvis a la branca existent del repositoris configurat per defecte)*

## Baixar canvis

La comanda *git pull* actualitza el repositoris local amb els canvis més recents del repositoris remot.

Aquesta comanda, realment està realitzant dues instruccions:

* *git fetch*: baixa els canvis al repositoris local
* *git merge*: integra aquests canvis al repositoris local

A partir d’un canvi manual afegit al repositoris remot (un fitxer nou):

1. Baixar els canvis

*git pull origin master*

1. Comprovar que s’han baixat els nous canvis

## Gestionar conflictes

Un conflicte succeeix quan dos desenvolupadors fan canvis en les mateixes línies d’un fitxer i intenten fusionar-los. Git necessita que resolguem el conflicte manualment.

1. S’ha afegit una nova línia en el fitxer user1File.txt en el repositoris remot.
2. Un desenvolupador modifica la mateixa línia del fitxer localment.

*code userFile.txt*

*git add userFile.txt*

*git commit -m “Nova línia afegia al fitxer userFile1.txt”*

1. Intentem pujar els canvis al repositori remot

*git push origin master (ens mostrarà un error de sincronització perquè el contingut el fitxer no és el mateix)*

1. Baixem els darrers canvis del repositori remot

*git pull origin master (detecta conflicte a resoldre)*

1. Resolem el conflicte

*code userFIle1.txt(avaluem les diferents opcions de resolució del conflicte)*

1. Pugem els canvis al repositori remot

*git add userFile.txt*

*git commit -m “Resolució de conflicte”*

*git push origin master*

# Resoldre conflictes i treballar amb remots

# Tancament i preguntes