Verziókövető rendszerek

Témakörök

- Bevezetés
- Verziókezelő fajták
- A Git CLI használata a saját gépen - Local Repository
- Remote Repository
- GitHub: Webes projekt-kezelés
- Összefoglalás

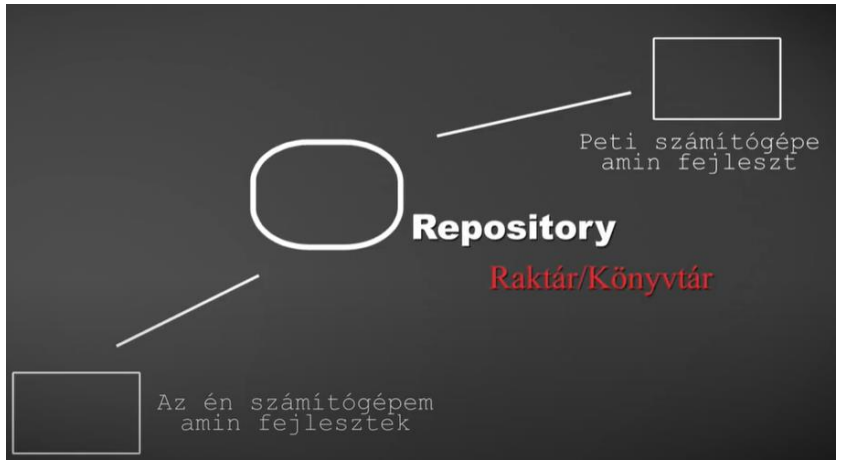
Bevezetés

- „Egyre több helyen használják a Git-et, mint technológát, tehát piacképes ez a tudás.
- Mire jó a Git? Hogyan tudunk egymással dolgozni?
- A szoftverfejlesztés (általánosságban is) világában fontos, hogy együtt tudjunk dolgozni a *csapaton belül* vagy a *más csapatokkal.*

Bevezetés

Dolgozhat *kis cégnél* vagy akár *nagy cégnél is*, alapvető különbség:

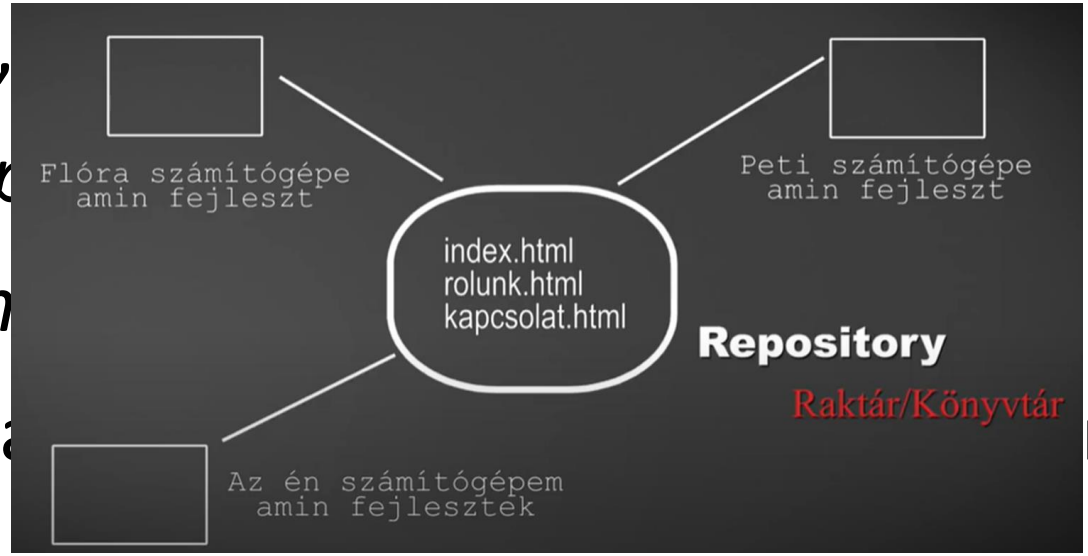
- Kis cégnél *elég pár embernek hozzáférést biztosítani* a kódhoz – fontos, hogy védje legyen az illetéktelen felhasználóktól.



Bevezetés

Példa: készítünk egy weboldalt, amely három *.html* fájlból áll.

- *index.html* – ezt megjelenítjük a webszerveren – ez tartalmaz egy linket,
- amely átirányít a *kap*
- van benne egy *rolun*



Tehát, minden oldalról a ra.

Bevezetés

Három fejlesztő dolgozik a weboldalon, és mindenkinek más-más feladata van.

Pl. egyik *a Kezdő képernyőt* írja meg, a másik a *Rólunk* részt, a harmadik *Kapcsolat* menüpontot.

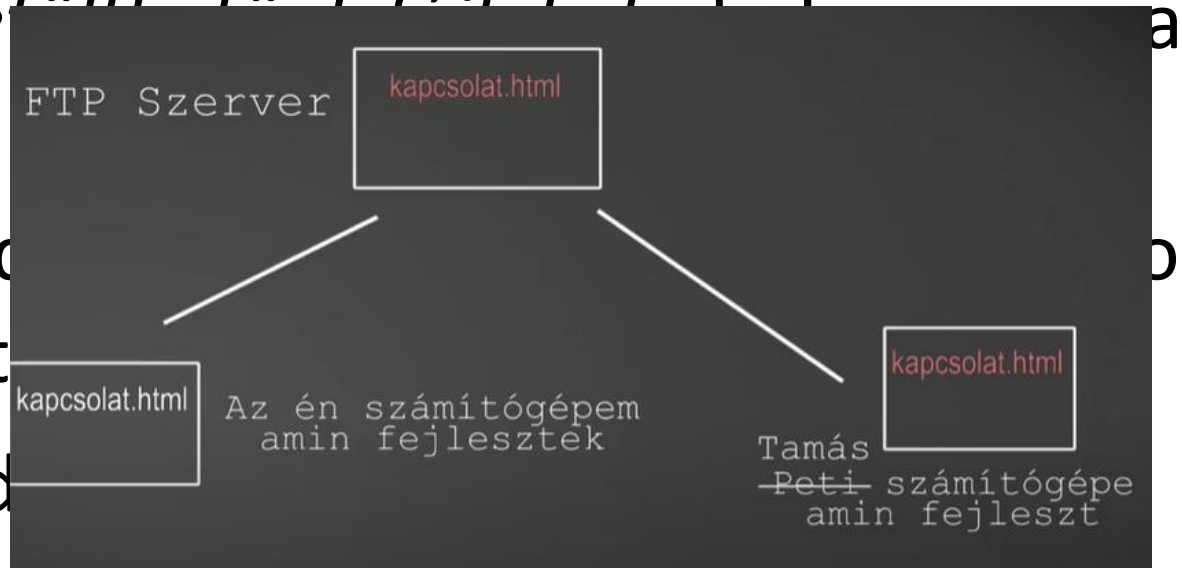
Bevezetés

Kérdés, hogyan lehet ezt egyszerűen megoldani: pl.: FTP szerverrel.

Felmásolhatunk és *le*[illegible]amokat nem tudunk futtatni,

A *kapcsolat.html* mód[illegible]osítás előtti rész az elveszet[illegible]

Tehát, ezzel a megold[illegible]

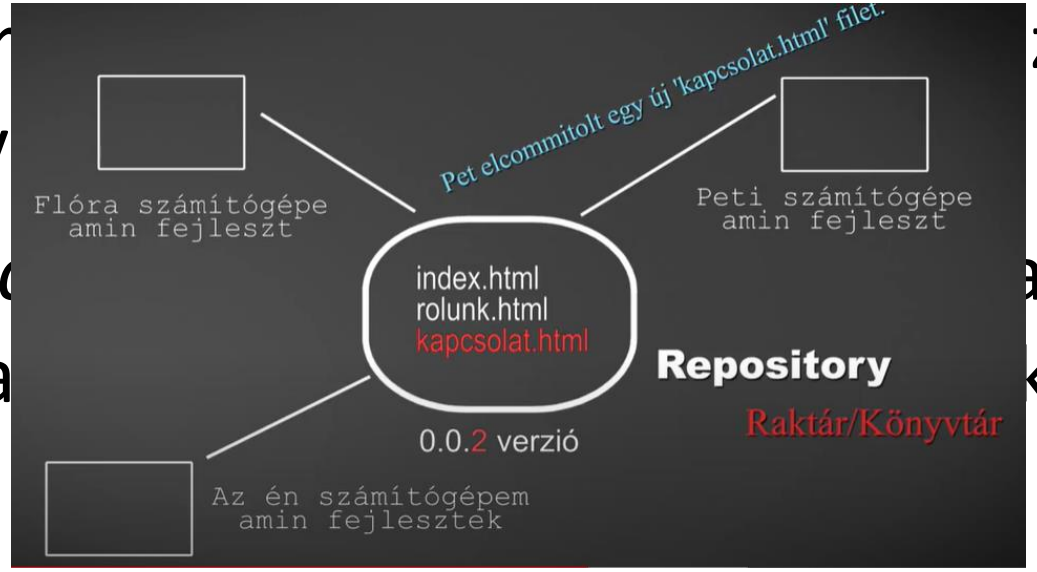


Bevezetés

***Megoldás**: Egy *verziókövető* rendszer ebben tud segíteni.*

A *verziókövető rendszer* zi a *repositoryba*, így az egy

Pl. ha *XY módosít* a *kapc* akkor a verziókövető rendszer a k: 0.0.**2** verzió.



Bevezetés – Mi a Git?

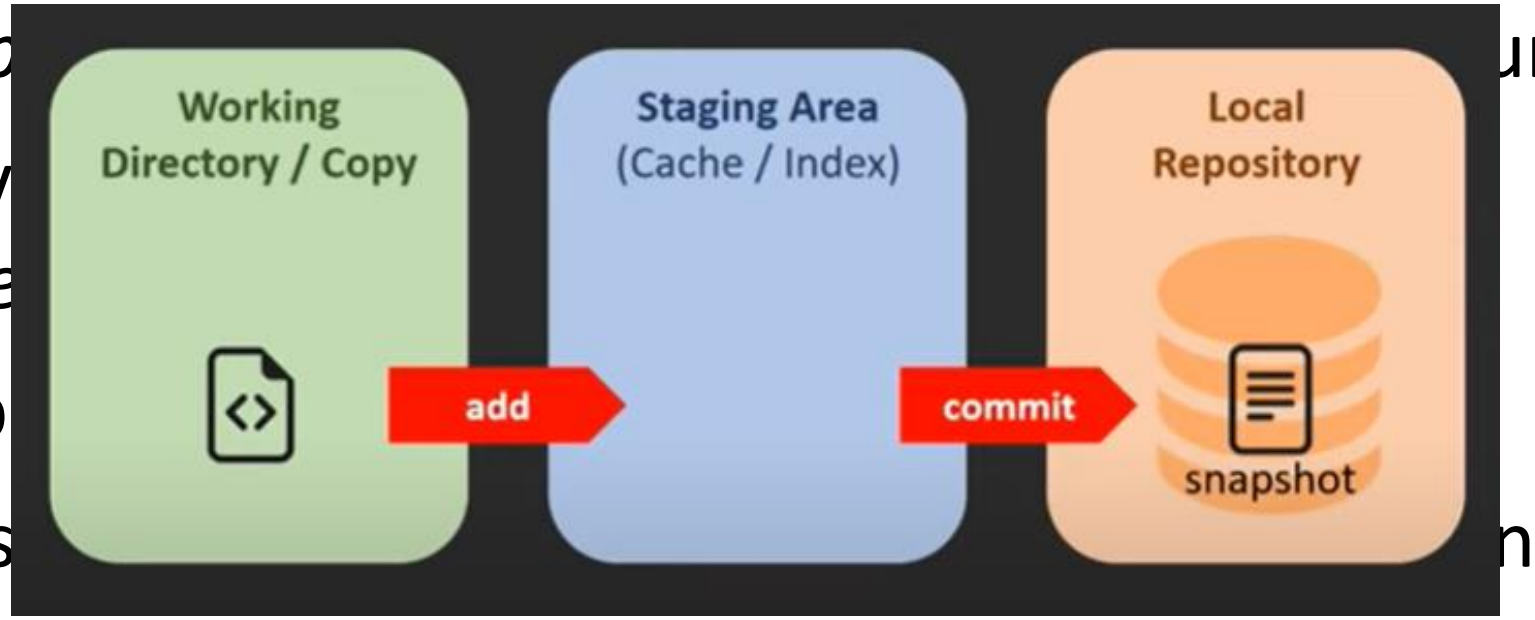
- A Git egy *nyílt forráskódú*, *verzió* és *szoftver* (forráskód) kezelő rendszer.
- A *forráskódról mentést készíthetünk*, melyet később *visszaállíthatunk*.
- *Korlátlan számú különböző verziót* képes kezelni, akár *elágazó (kísérleti)* verzió-ágakat is.
- A kódot egy *központi szerveren tároljuk* (GitHub), ahonnan a *saját gépünkre letölthetjük* az újabb verziót.

Bevezetés – Mi a Git?

- Ha saját gépen végeztünk a kód módosításával, *visszatölthetjük* a *módosított fájlokat* a *szerverre*.
- *Egyszerre többen*, akár *több ágon* is dolgozhatunk egy kódon, majd a végén *egymásba olvasztjuk*.

Git alapfogalmak

- *Repository:* az adatbázis, amely a fájlokat és azok összes verzióját tárolja.
- *Working cop* unk.
- *Revision:* a v *egyértelműe*
- *Head:* a rep
- *Branch:* más n hasznos.



Git alapfogalmak

- *Merge:* a funkció elkészült, ezzel a *művelettel fésüljük össze*.
- *Conflict:* ugyanabban a kódba két vagy több fejlesztő dolgozik.
- *Resolve:* ütközéseket fel kell oldani a verziók megfelelő összefésülésével. Lásd…még több info
- **URL:** https://betterexplained.com/articles/a-visual-guide-to-version-control/
- https://stackoverflow.com/questions/315911/git-for-beginners-the-definitive-practical-guide

Verziókövető rendszerek - történeti háttér

„A szoftverek *méretének* és *komplexitásának növekedésével* létrejött *szoftverkrízis következményeként* megnövekedett:

- a programok forráskódjának *mérete,*
- a szoftverprojektek megvalósításához *szükséges idő,*
- és szükséges *programozói erőforrás.*

Verziókövető rendszerek - történeti háttér

A szoftveripar fejlődésével egyre több alkalmazás készült:

- a *fejlesztések életciklusa* gyakran nem ért véget a program első *publikus verziójának kiadásával*,
- *karbantartási* és további *fejlesztési fázisok követték*.

A szoftverprojektek: *méretben, komplexitásban, időben és a résztvevő fejlesztők számában is növekedni kezdtek.*

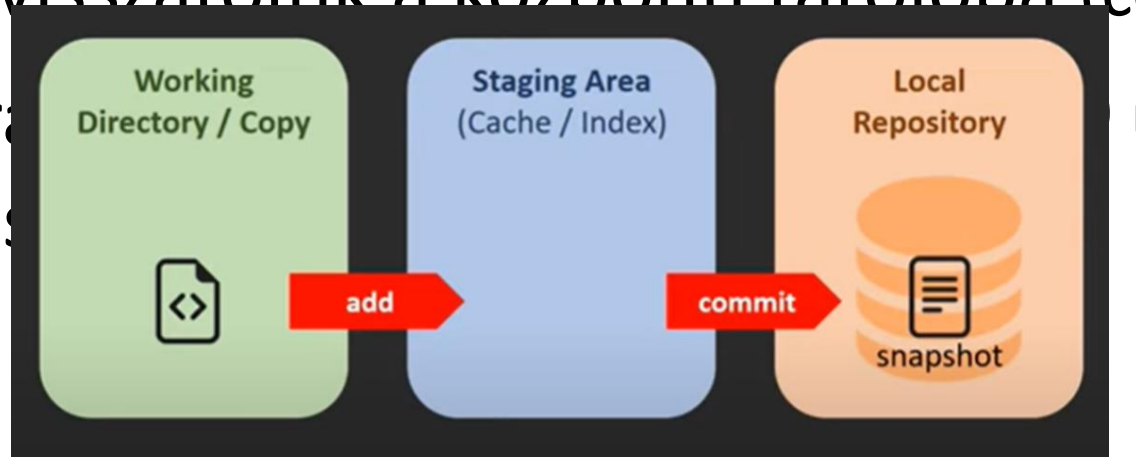
Verziókövető rendszerek - funkcionalitás

A szoftverfejlesztés általában:

- *több lépésben*,
- és sokszor *párhuzamosan zajlik*, szükséges, hogy
- az *egyes programállapotok*, *jól követhetőek legyen*ek, ezt a feladatot a *verziókövető rendszerek* látják el:
- pl. CVS, Apache Subversion (SVN), Mercurial, Git,
- egy közös tárolóban *(repository)* tartják kódokat,

Verziókövető rendszerek - funkcionalitás

- ezt a fejlesztők lemásolják egy *helyi munkakönyvtárba*, és amelyben dolgoznak (working copy)
- a módosításokat visszatöltik a központi tárolóba (commit),
- a munkakönyvtár után folyamatosan friss



Verziókövető rendszerek - modell

Verziókövető rendszerek -1. generáció

1. Lokális verziókövető rendszerek

a) *Forráskód változásainak követése*, a szoftver funkcióinak különböző kombinációjával készült kiadások kezelése:

- lokális tároló (de többen is elérhetik pl. *mainframe* esetén),
- fájl alapú műveletvégzés (1 verzió 1 fájl változásai),

Verziókövető rendszerek -1. generáció

b) Az 1970-es években lefektetésre kerültek az elméleti alapok:

- Source Code Control System (SCCS) – 1972
- Revision Control System (RCS) - 1982

Verziókövető rendszerek -2. generáció

2. Centralizált verziókövető rendszerek

Több fejlesztő által *párhuzamos szoftverfejlesztés* támogatásának *előtérbe kerülésre*:

- centralizált modellt megtartva, de *kliens-szerver architektúra,*
- fájlhalmaz alapú műveletek (1 verzió több fájl változásai),
- konkurenciakezelés jellemzően beküldés előtti egyesítéssel (*merge before commit*)

Verziókövető rendszerek - 2. generáció

Az 1990-es évektől terjedtek el:

- Concurrent Versions System (CVS)
- Subversion (SVN)
- SourceSafe, Perforce, Team Foundation Server, stb.

Hátrány: a szerver kitüntetett szerepe (pl. meghibásodás), továbbá a verziókezeléshez hálózati kapcsolat szükségeltetik.

Verziókövető rendszerek - 3. generáció

3. Elosztott verziókövető rendszerek

A klasszikus *verziókezelő műveletekről* leválasztásra kerül a *hálózati kommunikáció*, azok a felhasználó által *kezdeményezhető önálló tevékenységekként* jelennek meg:

- decentralizált, elosztott hálózati modell,
- minden kliens rendelkezik a *teljes tárolóval* és *verziótörténettel.*

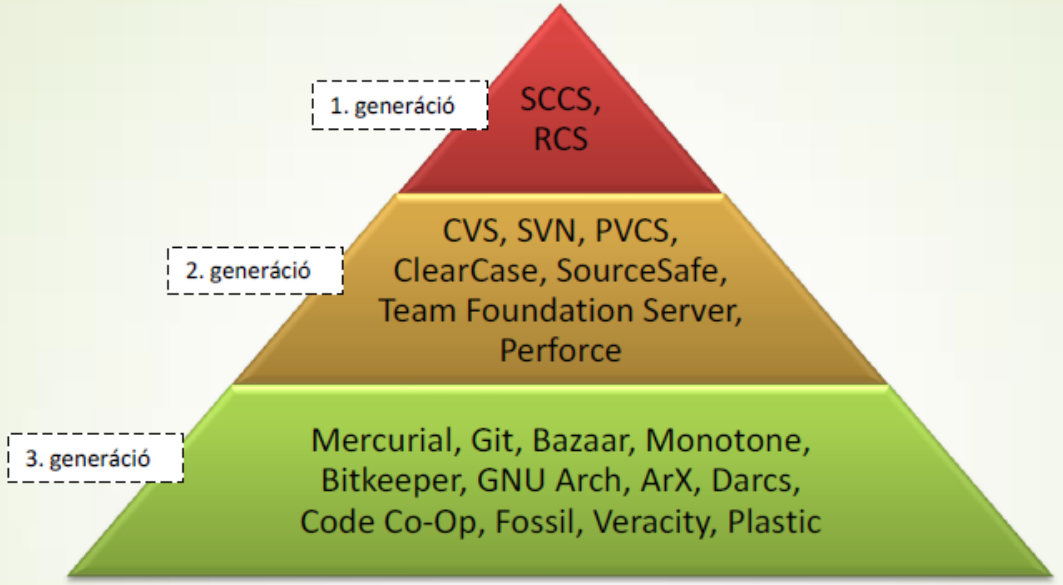
Verziókövető rendszerek - 3. generáció

- a *revíziókezelő eszköz* műveletei lokálisan, a *kliens tárolóján történnek,*
- a kommunikáció *peer-to-peer* elven történik, de kitüntetett (mindenki által ismert) szerverek felállítására van lehetőség,
- konkurenciakezelés jellemzően *beküldés utáni egyesítéssel* (*commit before merge*)

2000-es évek első felében jelent meg: Monotone, Darcs, Git, Mercurial, Bazaar, stb.

Verziókövető rendszerek

Generációs modell



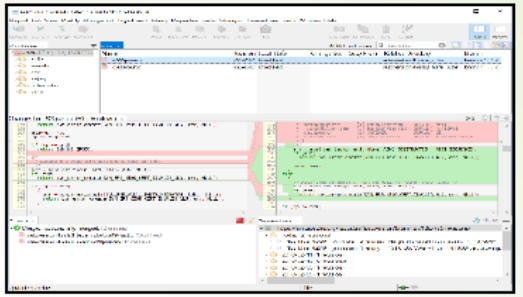
Generáció	Hálózati modell	Műveletvégzés	Konkurenciakezelés
Első	Lokális	Fájlonként (non-atomic commits)	Kizárólóagos zárak (exclusive locks)
Második	Központosított	Fájlhalmaz (atomic commits)	Egyesítés beküldés előtt (merge before commit)
Harmadik	Elosztott	Fájlhalmaz (atomic commits)	Beküldés egyesítés előtt (commit before merge)

Verziókövető rendszerek

SVN GUI Kliensek”

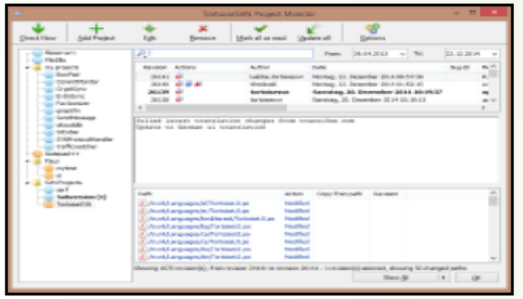
TortoiseSVN

- Windows



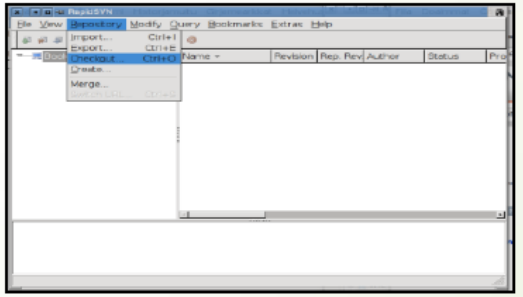
SmartSVN

- Linux, Windows, Mac



RapidSVN

- Linux, Windows, Mac



Irodalom

Cserép Máté: Verziókövető rendszerek

https://swap.web.elte.hu/2018192_pt2/ea03.pdf

Verziókezelő fajták

Két fajtát különböztetünk meg:

- *Centralized Version Control* system – Központosított verziókezelő pl. Apache Subversion (SVN), CVS (Concurrent Versions System).
- *Distributed Version Control* system – Elosztott verziókezelő pl. Git, Mercurial.

Ez a két rendszer áll egymás mellett.

Git Cliens letöltése, telepítése

Letöltés: https://git-scm.com/ or https://gitforwindows.org/

Itt automatikusan felismeri az OS-t és letölti a legfrissebb verziót.

`Git-2.46.1-64-bit`

Telepítés: célszerű rendszergazdaként telepíteni.

Az *útvonal maradjon* alapértelmezett: *C:\Program Files\Git*

Megadhatjuk, hogy az *Desktop*-ra is kerüljön ikon.

`Bash here` legyen bepipálva.

Git Cliens letöltése, telepítése

- *Válasszuk*: Windows parancssori konzolt használjuk a Git során.
- *Next:* alapértelmezett
- *Next:* szintén alapértelmezett
- Next: Use Windows default Consol window
- Install – kipipálni az indítást
- Megjelenik a parancssori konzol – hasonló a Windows p. consolhoz.

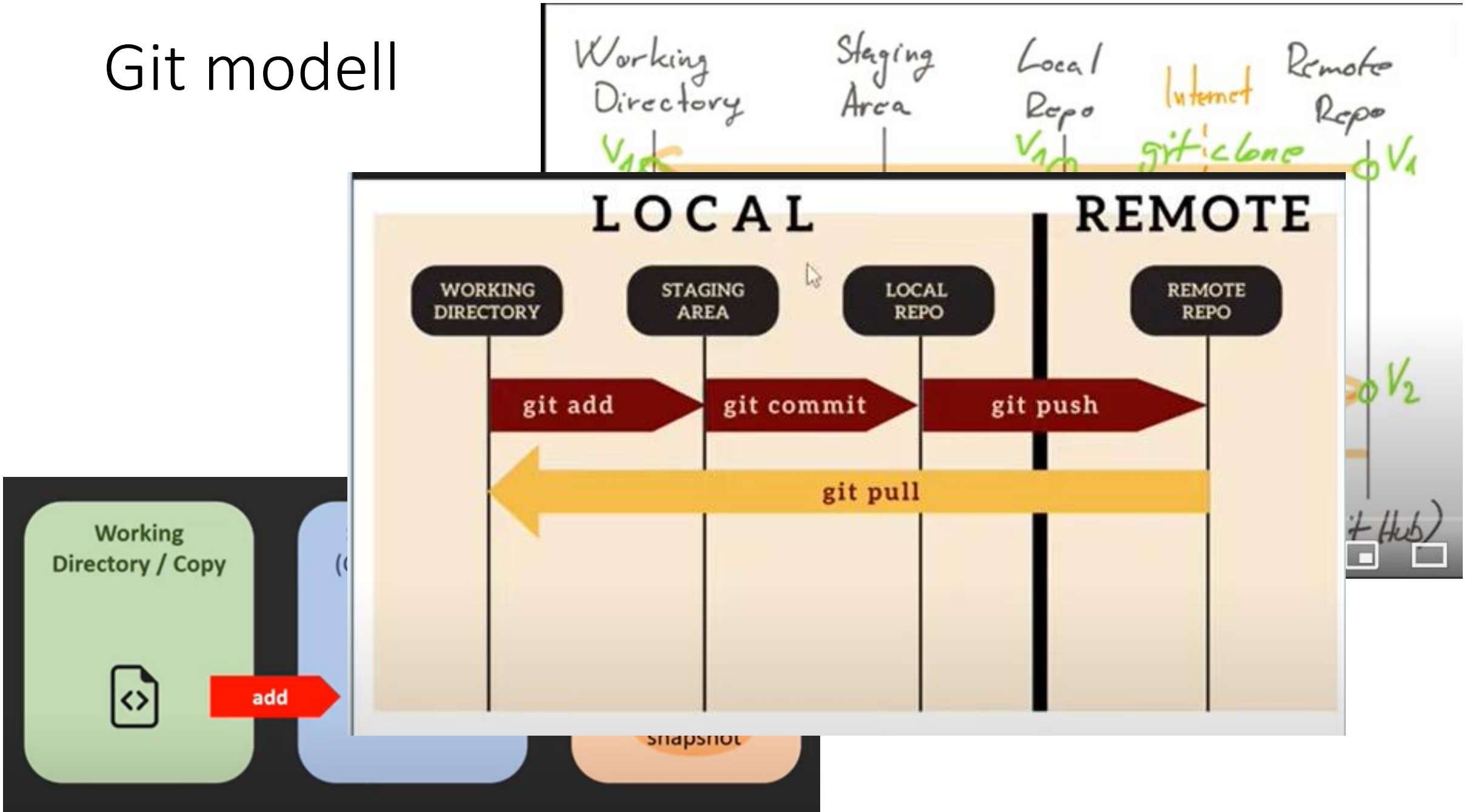
Git letöltése, telepítése

- Git Cliens aktuális versiójának lekérdezése: `git –version`
- Van lehetőség Mac OS X, Linux/Unix gépekre is letölteni/telepíteni a Git Clients-et.

Irodalom:

- https://guides.github.com/activities/hello-world/
- https://gitforwindows.org/

Git modell


Working Directory
Staging Area
Local Repo
Internet
Remote Repo
git clone
LOCAL
REMOTE
WORKING DIRECTORY
STAGING AREA
LOCAL REPO
REMOTE REPO
git add
git commit
git push
git pull
Working Directory / Copy
add
Snapshot

GitHub – felhasználói fiók létrehozása

Új fiók regisztrálása a GitHub-on

Legnépszerű felhő alapú menedzser: **GitHub**

Regisztráljon a rendszerbe, hozzon létre egy új fiókot.

URL: https://github.com/

GitHub – felhasználói fiók létrehozása

Regisztrálás a GitHubon

URL: https://github.com/

email cím: -

pw: ----

Felhasználói név megadása:

Érvényesítési kód: email- en kap.

GitHub: Webes projekt-kezelés

Eddig saját gépen dolgoztunk a projekten, de most következik a felhő alapú projekt-kezelés.

Új fiók regisztrálása a GitHub-on

Legnépszerű felhő alapú menedzser: **GitHub**

Regisztráljunk ebbe a rendszerbe, hozzunk létre egy új fiókot.

URL: http://github.com

Adjuk meg: *username, email, jelszó* – majd hitelesítsük.

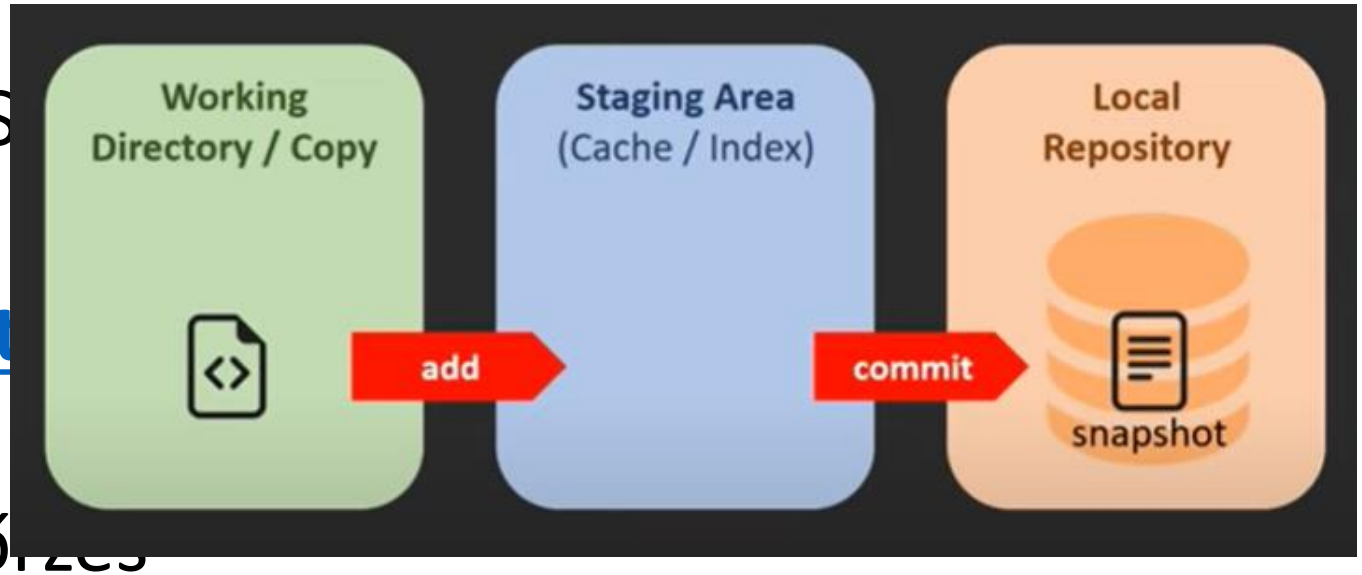
A GitHub szinkronizálása a saját gépen lévő projektünkkel

- Lépjünk be a GitHub programba és készítsünk *New Repository-t*.
- Adjunk nevet a projektnek: *NeptunkodInfOvAlk* – Create repository
- Itt megadják azokat a parancsokat, amelyek szükségesek egy új projekt létrehozásához.

GitHub

- Ha elkészült a repository (a weben) akkor le kell *Clone* -ni az általam megadott könyvtárba. pl. C:\XYGit
- **git clone https://github.com/username2024/NeptunkodInfOvAlk**
- Megjelenik egy ablak, meg kell adni: *username és a password.*
- Munkamenetenként csak egyszer kell megadni.
- A GitHub oldalon kattintsunk a **NeptunkodInfOvAlk**-re.
- Megjelenik a GitHub-on a saját gépen elkészített projekt.
- **.git**

GitHub - Local repos



Klónozás után a *XYGit/*[Nept
meghajtón

1. **git status** - állapot ellenő
2. **git add „Neptunkod_0923”** - feltöltés vagy ha több fájl van venne, akkor
3. **git add .** - feltölti az összes fájlt, ami a *Neptunkod_0923* -be van.
4. **git commit -m "3 fájl mentes"** - fel commit- toltuk a stage-re.

GitHub - Local repository

5. **git log** - látni milyen commit-tok készültek el.
6. **git shortlog** - csak annyi látható.

GitHub - Remote repository

Feltöltés a GitHub-ra

- **git push** - majd frissítés.
- **git pull** – változtatás esetén először ezt kell használni, azaz eltölteni.

GitHub - Rename repository

Rename repository

- *Settings – General - Repository name - Rename*

GitHub – fájl/mappa törlése a web-en

Fájl/mappa törlése a Web-en

Válasszuk ki a könyvtárat, majd a *More options* (3 pont) válasszuk a

Delete directory/file, majd a *Commit changes* - majd a *Commit changes* válasszuk – végleges törlés.

GitHub – Repository törlése –web-en

Repository törlése

- Kiválasztjuk a *repository*-t.
- Majd *Settings/Danger Zone - Delete this repository* - meg kell adni a *repository teljes nevét - törlés.”*