Projektek verziókezelése

Mely fájlokat érdemes verziókezelés alá vonni?

A Git verziókezelő rendszer a szöveges állományok, így tipikusan a forráskód fájlok változáskezelésében hatékony, így elsődlegesen a projekt forráskódját érdemes benne elhelyezni. Egy általános szoftver projekt esetén **nem** érdemes verziókezelés alá vonni:

- a fordítás során előálló köztes tárgykódot vagy a végső bináris állományokat (pl. DLL vagy EXE fájlok). Ezek a kód fordításával újból előállíthatóak, folyamatosan változó tartalmukat (bináris állományok révén) a Git nem tudja hatékonyan kezelni. A több *branchen* történő fejlesztés során könnyen ütközést (*merge conflict*) okoz, ami egyébként elkerülhető lehetett volna.
- a fejlesztő eszközök személyes beállításait (pl. Visual Studio esetén a `.vs/` vagy Netbeans esetén a `nbproject/private/` könyvtárak), amelyek eltérhetnek társaink beállításaitól, így folyamatos ütközést vagy egymás beállításainak felülírását fogják jelenteni, emellett személyes információkat is tartalmazhatnak.
- nagy méretű bináris állományokat (pl. videók, nagy méretű képek), amelyek kezelésében a Git nem hatékony. Bár a Git tárolók mérete jól skálázható, egy könnyen kezelhető *repository* mérete az 1-2 GB-os méretet nem haladja meg. Több népszerű projektvezető szolgáltatás (pl. GitHub, BitBucket) rendelkezik is a tárolók maximális méretére vonatkozó hasonló előírással.

Hogyan vonhatok ki adott fájlokat a verziókezelés alól?

Verziókezelés alá azok a fájlok kerülnek, amelyeket kifejezetten hozzáadunk (*git add*), azonban az esetleges véletlen hozzáadást elkerülendő megjelölhetjük azokat a fájlokat és könyvtárakat, amelyeket mellőzni szeretnénk. A mellőzendő állományokat egy speciális `.gitignore` elnevezésű állományban adhatjuk meg, és ezt a fájlt érdemes verziókezelés alá is vonni, hogy a fejlesztők között egységes legyen a beállítás.

A `.gitignore` minden sorában egy illeszkedési mintát adhatunk meg, hogy mely fájokat akarjuk kizárni a verziókezelés alól. A beállítás tranzitívan vonatkozik az alkönyvtárakra is, így gyakran elegendő lehet egyetlen `.gitignore` fájl létrehozása a projekt gyökér-könyvtárában. Néhány példa erre:

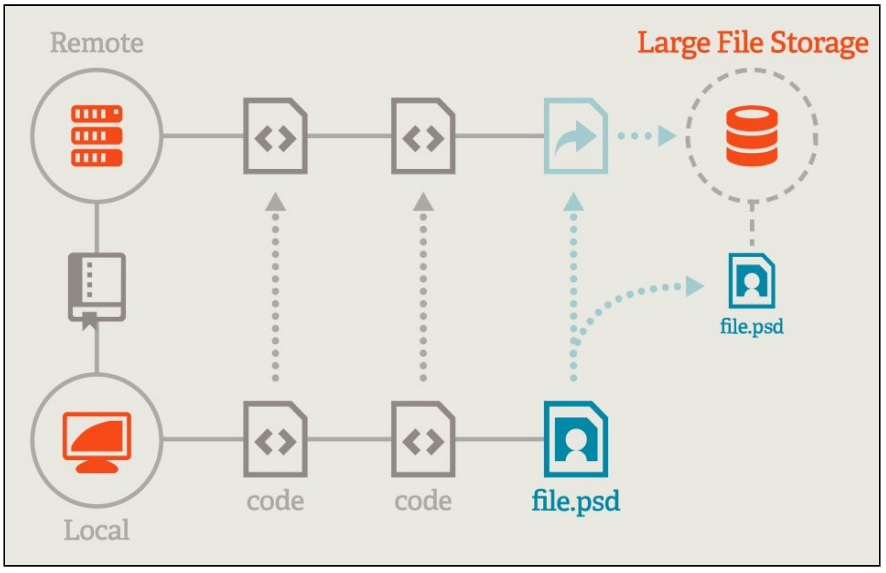
Minta	**Illeszkedés**	**Leírás**
program.exe	/program.exe /bin/program.exe	Minden `program.exe` fájlra illeszkedés.
/program.exe	/program.exe *de nem:* /bin/program.exe	Az adott könyvtárszinten lévő `program.exe` fájlra illeszkedés.
*.exe	/program.exe /bin/main.exe	Minden `exe` kiterjesztésű fájlra illeszkedés.

bin/	/bin/ /project/bin/ *de nem:* /logs/bin (fájl)	Minden `bin` könyvtárra illeszkedés (de `bin` nevű fájlokra nem!)
/bin/*.exe	/bin/program.exe /bin/main.exe *de nem:* /bin/program.dll /project/bin/program.exe	Az adott könyvtárban lévő `bin` könyvtárban lévő összes `exe` kiterjesztésű fájlra illeszkedés.
*.log !important.log	/application.log /logs/application.log *de nem:* /logs/important.log	Az összes `log` kiterjesztésű fájlra illeszkedés, kivéve, ha a fájl neve `important.log`.
logs/**/*.log	/logs/application.log /logs/runtime/main/01.log /project/logs/deploy.log *de nem:* /runtime.log	Minden olyan `log` kiterjesztésű fájlra illeszkedés, amely egy `logs` könyvtár alatt helyezkedik el (tetszőleges mélységben).

Számos programozási nyelvhez és IDE-hez érhető el általános esetekre megfelelő `.gitignore` állomány a GitHub-on, ezekből vagy ezek kombinációjából jó ötlet kiindulni.
URL: https://github.com/github/gitignore

Hova helyezzem a nagy erőforrás állományokat?

A nagy méretű videó, kép és hang erőforrás állományokat még akkor sem feltétlenül érdemes a Git tárolóban elhelyezni, ha amúgy ritkán változnak, mert jelentősen megnöveli a tároló helyi másolatának lekéréséhez szükséges hálózati forgalmat (*git clone*), valamint kezelésükben a Git kevésbé hatékony. Egy fejlesztőcsapatban a programozóknak nem feltétlenül van szükségük a fejlesztéshez a designerek által készített *assetekre*.



Forrás: git-lfs.github.com

A nagy méretű bináris állományok kezelése a Git Large File Storage (Git LFS) segítségével oldható meg, amely a nagy méretű bináris állományokat egy hivatkozással helyettesíti és magukat a fájlokat egy másik (akár távoli) szerveren tárolja, így a Git tárolónk mérete kezelhető marad. A szofttech.inf.elte.hu támogatja a Git LFS-t, így használatához csak a saját gépetekre szükséges a Git LFS telepítése.
Letöltés: https://git-lfs.github.com/
Használat: https://docs.gitlab.com/ce/workflow/lfs/manage_large_binaries_with_git_lfs.html

Mit tehetek a korábban már beküldött nagy méretű állományokkal?

A Git tárolóba már korábban tévesen beküldött nagy méretű vagy személyes adatokat (pl. jelszavakat) tartalmazó fájlokat ugyan törölhetjük és beküldhetjük a módosításokat (*git commit* majd *git push*), azonban a tároló verziótörténetében (*history*) ott maradnak az eltávolítani kívánt adatok. A már beküldött adatok azonban visszamenőlegesen utólagosan is eltávolíthatóak, ugyanis a Git utasításai révén támogatja (pl. *git filter-branch*), hogy egy tároló már megírt verziótörténetét is módosítsuk, újraírjuk az esetleges súlyos hibák utólagos kijavítása végett.

Mivel az utasításkészlet használata nem triviális, a nagy méretű vagy szenzitív információkat tartalmazó fájlok eltávolítására céleszközök is léteznek, pl. a *BFG Repo-Cleaner*.
URL: https://rtyley.github.io/bfg-repo-cleaner/

A verziótörténet módosításakor mindig óvatossággal kell eljárni, érdemes előtte másolatot készíteni a tárolóról annak klónozásával, ha esetleg mégsem megfelelően sikerülne a művelet. A verziótörténet ilyen átírásának velejárója lesz, hogy a távoli tárolóval a szinkronizálást *forced* módban kell végeznünk nekünk és fejlesztőtársainknak is a művelet megerősítéséhez (*git push --force* valamint *git pull --force*).