Git

Készítette: Czeglédi Dávid

Mi ez?

- A verziókezelés olyan eljárások összessége, amelyek lehetővé teszik egy adathalmaz változatainak (verzióinak) együttes kezelését. Szoftverek estében ez a szoftver életciklusa során a forráskódban végzett módosítások tárolását jelenti.

Miért van rá szükség?

- A fejlesztés során a forráskód sok iteráción megy keresztül, így szükséges lehet, hogy esetleges probléma eseten vissza lehessen térni egy korábbi verzióra.
- Egyszerű verziókövetés: minden változtatást elmentünk külön jegyzékekben. Megvalósítható, de nehézkes, időidényes, nem hatékony, nem átlátható.
- **Megoldás: verziókövető rendszerek (pl. Git, Mercurial, stb.).**
- **Lehet használni helyi (local) illetve távoli (remote) repokat, ilyen távoli repok például a github, bitbucket stb.**

Repository

- Ez maga a központi „tárhely”, ahol a projekt teljes története és aktuális állapota található.
  - Tartalmazza az összes fájlt, könyvtárat, valamint a változások (commitok) történetét is.
  - Olyan, mint egy könyvtár a felhőben, amelyben nemcsak a jelenlegi verzió, hanem az összes korábbi verzió is megtalálható.
- **Példa**:

GitHubon létrehozol egy repo-t „webshop” néven. Ez a repo tárolja a webshop összes forráskódját és verzióját.

Working copy

- Ez a helyi gépeden lévő példány, amin éppen dolgozol.
  - A repo tartalmát „lehúzod” (clone/checkout), majd azon szerkesztesz.
  - Ha módosítasz egy fájlt, az csak a working copyban változik, amíg nem commitolod.
- **Példa:**

Megnyitod a index.html fájlt, hozzáadsz egy új menüpontot. Ez még csak a te working copydban van, más nem látja.

Commit

- A kódon eszközölt változtatásokat úgynevezett kommitok formájában érvényesíthetjük a tárolókon belül. A tárolók mintegy pillanatképként tartalmazzák azokat, illetve projektünk aktuális állapotát. Célszerű minden nagyobb módosítást követően kommitolnunk. Az adott kommithoz általában megjegyzés is írható, hogy milyen módosítás történt az adott kommit hatására.
- **Példa:**

  git commit -m "Új menüpont hozzáadása a navbarhoz“
- Ezután bármikor vissza tudsz térni ehhez az állapothoz.

Revision (verzió)

- Minden commit kap egy egyedi azonosítót (hash).
  - Ez az adott „verziószám”.
  - A revision alapján pontosan meg tudod mondani, hogy a kód melyik állapotában vagy.
- **Példa:**

Egy commit azonosítója így nézhet ki: f5c2a9d. Ha erre hivatkozol, akkor biztosan ugyanazt a verziót éri el más is.

Checkout

- Egy adott commit/branch alapján létrehozott helyi másolat vagy váltás egy másik verzióra.
  - Segítségével nemcsak a legfrissebb kódot tudod nézni, hanem bármelyik régebbi állapotot is.

- **Példa:**

  Ha git checkout f5c2a9d-et írsz, akkor visszaugrasz ahhoz a régi commit állapothoz.

Head

- A HEAD az aktuálisan használt commitot mutatja.
  - Általában a legfrissebb commit az aktuális ágon.
  - Olyan, mint egy könyvjelző, ami mindig mutatja, hogy „hol tartasz most”.

Push

- A helyi commitokat feltöltöd a központi repóba (pl. GitHub).
  - Ettől lesz elérhető más fejlesztőknek is.
- **Példa:**

Ha otthon dolgozol és kész vagy egy új funkcióval → git push → kollégáid másnap le tudják húzni.

Pull

- Letöltöd a központi repóban történt változásokat a gépedre.
  - Gyakran tartalmaz új commitokat, amiket más fejlesztők toltak fel.
- **Példa:**
  - **Ha kollégád közben kijavította a hibás CSS-t, a git pull paranccsal lehúzod, és nálad is friss lesz.**

Diff / Change / Delta

- Megmutatja két verzió közötti különbségeket.
  - Lehet két fájl vagy két commit között.

- **Példa**:
  - A diff kiírja:

```
- background-color: red;
+ background-color: blue;
```

Branch (ág)

- Különálló fejlesztési vonal.
  - A main (főág) a stabil verzió, mellette hozhatsz létre új ágakat kísérletezéshez, funkciófejlesztéshez.
- **Példa:**

  feature-login ágban megírod a bejelentkezést, miközben a main stabilan működik tovább.

Merge

- Két ágat egyesítesz.
  - Ha kész a feature-login, akkor merge-ölheted a main ágba.
  - Ha nincs ellentmondás, automatikus; ha van, manuális feloldás kell (→ conflict).

Conflict

Akkor keletkezik, ha két ág ugyanazt a sort másképp módosította, és a rendszer nem tudja eldönteni, melyik legyen a helyes.

- Neked kell kézzel kijavítanod.

- **Példa: (index.html)**

egyik ágban:

```
<h1>Üdvözlet!</h1>
```

másik ágban:

```
<h1>Hello!</h1>
```

Merge után a Git megjelöli a konfliktust, és neked kell választani.

Clone

- Egy meglévő repo teljes tartalmának letöltése a gépedre.
  - A kezdő lépés, ha csatlakozol egy projekthez.
- **Példa:**

git clone https://github.com/valaki/webshop.git

Létrejön a „webshop” könyvtár, benne az összes fájl és a teljes történet.

Összefoglaló

- **clone** → repo letöltése első alkalommal
- **pull** → frissíted a helyi verziót mások módosításaival
- **commit** → saját változtatás mentése helyben
- **push** → feltöltöd a központi repóba, hogy más is lássa

Alapfogalmak

- **Repository:** röviden csak repo. Maga a tárolónk.
- **Working copy:** A kód egy részének egy példánya, amelyen a fejlesztő éppen dolgozik a saját gépén
- **Commit:** A kódon eszközölt változtatásokat úgynevezett kommitok formájában érvényesíthetjük a tárolókon belül. A tárolók mintegy pillanatképként tartalmazzák azokat, illetve projektünk aktuális állapotát. Célszerű minden nagyobb módosítást követően kommitolnunk. Az adott kommithoz általában megjegyzés is írható, hogy milyen módosítás történt az adott kommit hatására.

Alapfogalmak

- **Revision:** verzió
- **Checkout:** Lokális másolat készítése valamely verziókezelt fájlról.
- **Head:** a legfrissebb kommitot (verziót) jelöli, az aktuális ág teteje.
- **Push:** adatok feltöltése a központi repoba
- **Pull:** változások letöltése
- **Diff/Change/Delta:** két file között változás megtalálása/mutatása.

- **Branch:** fejlesztési ág
- **Merge:** összefésülés. A fejlesztési ágak létrehozása mellett lehetőségünk van ezek egyesítésére is.
- **Conflict:** Ágak összefésülése során keletkező jelenség. A két ág verziója olyan kódot tartalmaz, amit nem lehet automatikusan összefésülni
- **Clone:** repo tartalmának lehozása lokálisan (adott jegyzékbe)