

Verziókezelés a Git használatával

Vajna Miklós

2009. október 2.

Miről lesz szó?

- Miért jó a verziókezelés?
- Miért jó az elosztott verziókezelés?
- Miért jó a Git?
- A Gitről - alulról felfelé
- A Git használata külsősként
- A Frugalware mire használja a Gitet

Miért jó a verziókezelés?

- Mindenki használ verziókezelést, legfeljebb nem tud róla (Mentés másként, tarball + patch-ek, stb.)
- Kollaborációs munkához elengedhetetlen
- Hibakeresést segíti
- Dokumentációs eszköz

Miért jó az elosztott verziókezelés?

- A teljes repó elérhető helyben, gyors blame, log, diff, merge
- Nincs szükség hálózati kapcsolatra
- Nincs SPoF
- Megszűnhet a *committer* fogalma
- Backup jelentősége csökken
- Branch/merge egyszerűbbé válik

Miért jó a Git?

- A legtöbb előny természetesen az elosztottságból fakad
- merge-recursive (vö. Subversion)
- rerere
- blame - kódblokk-áthelyezés érzékelése (vö. explicit másolás/átnevezés)
- git grep
- combined diff

- Alacsony szinten egy tartalom szerint címezhető fájlrendszer
- 4 objektum-típus: blob, tree, commit, tag
- blob: egy fájl egy változata
- tree: lehet tree vagy blob, mindegyikből több, de összesen legalább egy
- commit: 0..sok parent, egy tree
- tag: van neve, és bármire mutathat (commitra szokott)

- ref, symref
- hook
- reflog
- config
- index

Merge vs. rebase

- Kiindulás:

```
      A---B---C topic
      /
D---E---F---G master
```

- rebase:

```
                A'--B'--C' topic
                /
D---E---F---G master
```

- merge:

```
      A---B---C---H topic
      /           /
D---E---F---G master
```

A Git használata külsősként

- Külsős: nincs commit joga, patch-eket küld
- Helyben persze git-ben dolgozik
- Rebase-el, nem merge-öl
- Interactive rebase: squash, darabolás, rendezgetés
- git format-patch, git am
- Bundle-ök

Git parancsok: sok van, melyik kell nekem?

- A Git 1.6.4 esetén 145 parancs
- Fő magas szintű parancsok
- Mellék magas szintű parancsok
- Alacsony szintű parancsok

- init, clone, add, rm
- status, diff
- commit, reset
- fetch, pull, push
- branch, checkout, rebase, merge
- log, tag, mv, show, grep, bisect

Fő magas szintű parancsok (példák)

- archive, bundle, am és format-patch
- cherry-pick és revert
- describe, shortlog
- gc, clean, stash, submodule

Mellék magas szintű parancsok (példák)

- Manipulálók: config, filter-branch
- Lekérdezés: blame, fsck, verify-tag
- Interakció más rendszerekkel: fast-import, fast-export, archimport, cvsimport, cvsexportcommit, quiltimport, svn

- Ha scriptelni szeretnénk
- Példa: log vs rev-list:

```
$ git log --pretty=oneline HEAD~2..
```

```
2920c0c vinagre
```

```
329aae5 gtk-vnc
```

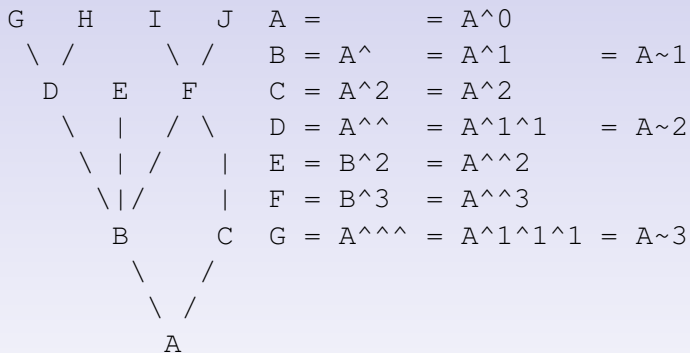
```
$ git rev-list HEAD~2..
```

```
2920c0c
```

```
329aae5
```

Commitok szimbolikus nevei

- Egy példa sokat segíthet:



$$H = D^2 = B^{^2} = A^{^^2} = A\sim2^2$$

$$I = F^{\wedge} = B^{3^{\wedge}} = A^{^^3^{\wedge}}$$

$$J = F^{\wedge 2} = B^{3^{\wedge 2}} = A^{^^3^{\wedge 2}}$$

- Probléma: egy fájlban két módosítás, de csak az egyiket szeretnénk commitolni
- Karbantartás esetén: merge-nél csak a conflict lenne az érdekes
- Megoldás: index, mint köztes réteg
- git diff, git diff --cached, git diff HEAD

- Az előbbi parancsok térbe helyezve:

```
diff
+-----+
|       |
+-----+
| Objektum- |
|   tároló   |
+-----+
diff HEAD | diff --cached
| +-----+
| | Index |
| +-----+
| | diff  |
+-----+
| Munka- |
| könyvtár |
+-----+
```

A Frugalware mire használja a Gitet

- A -current fa csomagleíróinak tárolására
- A 1.1-516-ge0b7c1e verziónál ez 4603 scriptet jelent, összesen 51263 commit, 46 fejlesztőtől
- Csomagkezelő
- Telepítő
- Dokumentáció és annak fordításai

- Innovations in git <http://gitster.livejournal.com/16077.html>
- Git for Computer Scientists
<http://eagain.net/articles/git-for-computer-scientists/>
- Összefoglaló szimbolikus nevekről: `man git-rev-parse`

- GIT honlap: <http://git-scm.com/>
- Levelezési lista: <http://vger.kernel.org/vger-lists.html#git>
- IRC: #git @ irc.freenode.net
- A diák elérhetősége: <http://vmiklos.hu/odp/>