Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Bevezetés a számítástechnikába #09 – Verziókezelés

2023. november 27–30.

Siklósi Bálint <siklosi.balint@itk.ppke.hu>
Naszlady Márton Bese <naszlady@itk.ppke.hu>

#09/1 – Verziókezelő rendszerek

A probléma

- Követtél-e el egy hibát a kódodban, amit utólag vissza akartál volna csinálni?
- Szükséged volt-e valaha arra, hogy valamiből több változatot tarts fenn egyszerre?
- Vissza akartad-e nézni valaha, hogy hogy fejlődött egy kód?
- Bizonyítani akarod, hogy egy bizonyos változás megjavítja/elrontja a kódod?
- Volt-e valaha, hogy elvesztettél egy projektet és nem volt biztonsági mentésed?
- Akartad-e követni, hogy mennyi munka készült el, hogy hol, illetve hogy ki által?
- Akartad-e valaha is más kódját kiegészíteni?
- Akartad-e valaha is megosztani a kódodat másokkal, vagy engedélyezni hogy veled párhuzamosan mások is fejlesszék azt?
- Akartál-e valaha is kísérletezni új módszerekkel anélkül, hogy a működő kódot elrontanád?

A megoldás

Ha az előzőek közül bármelyikkel volt már problémád, akkor javasolt valamilyen **verziókezelő rendszer** (Version Control System – VCS) használata. Ez lehet:

- CVS (Concurrent Version System)
- Git
- SVN (Subversion)
- TFS (Team Foundation Server)
- Perforce
- Mercurial
- …

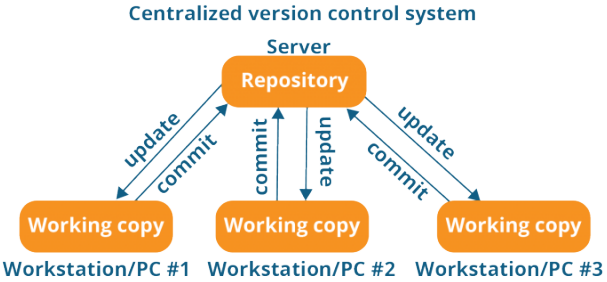
VCS

Verziókezelés:

- Különböző dokumentumok, programok, honlapok stb. fejlődése során keletkező változások kezelése.
- Ezek a változatok azonosítóval (revision number), illetve egyéb hasznos metaadatokkal vannak eltárolva (változat szerzője, keletkezés időpontja stb.)
- Ezek a változatok bármikor visszafejthetők, egymással összevethetők

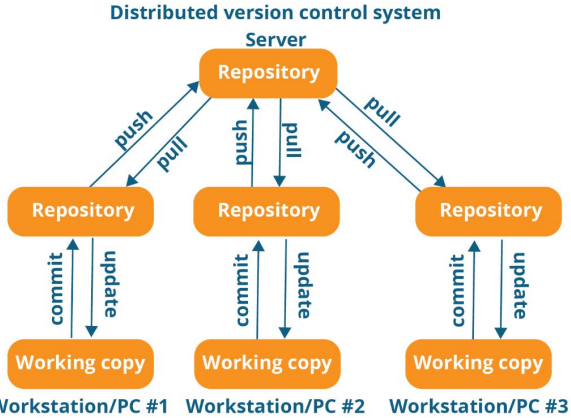
VCS típusai - Központosított

- Centralized Version Control System - CVCS
- Pl.:SVN



VCS típusai - Megosztott

- Distributed Version Control System - DVCS
- Pl.:Git



Git szerverek

- Webes szerverek, ahol könyvtárakat lehet létrehozni, elérni, hozzáadni stb…
- A tényleges szerveres adattárolás itt történik
- Példák:
  - GitHub
  - GitLab (pl. dev.itk.ppke.hu)
  - BitBucket
  - …

#09/2 – Feladatok

1. Feladat

Hozz létre egy saját Git repository-t a dev.itk.ppke.hu szerveren. Ehhez nyiss egy böngészőt, és lépj a `https://dev.itk.ppke.hu/` oldalra.

1. Add meg a Shibbolet-es felhasználónevedet és jelszavadat a belépéshez!
2. Belépés után kattints a "Create project" gombra!
3. Hozz létre egy üres projektet a "Create blank project"-re való kattintással.
4. A projekt neve a saját neved alapján legyen beállítva: (pl. *Béla repója*), a láthatóságot állítsd "internal"-ra.
5. A jelölőnégyzetek közül csak az első (readme fájl létrehozása) legyen bekapcsolva.

2. Feladat

Oldd meg, hogy ezt a GitLab-os projektet a cortex szerverről is elérd.
Ehhez először be kell állítani egy ún. ssh kulcsot, amivel a két szerver egymásra tud találni.

1. Add ki az alábbi parancsot a cortex.itk.ppke.hu szerveren:

   ```
   ssh-keygen -t ed25519 -f "$HOME/.ssh/id_rsa" -N ""
   ```

2. Lépj a böngészőben az alábbi oldalra: `https://dev.itk.ppke.hu/-/profile/keys`
3. Add hozzá a cortex-en futtatott parancs által generált publikus kulcsfájl tartalmát a táblázathoz az "Add new key" gombra kattintva:
   - 3.1 nézd meg a cortex-en futtatott parancs kimenetét, hova mentette a publikus kulcsot?
   - 3.2 írasd ki ennek a fájlnak a tartalmát pl. a `cat`-tel!
   - 3.3 másold ki a teljes tartalmat a terminálból
   - 3.4 illeszd be a tartalmat a böngészőben az űrlapba (Key mező)
   - 3.5 a többi beállítás maradjon változatlan
   - 3.6 mentsd el a kulcsot az "Add key" megnyomásával.
4. próbáld ki a kulcs működését, add ki a cortex szerveren az alábbi parancsot:

   ```
   ssh git@dev.itk.ppke.hu
   ```

   Ha működött, akkor *Welcome to GitLab, <neved>!* üzenetet kell látnod.

3. Feladat

Kezd el használni a repódat a cortex szerverről is (ne csak a böngészőből).

1. Keresd ki a projekt SSH-val való klónozásához szükséges URL-jét!
2. Add ki a cortex szerveren a következő parancsot:

   `git clone URL`

   ahol az URL rész az előző pontban megállapított URL.
3. Lépj be a repó mappájába, és nézd meg a fájlokat! Látnod kell a README.md nevű fájlt, ahogy azt a böngészőben is láttad a fájlok közt.
4. A repódon belül hozz létre egy szöveges fájlt (hello.txt) valamilyen tartalommal, add hozzá a git repohoz és commitold! (`git add hello.txt` és `git commit`) A commit üzenetben mindig értelmesen írd le, hogy milyen változás történt (akár csak egy sorban), hogy vissza tudd könnyen keresni.
5. Módosítsd az imént committolt fájlt, és commitold a változtatásokat! (Használhatod a `-m "<uzenet>"` kapcsolót is a gyorsabb kezeléshez)
6. Nézd meg a webes felületet. Látod a fájlokat és a változásokat? Miért nem?

4. Feladat

Szinkronizáld a helyi változtatásokat a gitlab szerverrel!

1. Pushold fel a lokális változásokat a szerverre! (`git push origin main`)
2. Nézd meg a webes felületet. Ott vannak a fájlok?
3. Nézd meg a webes felületen az oldalsó menüben a "Code > Repository Graph" lehetőségre kattintva előbukkanó idővonalat. Mit látsz rajta?
4. Add ki a cortex szerveren a `git log` parancsot. Hasonlítsd össze az idővonalat és a parancs kimenetét!
5. Állj vissza a cortex szerveren az egy committal előző állapotra, nézd meg a fájlod tartalmát, majd térj vissza a legvégső állapothoz! (`git checkout <commit>` és `git checkout main`)

5. Feladat

A következő feladatokat közösen (2-3 fős csoportokban) végezzétek! Legyen valaki, aki a "projektgazda" lesz, az ő repójában dolgozzanak a többiek is.

1. A projektgazda mondja el a git repójának URL-jét a többi közreműködőnek. A közreműködők is klónozzák le azt.
2. Hozzon létre mindenki egy új fájlt a saját nevével, és commitolja, pusholja azt. Mindenkinek sikerült?
3. A projektgazda adjon a közreműködőknek is jogot a repó írásához. A webes felületen a "Manage > Members" résznél kell meghívni a közreműködőket. A jogosultság legyen Maintainer.
4. Az immáron jogosulttá vált közreműködők is pusholjanak!
5. Mi történik akkor, ha egy fájlt mindketten szerkesztenek, és mindketten commitolják, pusholják? Miként lehet feloldani az ütközést? Hogyan lehetne eleve elkerülni az ilyen eseteket?

6. Feladat (pluszpontért)

Ha ezt a feladatot is végrehajtjátok, akkor pluszpontot kaphattok a helyes megoldásra.

1. Nézz utána annak, hogy mik azok a git branch-ek!
2. Készítsetek egy-egy saját branch-et, amin elkészíted a "fejlesztést"!
   2.1 `git branch <branch neve>`
   2.2 `git checkout <branch neve>`
3. Dolgozzatok a saját branch-eteken, hozzatok létre új fájlokat és szerkesszetek olyanokat is, amik már léteznek.
4. Commitoljátok és pusholjátok a szerkesztéseket
5. Nézzétek meg a gráfot a webes nézetben (Code > Repository Graph)
6. Hozzatok létre merge request-eket a branch main-be olvasztásához, amit a projektgazda fogadjon el.
7. Nézzétek meg ismét a gráfot és a kódbázist.

VÉGE

PÁZMÁNY

Pázmány Péter Katolikus Egyetem

Információs Technológiai és Bionikai Kar