

REGLAS PARA TRABAJAR CON GIT I SUBIR LOS CAMBIOS AL REPO

1. El teu company fa canvis a `login`, també toca `app.module.ts` per afegir l'import.
 - Fa `git add ., git commit -m "login component" i git push.`
 - Al repo de GitHub ara hi ha una versió nova del projecte.
2. Tu segueixes treballant al teu `skills`, però al teu local encara tens la versió antiga (no tens els canvis del company).

- Tu acabes i fas `git add ., git commit -m "skills component".`
- Si fas directament `git push` → Git et dirà que no pots perquè el teu repo local està desactualitzat respecte a GitHub.
En aquest punt has de fer:

```
git pull origin main
```

3.

- Això agafa els canvis nous de GitHub i els barreja amb els teus.
- Si heu modificat fitxers diferents → Git fusiona automàticament (merge sense conflictes).
- Si heu modificat el mateix fitxer (ex. `app.module.ts`) → Git et marca un conflicte.
Hauràs d'obrir el fitxer, veure què ha posat ell i què has posat tu, i decidir com deixar-ho.
Un cop resolt, fas:

```
git add .
```

```
git commit -m "resolve merge"
```

```
git push origin main
```

Exemple del flux complet:

Tots cloneu el repo al principi:

```
git clone https://github.com/aleixfal7/swapply.git
cd swapply
git checkout -b feature/skills
```

1.

Tu treballes a **skills** i el teu company a **login**.

Ell crea la seva branca:

```
git checkout -b feature/login
```

2.

Tu acabes **skills** → pushes la branca:

```
git push origin feature/skills
```

3. Vas a GitHub i fas el PR cap a **main**.

El teu company acaba **login** → pushes la seva branca:

```
git push origin feature/login
```

4. També fa PR.

Quan s'accepten els PRs, **main** queda amb els dos components.

Tothom fa:

```
git checkout main
git pull origin main
```

5.

Idees clau:

- Mai trebal·leu tots a **main** directament.
- Cada tasca → una branca (**feature/...** o **bugfix/...**).

- **Pull Request** per ajuntar canvis.
- **Git pull abans de començar** a treballar cada dia → així tens la versió més nova.