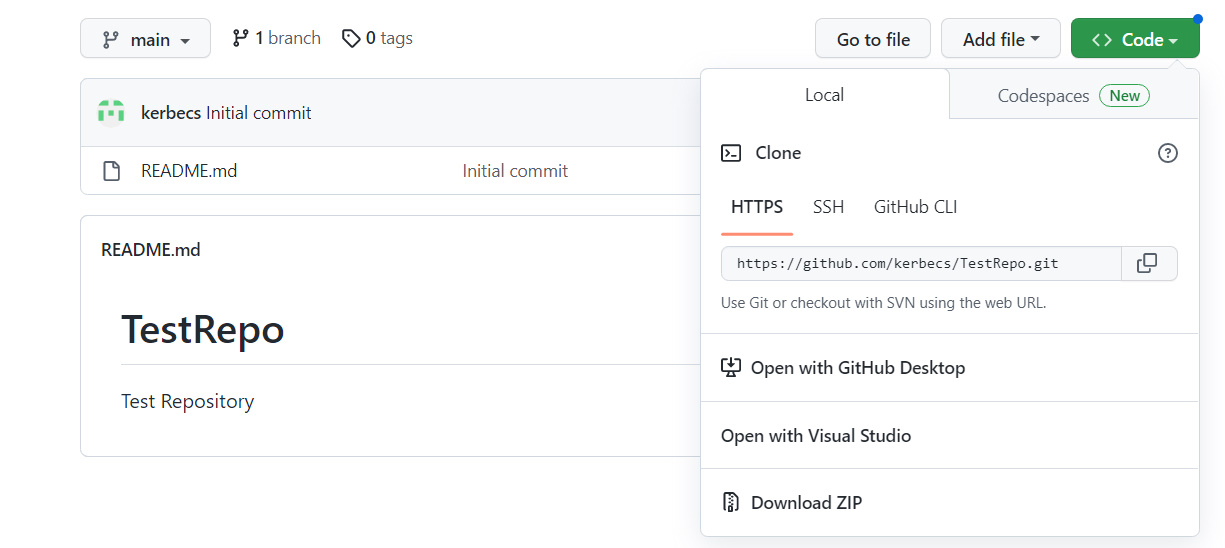
**GitHub**

* Git este un Distributed version control System
* Distributed – inseamna ca fiecare developer are o copie a intregului repozitoriu pe masina lor locala.
* Putem apoi sa hostam aceasta copie pe un server si sa il folosim ca un remote repository
* GitHub anume aici ne ajuta
* GitHub – este un web-based Git repository hosting service
* GitHub include si chestii suplimentare, inafara de control version system pentru Git, ca bug tracking, wikis, task management
* El ne permite sa hostam repozitorii si sa le accesam. Putem face clone ale acestora pe masina locala ca sa lucram cu ele.

**Clone**

* Putem clona orice repository de pe github cu comanda git clone url,chiar si care nu sunt ale noastre, si va trebui loginul si parola de pus

Putem sa apasam pe un buton pentru a vedea link, command etc.



**Remote**

* Cand cream o copie a unui repozitoriu de pe gitHub, cream un repozitoriu local care interactioneaza cu cel remote(hostat pe github)
* Local repository permit sa facem modificari in proiect fara sa afectam repozitoriul central direct, si fara sa modificam repozitoriile copii ale altora.
* Cu Git, putem lucra usor in echipa. De ex, daca la un joc lucreaza mai multi oameni, si fiecare are o parte din joc de care raspunde, fiecare copie remote repository de pe GitHub, face ce are de facut, si cand termina, muta modificarile facute in remote repositorty de pe GitHub. Toti trebuie din cand in cand sa foloeasca git pull pentru a se asigura ca local repository contine si munca altora.
* Nu uitam ca cand dam git push, se va pune in remote repository commitul nostru HEAD din branch in care scriem comanda, deci se vor adauga strict doar modificarile facute de noi in acel branch, asa desi pe pc avem tot proiectul, se va pune doar bucata la care noi am lucrat pe github,in branch respectiv. E problema insa cand 2 sau mai multe persoane fac modificari in acelasi loc si dau push. Acolo pot aparea merge conflicts
* Daca avem branches create in local repository, ele nu vor merge toate odata cu git push in remote repository, ci va merge doar acea din care scriem comanda.
* ATENTIE! Daca cream un branch local care nu exista in remote repsitory, va trebuie sa folosim alta comanda, nu git push(vezi mai jos), desi deobicei nu e necesar sa punem si branch create de noi ce nu sunt in remote, dar totul depinde de la caz la caz.
* **Daca cineva da push cat noi lucram, git ne va anunta ca e timpul sa facem update ca sa avem modificaril facute de altii in git status, git log, dar abea dupa git fetch**
* Cand vom face update, si apoi push,pot aparea merge conflicts, si va trebui sa le rezolvam.
* **Git push Nu va functiona daca nu avem cel mai nou commit din remote branch in local branch**

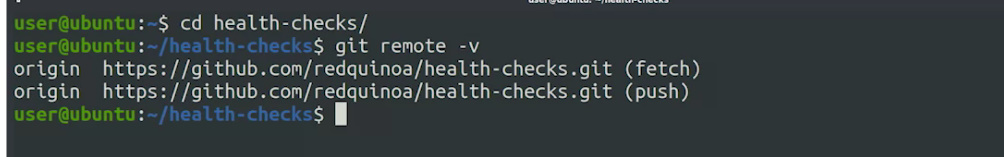
**ATENTIE!**

Nici-una din comenzile git status, git log, git log origin/main, etc. nu ne va arata date reale ACTUALE din remote repo pana nu vom da git pull sau git fetch(daca vrem intai sa analizam si apoi sa punem totul la noi).

**Unica comanda care ne arata daca remote branch e cu vreun commit inainte sau in urma e *git remote show origin*, chiar si fara git fetch**

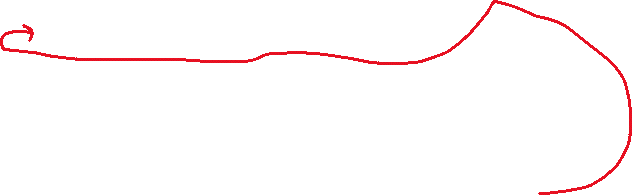
**More about remote**

* cand scriem git remote -v, apar 2 repository:

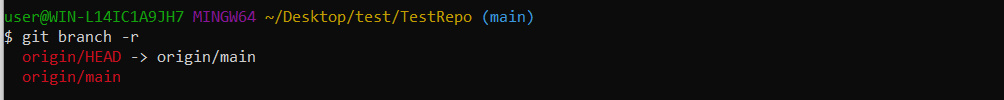


fetch – de la acest repository se iau datele

push – catre acest repository trimitem noi datele



* Deobice, push si fetch pointeaza catre aceeasi adresa, dar uneori putem avea un fetch cu HTTP doar pentru a citi date dintr-un repository si un push cu https sau ssh pentru a pune date intr-un al repository
* Orice remote repository au un nume default pus, anume *origin*
* Acest origin e folosit, deoarece putem avea mai multe repository in acelasi git directory
* Cand lucram cu remotes, git foloseste si remote branches(de pe github) pentru a pastra copii ale datelor pastrate in remote repository. **Remote branch e branch din remote repo, cu care local repo poate ineractiona.**
* Cand folosim git branch -r vedem doar remote branches la care repository al nostru e conectat, adica branches din remote repo:

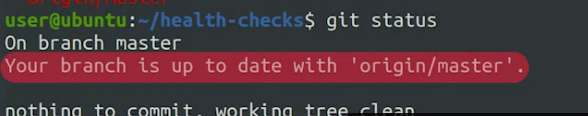


**Reguli inainte de git push**

1. Dam pull ca sa avem orice modificari facute in prooiect de altii puse la noi in local repository, sau fetch
2. Dam merge la aceste modificari cu ale noastre, daca e nevoie.
3. dam push la modificarile noastre in remote repository

E mereu necesar sa facem asa pentru orice remote branch!!!Asa vom fi siguri ca remote repository nu va avea conflicte sau probleme.

* Putem folosi git status pentru a vedea statusul modificarilor noastre si in remote branches. Asa vedem daca nu exista modificari in remote branch care noi nu le avem. Asta se poate intampla cand noi lucram si cineva pune vreun commit pe github



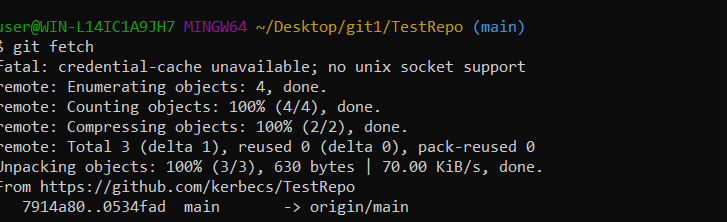
Acel mesaj ne spune ca, master branch in origin repository(cel remote) are aceleasi ultim commit ca master branch a noastra

**Metode de conectare la un remote repository**

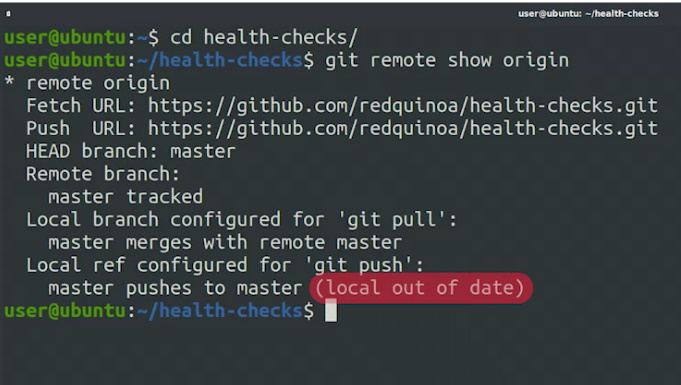
* HTTP – este mai mult folosit pentru a permite doar sa citim un repozitory.Deci permite doar sa clonam repozitory, fara a pune ceva in el
* HTTPS si SSH – permit autentificrea userilor si ne dau control de a seta cine poate adauga ceva in repositiory

**Fetching new Changes**

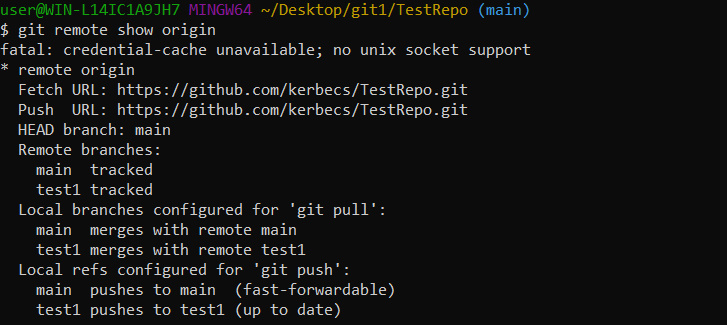
* fetch este doar ca o inlocuire pentru git pull! Nu si pentru git push
* **Deci, el se asigura ca sa se faca o copie la branch remote(deci downloand remote branch), insa noi putem doar sa o citim, nu o sa o si modificam in vreun fel. Cu push putem**
* Daca scriem git fetch, git ne va arata ce este nou in remote repo si nu avem pus in local repo. Insa, nu va face si invers!!!Adica nu arata ce avem nou in local repo si nu e remote. Comanda data e doar o substituire pentru git pull, nu si pentru git push



* git remote show origin – arata info despre origin si arata daca sunt diferente intre remote branches si local branches. Comanda arata mereu corect chiar si fara git fetch!!!



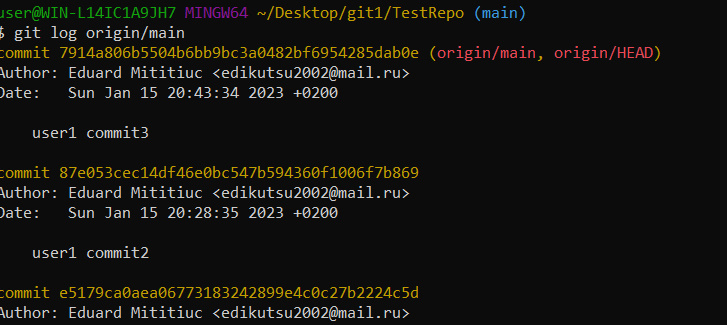
Aici vedem ca local branches(cele din local repository) nu au cele mai noi date introduse in remote repository. Deci, pe github sunt noi commituri in master branch.





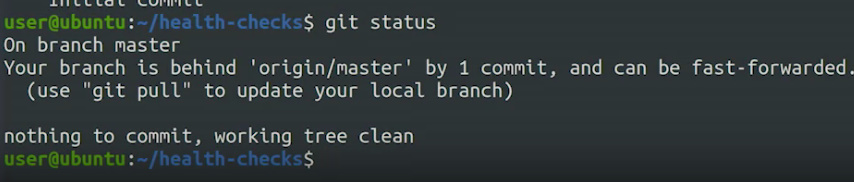
Aici dimpotriva, branch main din remote repo nu are cel mai recent commit din local main branch din local

* Git nu sincronizeaza automat local remote repo si branches. El asteapta ca noi sa dam fetch/pull si apoi update la date la aceste 2 tipuri de branches cum vrem noi, fie prin git pull/push sau prin git fetch + git merge ca inlocuitor pentru git pull
* Totusi, noile commituri nu sunt automat puse in local branches si nici in remote branches.
* git log origin/master – vedem log pentru master branch din remote



Vedem ca main local lipseste.

Iar daca rulam git status:



Aici git ne spune ca local master branch(locala adica) e in urma cu un commit fata de master branch din remote repository.

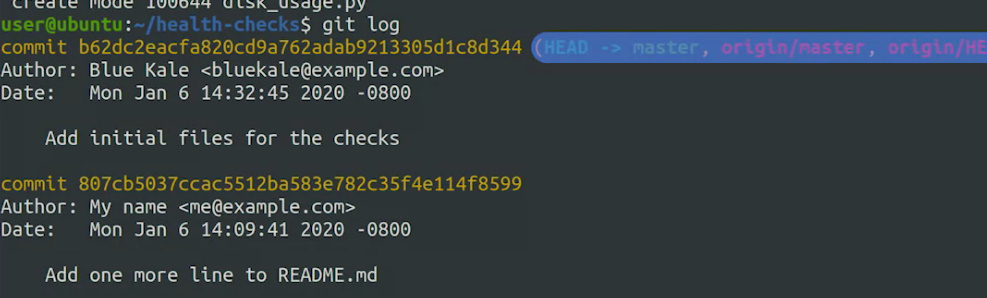


Aici git ne spune ca local master branch(locala adica) e in fata cu un commit fata de master branch din remote repository.

* Pentru a pune toate commiturile din branch master din remote in master branch din local folosim:

**git merge origin/master**

* git log ne va arata asta:



deci, vedem ca local branch are un ultim commit identic cu remote master branch

Atentie! Git merge origin/master nu va pune commiturile noi din master branch din local in master branch din remote repo.

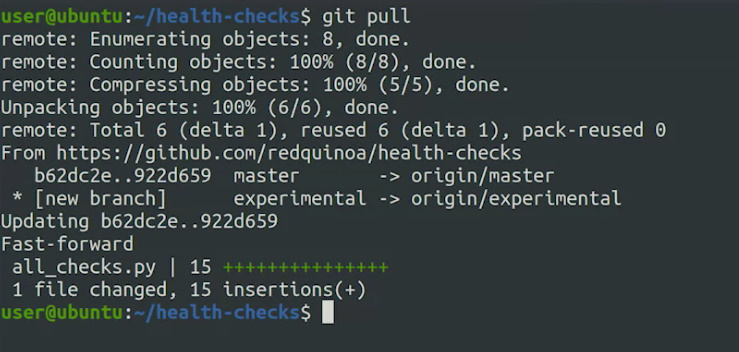
Comanda git fetch si git merge sunt o substitutie doar pentru git pull, nu si git push!!!

**git fetch vs pull**

git pull – pune deodata noile date la noi in repository

gut fetch – nu pune deodata noile date. Va necesita git merge. E bun ca sa vedem ce este nou in remote repository in acest branch. Daca ne place, dam merge. Chiar si fara a da merge, putem vedea ce e nou cu git log, caci acele commituri din branch master din remote vor fi aratate, dar git ne va atentiona ca ele nu sunt in branch local. Deci, el se asigura ca sa se faca o copie la branch remote, insa noi putem doar sa o citim, nu o sa o si modificam in vreun fel. Cu push putem

**git pull E combinatia dintre git fetch si git merge**





Vedem ca combina operatiile facute in ambele comenzi, doar ca doar prin folosirea uneia.

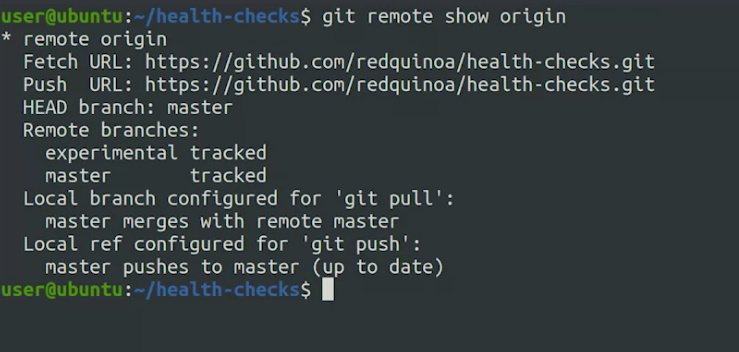


1 – git a depistat schimbarile din remote repo

2 – git a facut merge la branch

Dar, mai vedem ca este si un nou branch [new branch] cu numele experimental

Daca dam git remote show origin, vedem info despre origin:



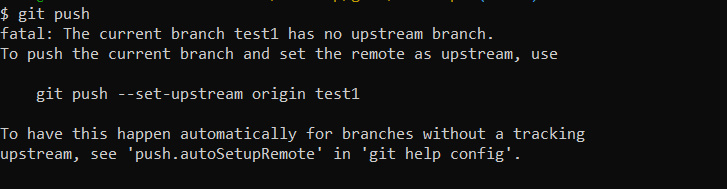
E un nou remote branche numita experimental

Aceasta branch noua nu o avem in branches ale noastre locale

Daca dam git checkout experimental, se va copia aceasta noua branch si se va conecta local la ea. Insa, daca avem noi un nou branch in local ce nu e in remote, e problema(vezi mai jos)

**Push new Branch**

* Daca in local repo avem un branch nou, care nu exista in remote, cand vom da git push, vom avea o eroare:



* Pentru a rezolva problema, facem asa cum ne spune git:

git push --set-upstream origin test1

Si noua branch va fi pusa in remote.

**Atentie la fecth/pull!**

Nici-una din comenzile git status, git log, git log origin/main,etc. nu ne va arata date reale din remote repo pana nu vom da git pull sau git fetch, daca vrem intai sa analizam si apoi sa punem totul la noi.

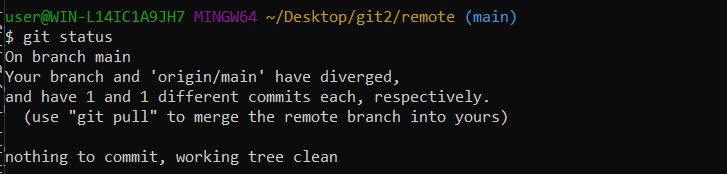
**Merge conflict**

Fie ca ambii useri, user1 si user2, copie un repository. Main commit are un fisier gol index.html

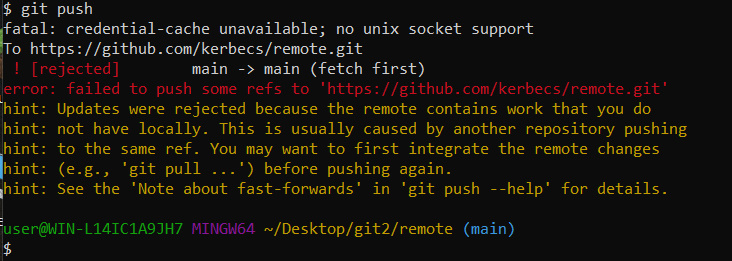
User1 modifica acest fisier si pune in el un cod. da commit si push

User2 nu a dat git pull si a decis sa modifice el fisierul deja dupa ce user1 a facut-o, dar el nu stie de asta. El inca are are commit de la remote branch unde fisiserul e gol.

Acum can user2 va da fetch, va vedea un mesaj de avertizare ca exista conflict:



* Dar, daca user1 de ex adauga un fisier nou,da push, dar user2 nu da pull, si adauga si el un fisier nou, cand va da push, se va intampla asta:



De aia e mereu nevoie sa dam git pull sau fetch inainte de a da git push

**Git nu da merge automat la 2 commituri diferite cand dam push, asa cum face la branches!!!**

**El ne va cere sa dam noi merge, si sa vedem cum stau lucrurile.**

**Daca sunt pur si simplu fisiere noi, ce la noi lipsesc, el va crea un nou format din commit remote + local.**

**Iata de ce, trebuie sa fim siguri ca avem cel mai recent commit din remote repo in local repo!!!!Altfel git nu ne va permite niciodata sa dam push. In schimb, va putea face auto merge la git pull.**