Git

Mire való?

Változások követése, kezelése (ki, mikor, mit)

Gyakorlatban:

Együttműködés

Különböző fejlesztések (és fejlesztők) elkülönítése

Egyes szoftver verziók azonosítása

Hibakezelő rendszerrel való integráció

Verziókezelés csoportjai:

CVCS (Centralized Version Control System)

* Subversion(SVN)
* CVS

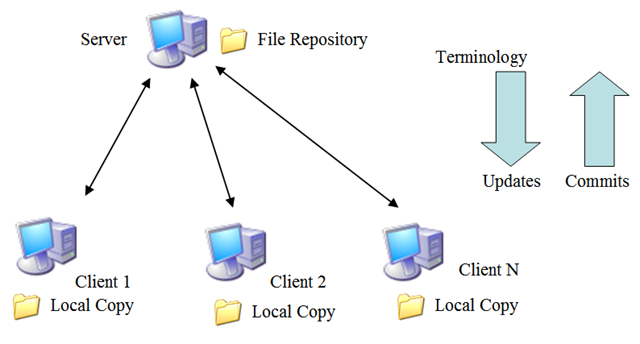
DVCS (Distributed Version Control System)

(Minden ami a mappában van, azt feltölti, letölti)

* Git
* Mercurial

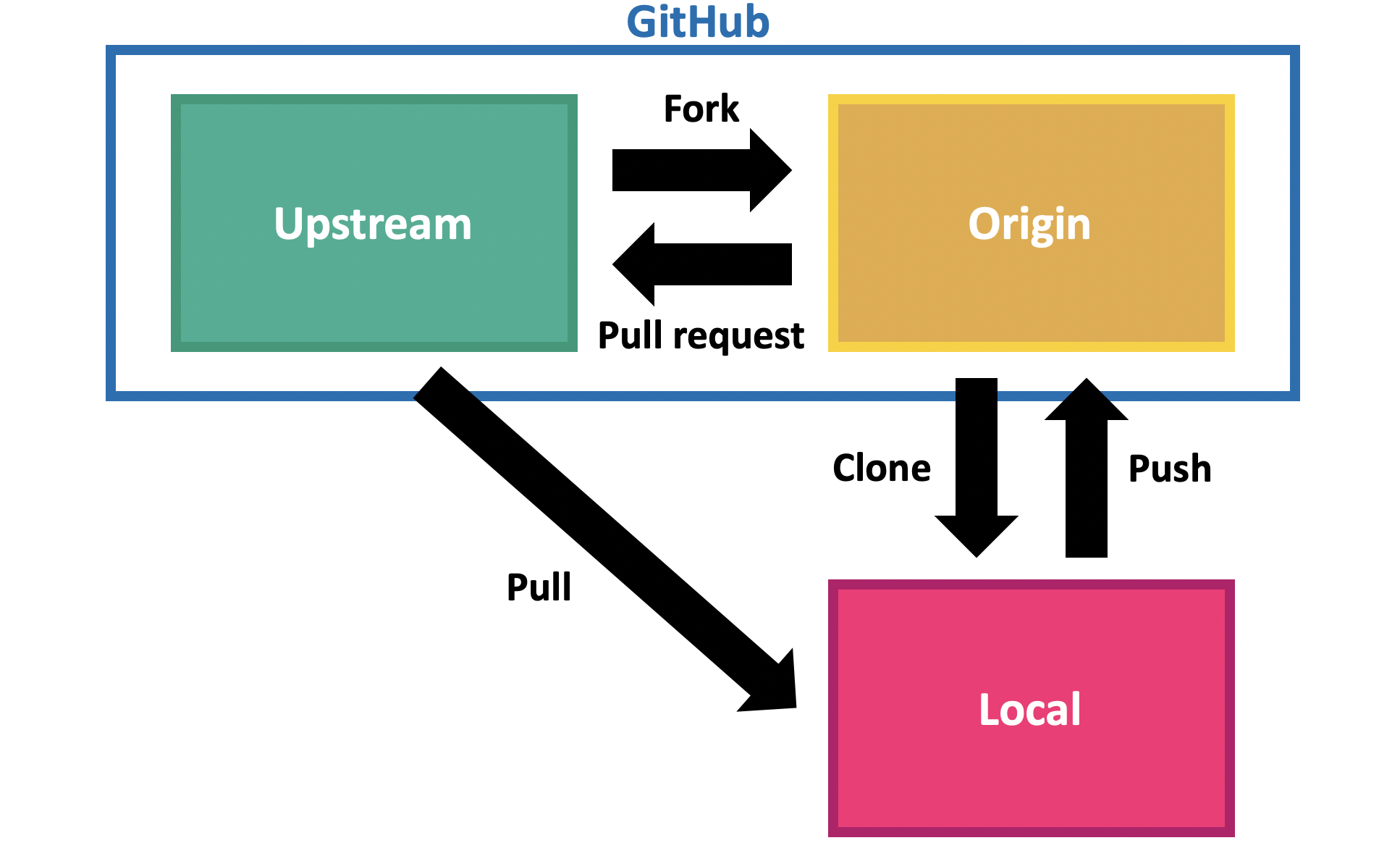
SVN:

Reposiroty -> “Raktár”



DVCS (Git)

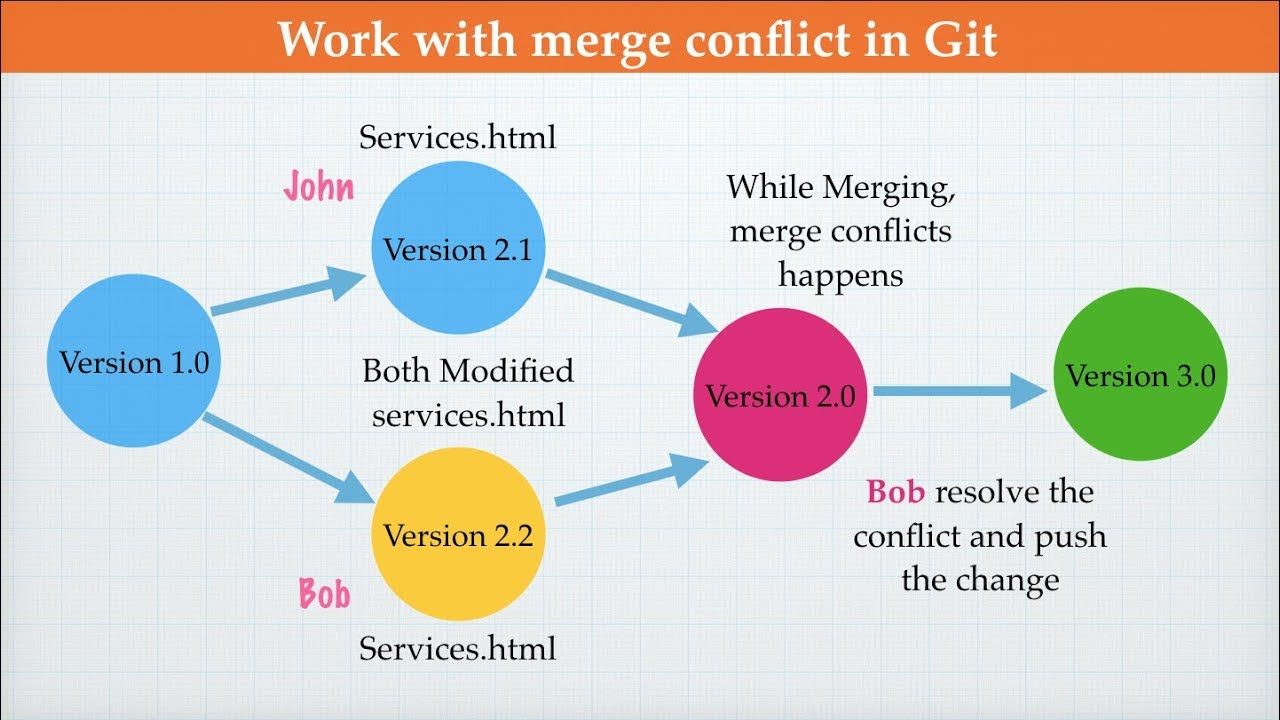
Fork/Pull model



* Alapfogalmak:
  + Repoaitory (repo): verziókövetés alatt álló állományok halmaza
  + Fork: egy repo ekvivalens másolata, más hozzáférési jogosultságokkal
  + Branch: egy repo-n belüli fejlesztési ág
  + Commit: változások atomi egysége
  + Conflict: két fejlesztő azonos sorban történő módosításának az ütközése a Repository-ban

Conflict:

Több ember egyszerre push-olja ugyan azt a fájl-t



Branch:



Parancsok:

git init: Új local repo létrehozása

git status: Infókat ad az adott repo-ról

git add: Fájlokat ad a repo-hoz

git commit: Alkalmazza a módosításokat, de nem push-ol

git log: A repo-ba fel commitelt fájlok listáját adja vissza (Új -> régi)

git push: A local repo anyagának feltöltése a szerverre

git clone: Egy létező repo-t lemásolja a gépünkre

git diff: Az előző commit-olt fájlhoz képest mi változott

git diff –staged: Local repo összehasonlítása a szerveren lévőkkel

git reset: A fájl visszavonása a repo-ról

git checkout - - {fájlnév}: A fájl visszaállítása az eredeti állapotára

git branch: A fejlesztési ágak listáját adja vissza

git branch {név}: Új ág létrehozása

git chechout {branch név}: Váltás az ágak között

git checkout –b {branch név}: Új ág létrehozása

git rm {mit szeretnénk törölni}: Törli a megadott fájlodat a branch-ből

git push –u origin {branch név}: Branch feltöltése a szerverre