

Introducció als repositoris de codi i el control de versions amb Git

Què és un repositori de codi?

- Un repositori de codi és una ubicació on es guarda tot el codi font d'un projecte. Permet als desenvolupadors col·laborar, compartir i gestionar els canvis en el codi de manera efectiva.
- Els repositoris de codi són fonamentals perquè els equips de desenvolupament puguin treballar de manera coordinada i mantenir un historial de les versions del codi.
- El control de versions i l'ús de repositoris són essencials per a un bon maneig del codi font en projectes de desenvolupament, permetent la col·laboració efectiva i la gestió ordenada de les versions del codi.

Què és el control de versions?

- El control de versions és el procés de gestionar i registrar els canvis en el codi font d'un projecte. Permet controlar qui ha fet quins canvis, quan s'han realitzat i revertir-los si cal.
- Hi ha múltiples alternatives per realitzar el control de versions

De tipus centralitzat

Consisteixen en un únic servidor amb totes les versions i permet que els programadors que treballen en el mateix projecte es puguin descarregar l'última versió al seu ordinador per a treballar-hi

CVS: <https://www.nongnu.org/cvs/>

Subversion: <https://subversion.apache.org/>

De tipus distribuït

Els programadors que treballen en un mateix projecte no només es descarreguen l'última versió sinó tot el conjunt de versions.

Si el servidor central falla, es poden recuperar totes les versions de qualsevol ordinador que hi treballi

GIT: <https://git-scm.com/>

Mercurial: <https://www.mercurial-scm.org/>

GitHub

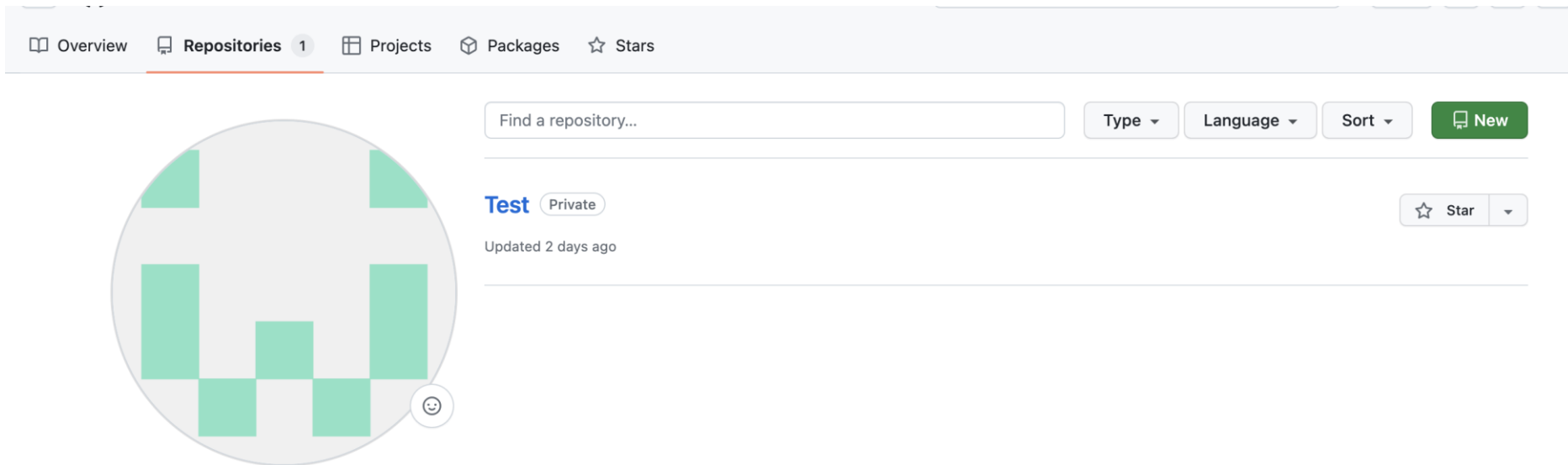
- GitHub és una plataforma en línia que ofereix serveis de control de versions i desenvolupament col·laboratiu de programari. És un dels llocs més populars per als desenvolupadors per emmagatzemar i compartir els seus projectes de codi font.
- És un lloc ideal per a l'aprenentatge, la col·laboració i la contribució a projectes de codi obert.
- **Algunes de les característiques clau de GitHub inclouen:**
 - **Repositoris públics i privats:** Permet als desenvolupadors triar si volen compartir el seu codi de forma pública o mantenir-lo privat.
 - **Control de versions amb Git:** GitHub utilitza el sistema de control de versions Git per gestionar els canvis en el codi font d'un projecte.
 - **Col·laboració:** Permet als membres de l'equip treballar conjuntament en projectes, gestionar tasques i comunicar-se mitjançant problemes i solucions.
 - **Seguiment de canvis.**
 - **Integracions i automatització:** GitHub ofereix integracions amb altres eines de desenvolupament.

Exemple: Crear un repositori a GitHub

Suposem que tenim un equip treballat format per dos desenvolupadors: IT1 i IT2.

Volem iniciar un nou projecte i per poder treballar conjuntament obrirem un repositori a GitHub quins passos cal fer?

1. Iniciem la sessió al nostre compte de GitHub: (<https://github.com>)
2. Dins la pestanya “Repositories” hi trobarem el botó **New**
Doneu un nom al repositori i opcionalment un descripció



Exemple: Crear un repositori a GitHub

Cal decidir si volem un repositori públic o privat. En aquest cas el crearem com a privat.

Posteriorment podrem afegir altres usuaris al repositori.

És recomanable afegir un arxiu **README** que descriurà el projecte

Afegirem també un arxiu **.gitignore** (per exemple amb la plantilla C++). En aquest arxiu s'especifiquen quins arxius o tipus d'arxius no volem que s'afegeixin al repositori. Per exemple no pujarem el executables ja que només hi volem el codi font

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

 marc-tenvinilo ▾

Repository name *

Great repository names are short and memorable. Need inspiration? How about [supreme-train](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: C++ ▾


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Exemple: Crear un repositori a GitHub

En el següent pas trobareu la **URL** del repositori GIT que utilitzarem per clonar el repositori.

Al clonar el repositori creareu una còpia local en la vostra màquina, on hi podreu desenvolupar i posteriorment pujar el canvis al servidor per a que tots els membres de l'equip pugui treballar sobre el mateix projecte.

Quick setup — if you've done this kind of thing before

 Set up in Desktop or HTTPS SSH

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# Test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/marc-tenvinilo/Test.git
git push -u origin main
```

Exemple: Començar a treballar en el repositori

Ara ja podem treballar amb el nostre repositori.

Si no tenim GIT instal·lat cal fer-ho abans: <https://git-scm.com/downloads>

- Ens situarem dins la carpeta en la que volem clonar el repositori.
- Crearem el nostre primer arxiu per exemple l'arxiu README.md
- Inicialitzem el repositori amb:

git init

Aquesta comanda inicia el repositori en la carpeta seleccionada creant tot el necessari per a que tot funcioni

Exemple: Començar a treballar en el repositori

Ara ja podem afegir els primers canvis al repositori. La comanda 'add' indiquem els arxius que volem afegir al repositori (podem utilitzar * per afegir-ho tot). Posteriorment el "commit" afegeix els arxius al repositori.

- Afegim els arxius al repositori

```
git add README.md
```

```
git commit -m "El meu primer commit"
```

- Creem la branca "master" en el repositori. Aquesta serà la branca principal del repositori:

```
git branch -M master
```

Exemple: Començar a treballar en el repositori

En aquest moment tots els canvis són locals. Cal doncs pujar-ho al servidor.

Afegim el nostre repositori de GitHub com a origen utilitzant la URL del repositori que hem creat a GitHub:

```
git remote add origin  
https://github.com/marc/Test.git
```

Ara ja podem pujar els canvis al servidor amb la comanda '**push**':

```
git push -u origin master
```

Ara el nostre company ja pot clonar el repositori en la seva màquina. Triarà la carpeta on vol clonar el repositori i executarà:

```
git clone https://github.com/marc/Test.git
```

Exemple: Començar a treballar en el repositori

Ara ja podem treballar sobre el repositori.

Per baixar el canvis que s'hi hagin pogut fer farem:

```
git pull
```

Aquesta comanda permet tenir el codi sincronitzat, ja que Baixa els canvis i els fusiona amb els arxius locals

Ara cada vegada que vulguem pujar canvis al repositori, farem "commit" i "push":

```
git add .
```

```
git commit -m "Descripció del commit"
```

```
git push
```

Git: Treballar amb branques

- Com altres controls de versions Git permet que els membres de l'equip pugui treballar sobre el mateix projecte.
- **Permet treballar amb branques** de manera que sempre tindrem una branca principal (master) que contindrà el codi “definitiu” i els programadors poden obrir branques per treballar sense trepitjar-se
- **Treballar amb branques en un sistema de control de versions com Git permet als desenvolupadors treballar en els canvis del codi de manera aïllada, sense afectar la branca principal del projecte.** Això ofereix diverses avantatges com la possibilitat de desenvolupar funcionalitats noves sense interferir en el treball dels altres membres de l'equip i provar canvis abans de fusionar-los amb la branca principal.

Git: Treballar amb branques

El procés de treball amb branques en Git es pot resumir en els següents passos:

1. **Crear una nova branca:** Amb el comandament ``git branch`` es pot crear una nova branca a partir de la branca actual o una altra existent.
2. **Canviar a la nova branca:** Amb ``git checkout`` es pot canviar de branca per començar a treballar en ella.
3. **Realitzar els canvis:** En la nova branca es poden fer els canvis necessaris, com afegir noves funcionalitats o corregir errors.
4. **Afegir i desar canvis:** Utilitzant ``git add`` i ``git commit`` es poden afegir i desar els canvis en la branca.
5. **Fusionar la branca:** Un cop els canvis estan provats i revisats, es pot fusionar la branca amb la branca principal utilitzant ``git merge``.
6. **Eliminar la branca:** Un cop la branca s'ha fusionat amb èxit, es pot eliminar-la amb ``git branch -d`` per mantenir el repositori net.

Exemple: utilització de les branques

Seguint l'exemple anterior, volem afegir noves funcionalitats al nostre codi.

- Volem afegir la classe `Exemple`, creant els arxius `Exemple.h` i `Exemple.cpp`
- Farem canvis a `main.cpp` per afegir el codi utilitza la nova classe
- A `main.cpp` volem fer canvis per modificar i millorar el menú d'inici

IT1 s'encarregarà dels dos primers punts i IT2 del tercer.

Si els dos membres de l'equip treballessin a l'hora sobre el mateixos arxius, es podrien perdre canvis o podrien quedar inconsistències en el codi.

Per evitar-ho l'equip treballarà amb branques:

- Branca **feature1** pels canvis que farà IT1
- Branca **feature2** pels canvis que farà IT2

Exemple: utilització de les branques

Si som el desenvolupador IT1 i tenim el repositori en la branca **master**, què hem de fer per fer la tasca?

- Creem una nova branca per afegir la nova funcionalitat. Crearem la branca "feature1"

git branch feature1

- Canviem a la nova branca amb:

git checkout feature1

- Fem els canvis en el codi. Afegim la nova classe i fem els canvis necessaris a main.cpp
- Guardem els canvis en el repositori:

git -a commit -m "Nova classe Exemple"

- -m : permet posar un missatge explicatiu al commit
 - -a : afegeix els arxius treballats al commit
- Pugem els canvis servidor principal, sobre la branca actual. Així evitem perdre la feina feta si tenim cap problema amb la nostra màquina.

git push

Exemple: utilització de les branques

Els punts 3 i 4 es poden anar repetint durant el procés de desenvolupament.

Una vegada hem acabat i verificat els canvis ja podem fusionar la branca de treball “feature1” amb la branca principal “master”

Per fer-ho utilitzarem la següent seqüència de comandes:

- Tornem a la branca principal amb
git checkout master
- Fem la fusió de la branca `feature1` amb la branca principal amb:
git merge feature1
- En aquest moment poden aparèixer conflictes si Git no pot determinar com fusionar els canvis en el arxius. Caldrà resoldre'ls i tot seguit continuar amb el merge.
- Un cop la nova funcionalitat està fusionada amb èxit a la branca principal, podem eliminar la branca `feature1` amb:
git branch -d feature1

L'altre desenvolupador de l'equip seguirà els mateixos passos i finalment a la branca “master” hi tindrem els canvis definitius

Git a Visual Studio

En aquest enllaç hi trobareu una guia completa de Git:

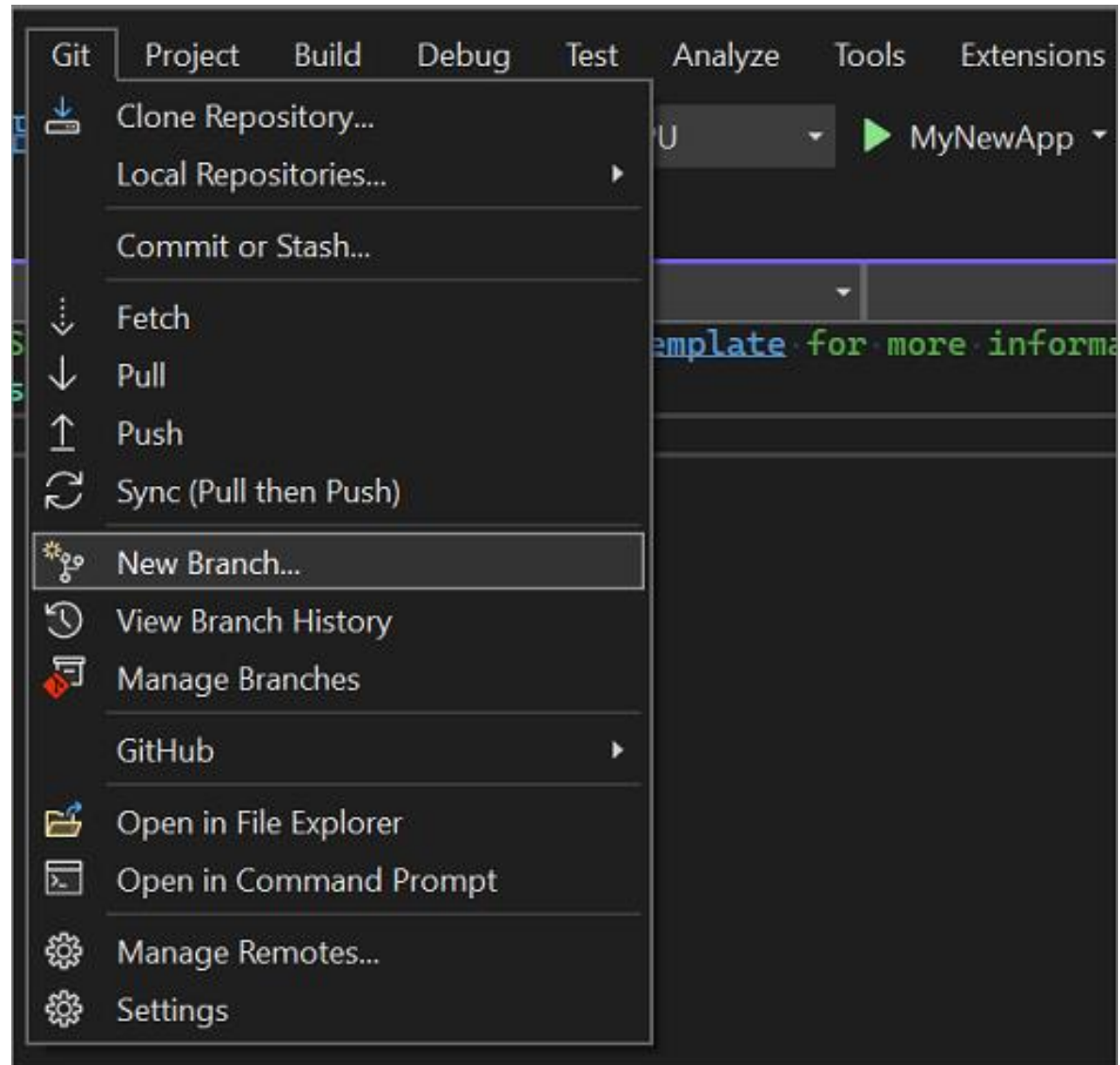
<https://git-scm.com/book/es/v2>

Podeu integrar Git i el vostre repositori de GitHub en el Visual Studio:

1. Obriu un Visual Studio.
2. Aneu al menú Git i seleccioneu **Clonar o Clonar Repositorio**.
 1. Si no veieu el menú Git caldrà afegir el complement: **Herramientas > Opciones > Control de código Fuente > Selección de complemento**
3. Poseu la URL del repositori.
4. Inicieu sessió a GitHub (doneu els permisos que es demanin).
5. I ja podeu clonar el repositori.

Git a Visual Studio

Dins el menú GIT trobareu les eines necessàries per gestionar el repositori pujar canvis, gestionar branques, etc:



Git a Visual Studio

Documentació de Visual Studio:

<https://learn.microsoft.com/es-es/visualstudio/version-control/git-with-visual-studio?view=vs-2022>

Apèndix: Comandes bàsiques de Git

git init: Inicialitza un nou repositori de Git a la carpeta actual del projecte.

git add: Afegeix els canvis realitzats als arxius, preparant-los per ser “comitats”, guardats en el repositori.

git commit: Guarda els canvis al repositori de Git juntament amb un missatge descriptiu.

git status: Mostra l'estat dels arxius en el repositori de Git, destacant els canvis pendents.

git push: Puja els canvis al repositori remot, com GitHub, per compartir-los amb altres membres de l'equip.

git pull: Baixa els canvis del repositori remot al repositori local.

git branch: permet crear, llistar o eliminar branques.

git checkout: permet canviar de branca. La branca activa és la branca sobre la que fem els canvis

git merge: permet fusionar els canvis d'una branca