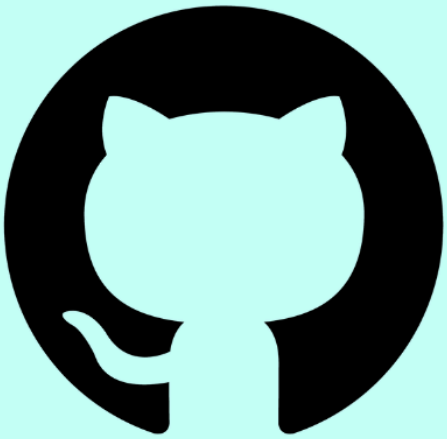
Creat per Marc Cachinero i Ruth Romero

MANUAL DE



GitHub

Índex

Què és GitHub?

GitHub és un lloc web on diferents persones poden treballar juntes en projectes digitals, com ara programes o aplicacions. Funciona com una mena de directori on es guarden totes les versions i canvis d'un projecte, perquè ningú perdi informació i sempre es pugui recuperar una versió anterior. A més, GitHub inclou eines perquè les persones es puguin organitzar, deixar comentaris i resoldre problemes fàcilment, fent-lo ideal per a treballs en equip o per a projectes oberts on tothom pot col·laborar.

Com funciona?

El funcionament de GitHub es basa en els següents conceptes clau:

- **Repositoris**: Un repositori és un lloc dins de GitHub on es guarda un projecte i tot el seu historial de canvis. És com una carpeta especial que permet veure què ha canviat i col·laborar amb altres persones.
- **Commits**: Un *commit* és com una foto dels canvis que es fan en un projecte. Quan es fa un *commit*, es guarda una nova versió amb una breu explicació dels canvis. Així es pot tornar a una versió anterior si cal.
- **Branques**: Una branca és una versió paral·lela del projecte on es poden provar noves idees sense afectar la versió principal. La branca principal sovint es diu *main* o *master*. Quan els canvis de la branca estan llestos, es poden unir a la versió principal.

Què és el control de versions?

El control de versions és una pràctica essencial en el desenvolupament de programari que permet gestionar i rastrejar els canvis realitzats en el codi font. Aquesta tècnica guarda un historial de totes les modificacions, facilitant tornar a versions anteriors si es cometen errors i permetent la col·laboració entre diversos desenvolupadors sense conflictes.

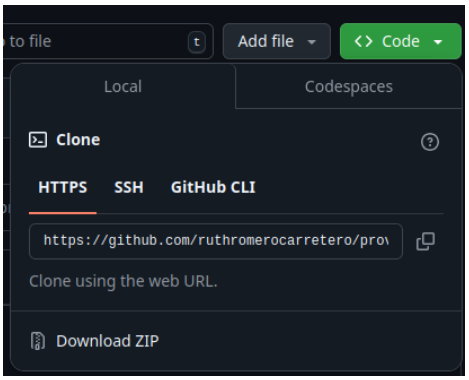
---

1. Llistat de comandes més utilitzats

1. `$git clone`: es fa servir per copiar un repositori remot a la teva màquina local. Això crea una còpia exacta del repositori, incloent-hi tots els commits, branques i etiquetes. És ideal per començar a treballar amb un projecte ja existent.

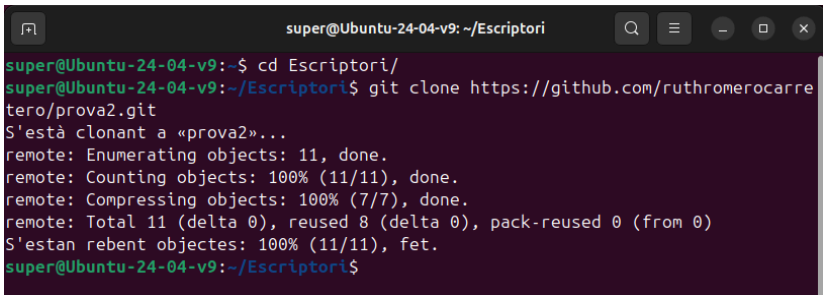
Exemple:

Anirem a la pàgina de github, del repositori que ja està creat i clicarem en el botó que posa <> Code:



I copiare'm el link que posa en l'apartat de "HTTPS".

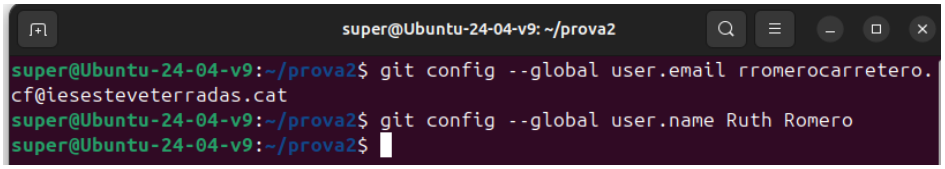
Anem a la terminal del nostre ordinador:



- **$cd Escriptori/:** la comanda cd és per navegar entre directoris en la línia de comandes. En aquest cas volem que estigui en l'escriptori del nostre sistema operatiu.
- **$git clone <enllaç_del_repositori>:** això farà que la clonació del repositori es faci dintre del nostre escriptori.

2. `$git config --global:`
   - `$git config --global user.email <email_que_has_utilizat_al_crear_l'usuari_de_github>`: li donarem el email que utilitzem en el Github
   - `$git config --global user.name <el_teu_nom>`: li posem el nostre nom.

Exemple:

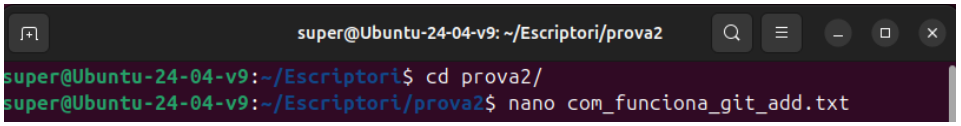


3. `$git init`: Crea un repositori en local, per després enllaçar-lo amb un repositori remot.

4. `$git add .`: S'utilitza per afegir fitxers que estan en local, dintre del control de versions, preparant el procés de pujar-lo al nostre repositori de GitHub.

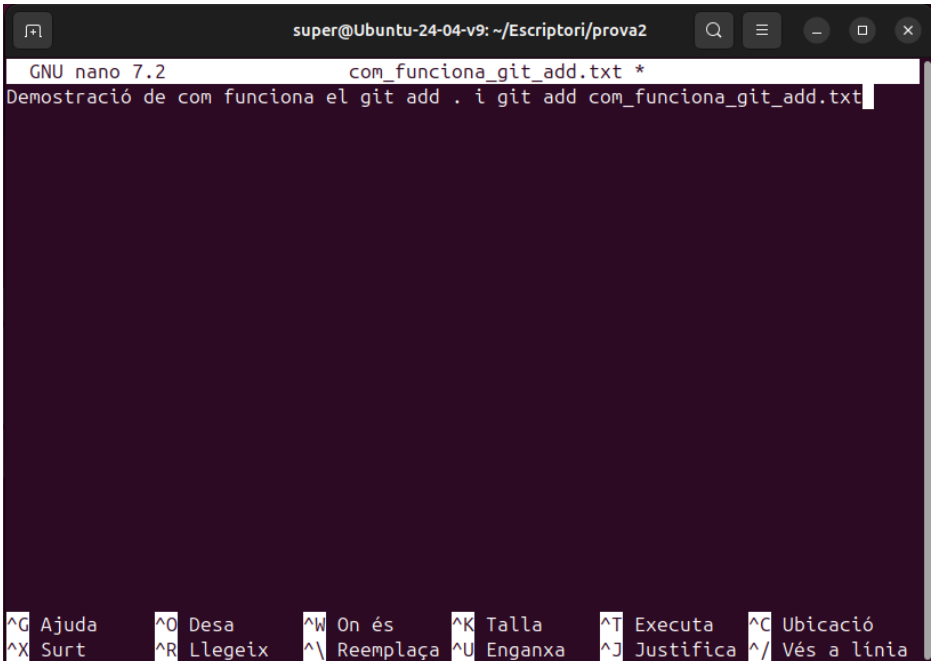
Exemple:

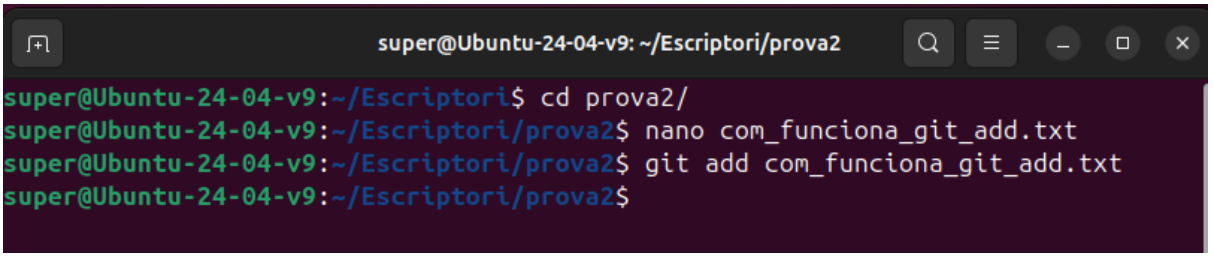
Creem un fitxer o afegim un fitxer, en el nostre cas ho crearem:



- nano <nom_fitxer>: és una comanda, per iniciar un editor de text per la consola de comandes.

Dintre del fitxer escribim lo que necesitem i fem Ctrl + S per guardar i Ctrl + X per sortir:





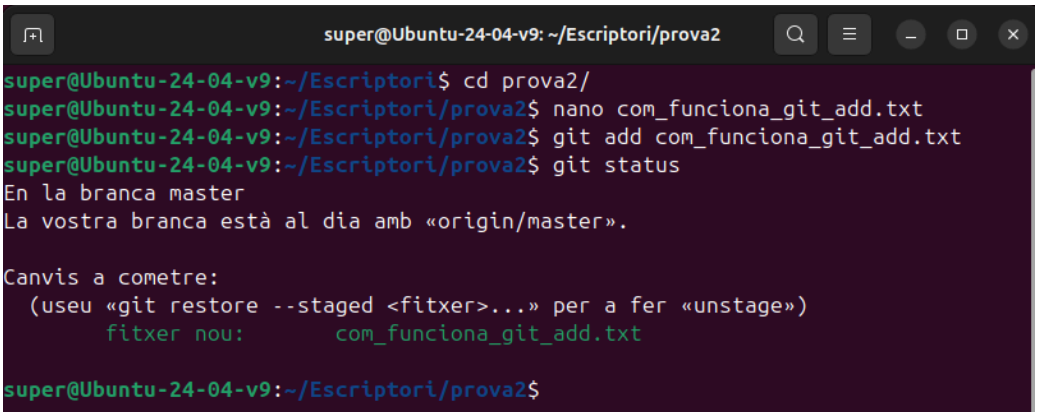
git add té dues formes:

- `$git add .`: que és per afegir tots els fitxers al staging area.
- `$git add <nom_del_fitxer>`: solament escogim que aquest fitxer en concret es mogui al staging area.

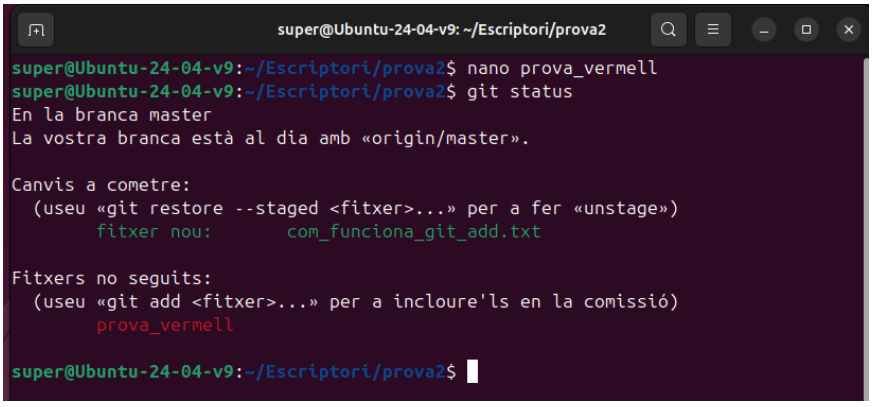
5. `$git status`: Podem veure en que part del procés estan els nostres fitxers, si l'executem abans de fer el git add ., ens sortirà en vermell, i a sobre ens posa que hem de fer una comanda per començar el procés de pujar-lo.

Exemple:

Com anteriorment s'havia executat el git add, al executar la comanda $git status podem veure de que el fitxer surt en verd, ja que ja s'ha afegit a l'staging area:

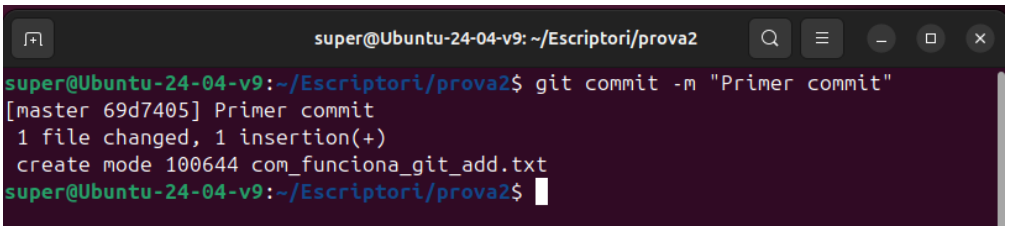


En el cas de no afegir-lo dintre del staging area, es veuria així:



6. `$git commit -m <missatge>`: S'utilitza per enregistrar els canvis realitzats en els arxius del repositori local.

Exemple:



Nota: S'executa després del $git add

7. `$git commit -a`: Aquesta comanda, combina el $git add . i el $git commit -m en una mateixa comanda (Només afecta als fitxers modificats).
   També existeix `$git commit -a -m` que es lo mateix que lo anterior, però podem afegir un missatge al commit (Només afecta als fitxers modificats).

Exemple:

Primer hem d'afegir el fitxer dintre, amb el $git add, perquè lo que fa es aplicar els canvis que fem en el fitxer, després executem la comanda $git commit -a:

I sobrirà aquest fitxer amb editor de text, on haurem d'afegit aquí el missatge del commit, una vegada l'establim per guardar-ho fem Ctrl+S i Ctrl+X per tancar-ho:

8. `$git push`: S'envien els canvis que s'han realitzat en la màquina local al Github.
    - `$git push -set-upstrem origin master`: S'utilitza en el cas de que el repositori es faci des de local, i volem pujar els canvis a Github. Exemple pràctic
    - `$git push origin <nom_branca>`: S'utilitza per empènyer els canvis locals d'una branca específica <nom_branca> al repositori remot anomenat origin.
    - `$git push -all origin`: Lo mateix que lo d'abans però amb la diferència que el –all indica que volem pujar totes les branques locals.

Exemple:

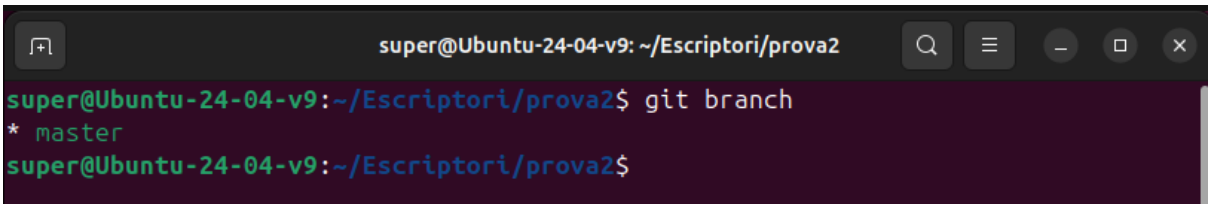
9. `$git pull`: Ens descarreguem els canvis que s'han realitzat als fitxers i que estàn registrats en GitHub (s'actualitza el repositori local, amb els canvis més nous).

Exemple:

Nota: en el nostre cas com no hi ha hagut cap canvi, posa "Ja està al dia", es sol fer quan hi han canvis de altres usuaris i vols tenir els canvis en el teu repositori remot.

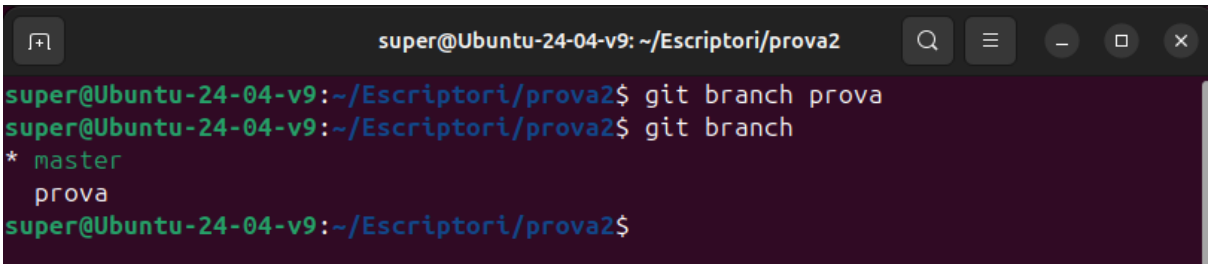
10. `$git branch`: Podem utilizar-la per veure les branques que existeixen en el repositori i en la que posa l'asterisc (*) es en la que estem situats actualment:

Exemple:



11. `$git branch <nom_branch>`: Per treballar en diferents línies de desenvolupament al mateix temps, utilitzem branques. Una branca permet treballar en noves característiques o correccions d'errors de manera aïllada del codi principal.

Exemple:



- `$git branch <nom_branca>`: crear una branca amb el nom que l'assignem.
- `$git branch`: per veure les branques i verificar si s'ha creat la nostra branca.

Nota: per veure en canvis haurem de fer un $git push.

12. `$git branch -d <nom_branch>`: Eliminar una branch.

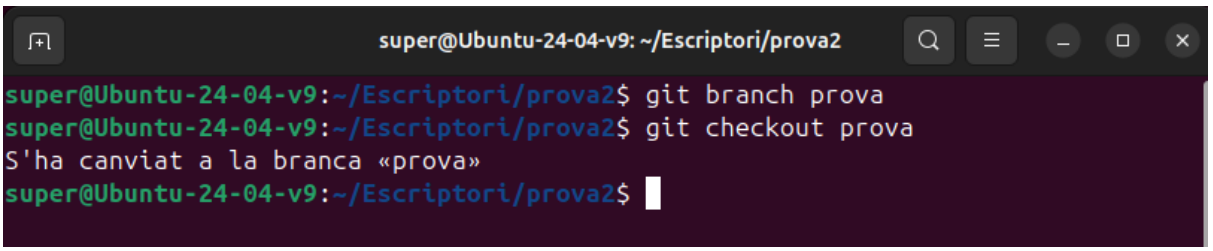
Exemple:

Nota: Executem dues vegades el $git branch, per verificar que s'ha esborrat correctament.

13. `$git checkout <nom_branch>`: Aquesta comanda, va lligada per l'anterior, l'executarem per poder navegar entre branches.

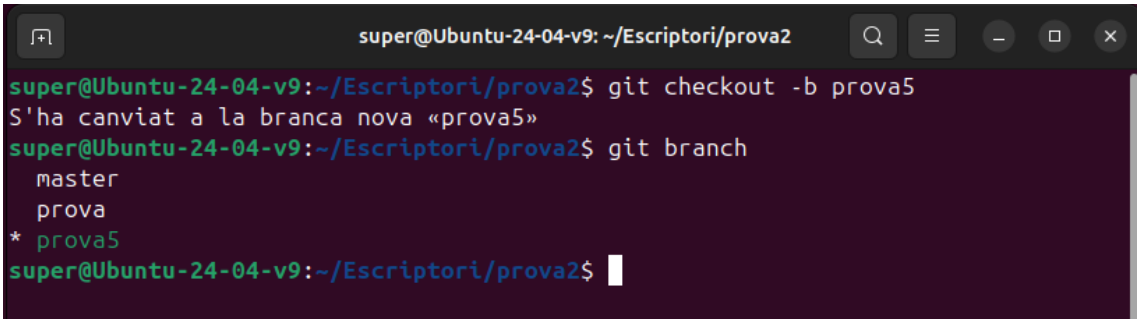
- `$git checkout -b <nom_branca>`: Crea una nova branca i canvia a ella.

Exemple `$git checkout <nom_branch>`:



Nota: s'ha creat una altre vegada la branch anterior, per poder canviar de branch, ara estariem treballant des de la branch prova. La següent comanda ($git merge) ens ajudarà a posar els canvis en la mateixa branca.

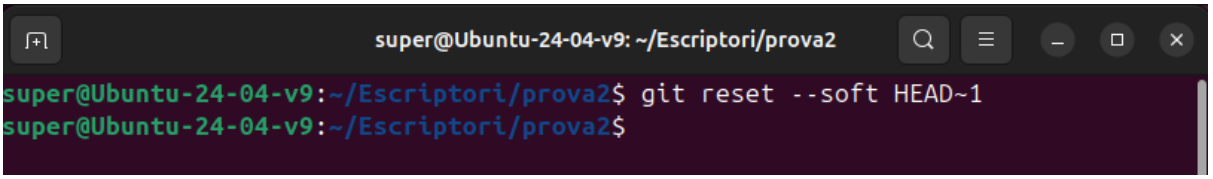
Exemple `$git checkout -b <nom_branca>`:



14. `$git merge`: s'utilitza per combinar canvis de diferents branques dins un projecte Git. Això permet fusionar el treball fet en diverses línies de desenvolupament, resultant en una única branca actualitzada amb tots els canvis aplicats.
    - `$git merge --abort`: cancel·la un merge en procés i retorna l'estat del repositori al moment abans de començar la fusió. És útil si es troben conflictes durant el merge i es decideix no continuar.

Exemple: Ho podeu veure en un exemple pràctic.

15. `$git reset`: s'utilitza per desfer canvis en el repositori. Té diferents maneres d'actuar segons el paràmetre que s'afegeix:
    - `--soft`: Manté els canvis a l'índex (staging area) i al directori de treball.

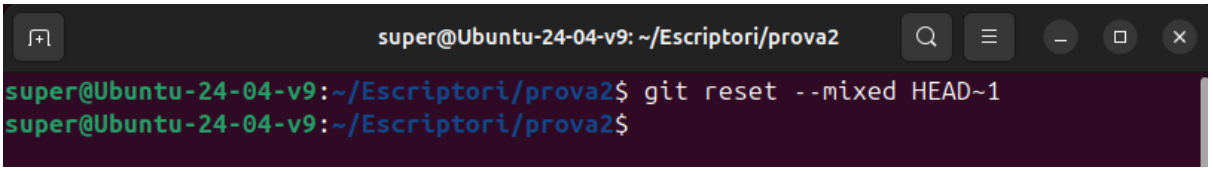


Nota:

- Resultat: Commit 3 es desfà, però els canvis encara estan preparats per fer un nou commit.

- `--mixed` (opció per defecte): Desfà els canvis a l'índex però els manté al directori de treball.

Exemple:

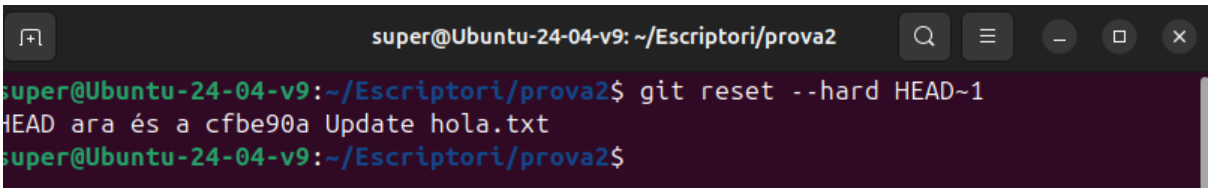


Nota:

- Historial de commits: Es desfa el Commit 3.
- Staging area: El Commit 3 es retira del staging area.
- Directori de treball: Els canvis del Commit 3 es mantenen al directori de treball, llestos per ser modificats, afegits o esborrats novament.

- `--hard`: Desfà tots els canvis a l'índex i al directori de treball, eliminant qualsevol canvi no desat.

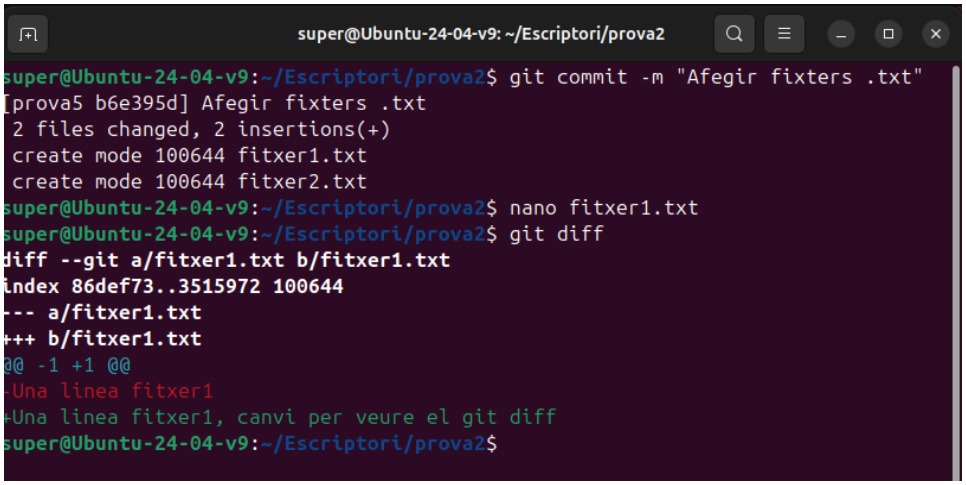
Exemple:



Nota:

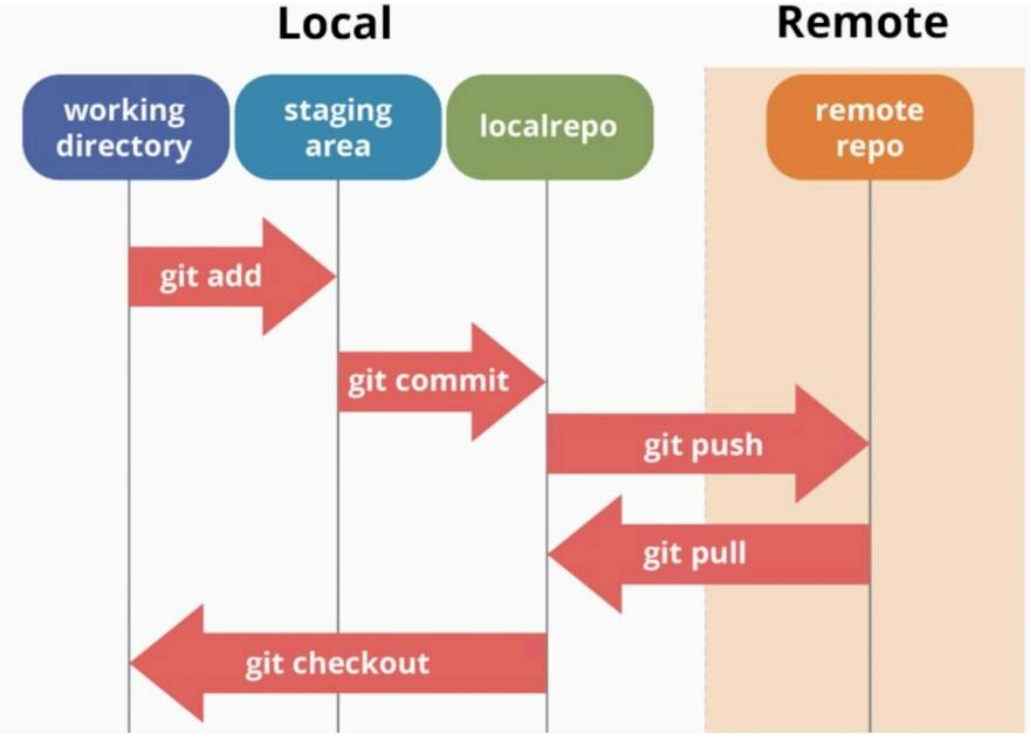
- Historial de commits: El Commit 3 es desfa completament.
- Staging area i directori de treball: Tots els canvis del Commit 3 es perden completament del directori de treball i de l'índex.

16. `$git diff`: Mostra les diferències entre els canvis no compromesos (unstaged, es dir, que no han entrat dintre del procés per pujar-ho) i els canvis (staged) en el directori de treball. Et dóna tota la visibilitat per veure que s'ha modificat, afegit o esborrat.
Nota: Fa la comparació respecte a l'últim commit. Si se executa sense fer un commit i canviar el fitxer no funciona.



Exemple `$git diff status` (no es necessari fer commit, es per comparar fitxers):

1.2 Comprenent els Conceptes Clau de Git



- **Working directory**: és el directori de treball on es troben els fitxers del projecte en l'estat actual.
- **Staging area**: és la zona intermèdia on es preparen els canvis abans de fer un git commit. S'activa en el moment que s'executa la comanda `$git add`, ja que els fitxers modificats es mouen a l'staging area. (és quan es fa un `$git status` que es veu en verd.)
- **Local repo**: una còpia del repositori de Git que es troba emmagatzemada localment en la nostra màquina.
- **Remote repo**: es lo mateix que el local repo, pero es quan s'emmagatzema en un servidor o un servei remot, en aquest cas es referim a GitHub.

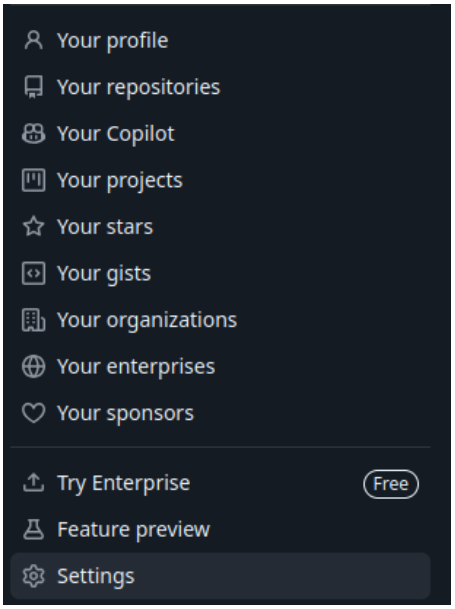
2. Què és el token d'autentificació de GitHub

El *token* de GitHub és una clau especial que serveix per identificar-te de manera segura quan vols fer canvis en els teus projectes de GitHub o connectar-hi altres aplicacions. És com una contrasenya temporal amb permisos específics que pots controlar, indicant què es pot fer i durant quant de temps.

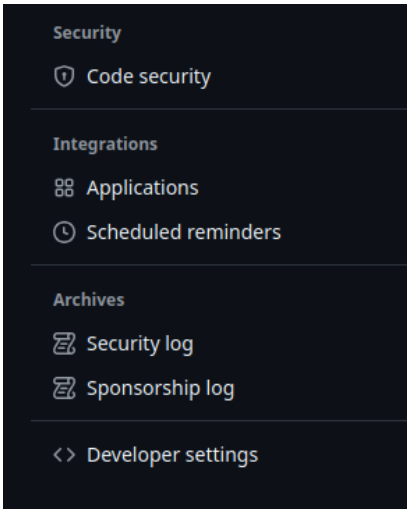
Aquest *token* és útil perquè et permet evitar l'ús de la contrasenya del teu compte en operacions automàtiques, fent que el teu compte sigui més segur.

2.1 Com s'aconsegueix

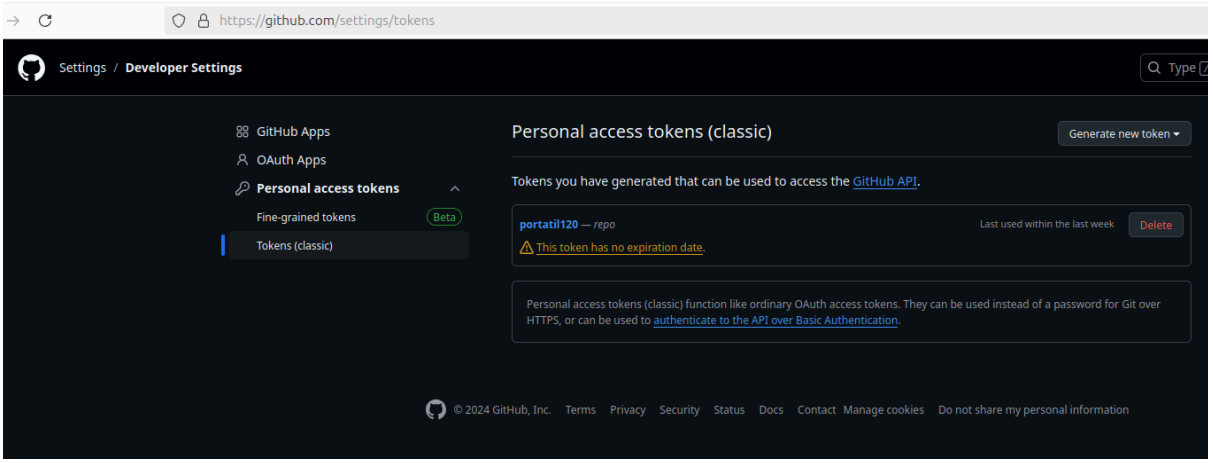
Li donem al nostre usuari, a dalt a la dreta, i en el menú que em vist abans li donarem a "settings":



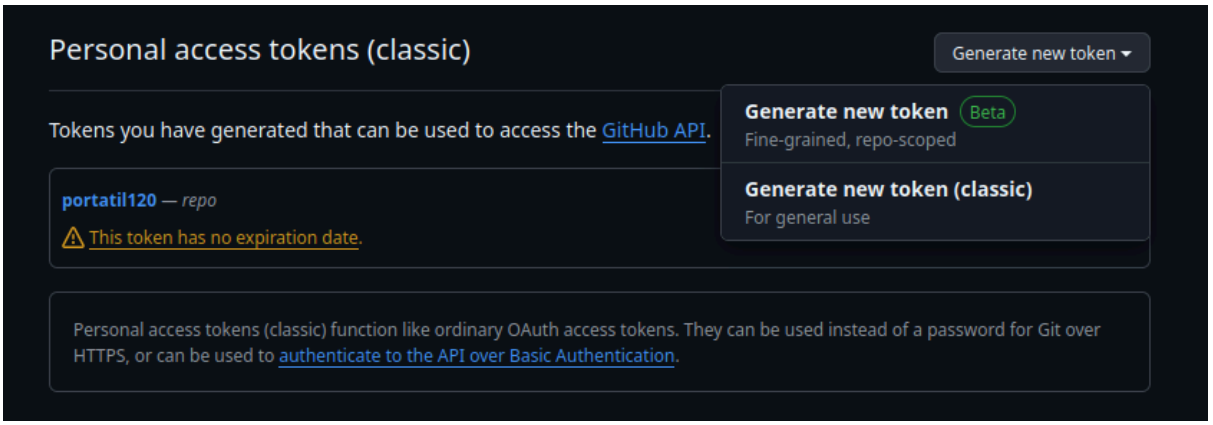
Dintre del menú que s'obrirà anem al final i cliquem a l'opció "<> Developer settings":

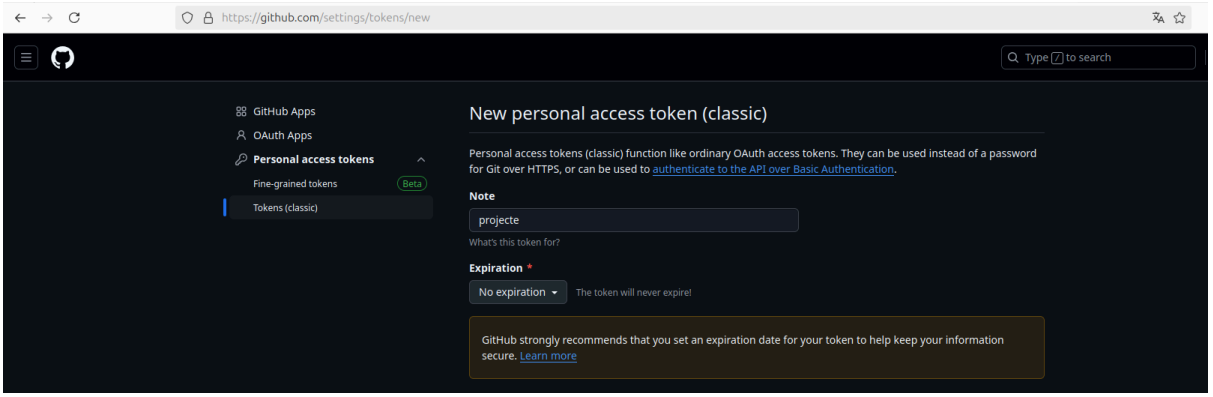


S’obrirà aquest submenú, que serà on obtindrem el Token, anirem a l’apartat “Personal access tokens” -> “Tokens (classics)”



Dintre d'aquí, cliquem en el desplegable que posa “Generate new token” -> “Generate new token (classic)”:

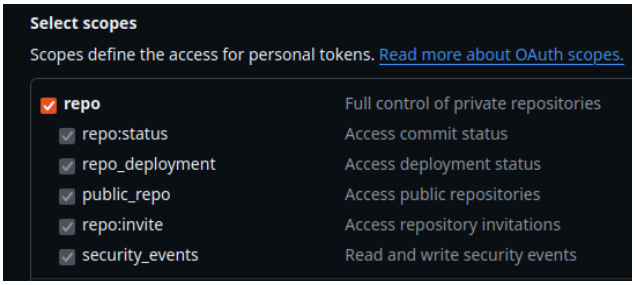




- **En la casella “Note”**: escriu el nom del teu token. Això serveix simplement per identificar-lo fàcilment més endavant.
- **Menú desplegable “Expiration”**: aquest menú et permet establir una data de caducitat per al token. Com que volem que el token no caduqui, seleccionarem l’opció “No expiration”.
- **Advertència de seguretat**: una vegada seleccionada aquesta opció, apareixerà una advertència indicant que, per motius de seguretat, és

recomanable establir una data de caducitat per al token. D'aquesta manera, podrem renovar-lo periòdicament, disminuint el risc de posibles accesos maliciosos.

- Select scopes:

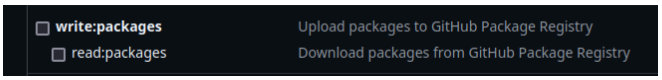


- Repo: Seleccionant la casella general, tindrem control total dels repositoris.
- repostatus: permet al token accedir a l'estat dels **commits** d'un repositori. Podem veure informació sobre si els commits han passat correctament les proves o si tenen algun error, i així seguir el progrés del codi dins del repositori.

-**workflow**: Són automatitzacions que es solen utilitzar quan hi ha tasques repetitives.

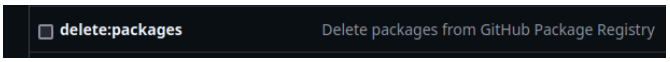
Si activem aquesta opció, podrem:

1. **Veure i accedir als workflows** d'un repositori.
2. **Executar o detenir workflows** manualment.
3. **Modificar les configuracions dels workflows** si el token té permisos d'escriptura.

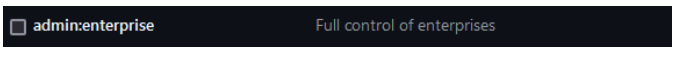


-**write:packages**: permet pujar, actualitzar i eliminar paquets en GitHub Packages.

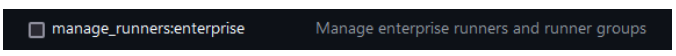
-**read:packages**: es per descarregar tots els paquets que hi ha dintre del registre.



-**delete:packages**: s'utilitza per eliminar el registre dels paquets.

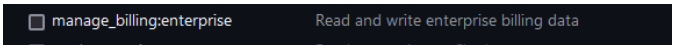


-**admin:enterprise**: Administració a nivell d'empresa per gestionar configuracions globals i recursos de l'organització.

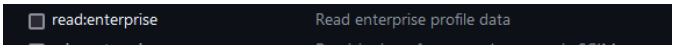


-**manage_runners:enterprise**: Són els servidors que fan tasques

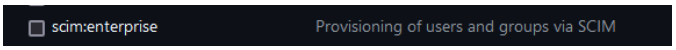
automatitzades com compilar, fer proves, o desplegar aplicacions.



-**manage_billing:enterprise:** Gestió de la facturació i subscripcions de l'empresa.



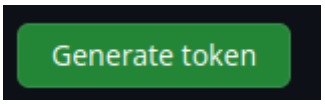
-**read:enterprise:** Permet accedir a la informació de l'empresa, incloent: configuracions, usuaris, equips i repositoris.



-**scim:enterprise:** Facilita la integració amb serveis de gestió d'identitat per automatitzar la gestió d'usuaris.

Hem explicat les especificacions que considerem importants, en el cas que tingueu qualsevol dubte sobre les següents: [manual GitHub tokens]

Després de tenir totes les especificacions que necessitem activats, anem al final de la pàgina i cliquem el botó “Generate token”:



Nota: Després de donar-li al botó, ens sortirà un codi, que haurem de copiar en un lloc de confiança, es importat que ningú sàpiga de la existencia del teu codi, ja que poden entrar amb el teu usuari de github y el teu codi y fer canvis dintre dels teus repositoris.
En el cas de que perdí el token, haurà de generar un nou, perquè no es pot recuperar.

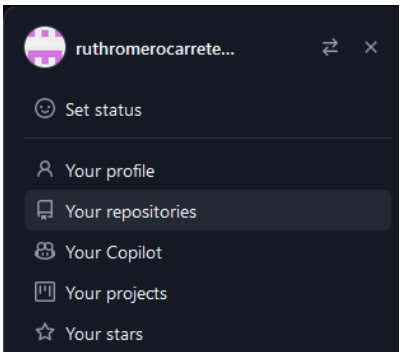
---

3. Com crear un repositori

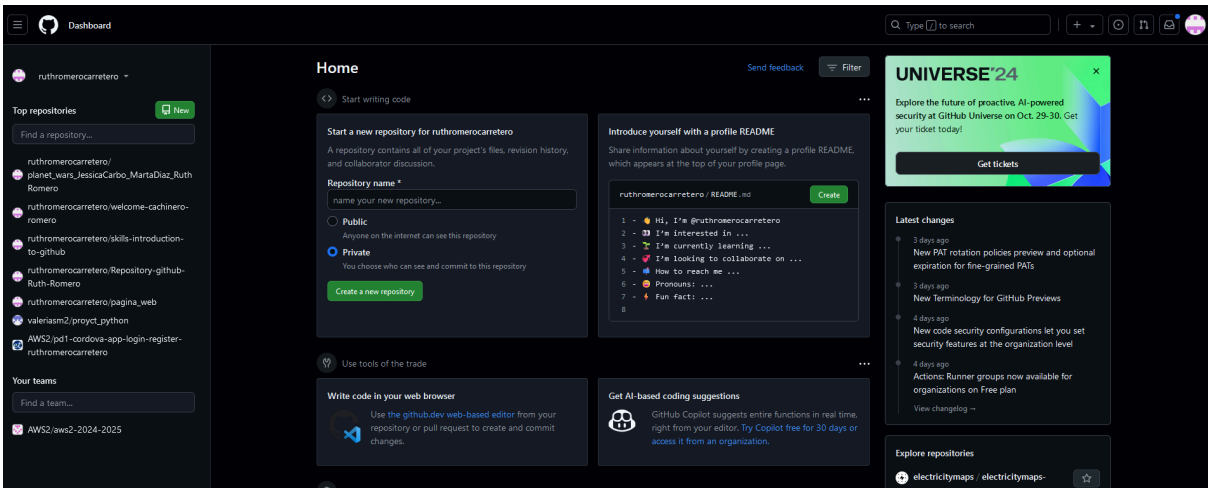
3.1 Crear un repositori des de GitHub

Una vegada iniciem sessió endins del github, ens desplatzem fins la part superior dreta, on estarà la nostra foto de perfil, li donem clic esquerra:

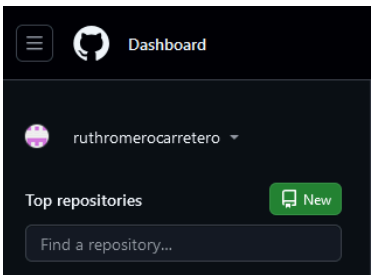
Seguidament després de clicar-ho sortirà aquest menú, on clicarem l'opció "Your repositories", per entrar dintre dels nostres repositoris, això es per poder accedir als repositories existents, en el cas que no tinguessim cap, no sortiría cap repositori dintre d'aquest submenú:



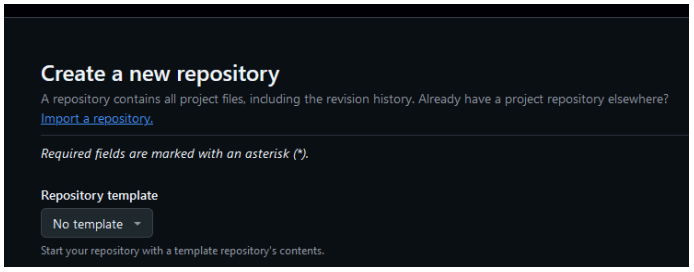
Una vegada estiguem en la página inicial, amb la sessió iniciada, anem a la secció de l'esquerra:



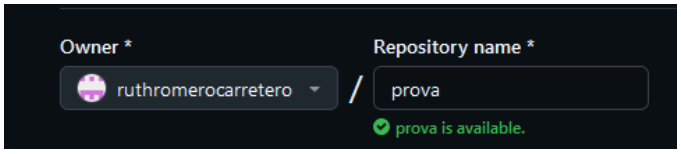
Pulsem el botó "New", per crear el repositori:



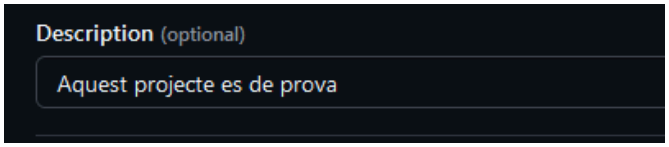
S’obrirà aquesta pàgina:



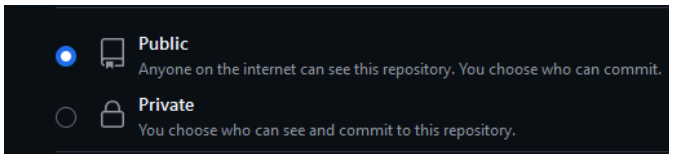
Repository template:
És en el cas que tinguem una plantilla d’un repositori, per si volem utilitzar el mateix.



Owner/Repository name:
Es la teva conta/el nom que tindrà el repositori. En aquest cas com no tenim cap repositori que es digui prova, em diu que es pot anomenar així.

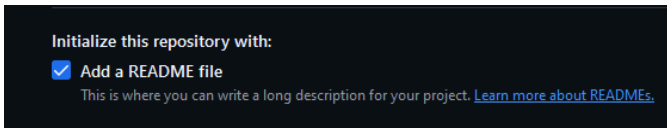


Description: Un espai on podem escriure una petita descripció del nostre projecte.



Public/Private:

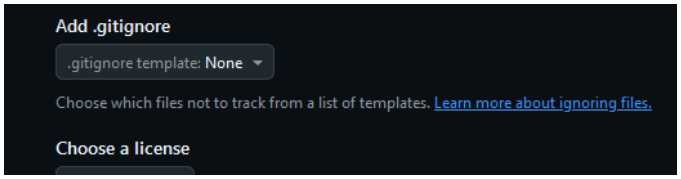
- Public: posar el nostre repositori visible per tot el món
- Private: posar el nostre repositori privat, visible per nosaltres i pels col·laboradors.



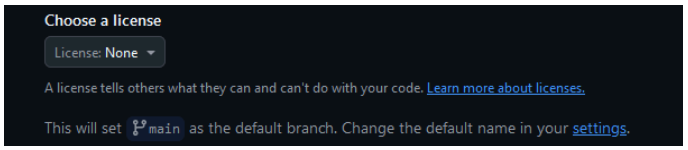
Add a README file:
Podem afegir un document, on posarem:

- Descripció del projecte.
- Instal·lació.
- Ús.
- Contribucions.
- Licencia.
- Crèdits.
- Dependencies.

El arxiu **.gitignore** és un arxiu de la configuració en Github que especifica què arxiu i directoris han de ser ignorats per Github i no ser rastretjats ni

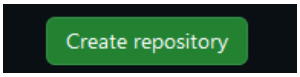


incluids en el repositori. Es útil per evitar que arxius innecesaris o confidencials es pujin al control de versions.

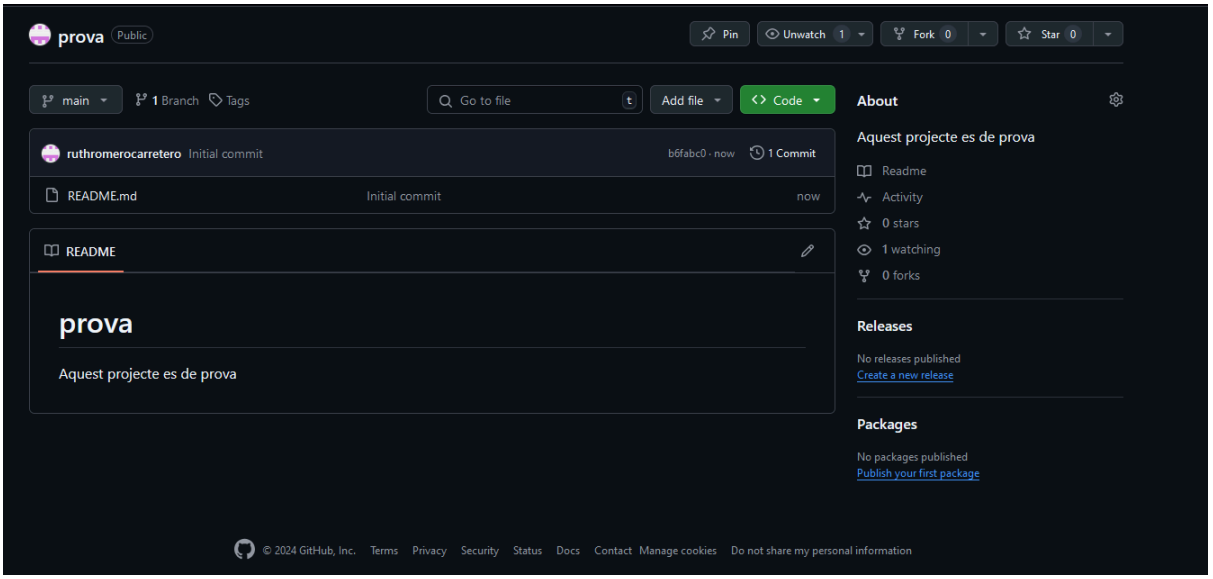


License
És un document que estableix els termes baix els quals el codi d'un projecte pot ser utilitzat.

Després de tota la configuració, li donem clic a "Create repository":



Una vegada li donem al botó, sortirà aquesta finestra:

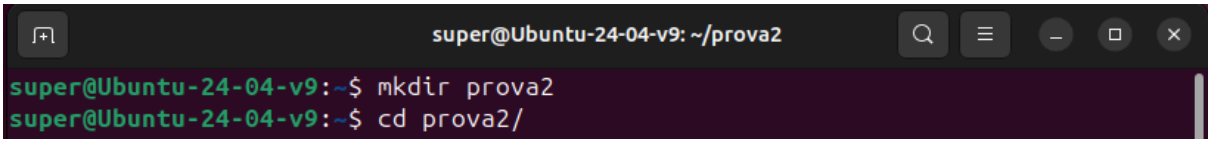


Nota: Dintre de dues seccions s'explica els passos per clonar un repositori de Git a la teva màquina local. Assegura't de tenir instal·lat Git abans de procedir. Clonar un repositori et permet obtenir una còpia exacta del projecte, incloent-hi tot l'historial de commits, branques i etiquetes. Aquesta operació és essencial per començar a treballar localment amb el codi del projecte.

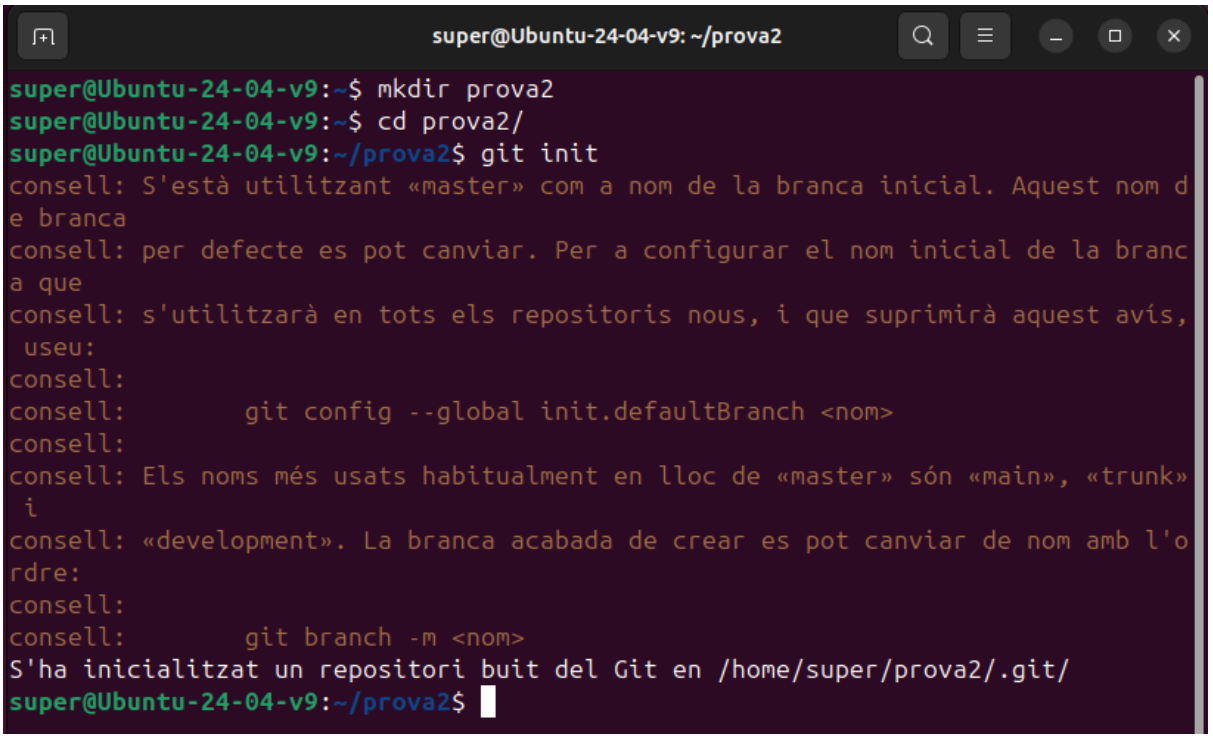
3.2 Crear un repositori des de local

Obrirem la terminal, crearem un directori on allotjarem el nostre repositori en local, les comandes són:

- `$mkdir <nom_directori>` : per crear el directori utilizarem mkdir i seguidament posarem el nom del directori
- `$cd <nom_directori>`: utilitzarem esta comanda per entrar dintre de la carpeta que acabas de crear.

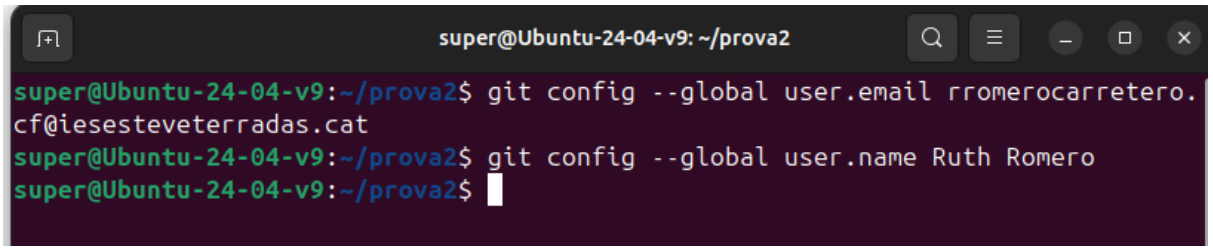


Seguidament, executarem la comanda:
`$git init`: Converteix la carpeta actual en un repositori Git buit. I estableix sol la primera branca del projecte.



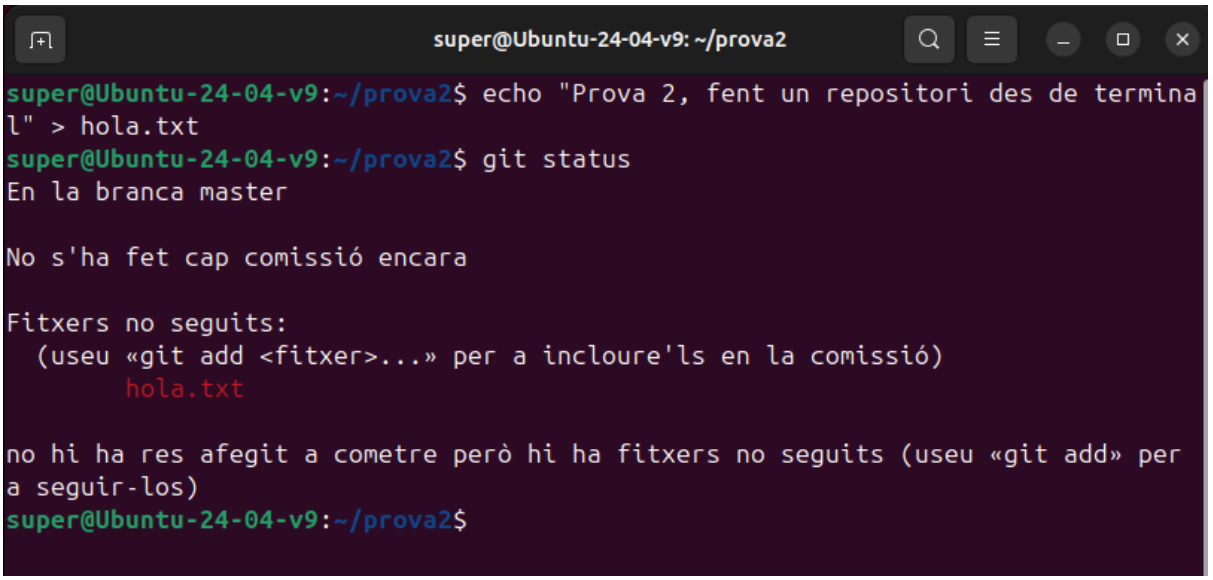
Nota: En el cas de que doni error, repeteix els passos anterior, i aprofita per iniciar sessió, amb el teu email i el teu usuari.

Iniciem sessió, ja que si ara fem un $git push (comanda que hem explicat amb anterioritat ens donarà un error, ja que no sap el nostre usuari, ni a que repositori ens estem referint), configurant el Git de l'ordinador:



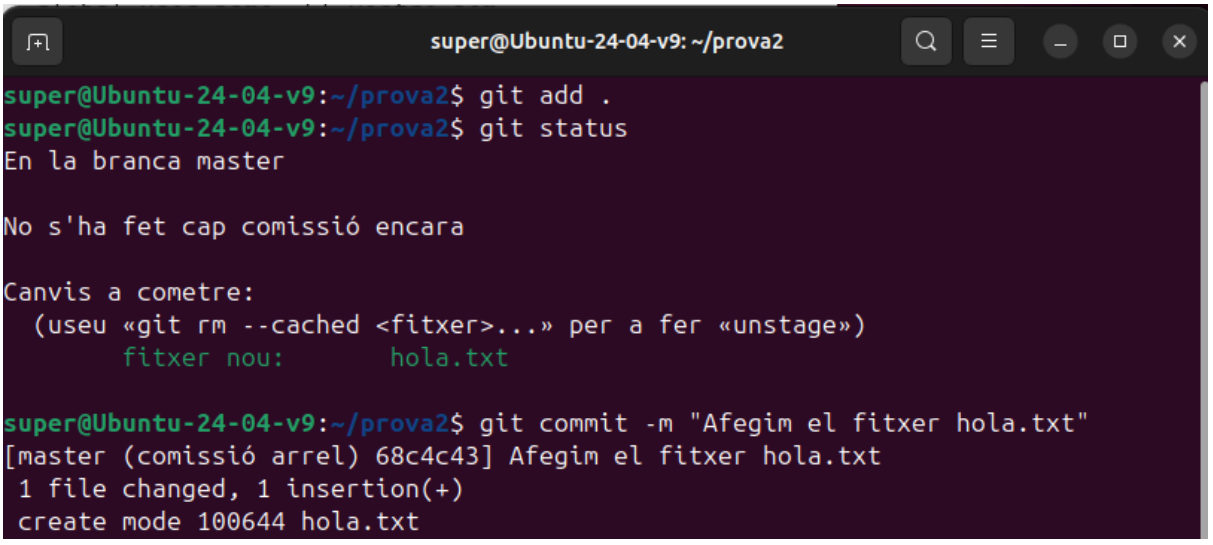
- `$git config -global user.email <email_que_has_utilizat_al_crear_l'usuari_de_github>`: li donarem el email que utilitzem en el Github
- `$git config -global user.name <el_teu_nom>`: li posem el nostre nom.

Per poder realitzar canvis, creem un arxiu, en el nostre cas es dirà hola.txt:



- `echo "<missatge_que_estara_dintre_del_fitxer>" > nom_fitxer.extensio`: creem un fitxer que té una frase, per poder pujar-lo.
- `$git status`: com no ho hem afegit amb el $git add . es pot veure com està de color vermell i a sobre del fitxer posa que hem de fer el `$git add .` per poder començar el procés de pujar-lo.

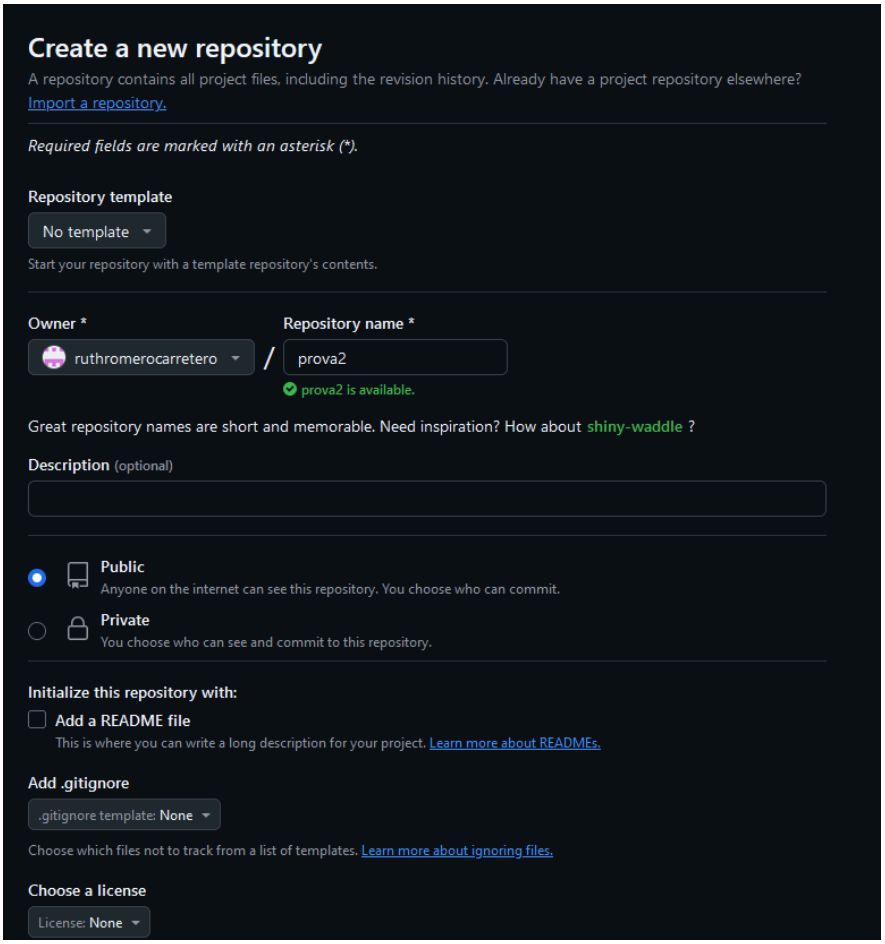
Comencem el procés de pujar el fitxer per poder veure els canvis dintre del repositori:



- `$git add .`: seleccionem el fitxer per poder fer commit.

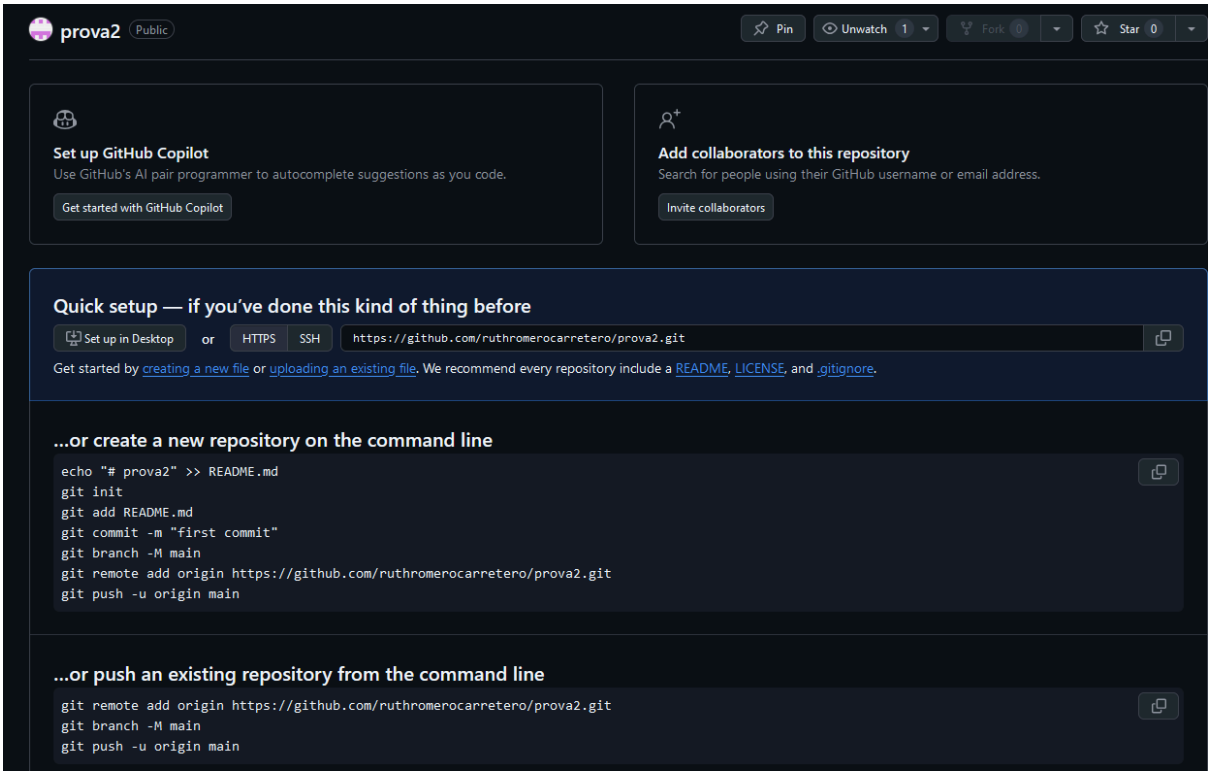
- `$git status`: veiem l'estat del fitxer, com anteriorment hem fet el git add ., podem veure de que ja no es de color vermell, sino que ha canviat a color verd.
- `$git commit -m "<missatge>"`: li assignem un missatge a tots els canvis anteriors.

Ara hauriem de fer un `$git push`, però com no existeix el repositori dintre de GitHub, hem d'anar dintre de la pàgina de GitHub, i crear un repositori nosaltres mateixos. Per això com anteriorment he descrit en la creació anterior de repositori, anirem fins aquesta finestra.

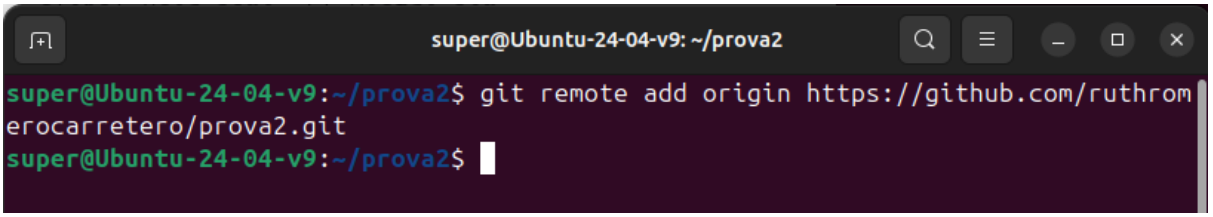


Nota: No podem afegir un README.md, GitHub iniciaria directament el repositori creat un commit, i en el moment quan executem la comanda `$git init`, ens donarà un error dient que ja hi ha un commit, donant conflictes.

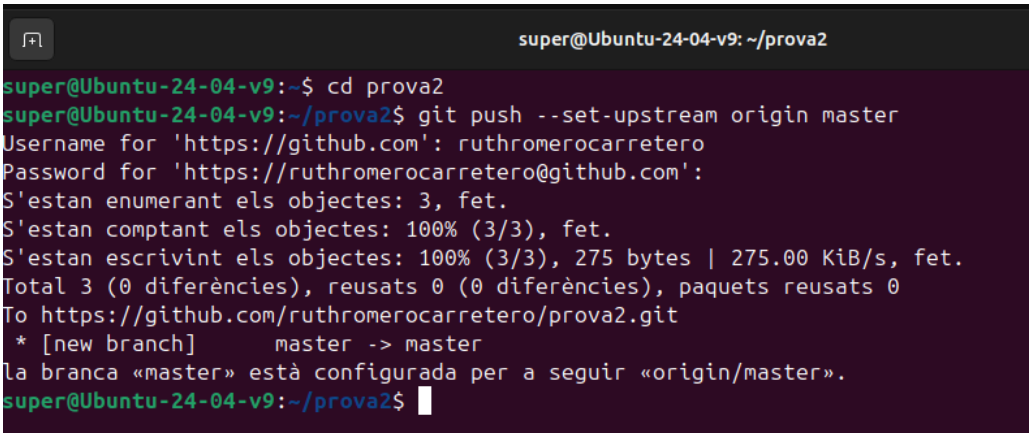
Seguidament, després d'haver creat el repositori, ens sortirà aquesta pàgina, on agafarem la primera comanda de l'últim quadrat de la captura:



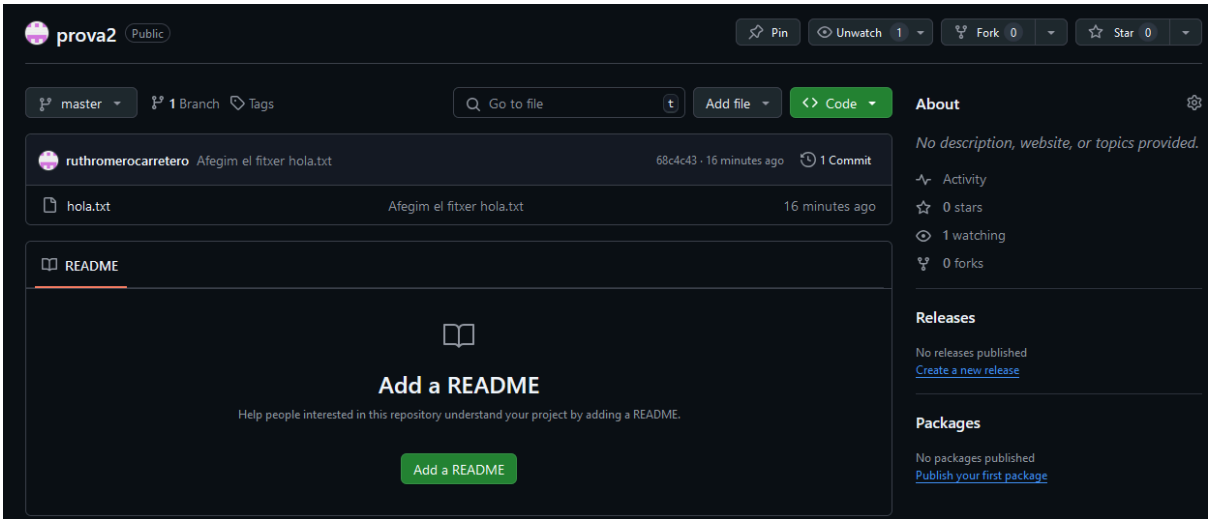
Anem a la terminal i li posarem la comanda, lo que farà aquesta comanda es enllaçar el repositori local amb el repositori que acabem de crear endins del GitHub:



Amb el `$git push -set-upstrem origin master`, li estem dient al GitHub que tingui en compte en les següent push que ho pujarà mitjançant origin (origin ho hem definit com GitHub):



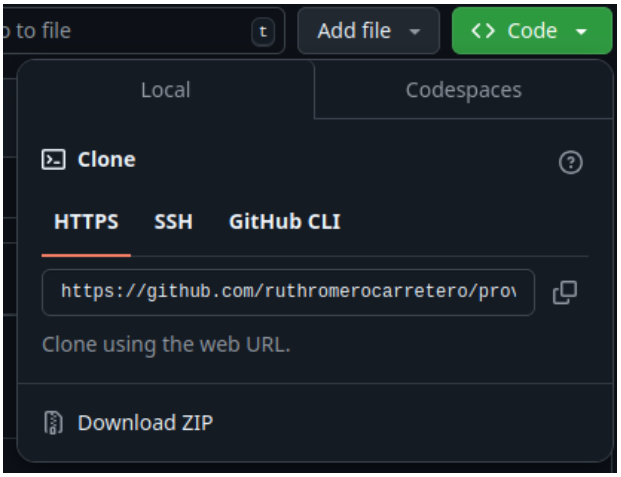
Entrem dintre del repositori de GitHub, i com es pot veure s'ha pujat correctament l'arxiu que hem creat abans:



---

4. Clonació del repositori a nostre màquina local

Li donem al botó verd que posa `<> Code` en cas que no estigui seleccionada la pestanya HTTPS, la seleccionarem nosaltres, i on està l'enllaç clicarem al botó que està posicionat a la dreta al costat de l'enllaç del https:



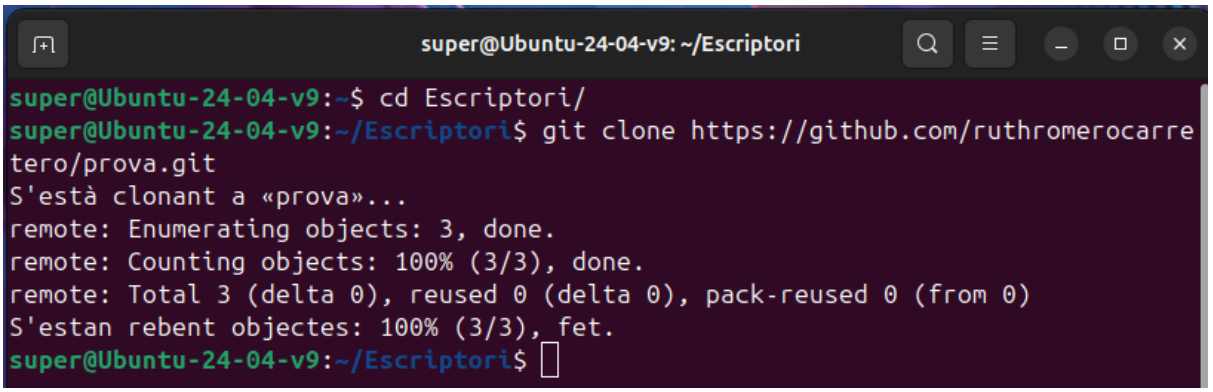
Tot seguit, obrirem la terminal del nostre sistema operatiu, en el nostre cas en ubuntu. Quan estigui obert, utilitzarem la comanda `$cd <directori_on_vulguem>/` per entrar dintre del directori on vulguem tenir el nostre repositori dintre de la nostra màquina.

Quan estiguem posicionats en el directori, executarem la comanda `$git clone (cliquem les tecles ctrl+shift+v per copiar l'enllaç) <enllaç_del_github>`, per clonar el nostre projecte dintre del directori en el nostre ordinador.

Nota:

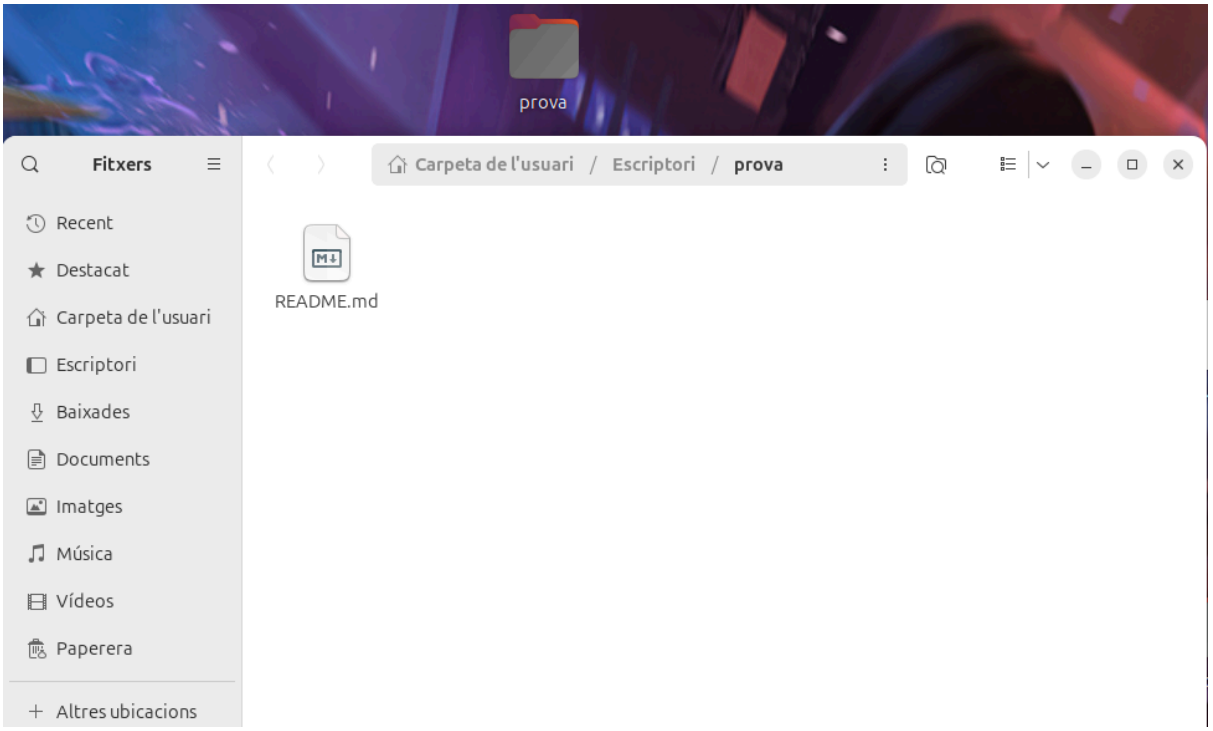
- `$cd <carpeta/directori>` es per navegar entre els directoris del nostre sistema.
- `$git clone <enllaç_del_github>`: es per clonar el repositori que acabem de crear dintre del nostre ordinador

Ens hauria de sortir un missatge com aquest:



Nota: En el cas que doni error, fer els passos d'abans o buscar dintre de la pàgina oficial de GitHub

Obrim la carpeta que s'ha creat al fer la clonació, per verificar que està dintre el fitxer que ja vam crear (README.md):



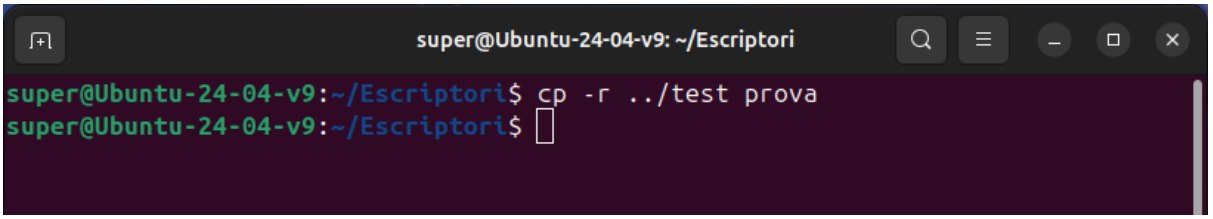
En el nostre cas, dintre d'aquest directori pujarem tota la informació del projecte que estem fent de nodejs:

`$cp -r <ruta_on_esta_el_fitxer_ruta_relativa>/fitxer.extensio <carpeta_repositori_local>`:

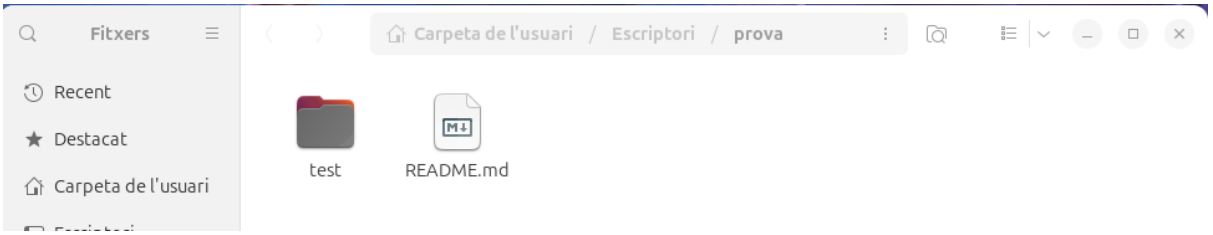
- Cp: s'utilitza per poder copiar fitxers, quan es posa la ruta, es perquè la màquina local sàpigui de quin fitxer estem parlant en concret on està guardat. Afegit a la ruta el nombre del fitxer que volem copiar.
- <carpeta_repositori_local>: posem la ruta entera del nostre directori local.

Nota: en el cas que sorti un error per permisos, hauriem de copiar el fitxer amb sudo. Es dir:

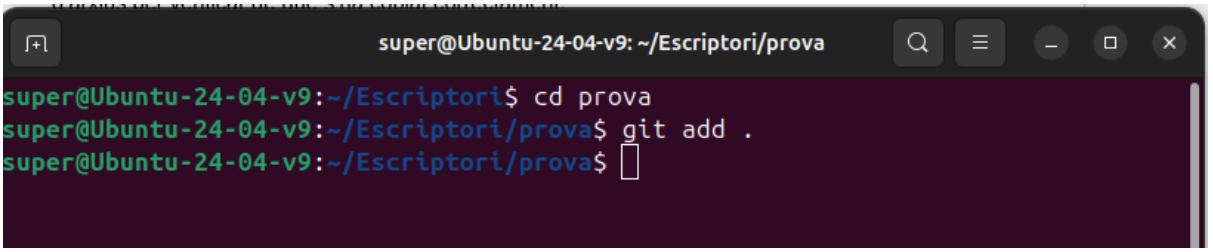
```
$sudo cp <ruta_on_esta_el_fitxer>/fitxer.extensio
<carpeta_repositori_local>
```



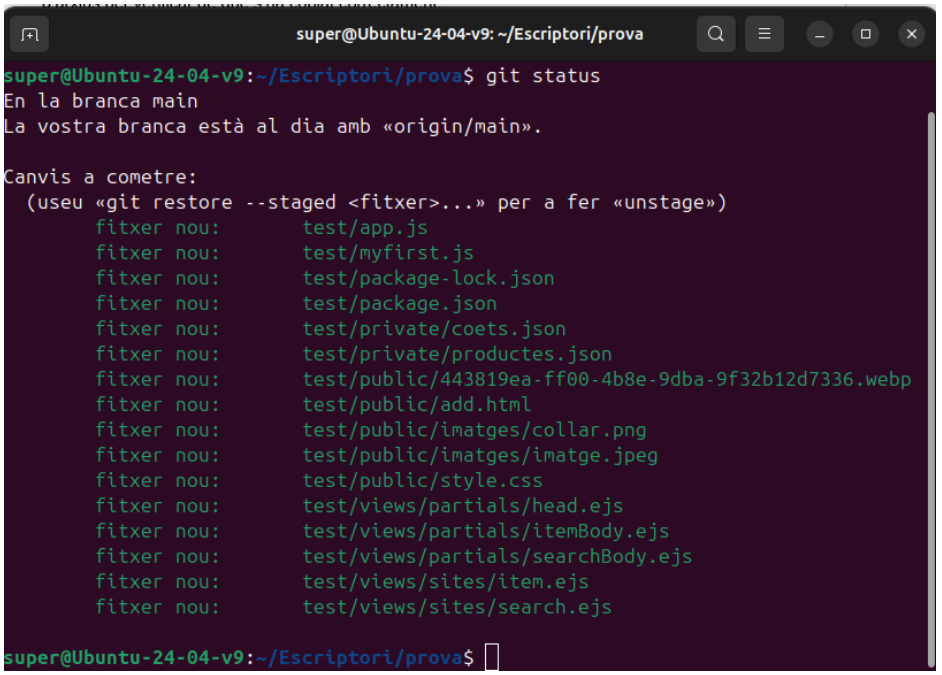
Després de copiar els fitxers dintre del nostre repositori local, entrem dintre de l'explorador d'arxius per verificar de que s'ha copiat correctament:



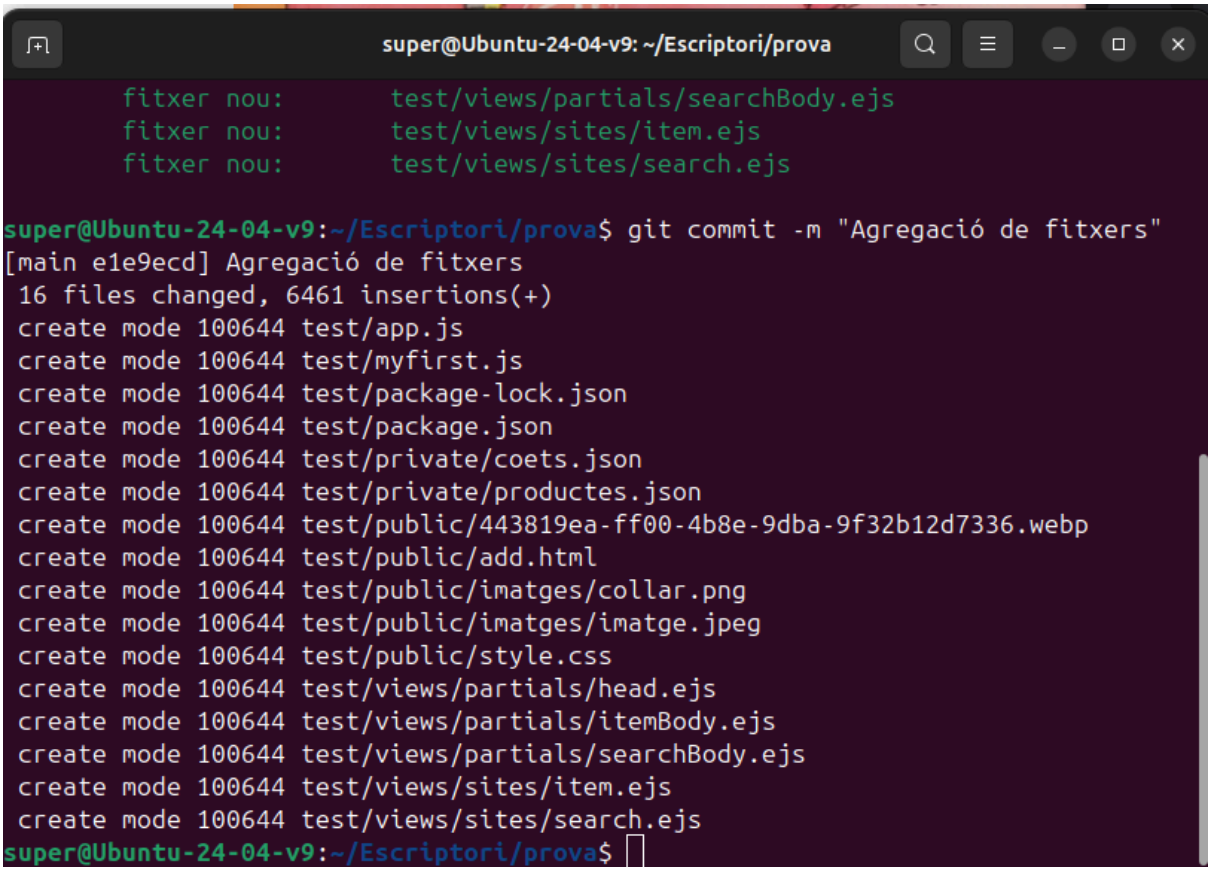
Entrem dintre de la consola de comandes, i fem un git add ., per introduir els fitxers al procés de pujada al github:



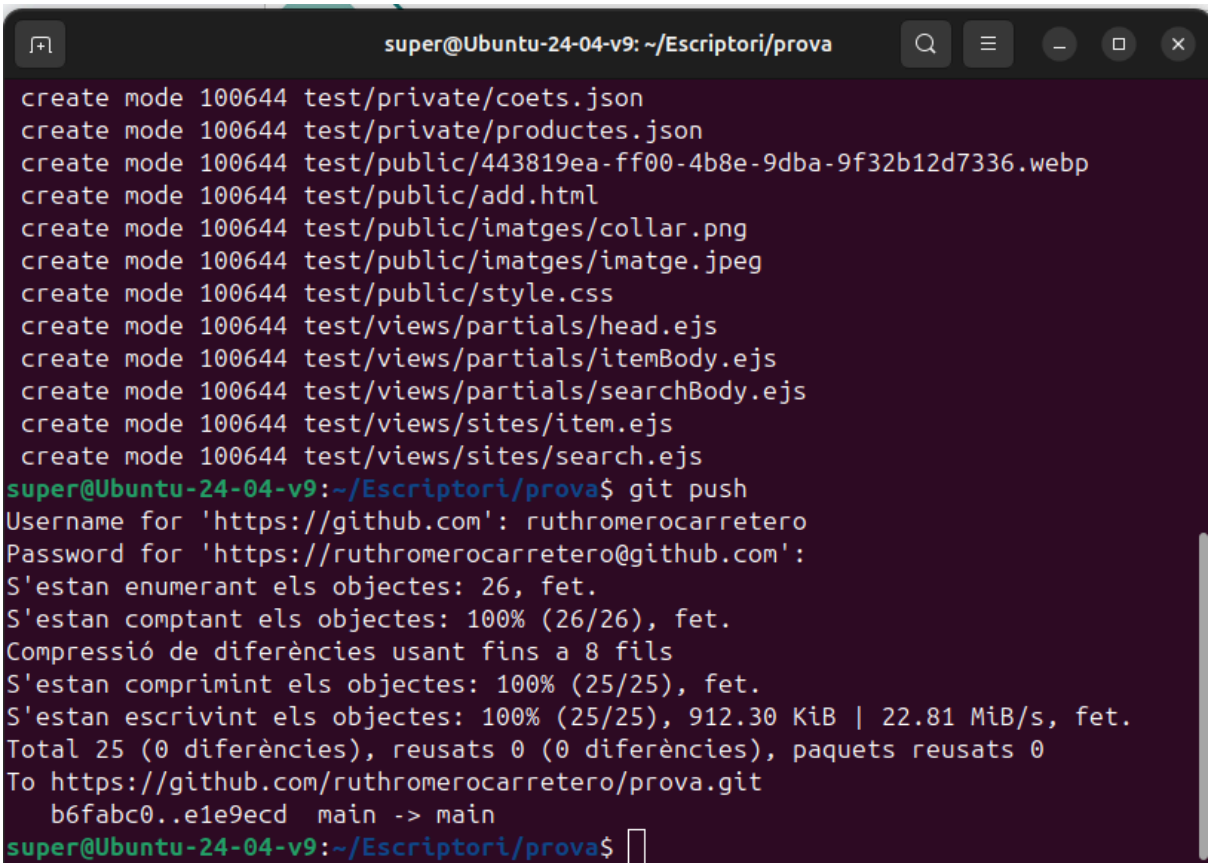
Executem la comanda `$git status` per verificar de que realment s'han afegit els fitxers al procés:



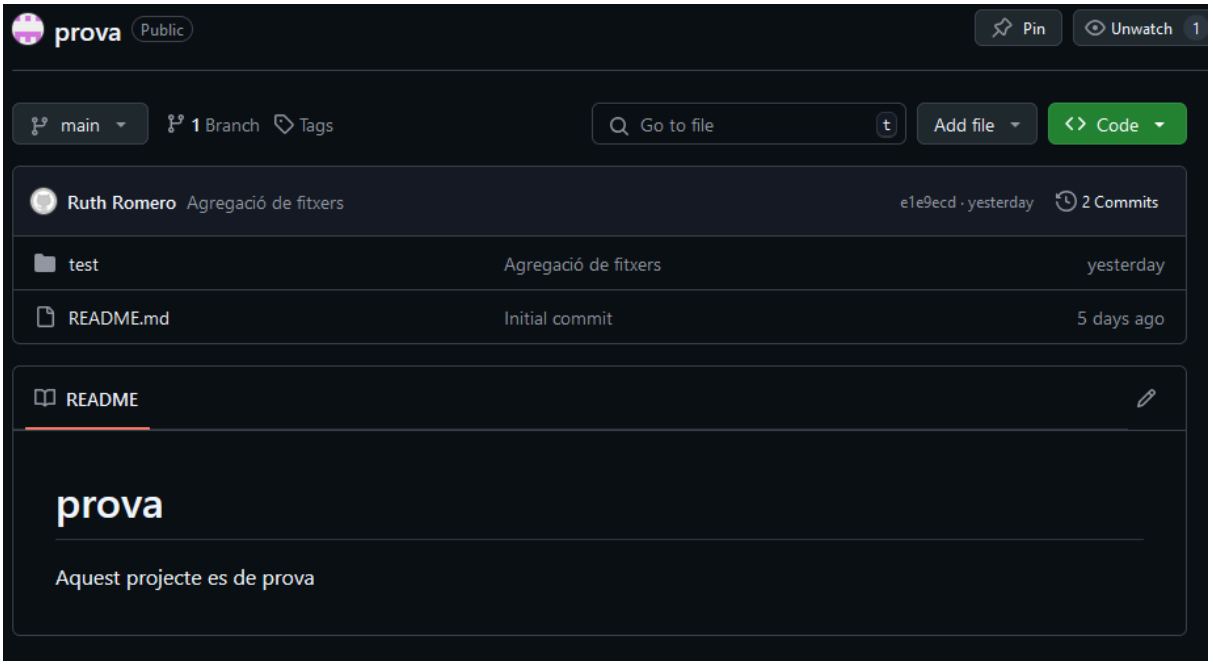
Seguit fem un `$git commit -m "<comentari>"`, per afegir un comentari a la pujada dels arxius i si surt una llista posant tots els fitxers afectats, com es pot veure en la captura, significa que és correcte:



I l'última comanda que executarem serà `$git push`, per pujar tots els canvis que hem realitzat en la màquina local a el repositori que tenim dintre del GitHub.
Ens demanarà el nostre usuari de GitHub i el *token* que hem generat anteriorment:

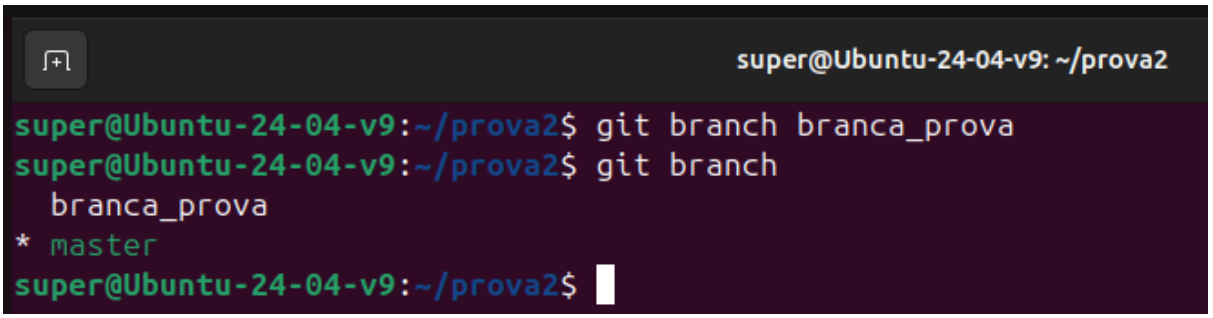


Entrem dintre de GitHub per veure si es poden ver els canvis i com es pot veure en la captura, tots els canvis son perfectament visibles:



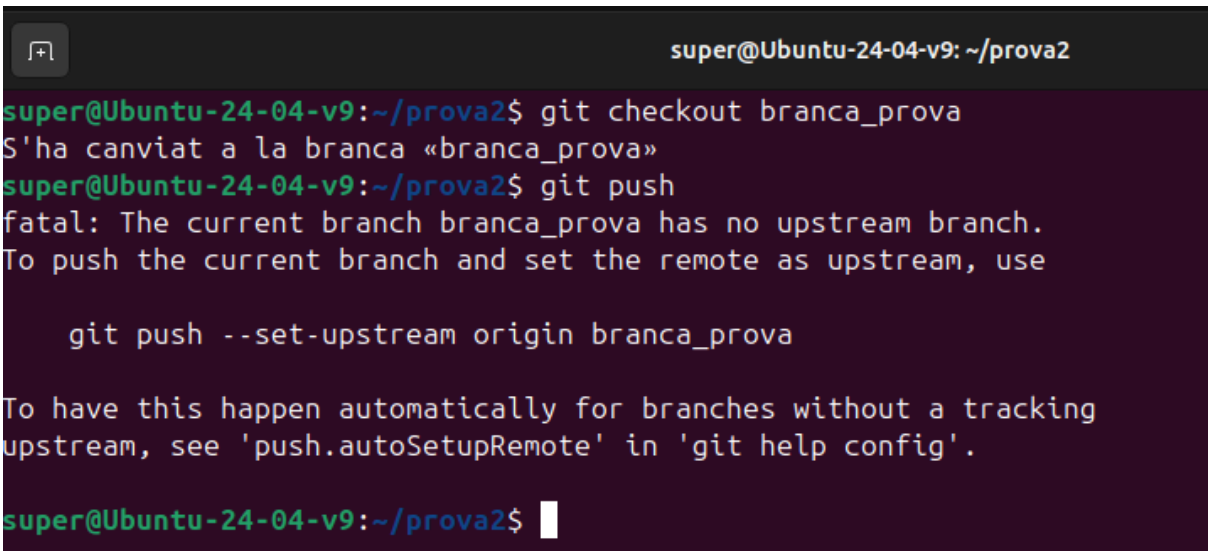
5. Creació de branches

5.1 Creació de branch des de terminal



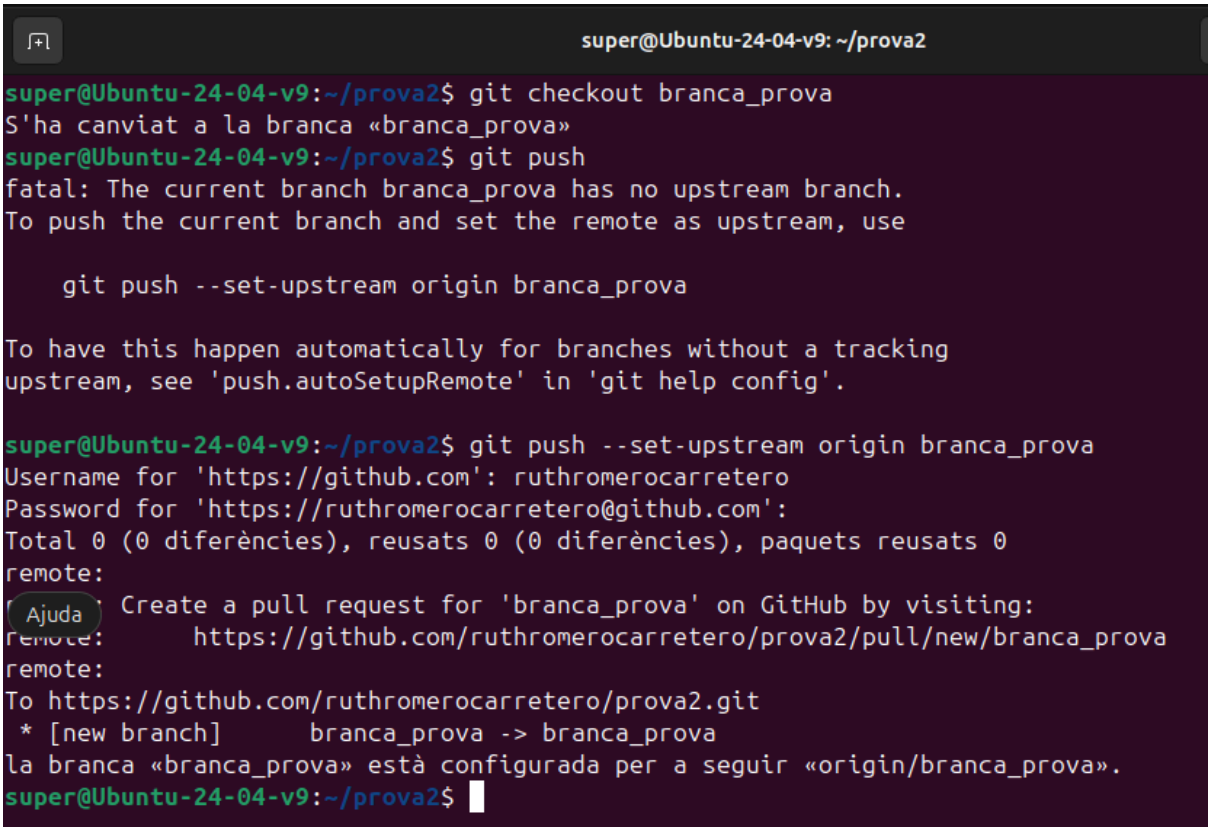
- `$git branch <nom_branch>`: aquesta comanda s'utilitza per crear una branca nova.
- `$git branch`: s'utilitza per sapiguer que branches existeixen.

Canviar de branca:

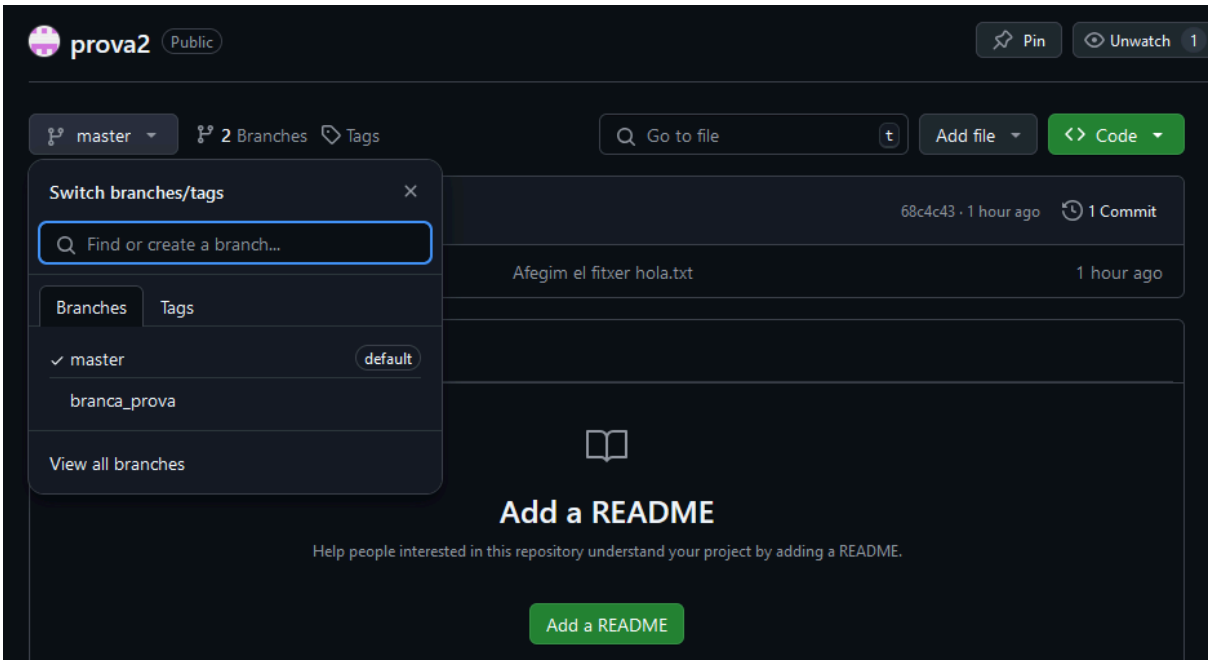


- `$git checkout <nom_branch>`: es per canviar de branch.
- `$git push`: per pujar-lo i com es pot veure, com es la primera vegada que fem un push en aquesta branca, haurem de executar la comanda que ens dóna el propi git ($git push –set-upstream origin branca_prova)

I com es pot veure, després de executar la comanda anterior, iniciem sessió amb el nostre usuari i funciona correctament:

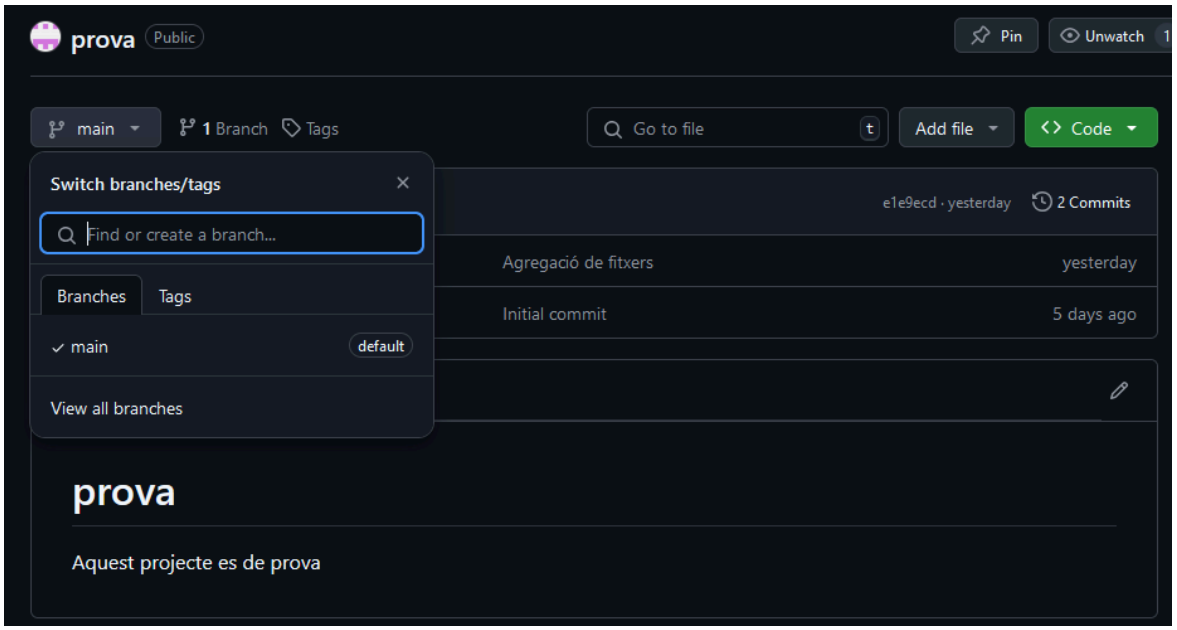


Entrem dintre de la pàgina de GitHub i ja surt la branca creada:

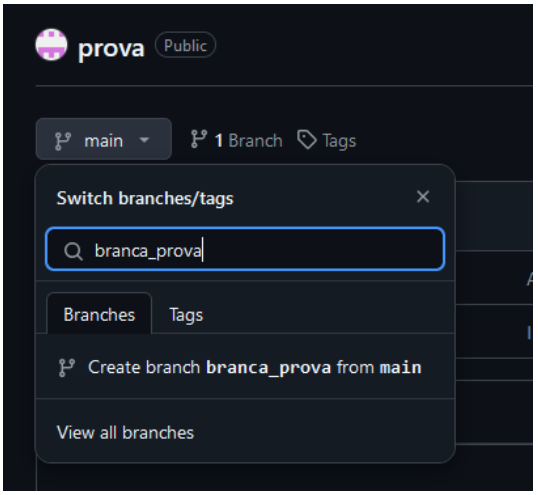


5.2 Creació de branch des de web

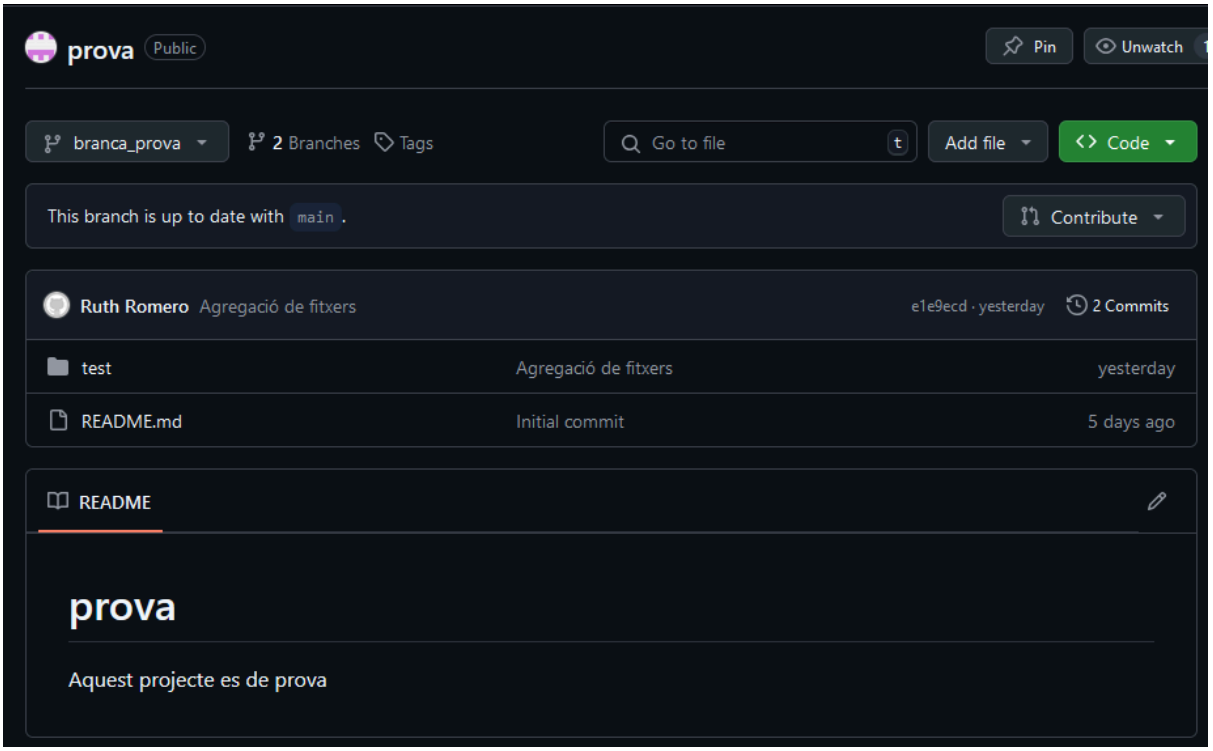
Anem dintre de la pàgina del nostre repositori, i on posa main:



Dintre del cercador, posem com volem que es digui la nostra branca:



I com es pot veure ja podem entrar dintre de la branca nova encara que no hi hagi cap fitxer dintre de la branch:



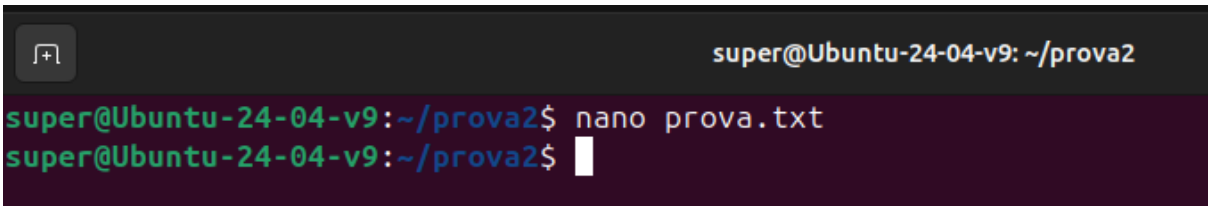
---

6. Què és un Merge?

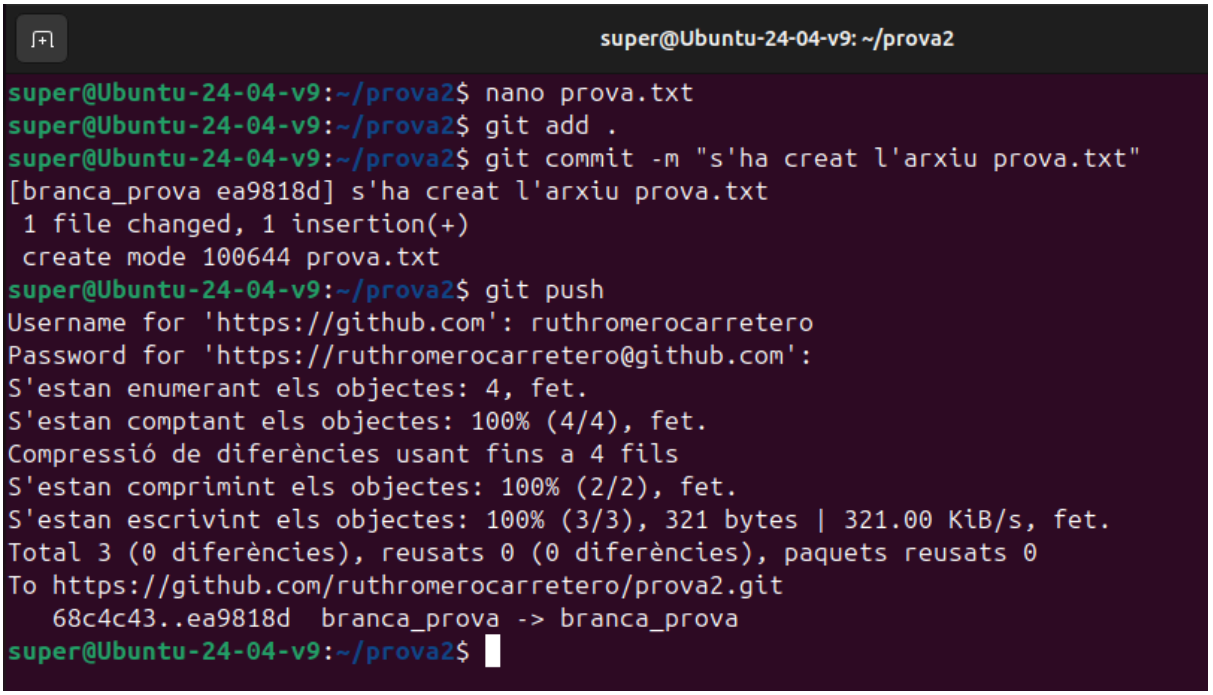
Es el procés de integrar els canvis de dos branques diferents en una sola branca:

```
$git checkout <nom_de_la_branca_base>
$git merge <nom_de_la_branca_objectiu>
```

6.1 Creació de fitxer en la branch branca_prova



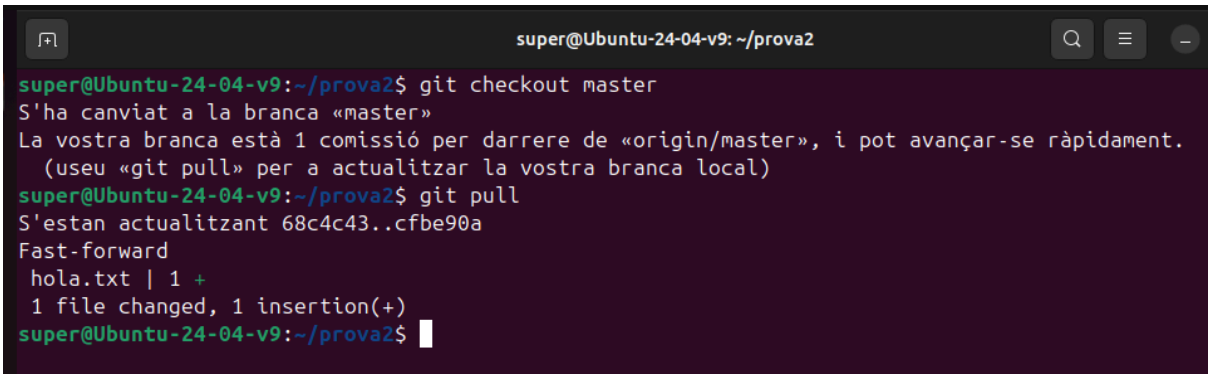
Introduïm informació dintre del fitxer i guardem:



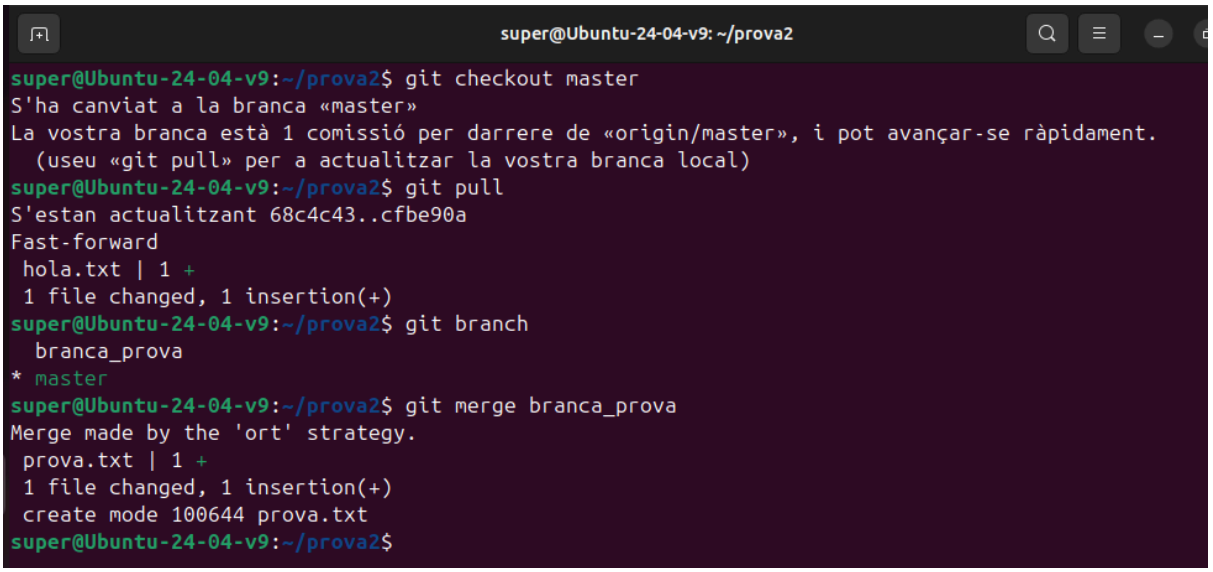
`$git add .`	Afegeix tots els canvis nous dins del directori actual al "stagging area".
`$git commit -m`	Guarda tots els canvis a l'stagging area amb un missatge.
`$git push`	Envia els commits al repositori remot.

Posem les nostres credencials de GitHub, tant el nom d'usuari com el *token* que hem generat abans.

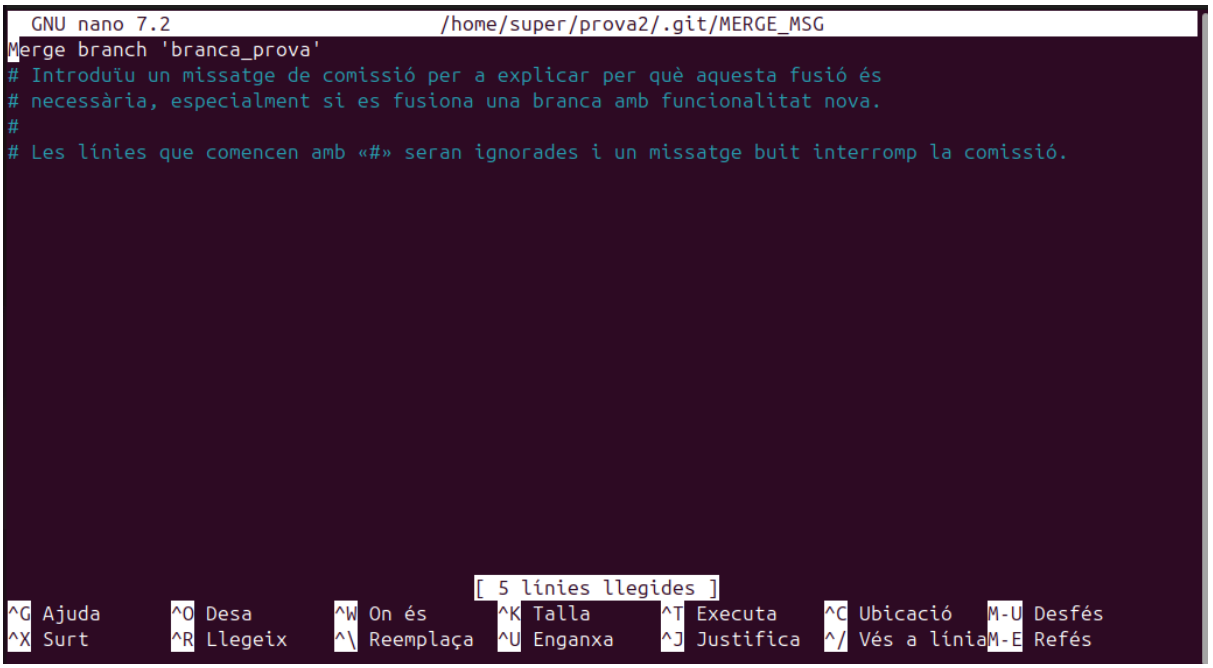
Seguidament fem un `$git checkout <nom_de_la_branca_principal>` per canviar de branca a la principal que en aquest cas es master. I fem un `$git pull` per actualitzar el directori en el cas de que hi hagi canvis:



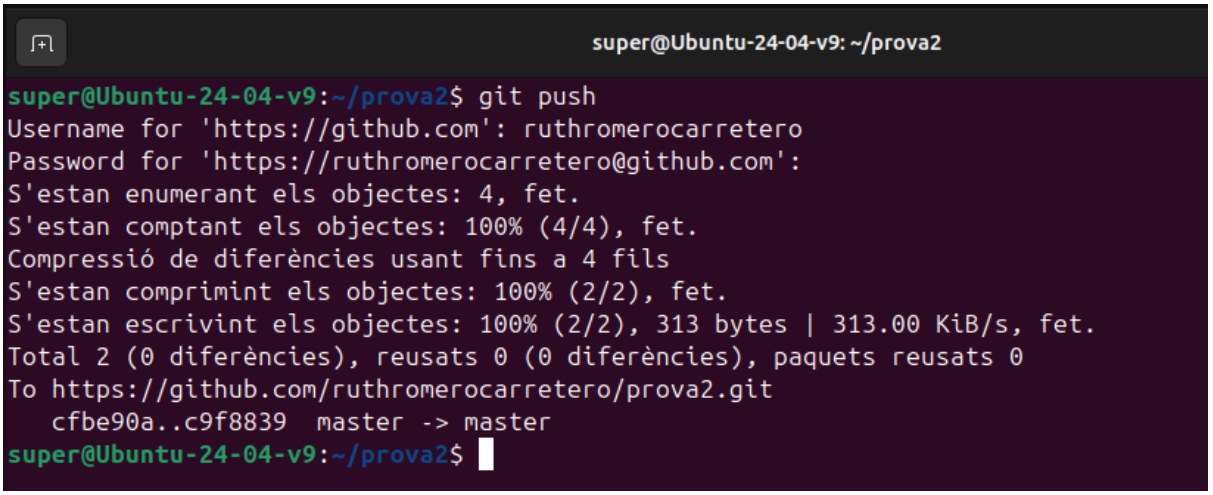
Executem `$git branch` per veure les branques que tenim dintre de la repo. Després executem `$git merge <nom_de_la_branca>` per combinar les línies de desenvolupament que has creat, sovint des de diverses branques:



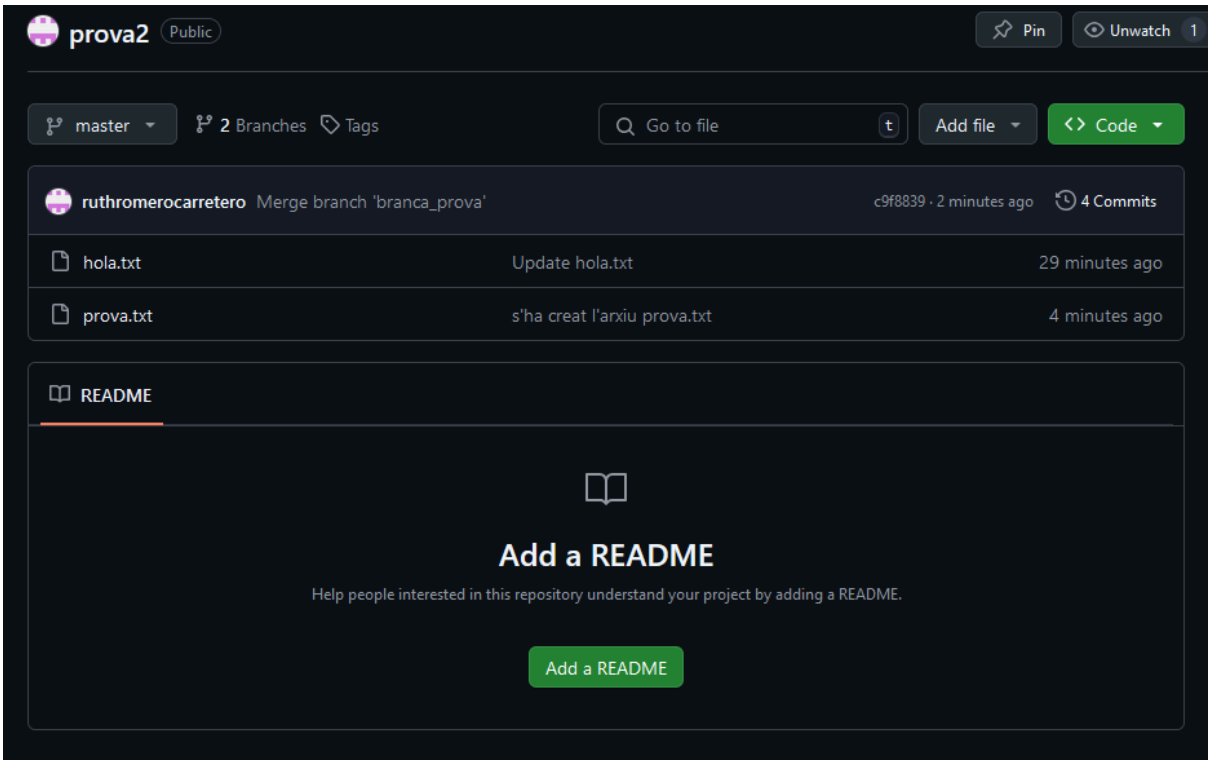
Al executar la comanda `$git merge <nom_de_la_branca>` Git genera automàticament un fitxer de missatge de fusió anomenat </home/super/prova2/.git/MERGE_MSG>. Aquest fitxer inclou els commits que es fusionen. Pots revisar i modificar aquest missatge abans de confirmar la fusió per assegurar-te que és clar i informatiu.



Si està tot correcte, tanquem el fitxer fent Ctrl + X, i executem la comanda `$git push` per pujar al GitHub el merge que acabem de fer.



Entrem dintre del nostre repositori en el navegador i ja podem veure com el visible el merge:



Webgrafia

1. Referència visual de com es veuria el manual i explicacions sobre com funciona GitHub: https://git-scm.com/docs

2. Ajuda per explicar les comandes i els conceptes bàsics: https://aprendeconalf.es/docencia/git/manual/manual-git.pdf

3. Explicació i implicació de la seguretat de autenticació i la creació dels *tokens*: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens