**Aspectes complementaris amb Git / GitHub**

**Main o master?**

És correcte i recomanable fer servir `main` en lloc de `master`. Aquesta pràctica s'ha estès per diverses raons:

**1. Normes d'inclusivitat**: El canvi de `master` a `main` forma part d'un esforç per utilitzar un llenguatge més inclusiu i neutre. Això ajuda a evitar connotacions negatives associades a termes com `master` en contextos que poden suggerir jerarquia o subordinació.

2. **Estàndard actual**: Des de l'octubre de 2020, GitHub utilitza `main` com a nom de la branca per defecte en els nous repositoris. Això ha convertit `main` en el nou estàndard de facto per a la branca principal en molts projectes.

**3. Compatibilitat i coherència**: Fer servir `main` permet que els nous projectes siguin consistents amb aquesta pràctica estesa, facilitant el treball col·laboratiu amb altres desenvolupadors que ja utilitzen aquest terme. A més, alguns serveis i eines d'integració contínua han actualitzat les seves configuracions per utilitzar `main`.

Recomanació: Si esteu treballant en projectes nous, utilitzeu `main` com a branca principal. Si esteu treballant en projectes antics o en entorns on encara es fa servir `master`, és recomanable canviar el nom de la branca principal a `main` per mantenir la coherència i adoptar pràctiques més modernes.

Per canviar el nom de la branca de `master` a `main` en un repositori existent, podeu fer-ho amb les següents comandes:

**git branch -m master main # Renombrar la branca local**

**git push origin --delete master # Opcional: esborrar la branca 'master' remota**

**git push origin main # Pujar la nova branca 'main'**

**git symbolic-ref refs/remotes/origin/HEAD refs/remotes/origin/main # Actualitzar la branca per defecte al remot**

**Els commits han de tenir obligatòriament un missatge?**

Sí, els commits a Git han de tenir un missatge associat, tot i que aquest missatge no ha de ser obligatòriament detallat. A la pràctica, el missatge de commit és molt important perquè ajuda a explicar i documentar els canvis que s'han fet al codi. Això és essencial per mantenir un historial clar i comprensible, especialment en projectes col·laboratius.

 Per què són importants els missatges de commit?

1. **Claredat**: Els missatges de commit permeten que altres desenvolupadors (i el mateix autor en el futur) entenguin què s'ha canviat i per què.

2. **Historial de canvis**: Faciliten la revisió de l'historial del projecte per entendre el propòsit de cada canvi.

3. **Solució de problemes**: Ajuden a identificar i a resoldre errors més fàcilment, ja que es pot veure quin commit va introduir un problema específic.

 Tipus de missatges de commit:

- Missatges mínims: Encara que Git permet fer commits amb missatges molt breus o fins i tot només amb un espai en blanc, això no és recomanable. Un exemple seria `**git commit -m ""**`, però això no proporciona cap informació útil.

- Missatges descriptius: El millor és proporcionar un missatge que resumeixi clarament el canvi, com per exemple: `**git commit -m "Corregeix l'error de validació en el formulari de contacte"**`.

 Conseqüències de no utilitzar missatges adequats:

1. Historial confús: Un historial ple de missatges com "update", "fix" o missatges buits no ofereix cap context, cosa que complica la gestió i comprensió del projecte.

2. Mala comunicació: Els col·laboradors poden tenir dificultats per saber què s'ha canviat, fent més difícil la revisió de codi i el treball en equip.

 Bones pràctiques per a missatges de commit:

- Ser breu però descriptiu: Escriure un resum clar del canvi en la primera línia.

- Format opcional: Per missatges més llargs, una primera línia breu seguida d'una línia en blanc i després una explicació més detallada.

- Utilitzar verbs en imperatiu: Escriure el missatge com si fos una ordre, com per exemple "Afegeix nova funcionalitat" o "Corregeix problema de rendiment".

Exemples de bones pràctiques:

**git commit -m "Afegeix una nova funció per gestionar usuaris"**

**git commit -m "Corregeix l'error de connexió a la base de dades"**

**Conclusió:**

Tot i que Git tècnicament permet fer commits amb missatges buits, és una mala pràctica. Proporcionar missatges de commit clars i descriptius és essencial per mantenir un historial de codi net i comprensible.

**Fer un `merge` de `main` a una branca de treball**

Fer un `merge` de `main` a una branca de treball és útil per diverses raons:

1. **Mantenir la branca sincronitzada**: Integrar els últims canvis de `main` ajuda a que la branca de treball estigui actualitzada amb el treball més recent d'altres desenvolupadors. Això minimitza sorpreses durant el `merge` final de la branca de treball a `main`.

2. **Reducció de conflictes futurs**: Fer un `merge` de `main` a una branca permet identificar i resoldre conflictes de codi més aviat en el procés de desenvolupament. Això fa que la integració final sigui més fluida i amb menys conflictes.

3. **Verificació de compatibilitat**: Amb aquest procés, els desenvolupadors poden comprovar com interactua el codi nou amb els canvis ja integrats a `main`, ajudant a detectar possibles problemes abans que es despleguin.

4. **Col·laboració eficient**: En equips de desenvolupament, mantenir les branques de treball actualitzades amb `main` facilita la col·laboració i la revisió de codi, ja que cada desenvolupador treballa amb una base de codi coherent.

 Pas a pas per fer un `merge` de `main` a una altra branca:

**1. Comprova que estàs a la branca de treball:**

 Abans de fer el `merge`, assegura't que estàs a la branca on vols integrar els canvis de `main`.

 **git checkout branca-de-treball**

**2. Actualitza la branca `main`:**

 Si estàs treballant en un entorn de col·laboració, és important tenir l'última versió de `main` abans de fer el `merge`.

 **git checkout main**

 **git pull origin main**

 **3. Torna a la branca de treball:**

 Canvia de nou a la branca de treball on vols fer el `merge`.

 **git checkout branca-de-treball**

**4. Fes el `merge` de `main` a la branca de treball:**

 Integra els canvis de `main` a la branca actual.

 **git merge main**

**5. Resolució de conflictes (si n'hi ha):**

 Si el `merge` genera conflictes, Git t'indicarà quins fitxers tenen problemes. Editeu aquests fitxers per resoldre els conflictes manualment. Els conflictes es marquen amb:

 **<<<<<<< HEAD**

 **Canvis de la branca de treball.**

 **=======**

 **Canvis de la branca `main`.**

 **>>>>>>> main**

 Un cop resolts els conflictes, afegeix els fitxers:

 **git add fitxer-conflictiu.txt**

 **git commit -m "Resolució de conflictes i integració de main a branca-de-treball"**

6. Opcional: Puja els canvis al repositori remot:

 Un cop completat el `merge`, envia els canvis al repositori remot.

 **git push origin branca-de-treball**