



MISKOLCI EGYETEM  
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

# Szoftvertechnológia gyakorlata

GEIAL316-B2

---

## Verziókezelés

**Dr. Tompa Tamás**

egyetemi adjunktus

Általános Informatikai Intézeti Tanszék

Miskolc, 2024

# Verziókezelés

---

- A verziókezelés olyan eljárások összessége, amelyek lehetővé teszik egy **adathalmaz változatainak (verzióinak) együttes kezelését**
  - **Adathalmaz:**
    - forráskód, dokumentumok stb.
    - szoftverek estében ez a szoftver életciklusa során a forráskódban végzett módosítások tárolását jelenti
    - nem csak a forráskódot lehet verziókövetni!

# Verziókezelés

---

- A fejlesztés során **a források sok iteráción megy keresztül**, így szükséges lehet, hogy esetleges probléma esetén **vissza lehessen térni egy korábbi verzióra**
- Nagy segítséget nyújt **csapatmunkában** és **egyéni** történő szoftverfejlesztés esetében is
- Elnevezések: Revision Control, Version Control System, Source Control

# Verziókezelés

---

## ○ Egyszerű „verziókövetés”

- minden változtatást külön jegyzékekben elmentünk
  - megvalósítható, de nehézkes
  - időidényes, nem hatékony
  - nem átlátható
  - és ha elvesznek?

**antipattern!** ->

FRIQ-framework_v05_210222_szabalyok_felcimkez..	zip
FRIQ-framework_v05_210222_szabalyok_felcimkez..	zip
FRIQ-framework_v05_210223_egyedi_szabalycimke	zip
FRIQ-framework_v05_210225_refactor	zip
FRIQ-framework_v05_210301_refactor_szabalykoze..	zip
FRIQ-framework_v05_210302_distEps_refactor	zip
FRIQ-framework_v05_210302_szabalyosszeolvaszt..	zip
FRIQ-framework_v05_210303_szabalyosszeolvaszt..	zip
FRIQ-framework_v05_210304	zip
FRIQ-framework_v05_210307	zip
FRIQ-framework_v05_210308	zip
FRIQ-framework_v05_210309	zip
FRIQ-framework_v05_210315	zip
FRIQ-framework_v05_210329	zip
FRIQ-framework_v05_210415	zip
FRIQ-framework_v05_210420	zip

## ○ Megoldás: verziókövető rendszerek

- pl. Git, Mercurial, SVN, stb.

# Alapfogalmak

---

- **Repository:** maga a tároló, röviden csak repo
  - lehet használni helyi (local) illetve távoli (remote) repokat, ilyen távoli repok például a Github, Bitbucket stb.
  - a tárolók pillanatképként tartalmazzák a változásokat, illetve a projekt aktuális állapotát
- **Working copy:** a kód egy részének egy példánya, amelyen a fejlesztő éppen dolgozik a saját gépén
- **Push:** adatok/commit-ok feltöltése a remote repoba
- **Pull:** változások letöltése a remote repo-ból
- **Diff/Change:** két file között változás megjelenítése

# Alapfogalmak

**Commit:** a kódon eszközölt változtatásokat commitok formájában érvényesíthetjük a tárolókon belül

- célszerű minden kisebb módosítást (ne 1000 soronként...) követően commitolni
- a commit egy lényegretörő megjegyzés, ami összefoglalja, hogy milyen módosítás történt

## antipattern!:

pici css móka committed 3 months ago	elfelejtettem ezt a portot committed 4 months ago
css committed 3 months ago	amúgy sem működik... committed 4 months ago
pár apróbb módosítás amit kért committed 3 months ago	d(*.*)b committed 6 months ago
little css committed 4 months ago	szeretek kétszer dolgozni committed 6 months ago
egy újabb committed 4 months ago	még egy fix committed 6 months ago
elkeseredett próbálkozás committed 4 months ago	fix the fix committed 6 months ago
ismét committed 4 months ago	

# Alapfogalmak

---

## ○ Commit típusok (scope)

- **build** - build folyamatával kapcsolatos változások
- **ci - CI** - (Continuous Integration) folyamattal kapcsolatos változások
- **chore** - build folyamattal kapcsolatos változások
- **docs** - dokumentációs változtatások
- **feat** - új funkció
- **fix** - hibajavítás
- **perf** - kódbeli változtatás, amely javítja a teljesítményt
- **refactor** - kódbeli változtatás, amely sem hibát nem javít, sem új funkciót nem ad hozzá, csak refaktorálás
- **revert** - visszavonások
- **style** - szóköz, formázás, hiányzó pontosvesszők stb.
- **test** - tesztek hozzáadása

# Alapfogalmak

---

## ○ Commit szabályok

- „scope: message” formátum
- angol, jelen időben
- 50 karakternél ne legyen hosszabb
- rövid, lényegretörő
- mit miért, és nem mit hogyan
  
- Példák:
  - `build: Update webpack configuration`
  - `ci: Add GitHub Actions workflow for automated testing`
  - `chore: Update dependencies using npm audit fix`
  - `feat: Add user authentication feature`
  - `fix: Fix issue with form validation not working on Safari`
  - `revert: Revert "feat: Add user authentication feature,"`
  - `test: Add unit tests for login component`



# Alapfogalmak

---

- **Revision:** verzió
- **Checkout:** branch váltás, commit visszaállítás
- **Head:** a legfrissebb kommitot (verziót) jelöli, az aktuális ág teteje
- **Snapshot:** adott időpillanatban fájlok aktuális állapota
- **Clone:** repo tartalmának letöltése (háttértárra lokálisan)
  - megadott jegyzékbe

# Alapfogalmak

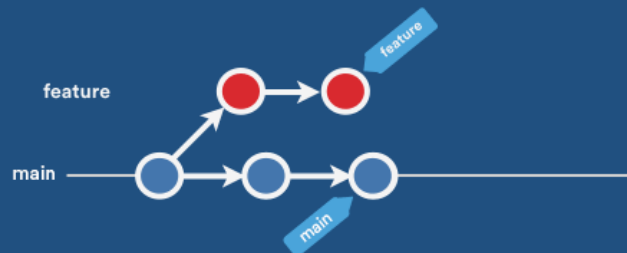
---

- **Merge:** összefésülés, fejlesztési ágak (branch) egyesítése
  - új commit (mindkét ág változásai)
  - megmarad a branch-ek története
  - conflict-okat jelzi

(animáció)

## What is a merge?

A process that unifies the work done in two branches



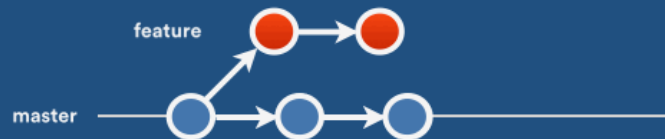
# Alapfogalmak

---

- **Rebase:** az aktuális ágban lévő commitok érvényesítése egy másik ágra
  - commit-okat áthelyezi a másik ágra
  - conflict esetén felfüggeszti a folyamatot (manuális feloldás)  
(animáció)

## What is a rebase?

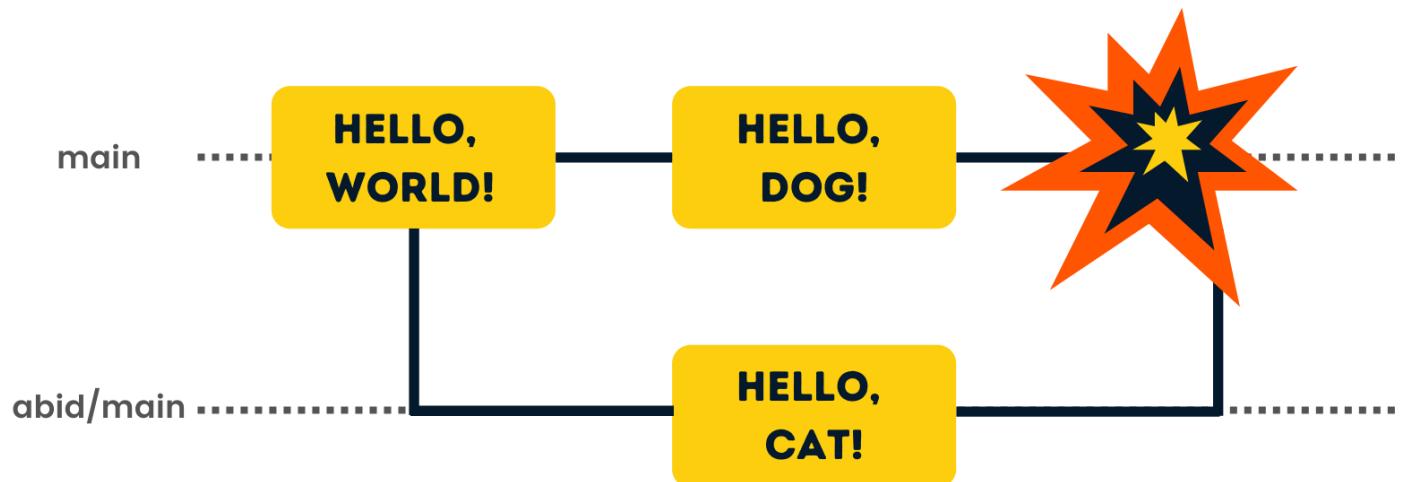
It's a way to replay commits, one by one, on top of a branch



# Alapfogalmak

---

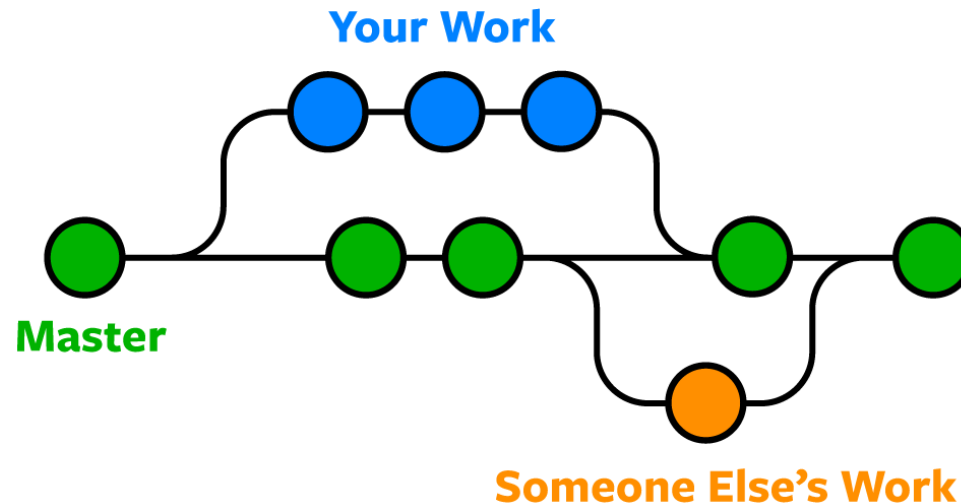
- **Conflict:** fejlesztési ágak (branch) összefésülése során keletkező jelenség, az adat ugyanazon részen történt módosítás több szereplő által, nem lehet automatikusan összefésülni
  - sokszor előfordul -> fel kell oldani
  - feloldás: konfliktus jelzők által manuálisan vagy IDE-ben



# Alapfogalmak

---

- **Branch:** fejlesztési ág, a fejlesztés ne csak egy „szálon” történjen
  - elterjedtebb branch-ek: master (main), develop, feature, release, hotfix stb.



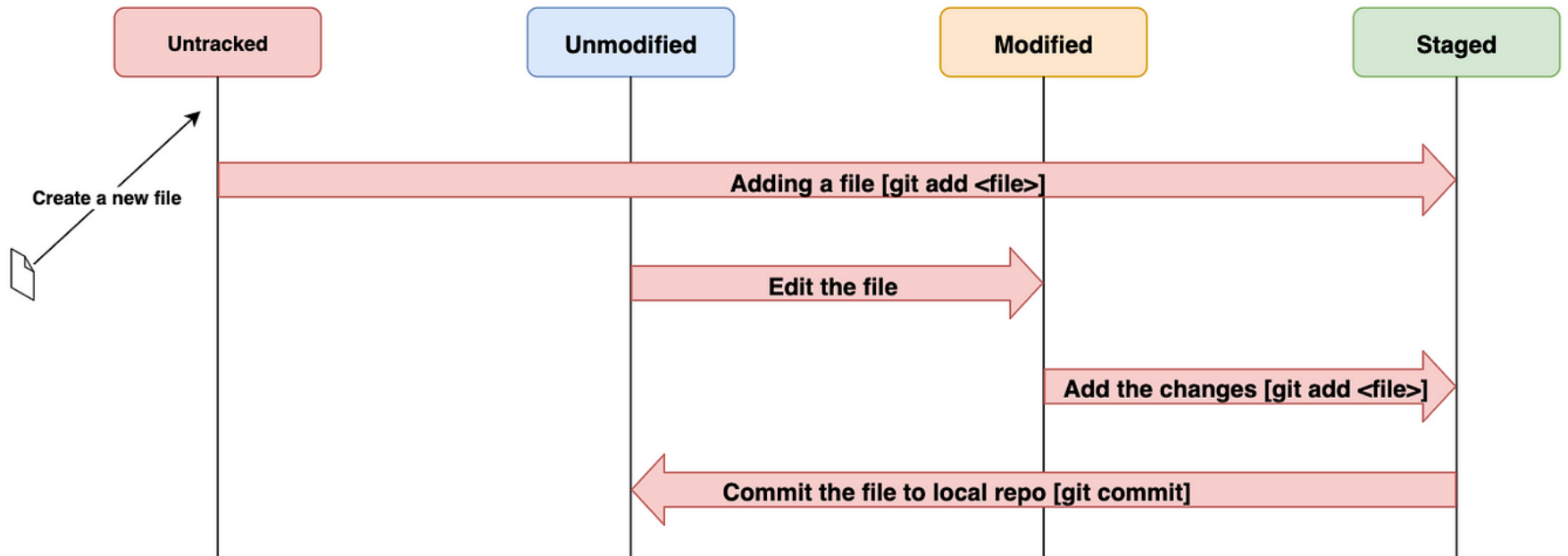
- ha lehet egy branch-en minél kevesebb időt kell tölteni
  - addig sokminden történik a master-en -> sok conflict
  - -> kis (1 napos) feladatok/ticket-ek

# Fájlok állapotai

---

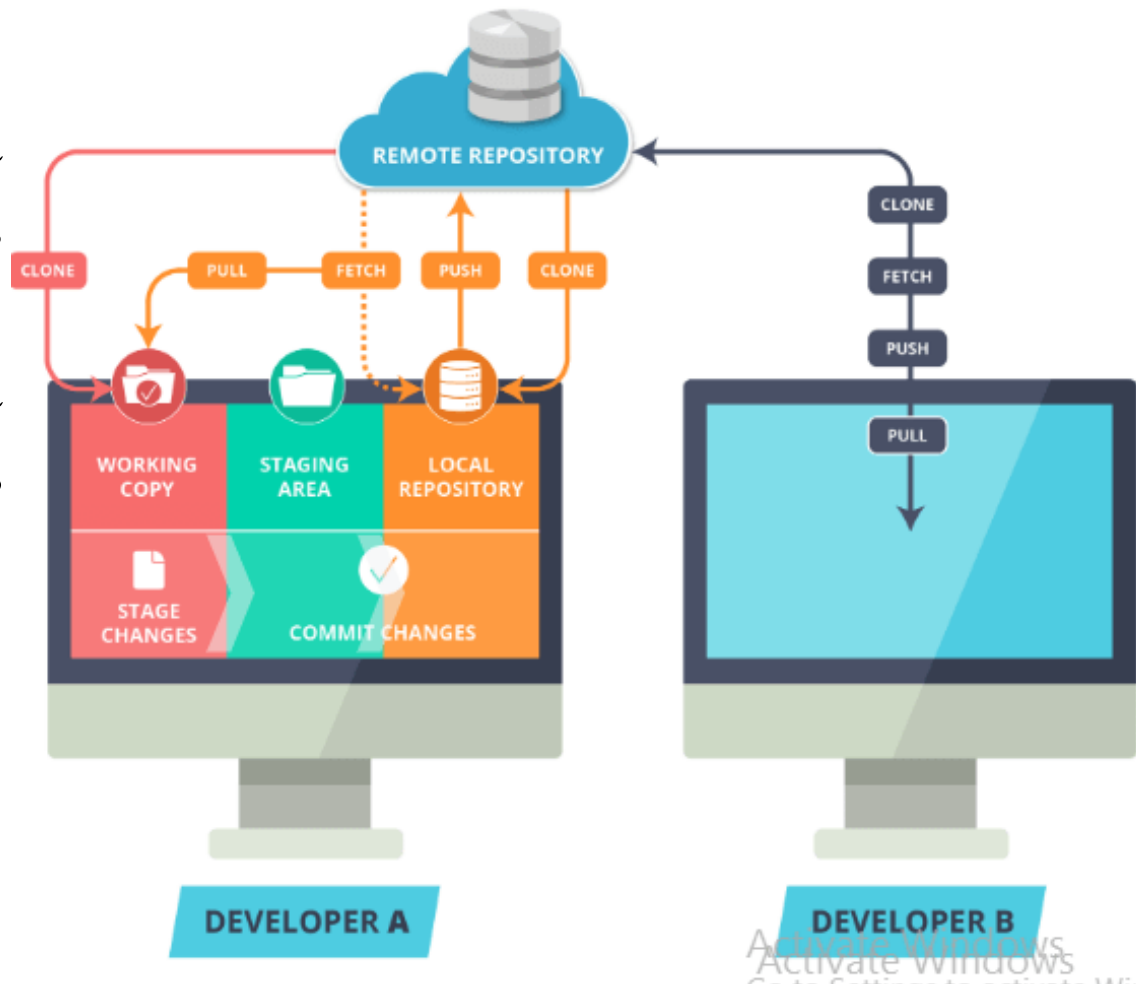
- **Untracked:** azok a fájlok melyek változásai nincsennek nyomonkövetve
- **Tracked:** azok a fájlok melyek változásai nyomon vannak követve
  - modified: a fájl módosult a legutóbbi commit óta
  - unmodified: a fájl nem módosult a legutolsó commit óta
  - staged: a fájl módosult a legutóbbi commit óta, a következő commit során el lesz tárolva a változása

# Fájlok állapotai



# Repository-k típusai

- **local:** változások a helyi fájlrendszeren, helyi repo-ban
- **remote:** változások a távoli szerveren, távoli repo-ban
- **public:** bárki hozzáférhet
- **private:** csak engedélyezett felhasználók férhetnek hozzá





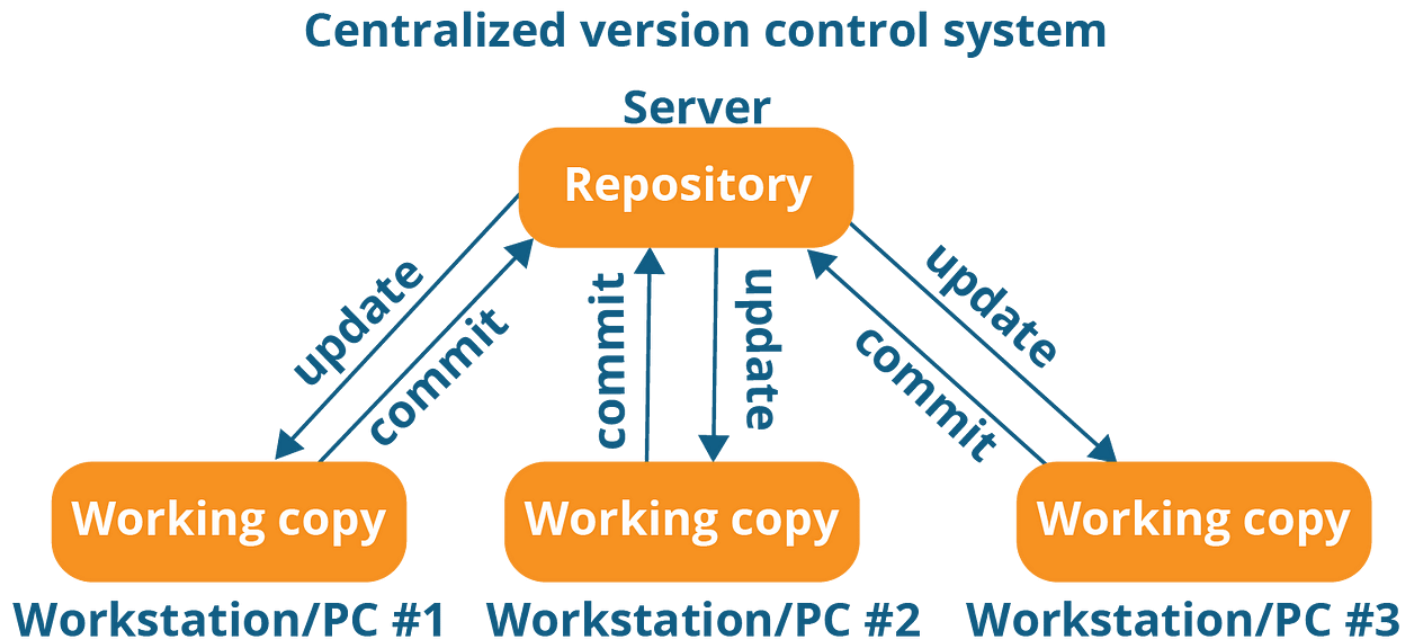
# Verziókezelő rendszer típusok

---

- **Centralizált (CVCS, központosított)**
  - közös központi repo
    - minden fejlesztő közvetlenül ehhez kapcsolódik
  - egységes verziókezelés
    - a fejlesztők mindig a központi repo változatával dolgoznak
  - minden művelet szerveren
    - mindig van kapcsolat a szerverrel
    - minden commit után frissítés kell, hogy a változások mindenkinél megjelenjenek
    - minden változtatást a központi repo-ba kell feltölteni, így az erősen kötődik a központi szerver elérhetőségéhez

# Verziókezelő rendszer típusok

- Centralizált (CVCS, központosított)



# Verziókezelő rendszer típusok

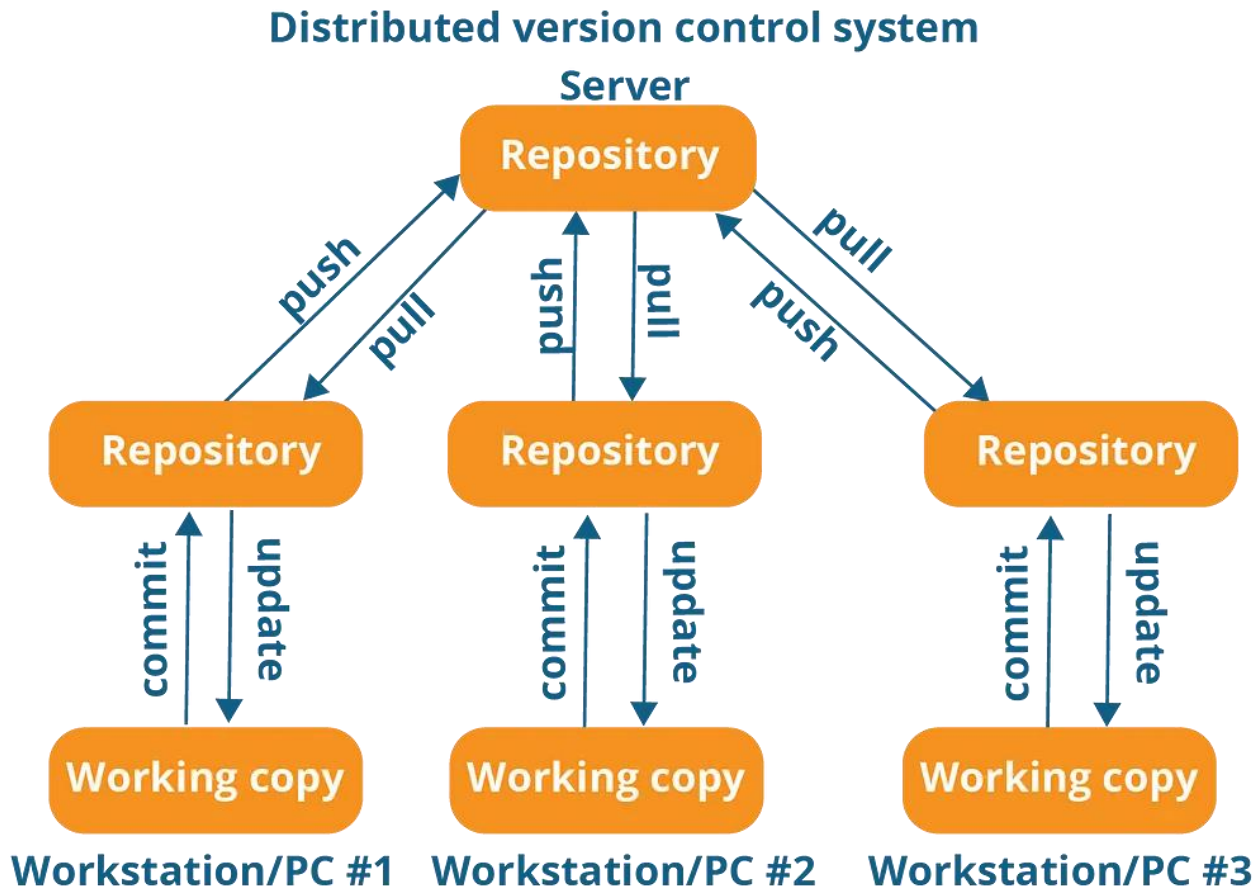
---

## ○ **Decentralizált rendszer (DVCS, elosztott)**

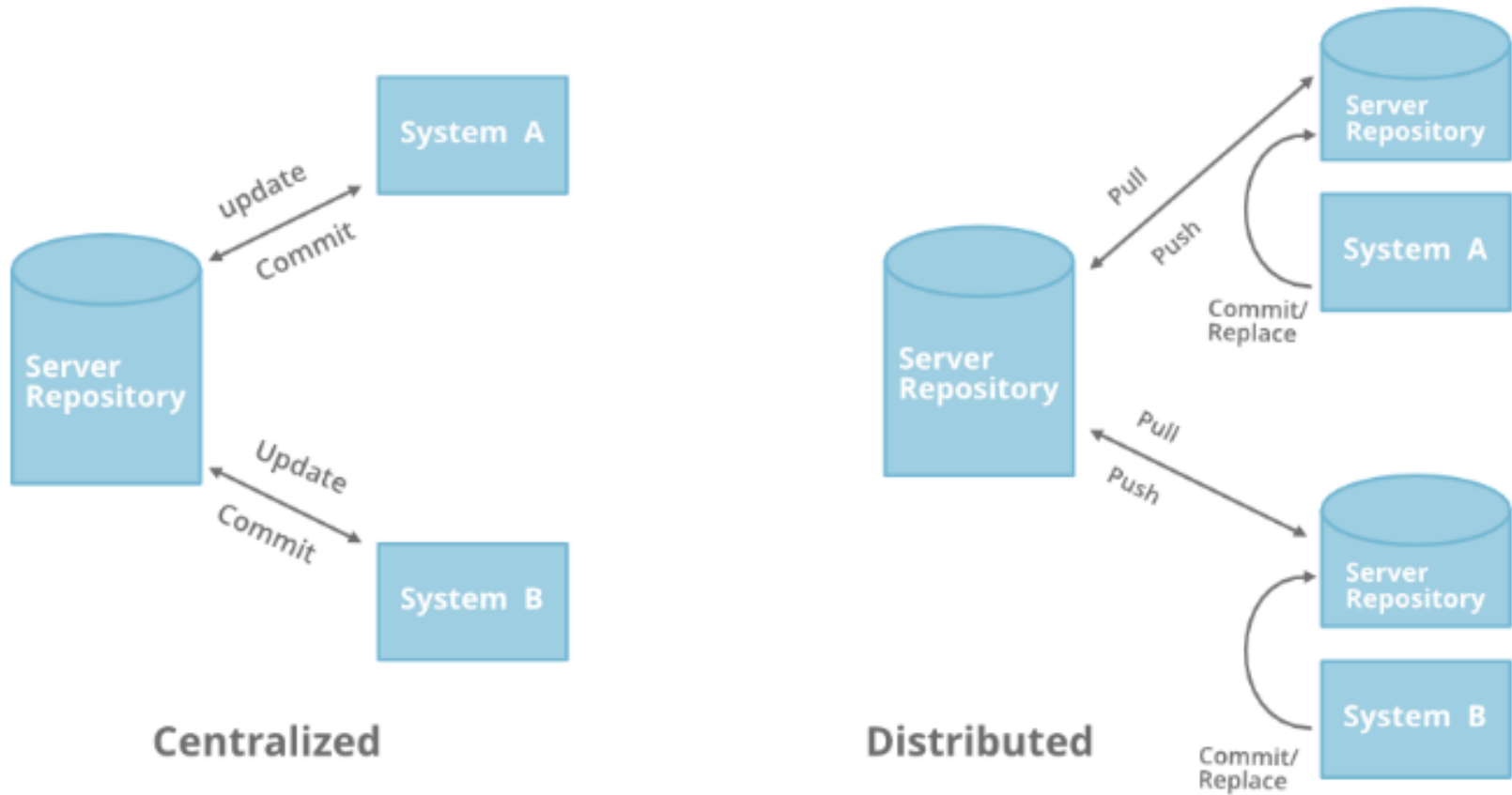
- nincs központi tároló
  - mindenkinek saját helyi repo-ja van
  - felhasználói gépek a tárolók (local repo)
  - de mellette lehet távoli repo is
  - a fejlesztők a saját helyi repository-jukban dolgoznak, és csak akkor szinkronizálják a változtatásokat a távoli repository-kkal, amikor készen vannak
  
- nem kell központi szerverrel kommunikálni
  - gyors
  - nem függ a szerver elérésétől -> nagyobb rugalmasság

# Verziókezelő rendszer típusok

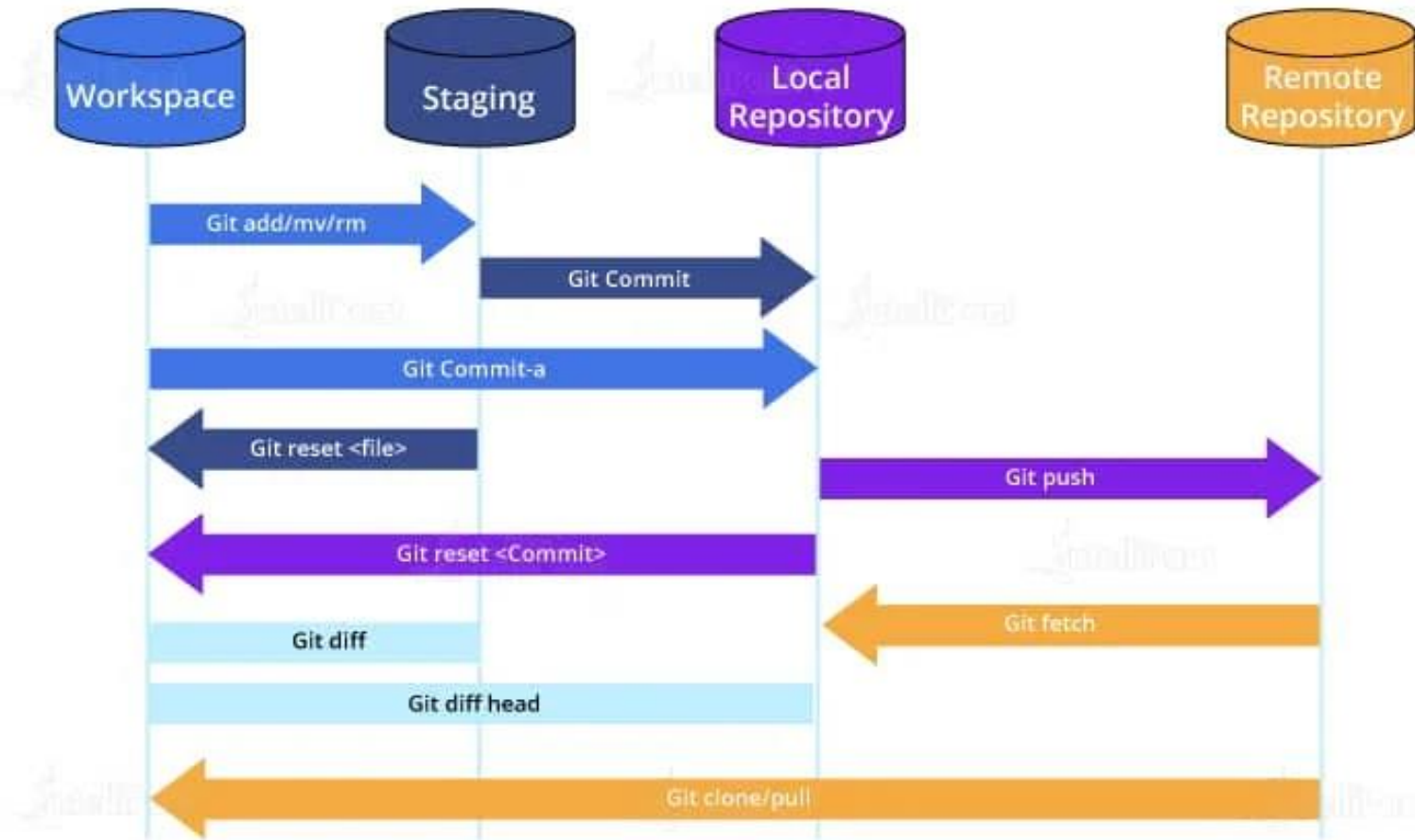
- Decentralizált rendszer (DVCS, elosztott)



# Verziókezelő rendszer típusok



# Git felépítése



# Git felépítése

## ○ Working copy (/directory)

- azok a fájlok összesége amiket verziókövet a rendszer (.git)

## ○ Staging area

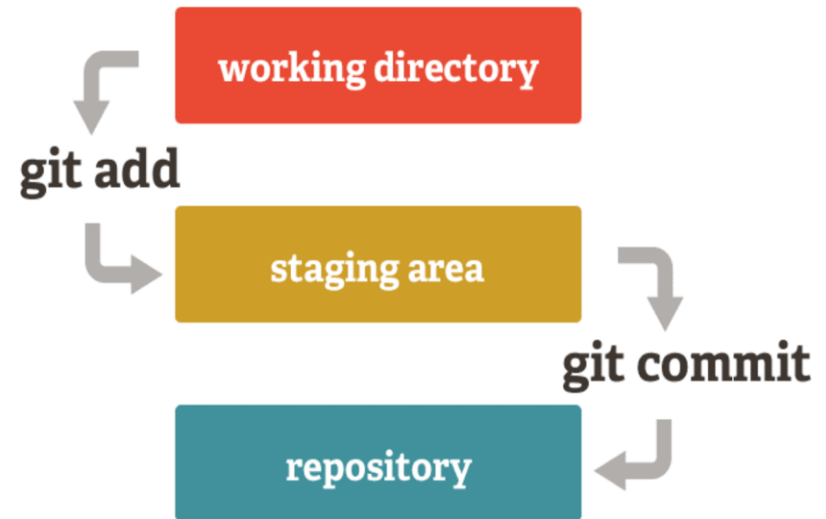
- azok a változások melyek a következő commit esetében érvényesítve lesznek

## ○ Local repository

- helyi repo

## ○ Remote repository

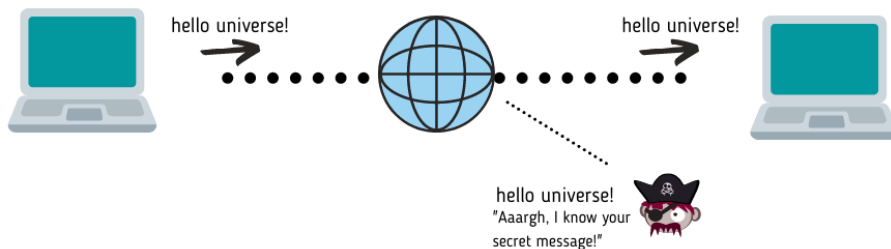
- távoli repo



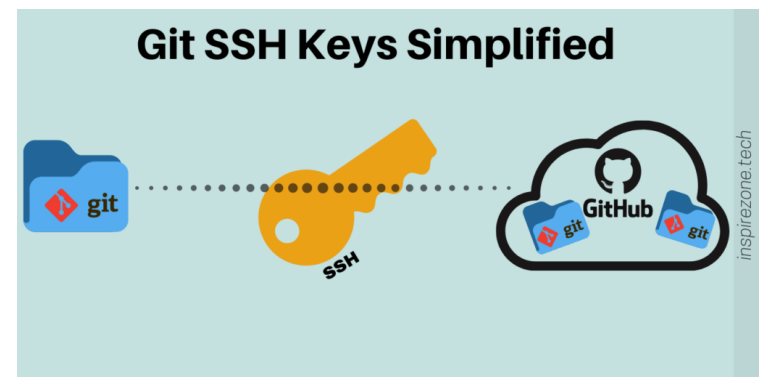
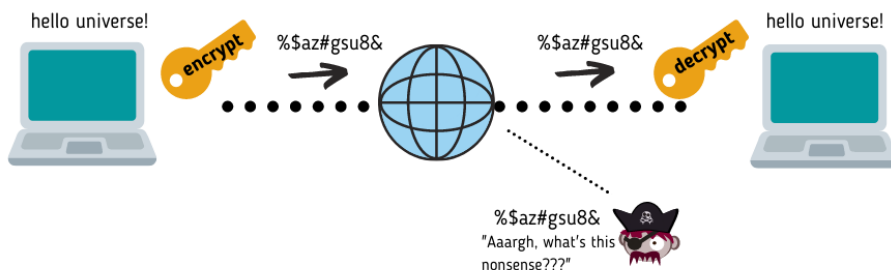
# Repository-k közötti kommunikáció

- local repo és remote repo között (Github, Bitbucket stb.)
  - csak szinkronizáláskor kell komm.
- HTTPS vagy SSH (mindkettő titkosított)
  - git clone <https://szolgaldato/felhasznalonev/repo-neve.git>
  - SSH esetében kulcspáros generálása: `ssh-keygen -t rsa`

## Unsecure channel



## Secure channel





# Gitflow

---

- Workflow model
  - Vincent Driessen, 2010, blogbejegyzés
- egy strukturált és jól definiált munkafolyamatot határoz meg
- alapja a különböző típusú ágak használata, amelyek különböző fejlesztési szakaszokat képviselnek
  - Master
  - Develop
  - Feature
  - Release
  - Hotfix

# Gitflow

---

- **Master ág (master branch):** ez az ág a stabil verziókat tartalmazza, amelyek készen állnak a kiadásra
- **Develop ág (develop branch):** a fejlesztés fő ága, amelyben az aktív fejlesztési munka zajlik. Az új funkciók és fejlesztések a develop ágon vannak fejlesztve
- **Feature ágak (feature branches):** Minden új funkciót vagy fejlesztést külön feature ágon fejlesztenek ki a develop ágból kiindulva. Amikor a funkció kész, a feature ág visszakerül a develop ágba

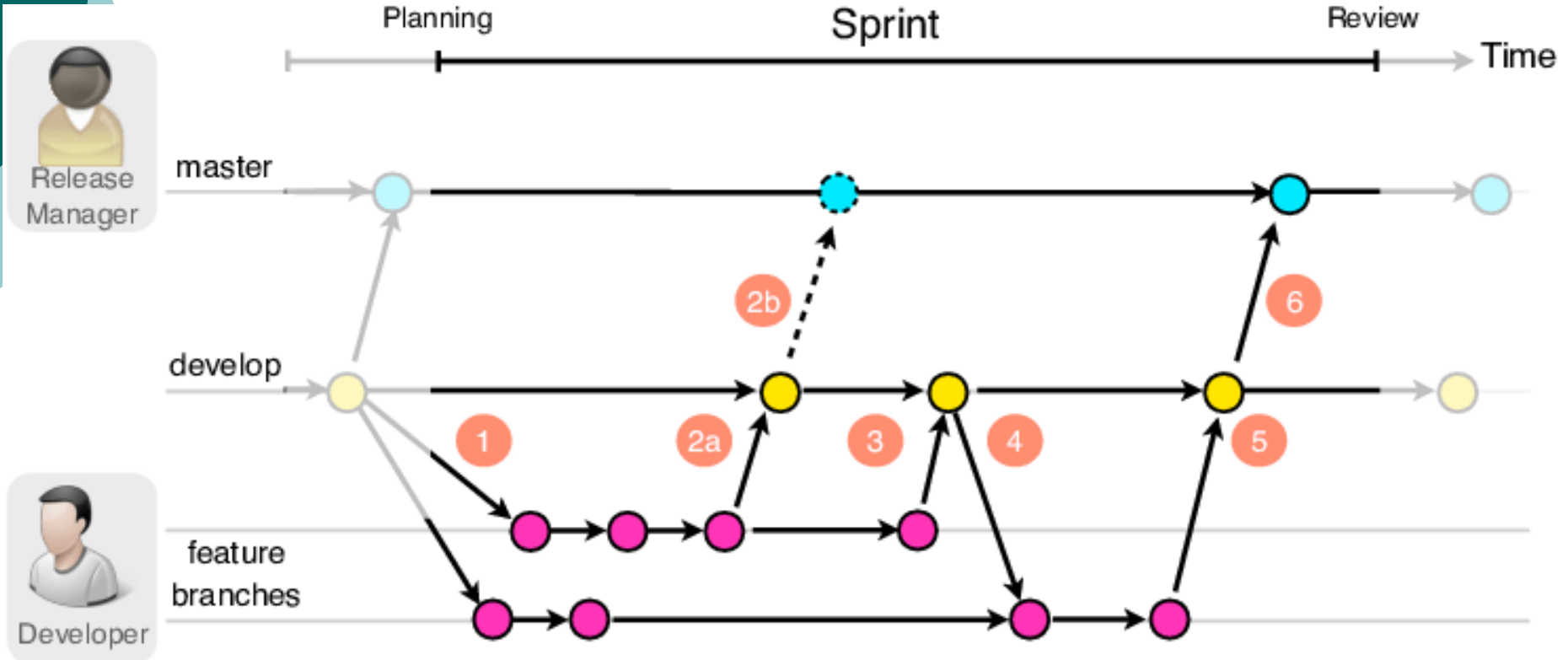


# Gitflow

---

- **Release ág (release branches):** Amikor egy új verzió kész a kiadásra, létrehozunk egy release ágat a develop ágból, ahol a kiadásra vonatkozó utolsó finomítások és javítások történnek. A release ág végül össze lesz vonva a master ággal, és a kiadás létrejön
- **Hotfix ágak (hotfix branches):** Amikor hirtelen és sürgős javításokra van szükség egy már kiadott verzióban, létrehozunk egy hotfix ágat a master ágból. Ez a javítás elvégzése után visszakerül a master és a develop ágba is

# Gitflow



Ábra forrása:

Krusche, Stephan & Alperowitz, Lukas. (2014). *Introduction of Continuous Delivery in Multi-Customer Project Courses*. 36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings. 10.1145/2591062.2591163.

# Felhasznált irodalom

---

- *Dr. Mileff Péter, Szoftverfejlesztés, Verziókövetés, Verziókövető rendszerek, jegyzet*
  - [https://users.itt.uni-miskolc.hu/~mileff/szf/Verziokezeles\\_V5.pdf](https://users.itt.uni-miskolc.hu/~mileff/szf/Verziokezeles_V5.pdf)
- *Mark Groves: Introducing Gitversion control*
- <https://www.atlassian.com/git/tutorials>
- <https://git-scm.com/docs/gittutorial>



---

Köszönöm a figyelmet!

*thank you* 😊