OS X.

Operációs rendszerek alapjai
X. gyakorlat

verziókezelés + gyakorlás

Verziókezelés - motiváció

OS X.

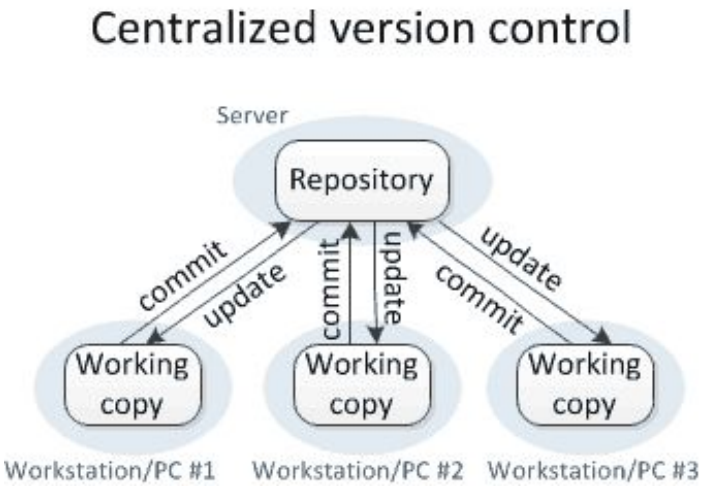
- Programozás során előfordul, hogy vissza akarunk térni egy olyan állapothoz, ami már egyszer biztosan működött - mégsem szeretnénk, hogy legyen egy csomó biztonsági másolatunk, mert az nehezen áttekinthető
    - pl.: szakdoga_v99_final_52_teljes_kesz_vegleges
- Több gépről is szeretnénk ugyanazon a projekten dolgozni
- Több ember szeretne egy közös projekten egyszerre párhuzamosan dolgozni
    - A különböző változtatásokat megfelelően össze szeretnék hangolni

Verziókezelés - típusok

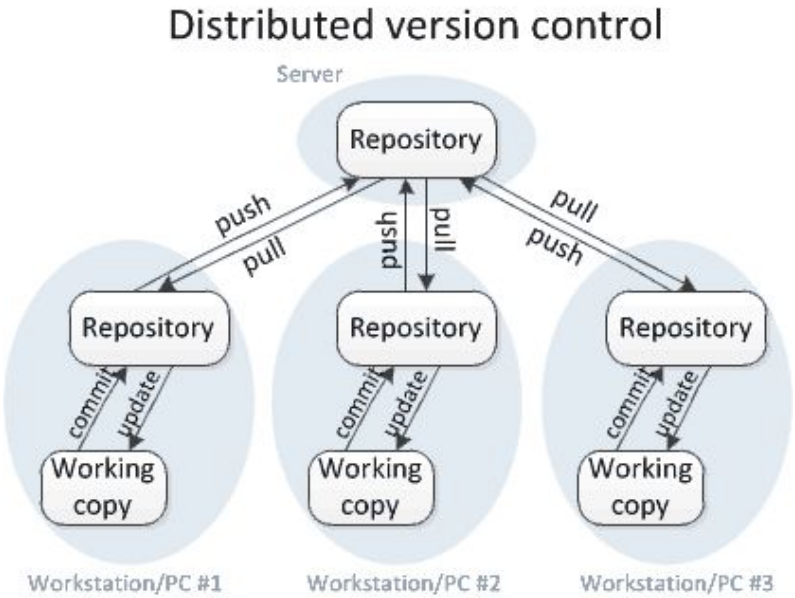
- Központosított
  - Centralized Version Control System - CVCS
  - Pl.: SVN
- Megosztott
  - Distributed Version Control System - DVCS
  - Pl.: Git

Verziókezelés - CVCS

OS X.



Verziókezelés - DVCS



Git

OS X.

- Elosztott verziókezelő szoftver
- Nyílt forráskódú
- Linus Torvalds fejlesztette ki a Linux kernel fejlesztéséhez
- https://git-scm.com/downloads

Git parancsok

OS X.

- git init
  - helyi git repository létrehozása
- git status
  - az adott mappában lévő git repositoryról ad információt
- git add
  - fájl hozzáadása a stage-hez, még nincs a repoban
- git commit
  - hozzáadás a repohoz, -m “message”

Git parancsok

OS X.

- git log
  - megadja a korábbi commitolásokat dátum szerint rendezve
    - commit azonosítója
    - ki csinálta a módosítást
    - mi az üzenet a commitnál

Git szerverek

OS X.

- Webes szerverek, ahol repositorykat lehet létrehozni, elérni, hozzáadni, stb...
- A tényleges szerveres adattárolás itt történik
- Példák:
  - **GitHub**
  - GitLab
  - BitBucket
  - stb...

GitHub

OS X.

- Egyik leggyakrabban használt szerver szolgáltató
- Ingyenesen használható, korlátlanul csak nyílt projektekhez → bárki megtekintheti a feltöltött kódokat
- Privát projekteket is létre lehet hozni, de ingyenesen max 3 tagnak
- https://github.com

GitHub - új repo létrehozása

OS X.

- A GitHub-on létre tudunk hozni repositorykat, melyeket össze tudunk kapcsolni lokális repokkal:
  - git remote add origin https://github.com/user/test_repo.git
    - lokális és központi repo összekapcsolása
  - git push -u origin master
    - feltölti a központi repoba a lokális repoban lévő változtatásokat
  - git pull
    - a központi repoban lévő változásokat tölti le a lokálisba

GitHub - repo klónozása

OS X.

- A GitHub-on létrehozott repositorykat le tudjuk klónozni a saját gépünkre egy lokális repoba
  - git clone https://github.com/user/test_repo.git
    - az adott mappába letölti a test_repo mappa teljes tartalmát

Git parancsok

OS X.

- git diff
  - megmutatja, hogy milyen változtatások illetve különbségek vannak a fájlokban az előző commithoz képest
- git reset
  - az addolt fájlokat vissza lehet hívni a stage-ről a reset paranccsal
- git checkout
  - visszaállítja (azaz TÖRLI a módosításokat)
  - *git checkout -- vmi.txt*

Ökölszabályok

OS X.

- Először addoljuk azokat a változtatásokat, amelyeket commitolni szeretnénk, ezután jön a commit
  - de ez még csak a lokális repora hat
- Ha a központi repoba akarjuk feltölteni a változtatásainkat (azaz pusholni akarunk), előtte mindig pull-olni kell, azaz le kell kérni a központi repo legfrissebb állapotát
  - ha a mi reponk nem a legfrisebb, akkor nem is fogja engedni a push-t

Branch létrehozása

OS X.

- Lehetőség van az eredeti repoval párhuzamos állapotokat létrehozni, vagyis elindulhat “oldalirányban” egy új fejlesztés, ezek a branchek
  - természetesen ezek később beépíthetők az eredeti ágba
- git branch test_branch
  - létrehoz az aktuális branchből egy másolatot, jelen esetben test_branch néven
- git branch
  - kilistázza a brancheket
  - csillaggal és zölden jelöli az éppen aktuálisat

Branch használata

OS X.

- *git checkout test_branch*
  - átvált a test_branch-re
  - ezután minden amit csinálunk csak a test_branchen hajtódik végre, az eredeti változatlan marad
- össze lehet fésülni a brancheket
  - *git merge test_branch*
  - mindig onnan hívjuk meg ahova fésülünk

Branch használata

OS X.

- törölni is lehet a brancheket természetesen:
  - *git branch -d test_branch*
  - *git push origin --delete test_branch* (szerverről törli)

Ökölszabályok II.

OS X.

- Van egy master branch
  - Ez lesz a release
  - ebbe csak ellenőrzött, letesztelt, működő kódot töltünk fel
- Az elkülönülő részfeladatokat külön branchekben készítjük
  - egymástól független
- Az elkészült funkciókat, ha működik, akkor a masterbe lehet mergelni

G10F01

OS X.

- Készítsünk egy scriptet, ami az adott mappában található fájlok kiterjesztését módosítja
  - A script
    - első paramétere az átírandó fájlkiterjesztés legyen
    - második paramétere pedig a kiterjesztés legyen, amire át akarjuk írni
    - ha nem megfelelő a paraméterek száma, írjon ki hibaüzenetet
    - miután lefutott írja ki, hogy hány darab fájlt módosított

G10F02

OS X.

- A Unix(-szerű) rendszerek rendszerbetöltést követő állapotát a runlevel határozza meg. A runlevel-ekhez tartoznak scriptek is, amik az adott runlevel betöltésével lefutnak.
- Az adott runlevelhez tartozó scriptek a /etc/rcX.d elérési út alatt találhatóak, ahol X az adott runlevel száma.
- Írj egy scriptet, ami kilistázza a paraméterben megadott runlevel-hez tartozó scripteket, midegyik után megjelenítve a fájlban található rövid leírást (Short-Description), illetve a végén kiírja az adott runlevelhez tartozó scriptek számát!
    - Bemenet pl.: G10F02 3
    - Kimenet tartalmazza:
        - az adott runlevelhez tartozó scriptek nevét
            - az egyes scriptek neve alatt a hozzájuk tartozó rövid leírás
        - Az adott runlevelhez tartozó scriptek számát