Verzió követő rendszerek (SCM)

SIPOS Péter
Szoftverfejlesztési csoportvezető
peter.sipos@neti.com
+36309280477

NETI
Information is our service

Tartalom

- Mire jók a verziókövető rendszerek?
- Mik az alapvető utasítások a verziókövető rendszerek esetében.
- Ismerkedés a GIT-el röviden.
- Hogyan javítsunk a kódminőségen SCM-el?
- Milyen workflow-t alkalmazzunk?

Verziókövető rendszerek

- SCM – **S**ource **C**ontrol **M**angement
- Kódváltozások archiválása
- Kooperáció a csapaton belül

Kódváltozások archiválása

„Nobody actually creates perfect code the first time around, except me. But there's only one of me. ”

Linus Torvalds, 2007.

- Könnyebb hibajavítás, ha valaki eltörte a kódot
- Mindennek nyoma marad, nincs véletlenül kidobott kód
- Centralizált esetben a kód biztonságos szerveren van
- Egymástól eltérő kódbázis fenntartása közös alapokon

Kooperáció a csapaton belül

- Külön fejlesztési ágak létrehozása
- Leegyszerűsített code review
- Könnyebb áttérés egy másik munkára, hibajavításra
- Egymás kódjának megtekintése, azonnali közreműködés
- Release és tesztelési folyamat leegyszerűsödik

Mindig vannak hibák. Az SCM lehetővé teszi ezek nyomon követhetőségét.

De hogyan is „verziókövessünk”?

- Nem csak magunknak végezzük, sőt elsősorban másoknak
- Mindent részletesen írjunk le
- Alapvető utasítás: „commit”
- Legyen minél gyakoribb és egyértelmű a commit
- Figyeljünk rá, hogy minden lényegeset feltegyünk, a szemetet pedig ne (fordító fájljai, ideiglenes object-ek, stb…)

A „Commit”

- Lehető leggyakrabban, amikor kis funkció elkészül (akár 10 soronként)
- Csak összetartozó kódok legyenek egy commit-ban (interface módosítás, új funkció)
- Nem gond, ha a commit nem működik
- Legyen beszédes és egyértelmű
- Legyen angol nyelvű

The good commit message*

- Separate subject from body with a blank line
- Limit the subject line to 50 characters
- Capitalize the subject line
- Do not end the subject line with a period
- Use the imperative mood in the subject line
- Wrap the body at 72 characters
- Use the body to explain *what* and *why* vs. *how*

** Source: http://chris.beams.io/posts/git-commit/*

Other commands

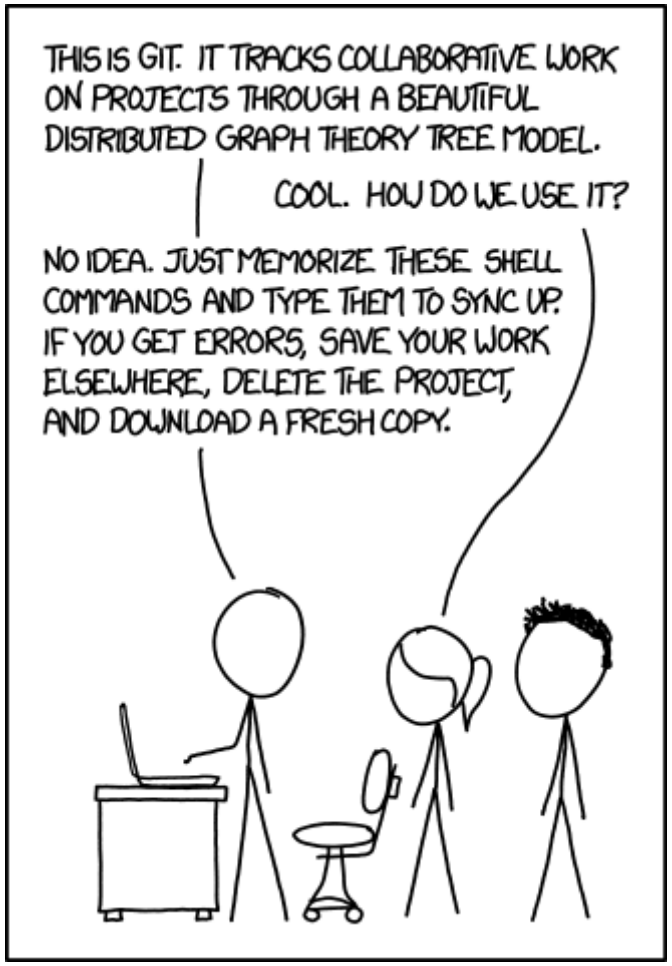
- Clone - saját másolat készítése
- Tag - megcímkézés
- Push - feltöltés
- Pull - letöltés
- Merge – összeolvasztás
- Checkout - egy állapot megtekintése

GIT verziókövető rendszer

Ingyenes.

- Nem kifejezetten centralizált
- A változások csak munkamásolatként léteznek (working copy)
- A commitra jelölt változások a stage-be kerülnek
- Lehetőség van „remote” beállítására
- Init, clone, commit, add, tag, push, pull, fetch, merge, revert, …

http://pcottle.github.io/learnGitBranching/



Source: http://explainxkcd.com/wiki/index.php/1597:_Git

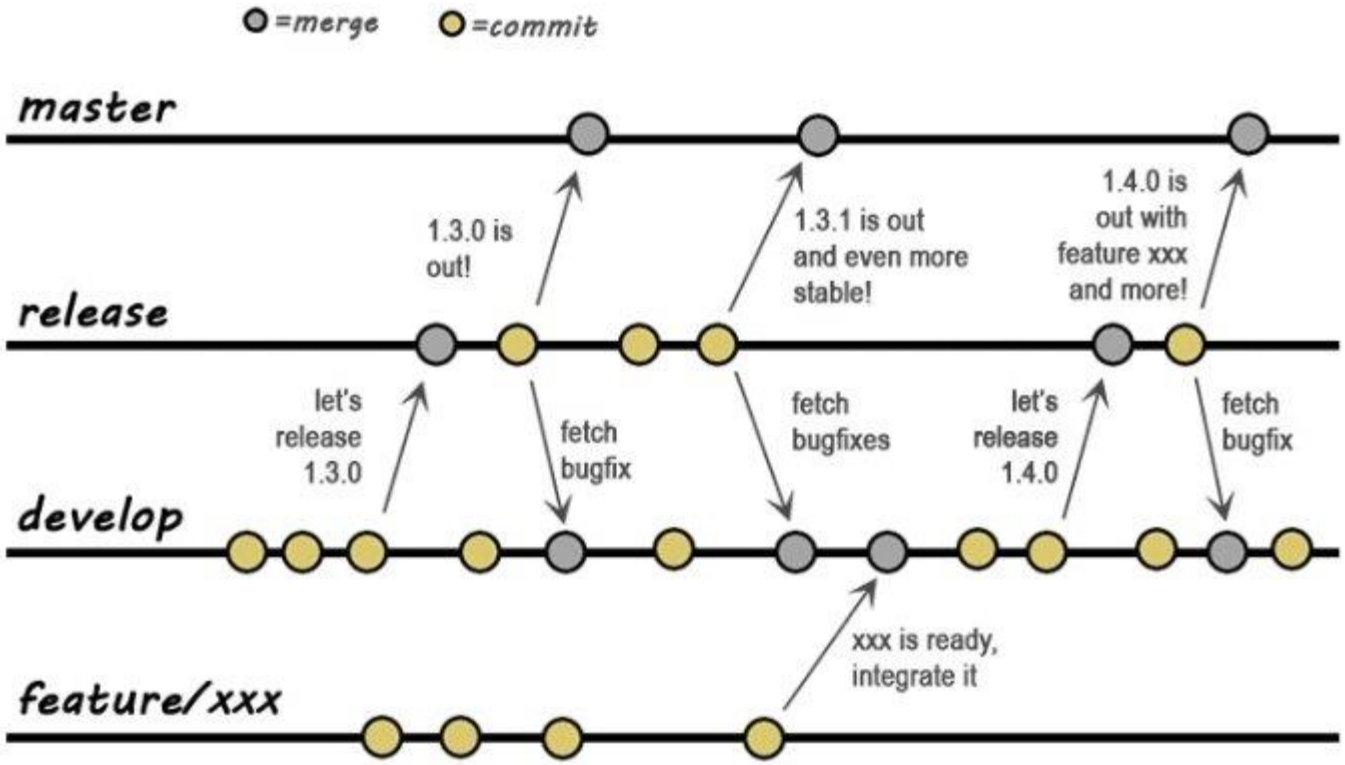
Kódminőség javítása

- Visszakövethetőség
- Könnyű hibajavítás
- Könnyű megtalálni egy sor felelősét
- Könnyű code review, így uralni lehet a kódot
- A commitokra bontás átgondolttá teszi a folyamatot

Workflow

- Stabil (védett) ágak működő verzió érdekében
- Minden funkció külön ágok (branch) fejlődik
- A funkció ágak csak ellenőrzés után kerülnek be a stabil ágba

Worklow



Soruce: https://www.linkedin.com/pulse/git-merge-resolving-git-merge-conflict-code-branches-gaurav-aggarwal

Tools

- Gitlab*
- Gitorious
- GitHub

Kérdések?

Köszönöm a figyelmet!

Péter SIPOS
Software Development Team Leader
peter.sipos@neti.com
+36 30 928 0477

NETI
Information is our service