Git

Índex

1. Què és Git?

Git és un sistema de control de versions (VCS, *Version Control System*) distribuït, creat l'any 2005 per **Linus Torvalds** (el creador del nucli Linux) per gestionar el desenvolupament del propi kernel després que es retirés la llicència de l'eina propietària que s'utilitzava fins aleshores (BitKeeper).

Figura 1: Git logo

Un sistema de control de versions permet:

- Registrar l'historial de canvis d'un projecte (fitxers de codi, documentació, configuració, etc.).
- Tornar a qualsevol versió anterior en qualsevol moment.
- Treballar en paral·lel amb altres persones sense sobreescriure el treball dels altres.
- Comparar versions, saber qui ha canviat què i quan, i per què.

A diferència d'altres sistemes més antics (com CVS o Subversion), que són **centralitzats** (hi ha un únic servidor amb l'historial complet i els clients només tenen una còpia de treball), Git és **distribuït**: cada còpia local del dipòsit conté tot l'historial complet del projecte. Això té avantatges importants:

- Es pot treballar sense connexió a internet (commits, branques, historial… tot és local).
- No hi ha un únic punt de fallada: qualsevol còpia clonada és una còpia de seguretat completa.
- Les operacions habituals (veure l'historial, crear branques, fer commits) són molt ràpides perquè no cal contactar amb cap servidor.

Git no s'ha de confondre amb **GitHub**, **GitLab** o **Bitbucket**: aquests són serveis web (plataformes d'allotjament de dipòsits) que utilitzen Git per sota, però hi afegeixen funcionalitats extra (interfície web, gestió d'incidències, *pull requests*, CI/CD, wikis, etc.).

És l'eina estàndard de facto per al control de versions en el desenvolupament de programari i la gestió de qualsevol projecte basat en fitxers de text (codi, configuracions, documentació…). El seu model distribuït, la rapidesa de les branques i la integritat garantida de l'historial l'han convertit en una peça imprescindible tant en entorns professionals com educatius. La instal·lació a Ubuntu és immediata mitjançant `apt`, i una configuració inicial mínima (nom, correu, editor i clau SSH) és suficient per començar a treballar-hi amb comoditat.

2. Conceptes bàsics

Concepte	Descripció
Dipòsit (repo)	Directori on Git emmagatzema tot l'historial d'un projecte (a la carpeta oculta `.git`).
Commit	Una “fotografia” de l'estat dels fitxers en un moment donat, amb un missatge descriptiu i un identificador únic (hash SHA-1).
Working directory	Els fitxers tal com els veus i edites al disc.
Staging area (índex)	Zona intermèdia on es preparen els canvis abans de confirmar-los amb un commit.
Branca (branch)	Una línia de desenvolupament independent. La branca per defecte sol anomenar-se `main` (abans `master`).
Remot (remote)	Una còpia del dipòsit allotjada en un altre lloc (per exemple, a GitHub), amb la qual se sincronitzen canvis.
HEAD	Punter que indica en quin commit/branca et trobes actualment.
Merge	Combinar els canvis de dues branques diferents.
Clone	Descarregar una còpia completa d'un dipòsit remot.

Flux de treball bàsic

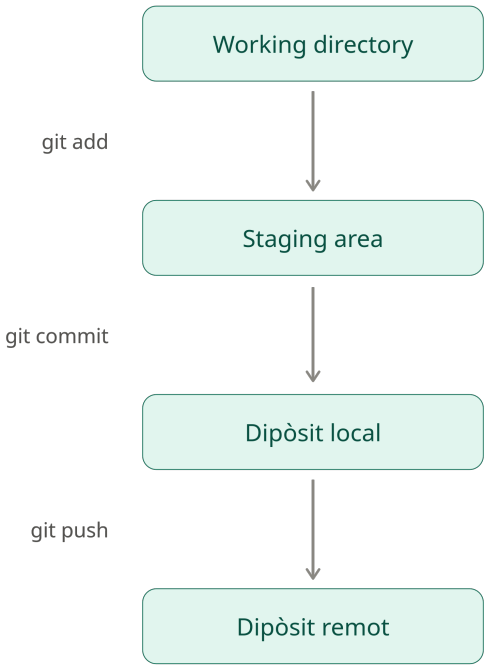

Figura 2: Flux de treball

3. Funcionalitats principals de Git

- **Control de versions distribuït**: tot l'historial és local a cada còpia.
- **Ramificació (branching) i fusió (merging) lleugers i ràpids**: crear una branca és una operació immediata i barata, cosa que fomenta treballar amb moltes branques en paral·lel (per funcionalitats, correccions d'errors, experiments…).
- **Integritat de les dades**: cada commit s'identifica amb un hash SHA-1 calculat a partir del contingut, cosa que fa pràcticament impossible modificar l'historial sense que es detecti.
- **Rendiment**: la majoria d'operacions es fan en local, sense latència de xarxa.
- **Àrea de preparació (staging/index)**: permet triar exactament què s'inclou a cada commit, encara que hi hagi més canvis al directori de treball.
- **Historial complet i eines de comparació**: `git log`, `git diff`, `git blame` per veure qui ha canviat cada línia i quan.
- **Etiquetes (tags)**: per marcar punts concrets de l'historial, típicament versions publicades (`v1.0.0`).
- **Stash**: per desar temporalment canvis sense confirmar-los, i recuperar-los més tard.
- **Ganxos (hooks)**: scripts que s'executen automàticament en certs esdeveniments (abans d'un commit, després d'un push, etc.), útils per a validacions automàtiques o integració amb CI/CD.
- **Submòduls**: per incloure altres dipòsits Git dins d'un dipòsit principal.
- **Compatibilitat amb serveis remots**: GitHub, GitLab, Bitbucket, o un servidor Git propi (per exemple, autoallotjat amb `git daemon`, Gitea o Gitolite).

4. Instal·lació a Ubuntu 26.04

Ubuntu inclou el paquet `git` als dipòsits oficials, de manera que la instal·lació és molt senzilla.

4.1. Instal·lació des dels dipòsits oficials (recomanat)

Actualitza la llista de paquets:

```
sudo apt update
```

Instal·la Git:

```
sudo apt install git
```

Comprova la versió instal·lada:

```
git --version
```

4.2. Instal·lació de l'última versió (PPA oficial del projecte)

Els dipòsits d'Ubuntu no sempre porten la versió més recent de Git. Si es necessita l'última versió estable, es pot utilitzar el PPA mantingut pels desenvolupadors de Git:

```
sudo add-apt-repository ppa:git-core/ppa -y
sudo apt update
sudo apt install git -y
git --version
```

4.3. Interfícies gràfiques opcionals

Encara que aquest document se centra en la línia d'ordres (la manera més habitual i completa de treballar amb Git), hi ha clients gràfics disponibles als dipòsits si es vol una interfície visual:

```
sudo apt install gitk git-gui
```

- `gitk`: visualitzador d'historial i branques.
- `git-gui`: eina gràfica per fer commits, veure diferències, etc.

5. Configuració inicial

Un cop instal·lat, cal configurar com a mínim el nom i el correu electrònic, ja que Git els incrusta a cada commit.

5.1. Identitat de l'usuari

```
git config --global user.name "Ramon López"
git config --global user.email "ramon@exemple.cat"
```

L'opció `--global` desa la configuració a `~/.gitconfig`, aplicant-se a tots els dipòsits de l'usuari. Sense `--global`, la configuració només s'aplica al dipòsit actual (es desa a `.git/config`).

5.2. Editor de text per defecte

Per als missatges de commit llargs o operacions com `git rebase -i`:

```
git config --global core.editor "nano"      # o "vim", "code --wait",
 ↪  etc.
```

5.3. Nom de la branca per defecte

Des de Git 2.28, es pot triar el nom de la branca inicial (habitualment `main` en lloc de l'antic `master`):

```
git config --global init.defaultBranch main
```

5.4. Colors a la terminal

```
git config --global color.ui auto
```

5.5. Àlies útils

```
git config --global alias.st status
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.cm "commit -m"
git config --global alias.lg "log --oneline --graph --all --decorate"
```

5.6. Comprovar tota la configuració

```
git config --list
git config --list --show-origin   # mostra a quin fitxer prové cada
 ↪  valor
```

Els fitxers de configuració es llegeixen en aquest ordre de prioritat (de menor a major):

1. `/etc/gitconfig` --- configuració de tot el sistema (`git config --system`).
2. `~/.gitconfig` o `~/.config/git/config` --- configuració de l'usuari (`git config --global`).
3. `.git/config` --- configuració específica del dipòsit (`git config --local`, per defecte).

5.7. Autenticació amb dipòsits remots (SSH)

Per treballar amb GitHub/GitLab sense haver d'introduir la contrasenya cada vegada, es recomana generar i configurar una clau SSH:

```
ssh-keygen -t ed25519 -C "ramon@exemple.cat"
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
cat ~/.ssh/id_ed25519.pub    # copiar i enganxar a GitHub/GitLab
 ↪  (Settings > SSH Keys)
```

Comprovar la connexió:

```
ssh -T git@github.com
```

6. Ordres bàsiques d'ús diari

```
# Crear un dipòsit nou
git init

# Clonar un dipòsit existent
git clone https://github.com/usuari/projecte.git

# Veure l'estat del directori de treball
git status

# Afegir canvis a l'àrea de preparació (staging)
git add fitxer.txt
git add .                     # afegeix tots els canvis

# Confirmar els canvis (crear un commit)
git commit -m "Missatge descriptiu del canvi"

# Veure l'historial de commits
git log
git log --oneline --graph --all
```

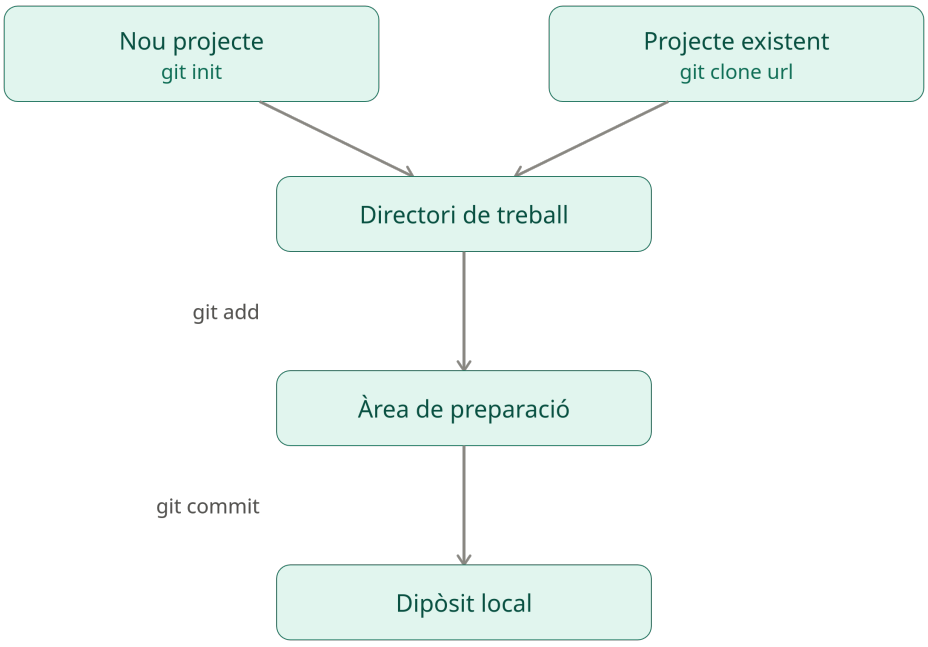

Figura 3: Flux d'ordres bàsiques (1)

```
# Crear i canviar de branca
git branch nova-funcionalitat
```

```
git checkout nova-funcionalitat
# o, en una sola ordre:
git checkout -b nova-funcionalitat

# Fusionar una branca amb la branca actual
git merge nova-funcionalitat
```

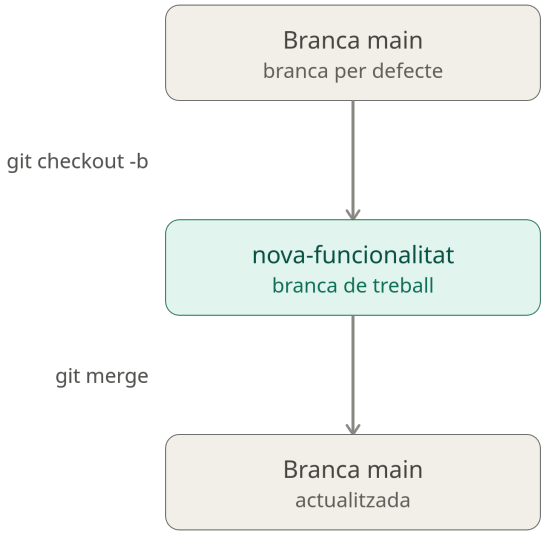

Figura 4: Flux d'ordres bàsiques (2)

```
# Sincronitzar amb el dipòsit remot
git pull                    # descarregar i fusionar canvis
git push                    # pujar els commits locals
git fetch                   # descarregar canvis sense fusionar-los
```

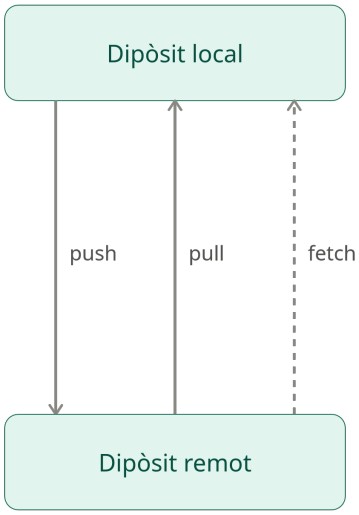

Figura 5: Flux d'ordres bàsiques (3)

```
# Desar canvis temporalment
git stash
git stash pop

# Desfer canvis
git checkout -- fitxer.txt    # descartar canvis no confirmats
git reset HEAD~1              # desfer l'últim commit (mantenint els
 ↪  canvis)
git revert <hash>             # crear un commit nou que desfà un
 ↪  commit anterior
```

7. Recursos i documentació de referència

- **Documentació oficial de Git**: https://git-scm.com/doc
- **Pro Git (llibre gratuït, disponible en espanyol)**: https://git-scm.com/book/es/v2
- **Referència completa d'ordres**: https://git-scm.com/docs
- **Git Cheat Sheet oficial (PDF)**: https://education.github.com/git-cheat-sheet-education.pdf
- **Tutorial interactiu "Learn Git Branching"** (molt útil per visualitzar branques i merges): https://learngitbranching.js.org/
- **Documentació d'Ubuntu sobre Git**: https://help.ubuntu.com/community/Git
- **GitHub Docs**: https://docs.github.com/es/get-started

Índex de figures

Versions d'aquest document

- HTML
- PDF
- ODT
- MD