Git verziókezelő

| **?** | **Központosított** | **Elosztott** |
| --- | --- | --- |
| rövidítve | **CVCS** | **DVCS** |
| angolul | Centralized Version Control System | Distributed Version Control System |
| modell | központosított | elosztott |
| architektúra | kliens-server | központi és local repó |
| szerver van? | igen | igen + local repó |
| biztonság | mentést igényel | munkamásolatok is vannak |
| alkalmazások | SVN | GIT, Mercurial |
|  |  | vannak zárt és vannak open source rendszerek |

A verziókezelés annyit jelent, hogy egy speciális program (Git) egy mappán belül saját fájlokat tárol ahhoz, hogy egy projekt fájljainak különböző verzióit hozhassuk létre, és ezen verziók között váltogatni tudjunk.

Pl. létrehozok egy fájlt, hozzáadom az első verzió nevű projekthez, majd készítek egy új projekt verziót második néven, amiben a fájlt megváltoztatom vagy kitörlöm. A két verzió közötti váltás során azokat a fájlokat fogom látni a mappában, amelyek az adott verzióban léteznek. Vagyis a második verzióban kitörölt fájl nem veszik el, csak az második verzióból, viszont az első verzióban látszik (amikor arra átváltok) sértetlenül.

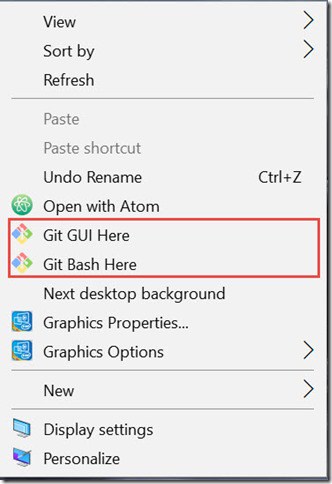
A GitHub ennek a rendszernek online elérhető, és a projekt fájlokat felhőben is tároló és menedzselhető változata, számos kényelmi szolgáltatással.

Parancsok használatához, illetve helyi projekt **(repository)** létrehozásához szükséges a **git** telepítése a gépre (GitHub használatához ez nem szükséges):

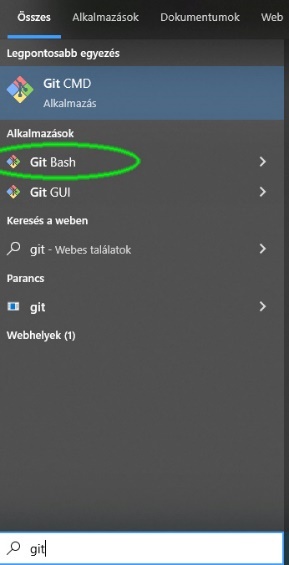
<https://git-scm.com/download>

|  |  |
| --- | --- |
| a telepítésnél a célszerű bepipálni a **windows integráció**t: | illetve, ha valaki használ **windows terminál** programot, akkor ennek a kiegészítőjét is: |

# Helyi projekt (repository) létrehozása

A gép egy mappájában jobb gomb menüben „**Git Bash here**” parancsot választva megnyitja a **bash terminált** az aktuális mappában, ahol a parancsok kiadhatók. *(A grafikus felület is használható, de az alapok megértéséhez a parancskörnyezetet használjuk.)*

Ha ez a menüpont nem elérhető, akkor kihagytuk a windows integrációt telepítésnél. Ekkor a **bash terminált** elindítva be kell navigálnunk a megfelelő mappába (*cd parancs*).

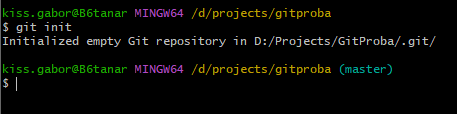


Innentől fogva bármilyen *fájlt, mappát* másolunk a **főmappába** (projects/gitproba), az hozzáadható a verziókezelőhöz.

# Parancsok

git init

Az aktuális mappában létre hoz egy új git projekt környezetet, amit **repository**-nak hívunk. A repository neve a példában **master**, de **main** néven is szokás elnevezni.



A **.git** mappa tartalmazza a verziókezeléshez szükséges fájlokat. A mappa rejtett.

git config

Környezeti változók beállítására alkalmas. Ezek közül az egyik legfontosabb a felhasználó megadása, aki a repository-n dolgozik.

**git config --local user.name ”Kiss Gabor”**

*(Használható e-mail cím is név helyett, ekkor a* ***user.email*** *paramétert kell használni a user.name helyett.)*

**--local** csak az aktuális repository-ban használt felhasználót állítja be

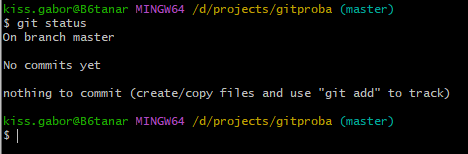
**--global** az összes repository-ban használt felhasználót állítja be



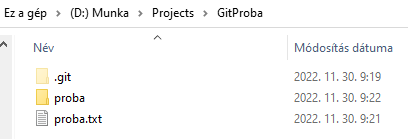
**git config --list** kilistázza a git beállításokat.

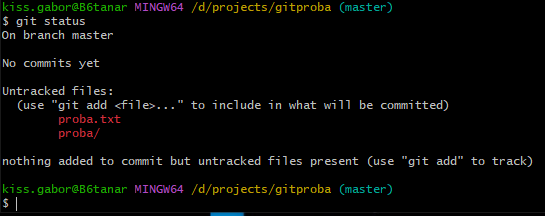
git status

Az aktuális **verzió ágról** (továbbiakban **Branch**) kapunk információkat (branch neve, hozzáadások listája, várakozó fájlok stb.).



Fájlok és mappák hozzáadása után a parancs jelzi az új, de a projekthez még nem hozzáadott fájlokat, mappákat.





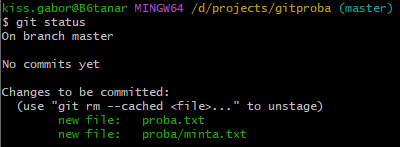
git add

A projektnek van egy úgynevezett **staging area** nevű területe, amelyben lévő fájlok, mappák átmenetileg tartózkodnak, hogy hozzá lehessen adni végleg a projekthez. A **git add** hozzáadja a fájlokat, mappákat a staging area-hoz. *(Használható helyettesítő karakter is pl. git add \*.txt, amely az összes mappában található txt fájltokat hozzáadja. A* ***git add .*** *minden fájlt és mappát hozzáad.)*

Szóközzel felsorolva több fálj, illetve mappa (teljes tartalmával) hozzáadható. A *proba* nevű mappát és a *proba.txt* fájlt adja hozzá.

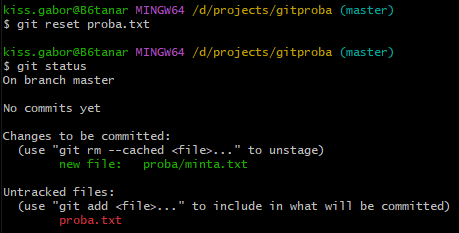


A git status ezután jelzi, hogy a fájlok a projekthez csatolhatók. Ezt hívjuk **commit**-nek.



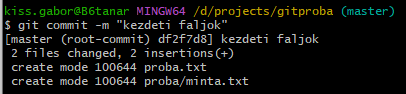
git reset

A **staging area**-ból visszavonja a megadott fájlokat, mappákat.



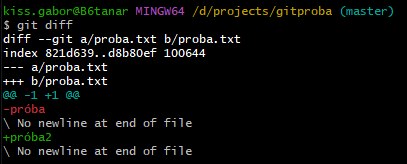
git commit

A **staging area**-ban lévő fájlok és mappák egy akcióval a projekthez adódnak (commit). A commit-nek kell egy **címke** is, amit **-m** kapcsoló után adhatunk meg.



git diff

Ha **változtatunk** egy fájlon (tartalom vagy név, esetleg töröljük), akkor a változásokról kapunk listát a **diff** paranccsal. *(pl. a proba.txt fájlban a „próba” szöveg „próba2”-re változott. - ami kikerült, + ami bekerült)*



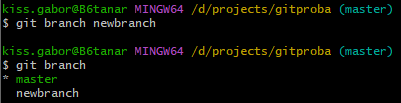
# Verzió ágak (Branch)

git branch

A létező összeg branch-et **kilistázza**. *(Jelenleg egy van, a master)*

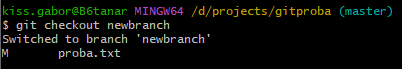


**git branch newbranch** létre hoz egy új branch-et „newbranch” néven.

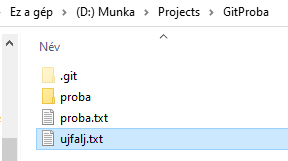
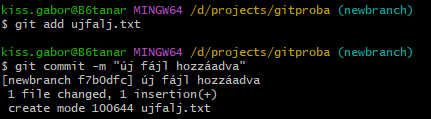


git checkout

**Átlép** a megadott nevű branch-re, és innentől ebben dolgozunk.

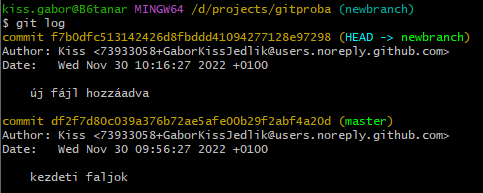


Bármely változtatás a fájlokon ebben a branch-ben történik mentésre (ezeket a fájlokat látja a fájlkezelő is), a master-ben lévő fájlok pedig sértetlenek maradnak. Amint visszaváltunk a master branch-re, a fájlkezelőnk azokat a fájl verziókat fogja látni, amelyek a master-ben kerültek mentésre.

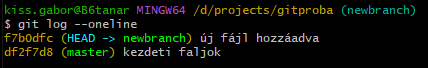
 

git log

**Kilistázza az összes commit-ot** minden branch-re, időrendben visszafelé.



Tömörebb (egysoros commit-ok) kimenet



git merge

**Összevonja** a megadott branch-et a master branch-el. Azaz egy branch-ben eszközölt változásokat beolvasztjuk (élesítjük) a fő projektbe.

**Át kell lépni a master branch-re!** Ezután olvasztható be egy megadott branch.

