Systémy pro distribuovanou správu verzí, GIT

Michal Wiglasz

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole
iwiglasz@fit.vutbr.cz

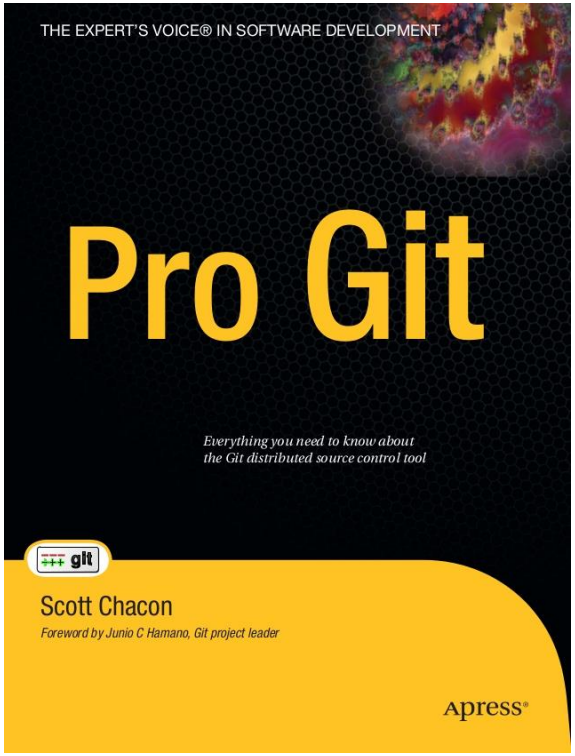
28. 2. 2017
Praktické aspekty vývoje software (IVS)

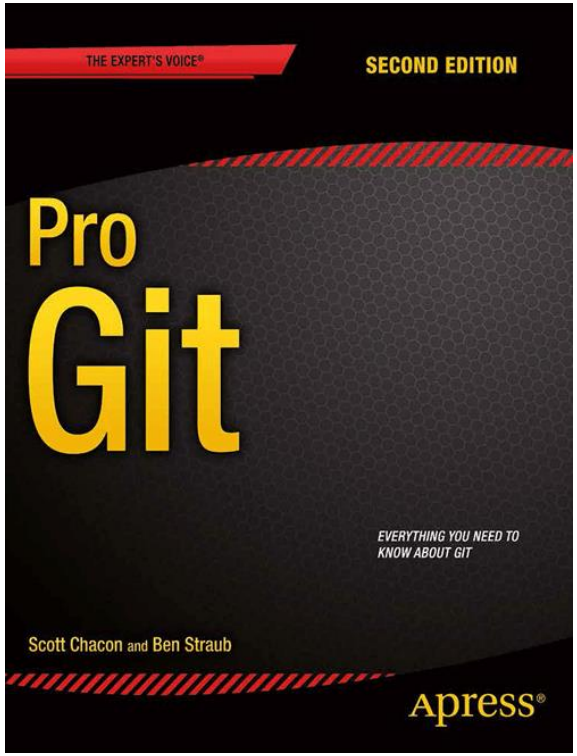
Způsoby verzování a sdílení

- Nic
- diplomka_v1.doc, diplomka_v2.doc, diplomka_s_jinym_uvodem.doc, diplomka_bez_fotky_muflona.doc, ...
- Flashdisk
- Cloudové úložiště
  - Dropbox, Google Drive, OneDrive, ...

- Centralizované verzovací systémy
  - CVS, Subversion (SVN), ...
- Distribuované verzovací systémy
  - Git, Mercurial, ...

- Scott Chacon, Ben Straub: Pro Git
- https://git-scm.com/book/en/v2

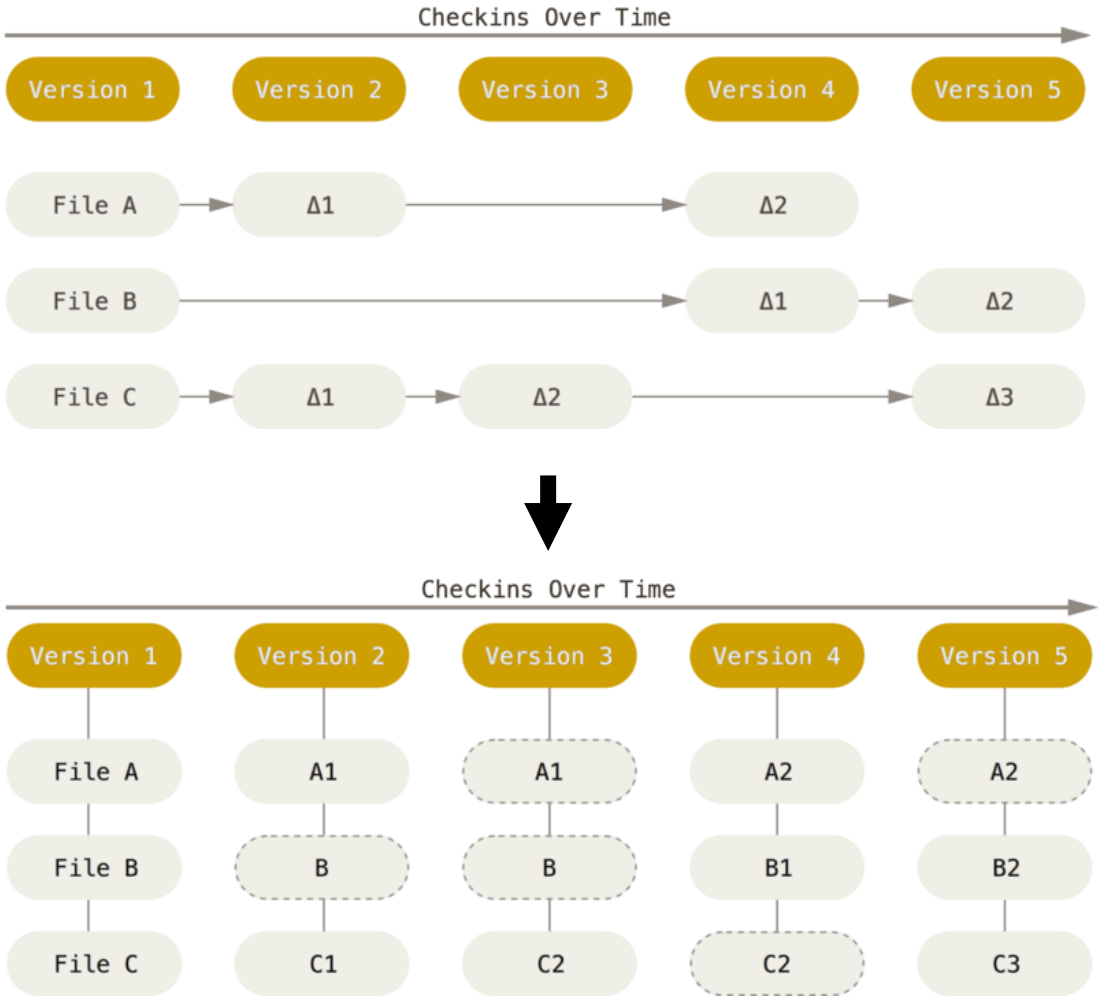




- https://git-scm.com/
- yum install git
- apt-get install git

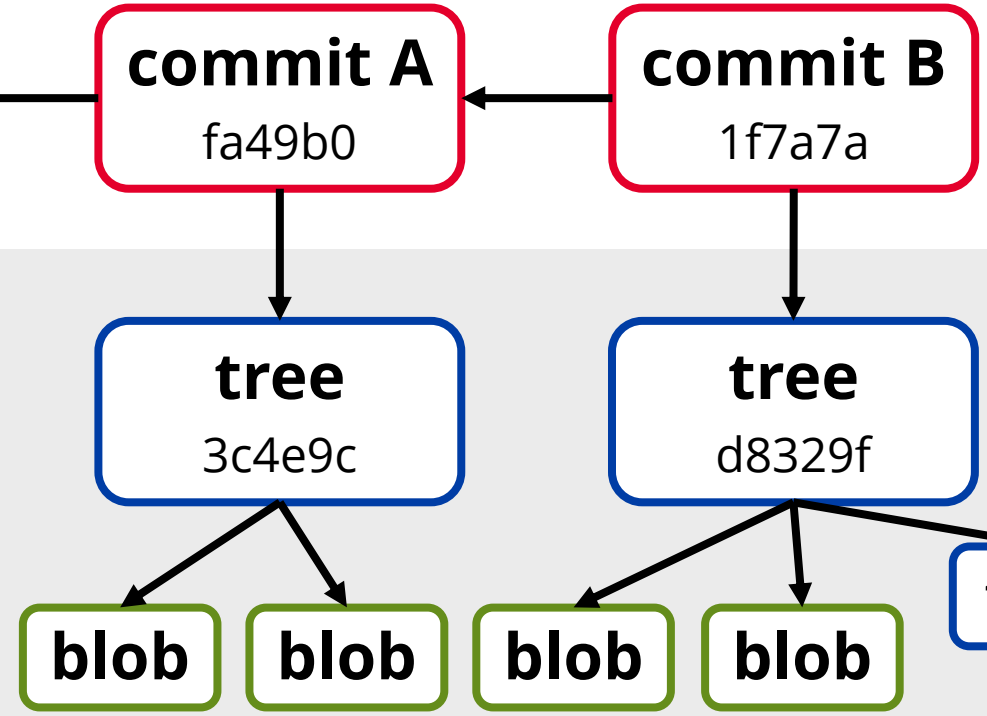
- Distribuovaný verzovací systém
- Vše lokálně, lze pracovat i ze zaoceánského parníku
- Neexistuje centrální repozitář, každý vývojář má úplnou kopii (a tedy i zálohu)
  - velké soubory → Git Large File Storage
- Ignoruje prázdné složky!
- Nepoužívá inkrementy, ale snapshoty

Inkrementy vs snapshoty:



Struktura repozitáře

- Vše, co potřebuje git k životu, je ve složce .git
- Struktura podobná filesystému
- Vše identifikováno kontrolním součtem obsahu (SHA-1)



blob ~ soubor
strom (tree) ~ adresář
revize (commit) ~ speciální adresář

Základní příkazy

- Vytvoření repozitáře:
  - **git init:** nový lokální
  - **git clone:** klon vzdáleného
- Kontrola stavu
  - **git status**
  - **git diff**
  - **git log**
- Vyčkávací prostor a revize:
  - **git add:** přidání souboru do vyčkávacího prostoru
  - **git mv:** přesun souboru
  - **git rm [--cached]:** smazání souboru [jen z vyčkávacího prostoru]
  - **git commit [--amend] -m "msg":** vytvoření commitu [oprava]
- Dále:
  - **git help [příkaz]**
  - **git config**
  - **git gui**

Vytvoření repozitáře

git init

- vytvoří lokální repozitář v aktuální složce
- metadata ve složce .git
- pro pracovní repozitáře

git init --bare

- vytvoří repozitář bez pracovní kopie
- metadata přímo ve složce repozitáře
- pro sdílené repozitáře

git clone <remote_repo>

- naklonuje vzdálený repozitář

git status

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
        new file: soubor-přidaný-přes-git-add
        deleted: soubor-smazán-přes-git-rm

Changed but not updated:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
        modified: upravený-soubor
        deleted: soubor-smazán-obyčejným-rm

Untracked files:
  (use "git add <file>..." to include in what will be committed)
nesledovaný-soubor
```

git status

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
        new file: soubor-přidaný-přes-git-add
        deleted: soubor-smazán-přes-git-rm

Changed but not updated:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
        modified: upravený-soubor
        deleted: soubor-smazán-obyčejným-rm

Untracked files:
  (use "git add <file>..." to include in what will be committed)
nesledovaný-soubor
```

git status

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
        new file: soubor-přidaný-přes-git-add
        deleted: soubor-smazán-přes-git-rm

Changed but not updated:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
        modified: upravený-soubor
        deleted: soubor-smazán-obyčejným-rm

Untracked files:
  (use "git add <file>..." to include in what will be committed)
nesledovaný-soubor
```

git status

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
        new file: soubor-přidaný-přes-git-add
        deleted: soubor-smazán-přes-git-rm

Changed but not updated:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
        modified: upravený-soubor
        deleted: soubor-smazán-obyčejným-rm

Untracked files:
  (use "git add <file>..." to include in what will be committed)
nesledovaný-soubor
```

git diff

- změny, které ještě nejsou ve vyčkávacím prostoru
- neobsahuje nesledované soubory

git diff --cached

- vyčkávací prostor vs. poslední commit

git log

```
commit 1d33b1b1b7530a1168d7c3adc44063bfe32592eb
Author: Random <jrh@example.net>
Date: Sat Sep 12 15:52:20 2009 -0400

    třetí commit

commit 895740c342fdbc03ce26b6417051ab6f2188a6d0
Author: Random <jrh@example.net>
Date: Sat Sep 12 14:53:20 2009 -0400

    druhý commit

commit db35c00c63a7e98c4a89e180d317c9fb08e40f4b
Author: Random <jrh@example.net>
Date: Sat Sep 12 10:08:20 2009 -0400

    první commit!
```

git log --stat

```
commit 1d33b1b1b7530a1168d7c3adc44063bfe32592eb
Author: Random <jrh@example.net>
Date: Sat Sep 12 15:52:20 2009 -0400

    třetí commit

 test | 15 ++++++++++++---
 1 files changed, 12 insertions(+), 3 deletions(-)
```

git log --oneline

```
1d33b1b třetí commit
895740c druhý commit
db35c00 první commit!
```

git add <soubor/složka>

- přesun změn do vyčkávacího prostoru

git mv <from> <to>

- přejmenování složky či souboru

git rm <soubor/složka>

- odstranění z disku a označení, že má být v commitu odstraněno

git rm --cached <soubor/složka>

- pouze označení, že má být v commitu odstraněno

Tvorba revizí

- Commit je jedna revize
- Identifikován hashem (SHA-1)
  - **bf2ca58**1680417f466fbedf35f1a4aa1c4c6c5e9
- Obsahuje zejména zprávu, autora a datum a čas
- Častá konvence pro zprávy:
  - první řádek = stručné shrnutí
    - je vidět ve všech výpisech historie
  - druhý řádek prázdný
  - další řádky = podrobnější informace, např:
    - proč se dělá změna
    - jak je to vyřešeno
    - vedlejší efekty

git commit [-m message]

- Pokud se nezadá zpráva, git spustí textový editor
  - $VISUAL, $EDITOR, vi

git commit --amend [-m message]

- Oprava předchozího commitu → sloučení změn

git commit -a [-m message]

- všechny soubory, i ty mimo vyčkávací prostor

git commit <soubor> [-m message]

- pouze vybraný soubor

git config <key> <value>

- konfigurace repozitáře, má přednost před globální

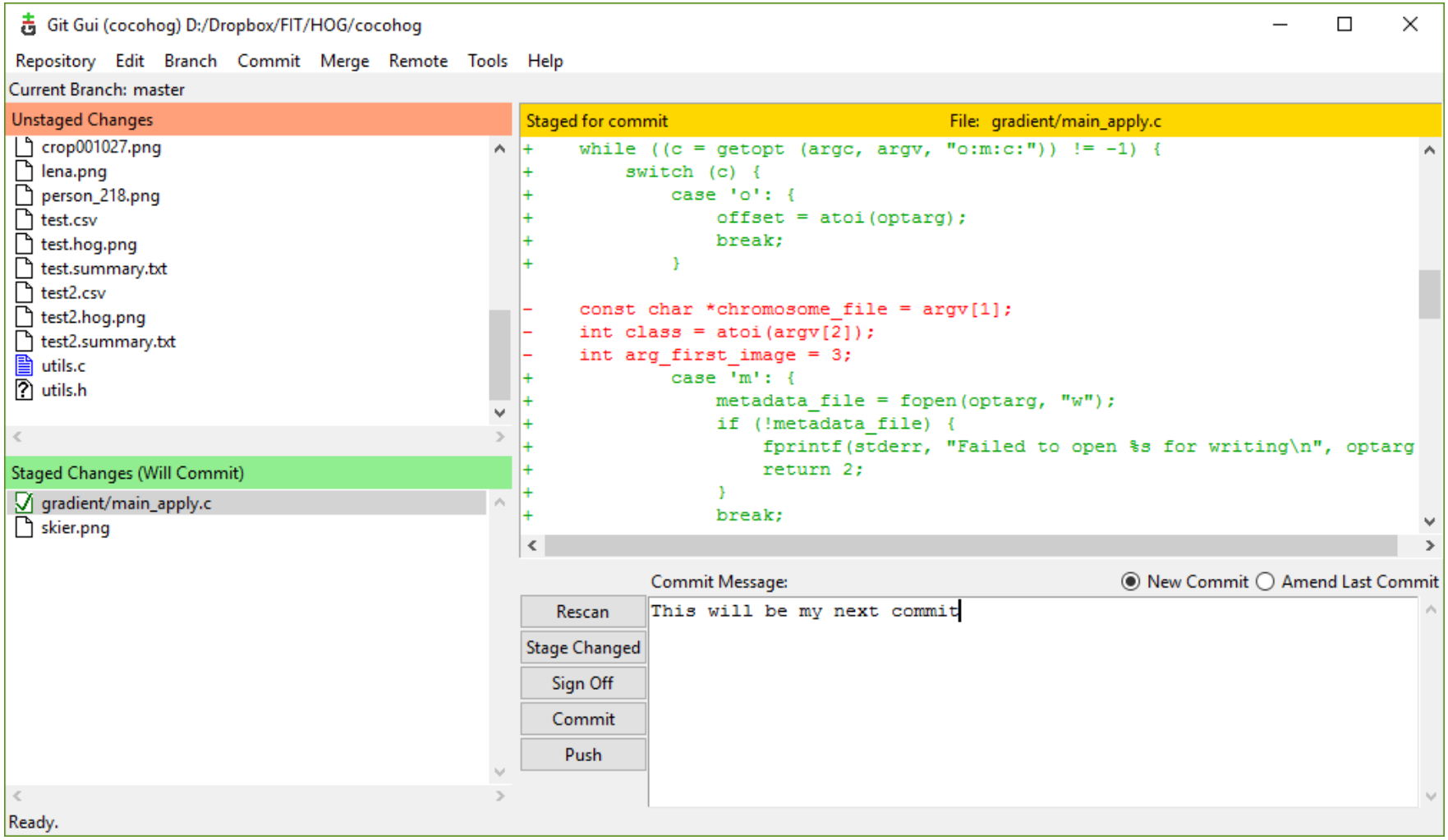
git config <key> <value>

- globální konfigurace

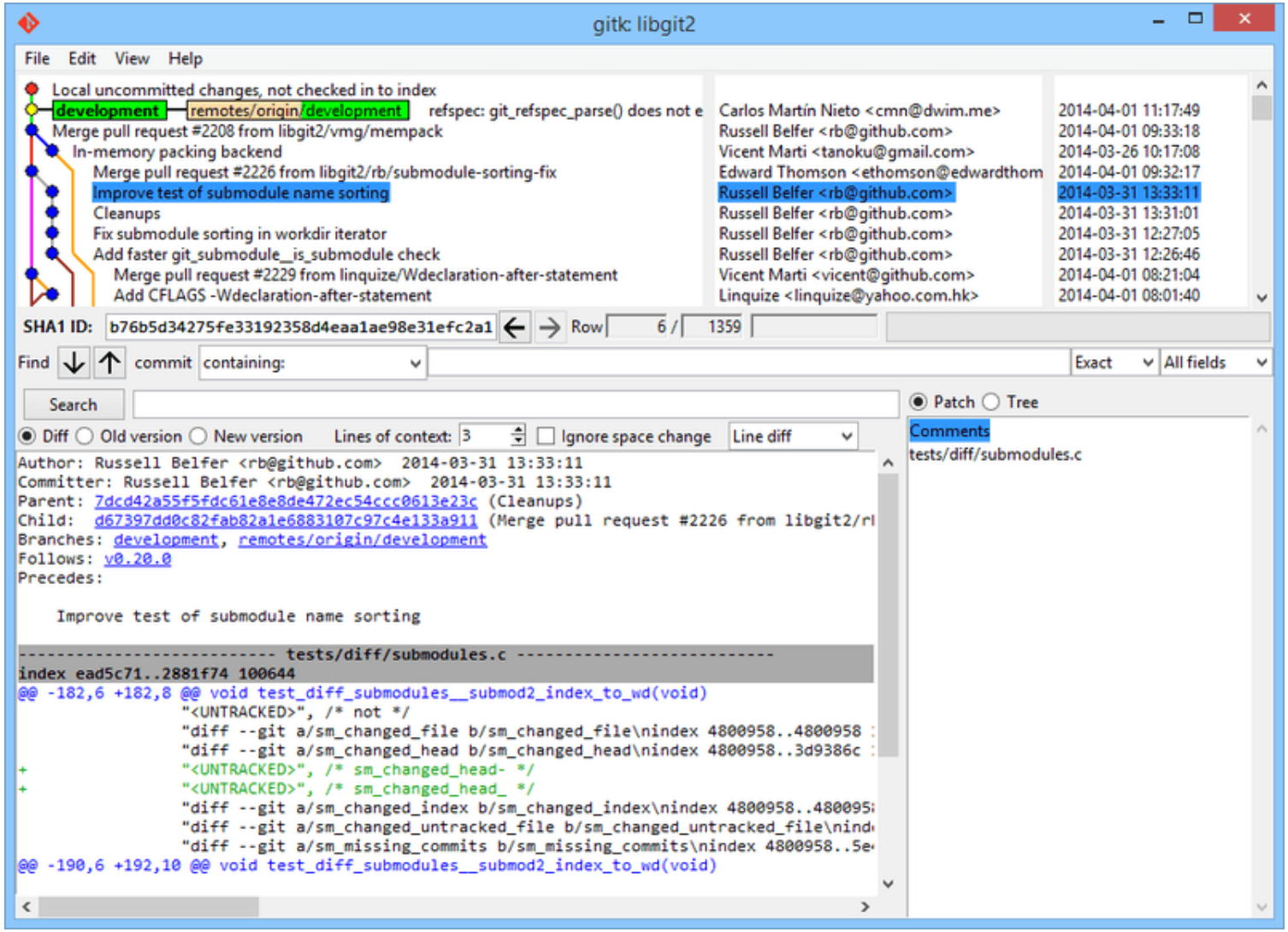
Užitečná nastavení:

```
git config --global user.name "Jan Novák"
git config --global user.email "jan.novak@firma.cz"
git config --global color.ui auto
```

git gui


Git Gui (cocohog) D:/Dropbox/FIT/HOG/cocohog
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: master
Unstaged Changes
crop001027.png
lena.png
person_218.png
test.csv
test.hog.png
test.summary.txt
test2.csv
test2.hog.png
test2.summary.txt
utils.c
utils.h
Staged Changes (Will Commit)
gradient/main_apply.c
skier.png
Staged for commit
File: gradient/main_apply.c
+ while ((c = getopt (argc, argv, "o:m:c:")) != -1) {
+ switch (c) {
+ case 'o': {
+ offset = atoi(optarg);
+ break;
+ }
- const char *chromosome_file = argv[1];
- int class = atoi(argv[2]);
- int arg_first_image = 3;
+ case 'm': {
+ metadata_file = fopen(optarg, "w");
+ if (!metadata_file) {
+ fprintf(stderr, "Failed to open %s for writing\n", optarg
+ return 2;
+ }
+ break;
Commit Message:
New Commit
Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push
This will be my next commit
Ready.

gitk


gitk: libgit2
File Edit View Help
Local uncommitted changes, not checked in to index
development
remotes/origin/development
refspec: git_refspec_parse() does not e
Merge pull request #2208 from libgit2/vmg/mempack
In-memory packing backend
Merge pull request #2226 from libgit2/rb/submodule-sorting-fix
Improve test of submodule name sorting
Cleanups
Fix submodule sorting in workdir iterator
Add faster git_submodule__is_submodule check
Merge pull request #2229 from linquize/Wdeclaration-after-statement
Add CFLAGS -Wdeclaration-after-statement
Carlos Martín Nieto <cmn@dwim.me> 2014-04-01 11:17:49
Russell Belfer <rb@github.com> 2014-04-01 09:33:18
Vicent Marti <tanoku@gmail.com> 2014-03-26 10:17:08
Edward Thomson <ethomson@edwardthom 2014-04-01 09:32:17
Russell Belfer <rb@github.com> 2014-03-31 13:33:11
Russell Belfer <rb@github.com> 2014-03-31 13:31:01
Russell Belfer <rb@github.com> 2014-03-31 12:27:05
Russell Belfer <rb@github.com> 2014-03-31 12:26:46
Vicent Marti <vicent@github.com> 2014-04-01 08:21:04
Linquize <linquize@yahoo.com.hk> 2014-04-01 08:01:40
SHA1 ID: b76b5d34275fe33192358d4eaa1ae98e31efc2a1
Row 6 / 1359
Find commit containing:
Exact
All fields
Search
Patch Tree
Diff Old version New version Lines of context: 3 Ignore space change Line diff
Comments
tests/diff/submodules.c
Author: Russell Belfer <rb@github.com> 2014-03-31 13:33:11
Committer: Russell Belfer <rb@github.com> 2014-03-31 13:33:11
Parent: 7dcd42a55f5fdc61e8e8de472ec54ccc0613e23c (Cleanups)
Child: d67397dd0c82fab82a1e6883107c97c4e133a911 (Merge pull request #2226 from libgit2/rl
Branches: development, remotes/origin/development
Follows: v0.20.0
Precedes:
Improve test of submodule name sorting
tests/diff/submodules.c
index ead5c71..2881f74 100644
@@ -182,6 +182,8 @@ void test_diff_submodules__submod2_index_to_wd(void)
"<UNTRACKED>", /* not */
"diff --git a/sm_changed_file b/sm_changed_file\nindex 4800958..4800958
"diff --git a/sm_changed_head b/sm_changed_head\nindex 4800958..3d9386c
+ "<UNTRACKED>", /* sm_changed_head- */
+ "<UNTRACKED>", /* sm_changed_head_ */
"diff --git a/sm_changed_index b/sm_changed_index\nindex 4800958..480095
"diff --git a/sm_changed_untracked_file b/sm_changed_untracked_file\nind
"diff --git a/sm_missing_commits b/sm_missing_commits\nindex 4800958..5e
@@ -190,6 +192,10 @@ void test_diff_submodules__submod2_index_to_wd(void)

Naučme se git za 5 minut

1. Založím lokální repozitář
   - mkdir mujprogram && cd mujprogram
   - **git init**
2. Vytvořím první revizi (commit)
   - **git commit** -m "initial commit"
3. Tvořím
   - vim main.c / emacs main.c / subl main.c
   - **git add / git rm / git mv**
4. Vytvořím další revizi (commit)
   - **git commit** -m "add main.c"

V praxi to je ale často takto:

1. Tvořím
   - mkdir mujprogram && cd mujprogram
   - vim main.c / emacs main.c / subl main.c
2. Nějak to roste, možná bych to měl verzovat
   - **git init**
   - **git commit** -m "initial commit„
3. Možná bych to měl dát na GitHub
   - git remote add origin https://github.com/...
   - git push -u origin master

Základní pravidla

- Nastavte si své jméno a email
- Každá revize je ucelený balík změn řešící jednu věc
  - 1 oprava, 1 nová funkce, …
  - Raději více menších než jedna bomba („code bomb“)
- Pište k revizím smysluplné zprávy
  - **NE:**
    - fixes, stuff, bflmpsvz
    - vulgarismy
    - Updated main.c
  - **ANO:**
    - Fixed blue bar not disappearing after midnight (closes #123)
    - Allow to set predictor size for baldwin mode

Co je dobré mít v repozitáři

- Konfigurace nástrojů
  - statická analýza (linter)
  - testovací framework
  - překladač (např. Makefile)
  - …
- README
- .editorconfig
  - nastavení konců řádků a odsazování
- .gitignore
  - seznam ignorovaných souborů a složek

Co je dobré nemít v repozitáři

Ne vše je vhodné verzovat, například:

- vše, co lze vygenerovat
  - binárky
  - generovaná dokumentace
  - vysázené PDF z LaTeXu
- lokální konfigurace (specifická pro jednoho vývojáře)
  - nastavení IDE
  - přístupy k testovací databázi
- dočasné soubory
- logy

- Jméno projektu
- Autoři
- Stručný popis funkce
- Licence
- Postup instalace
- Jak přispívat (hlavně u open source)
- A cokoliv dalšího, co dává smysl

- Dá se najít spousta šablon a příkladů:
  - https://github.com/matiassingers/awesome-readme

- Seznam složek a souborů, které se nemají verzovat
- https://github.com/github/gitignore

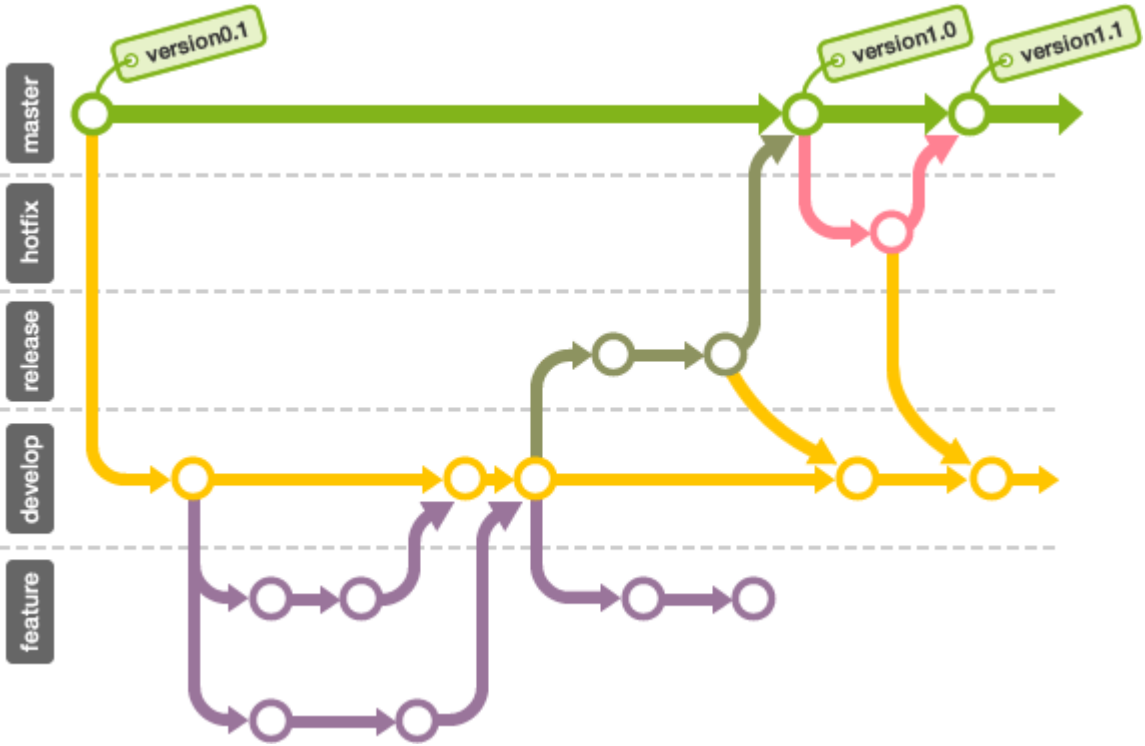
```
# ignoruj soubory .a
*.a

# ale s vyjimkou lib.a
!lib.a

# ignoruj soubor TODO v teto slozce, ale ne v podslozkach (./xyz/TODO)
/TODO

# ignoruj vse ve slozce build/
build/

# ignoruj doc/notes.txt, ale ne doc/server/arch.txt
doc/*.txt

# ignoruj vsechna pdf ve slozce doc/ a jejich podslozkach
doc/**/*.pdf
```

To nejdůležitější!

VĚTVENÍ A TAGY

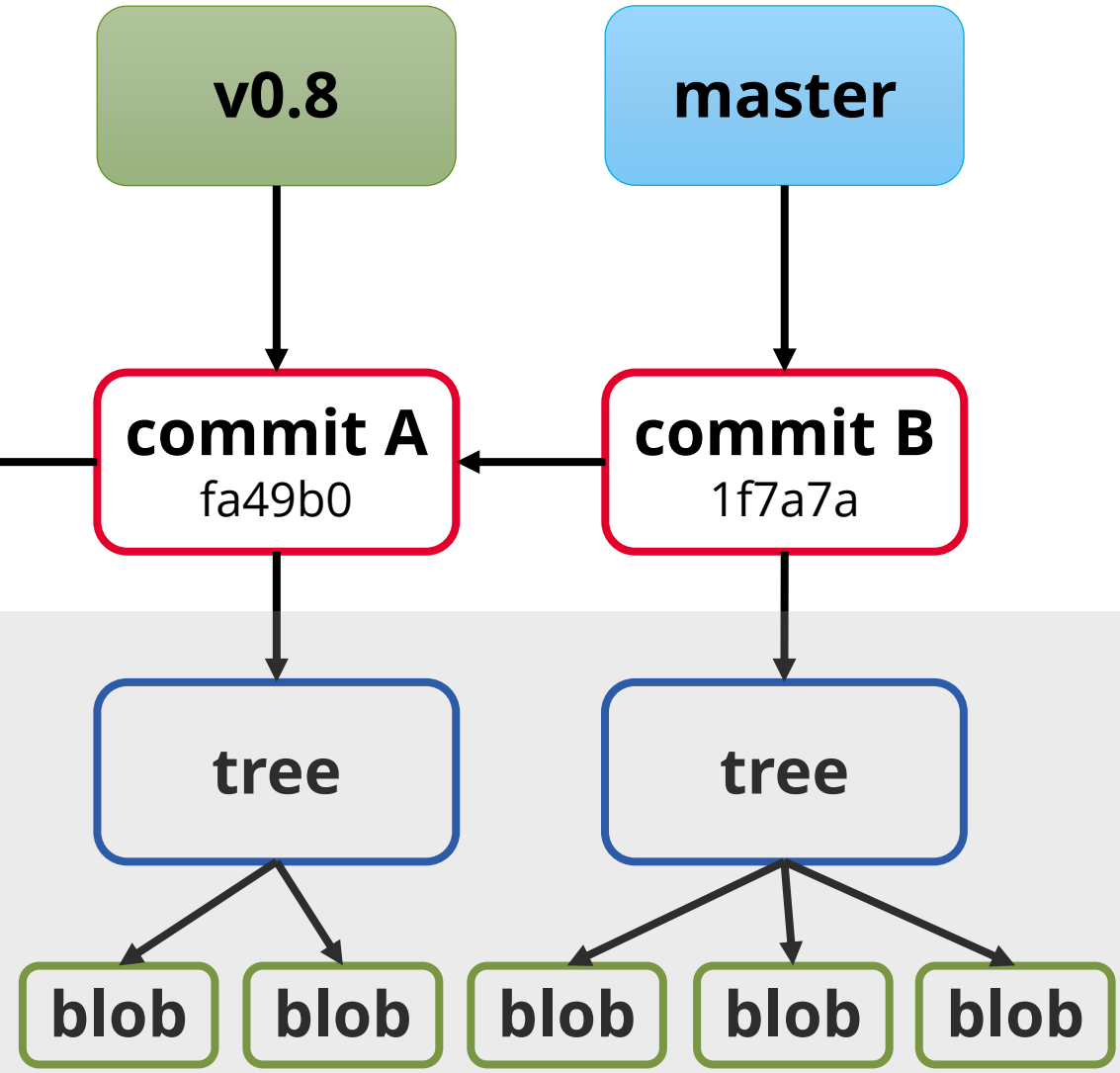
Větve a tagy

- Větev
  - Odbočka od hlavní linie vývoje
  - Opravy chyb, nové funkce, experimenty…
- Tag
  - Speciální pojmenování pro určitou revizi



V gitu **velmi** rychlé!

Větve a tagy



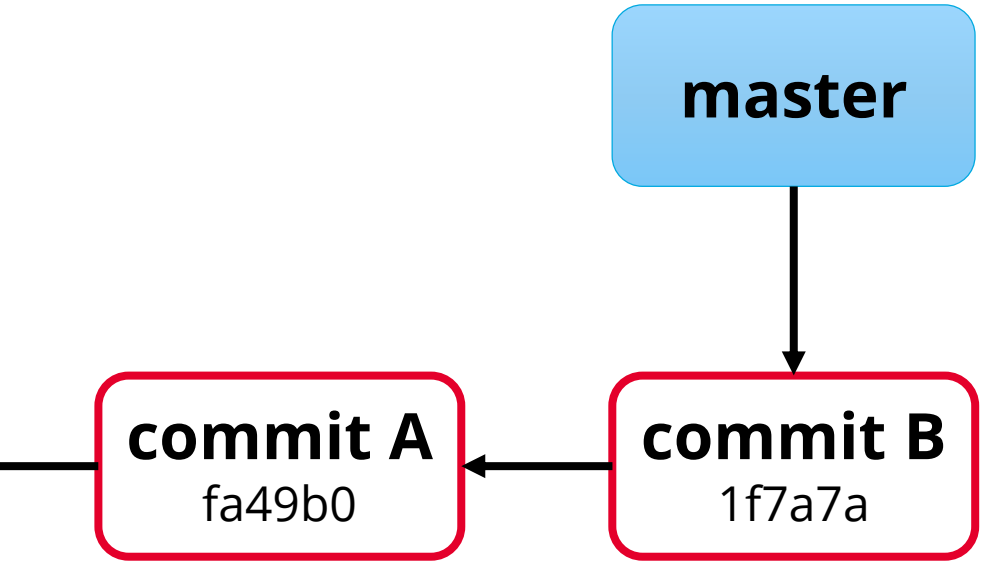
Větve a tagy jsou jen ukazatele na konkrétní revizi

Příkazy pro větve a tagy

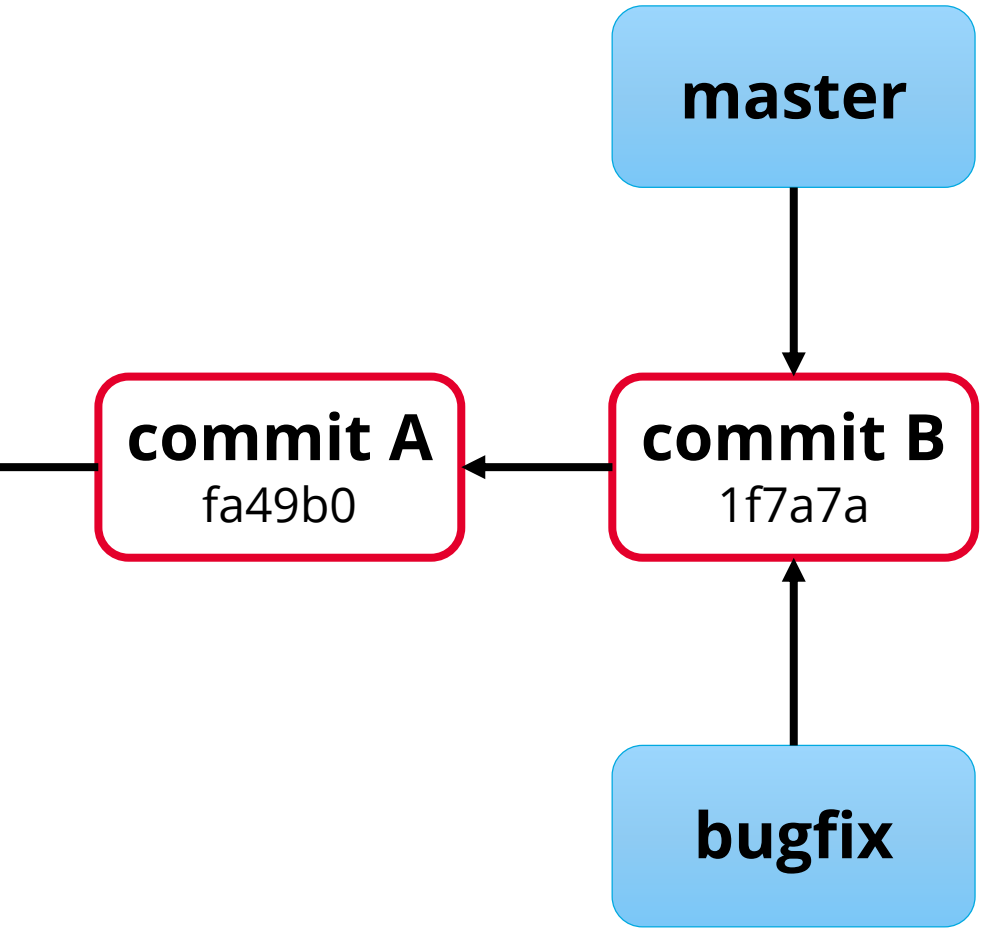
- **git branch:** výpis větví
- **git branch XYZ:** vytvoř větev XYZ
- **git checkout XYZ:** přepni na XYZ
- **git checkout -b XYZ:** vytvoř větev XYZ a přepni
- **git merge:** slučování
- **git rebase:** přeskládání
- **git reset [--hard | --soft]:**

- **git tag:** výpis tagů
- **git tag -m "v0.8":** vytvoření tagu

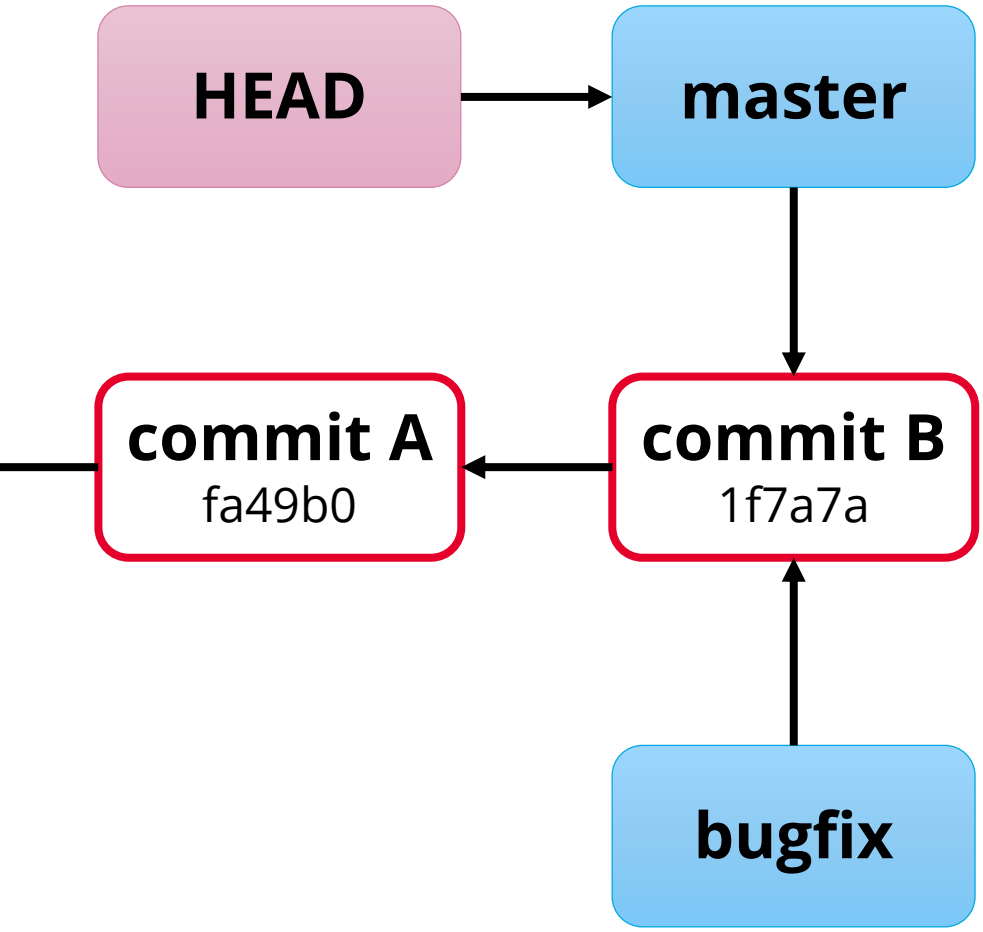
```
$ git branch bugfix
```



```
$ git branch bugfix
```

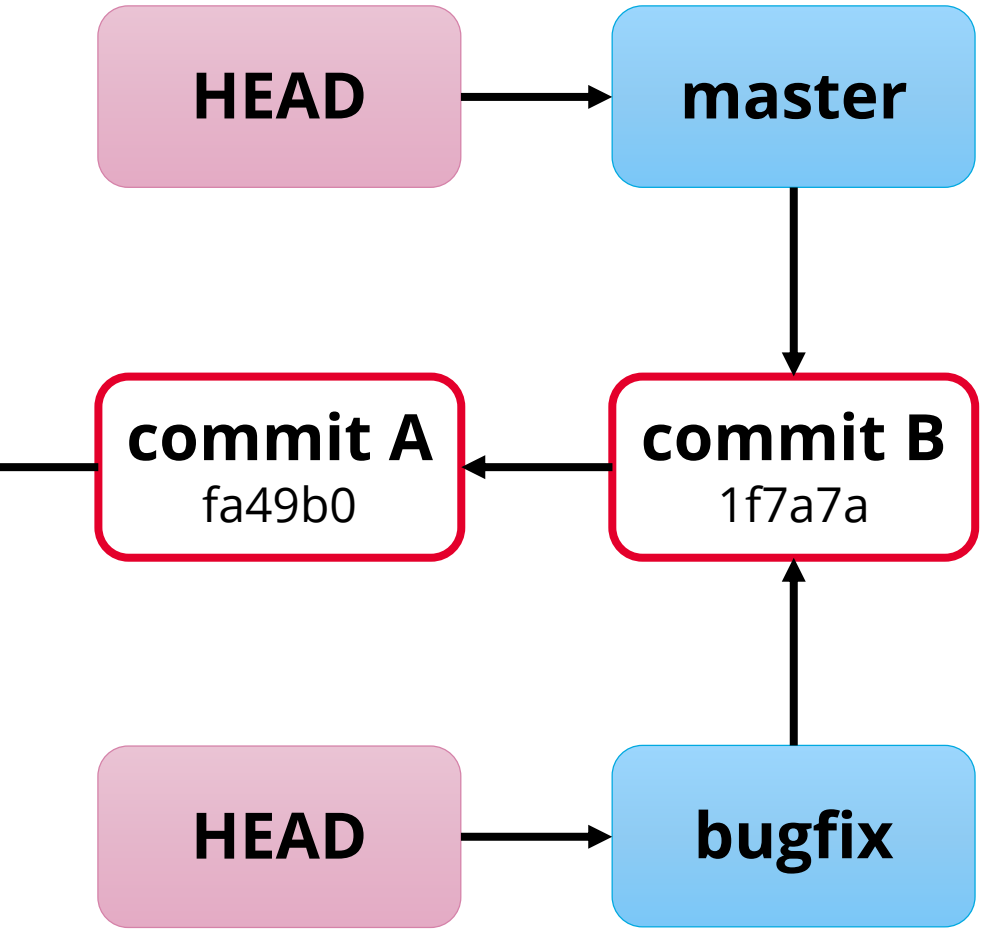


Jak ale git ví,
která větev je aktivní?



Ukazatel HEAD

Může ukazovat i na konkrétní commit (*detached head*, něco jako nepojmenovaná větev)

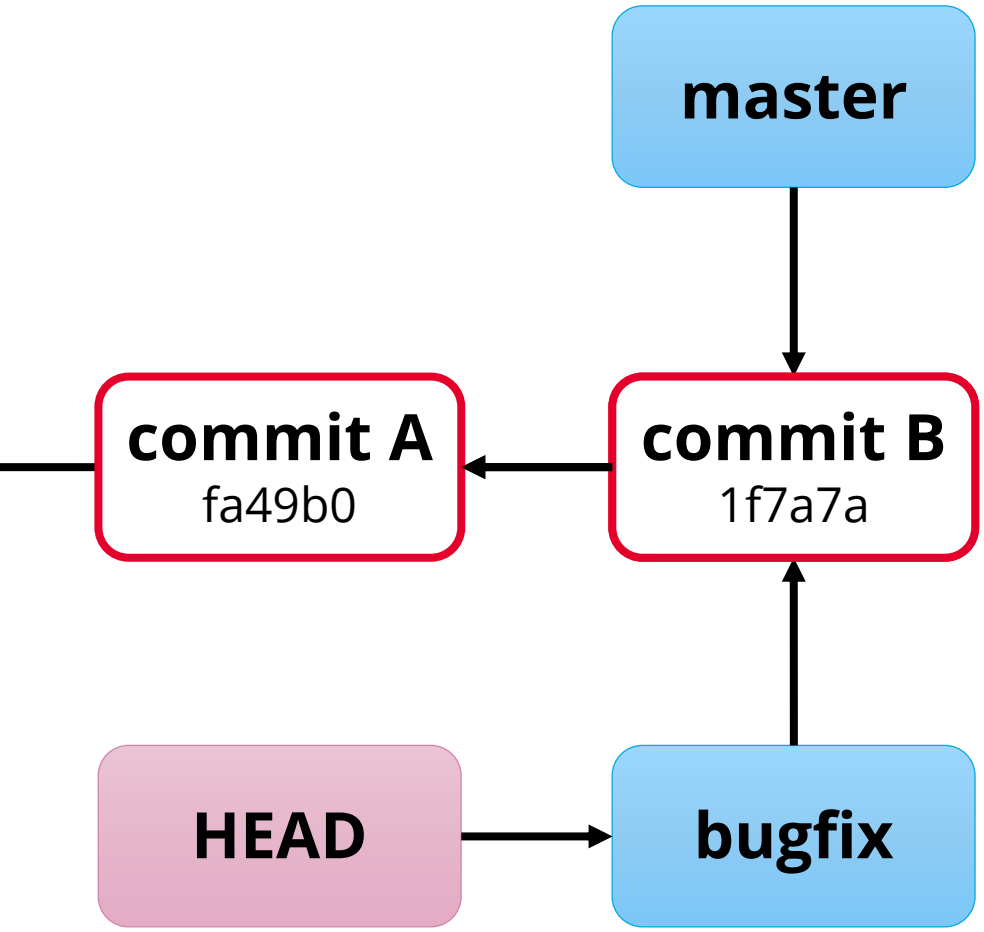



```
$ git checkout bugfix
```

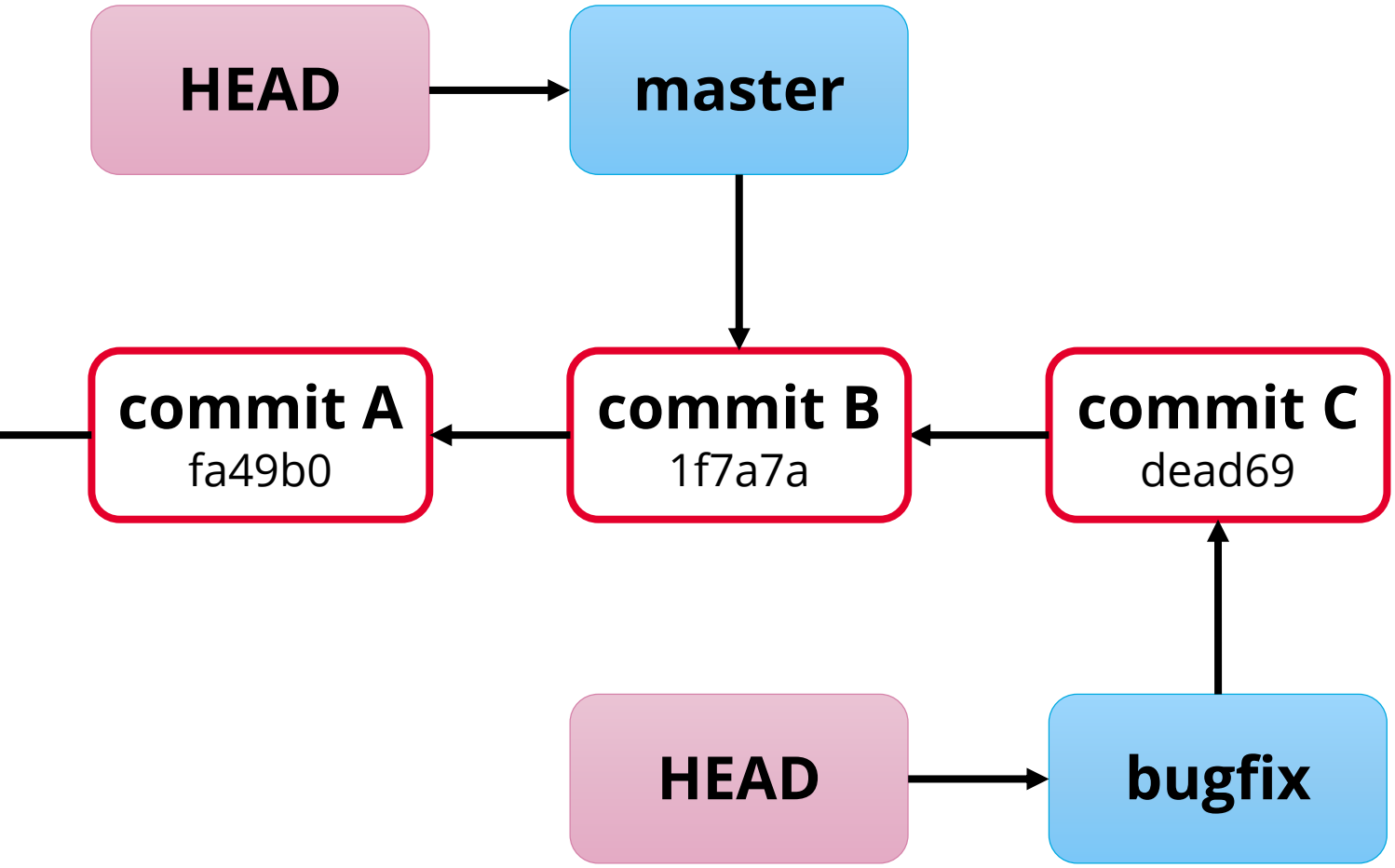
Vytvoření a přepnutí větve v jednom kroku:

```
$ git checkout -b bugfix
```

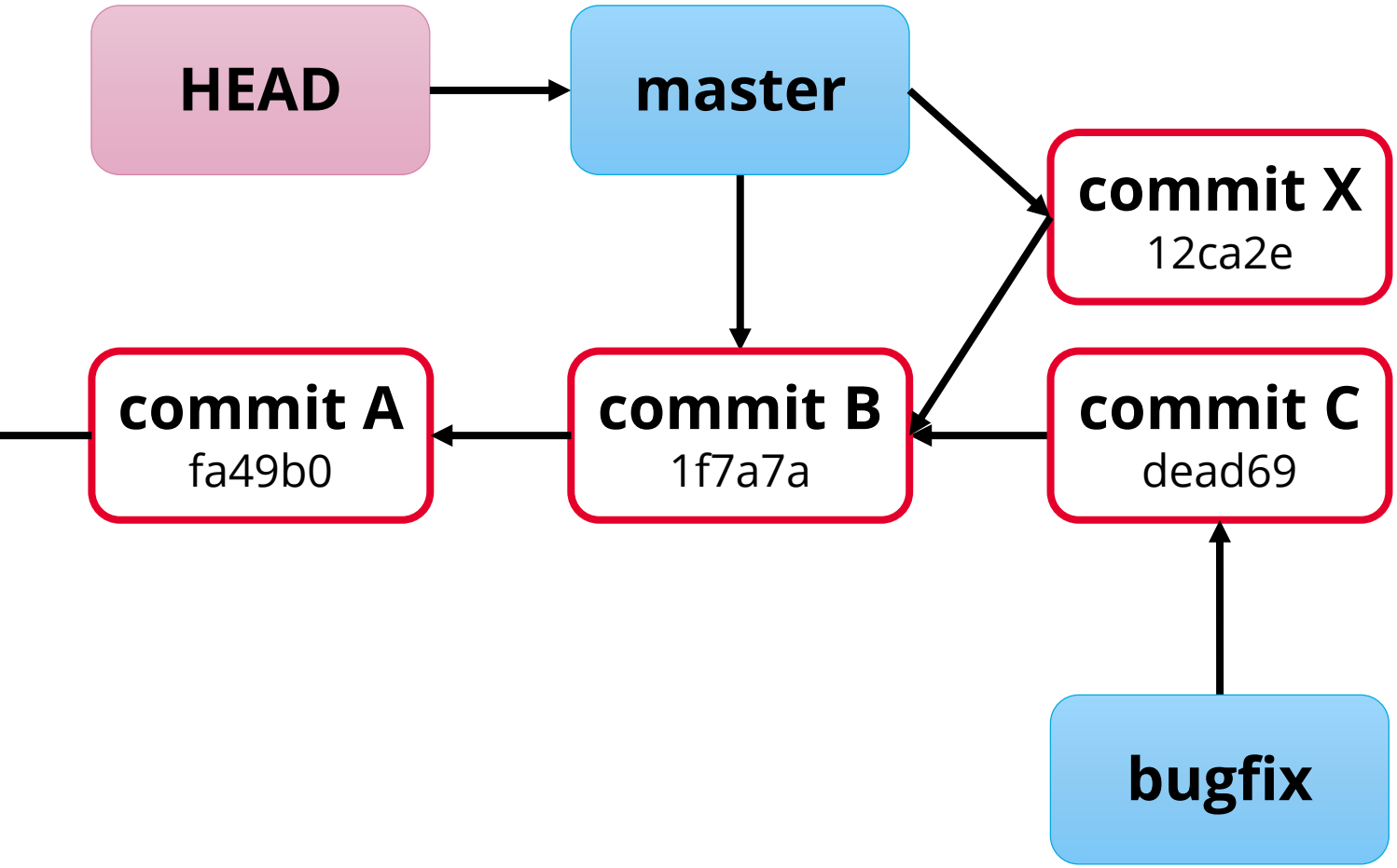
```
$ git commit -m "C"
```

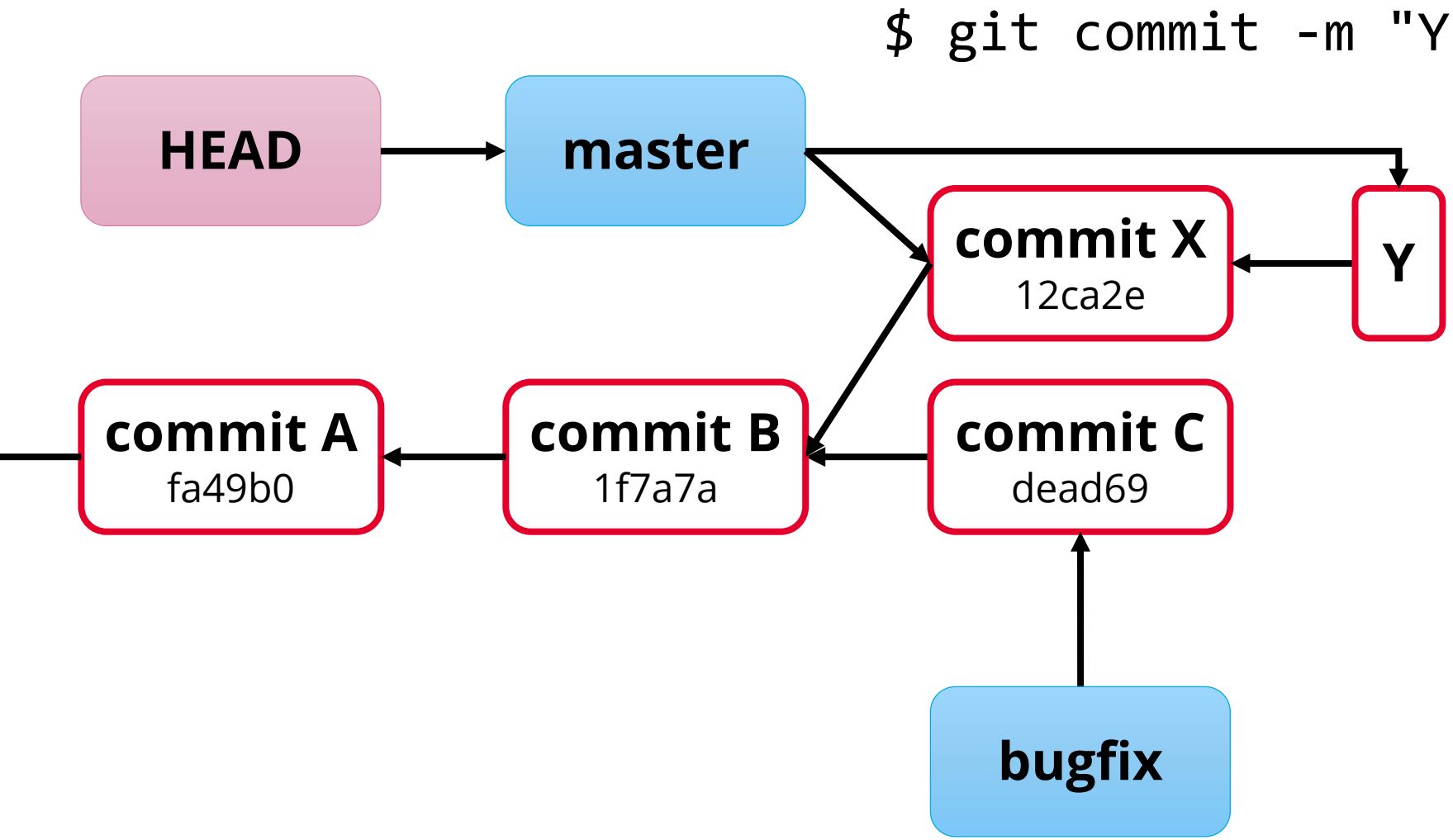


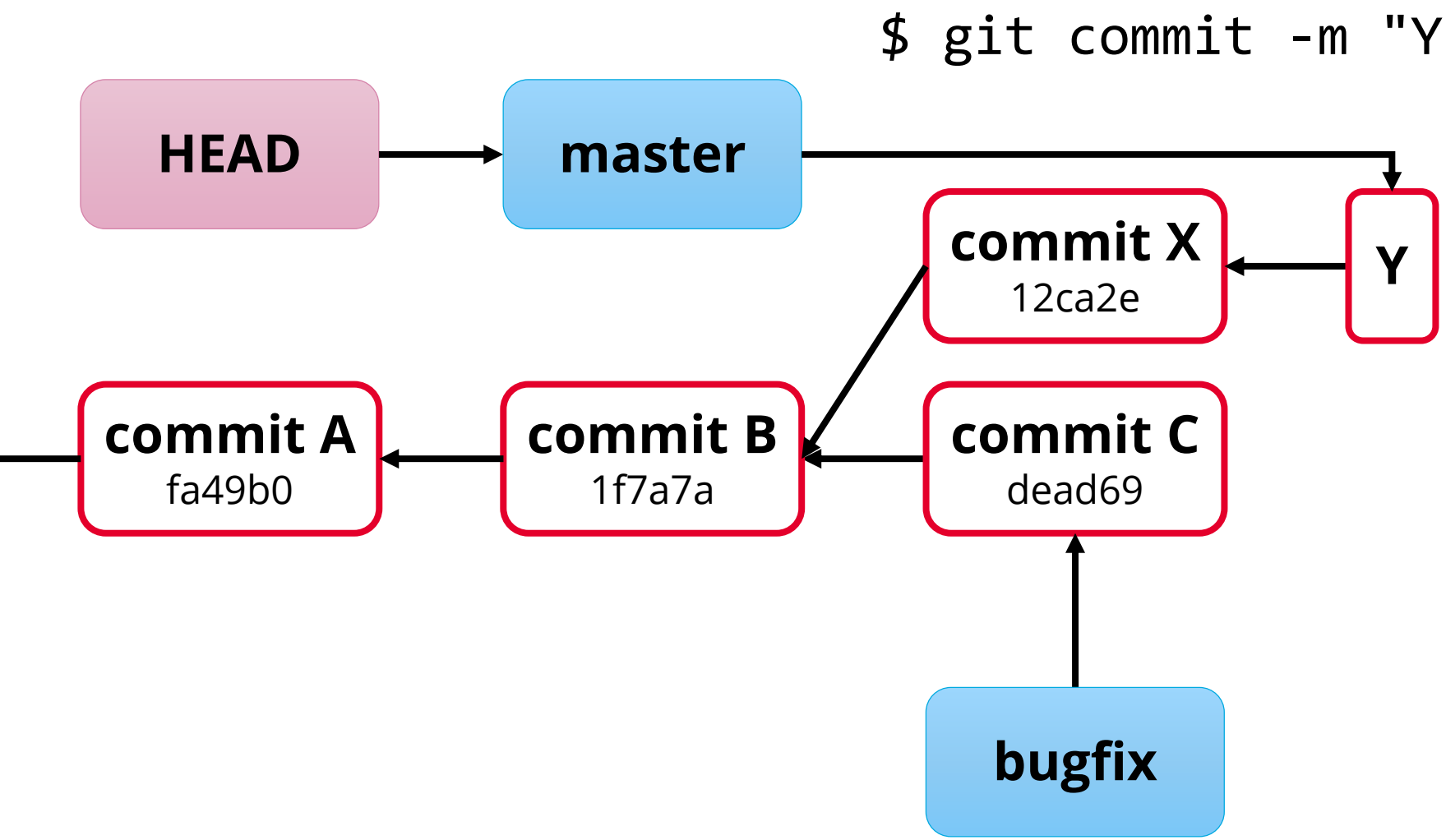
```
$ git checkout master
```

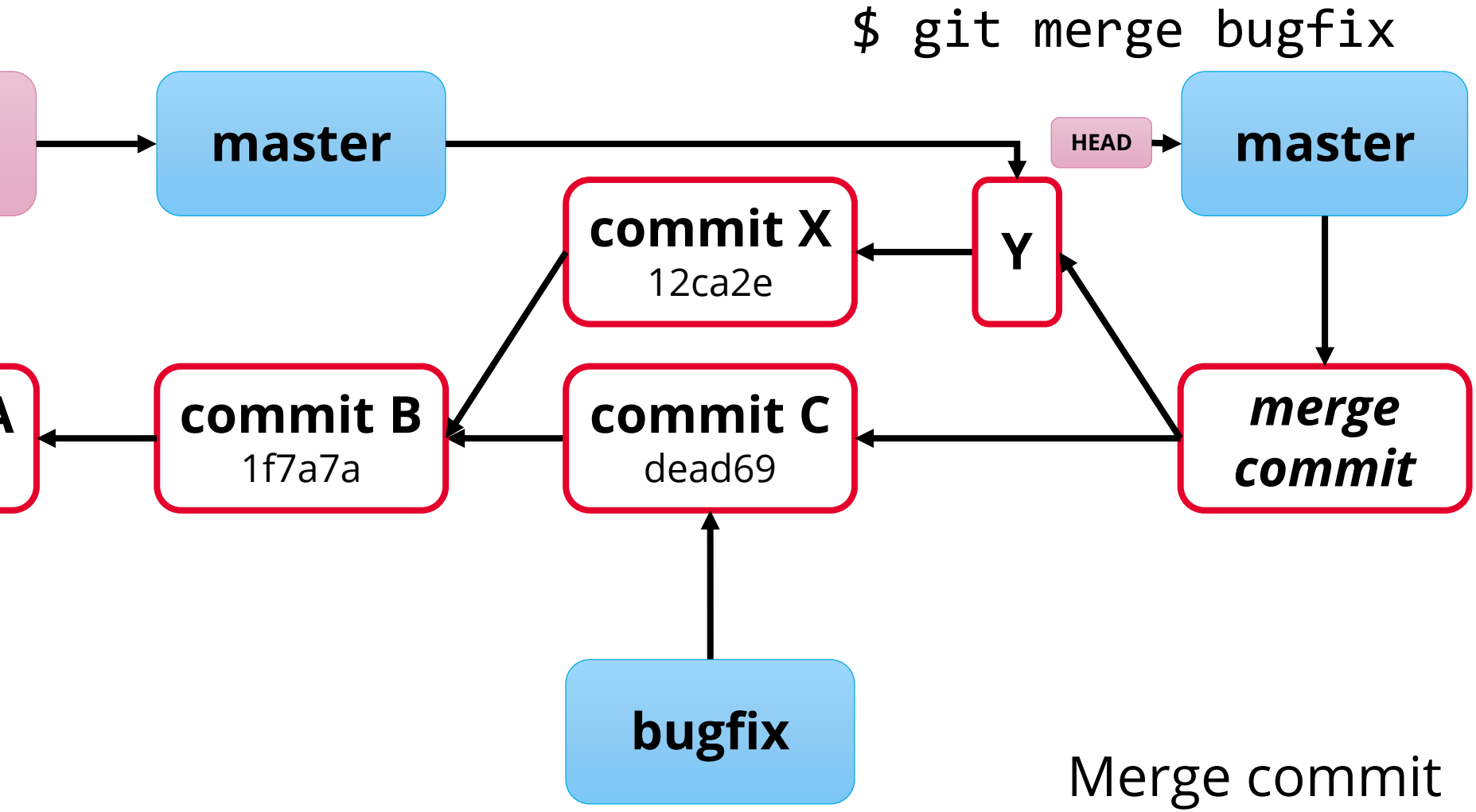


```
$ git commit -m "X"
```



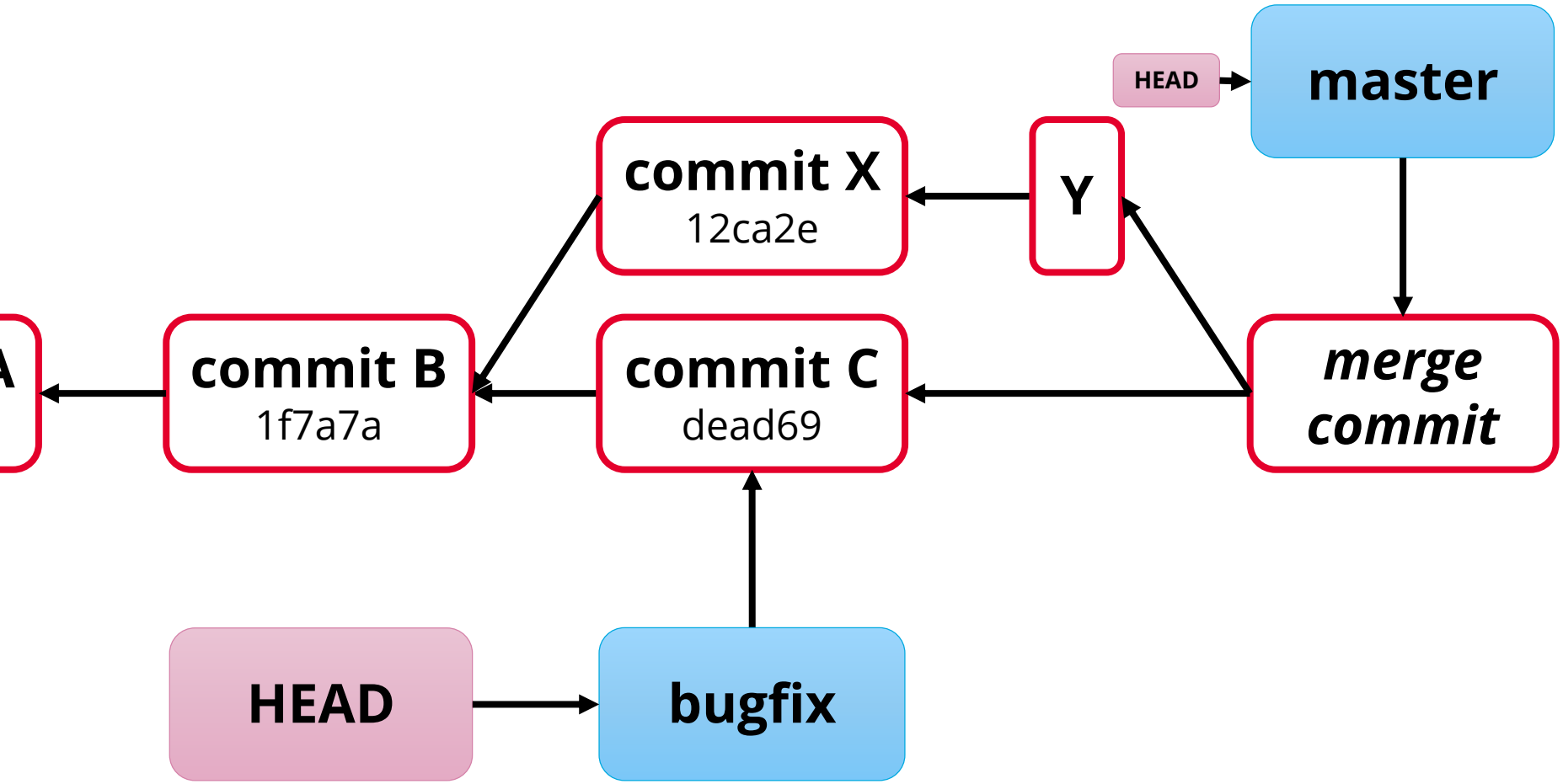

$ git commit -m "Y"
HEAD
master
commit X
12ca2e
Y
commit A
fa49b0
commit B
1f7a7a
commit C
dead69
bugfix


$ git commit -m "Y"
HEAD
master
commit X
12ca2e
Y
commit A
fa49b0
commit B
1f7a7a
commit C
dead69
bugfix

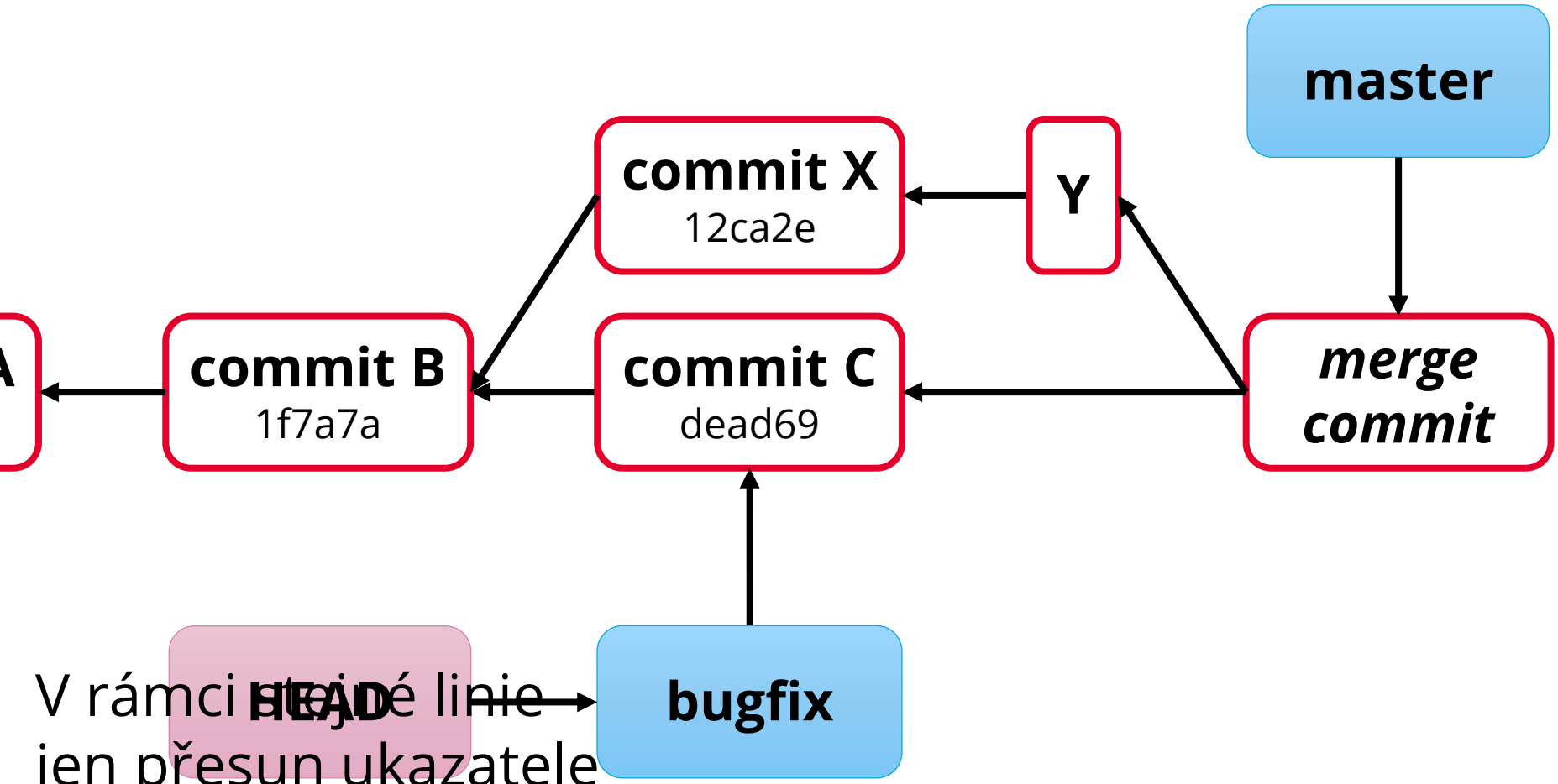


Merge commit má dva rodiče

```
$ git checkout bugfix
```



```
$ git merge master
```



V rámci stejné linie jen přesun ukazatele

Fast-forward

Fast-forward

- slučované větve jsou následovníky linii
- jen se přesune ukazatel

Merge commit

- slučované větve nejsou následovníky
- je třeba nalézt společného předka
- mohou vzniknout konflikty

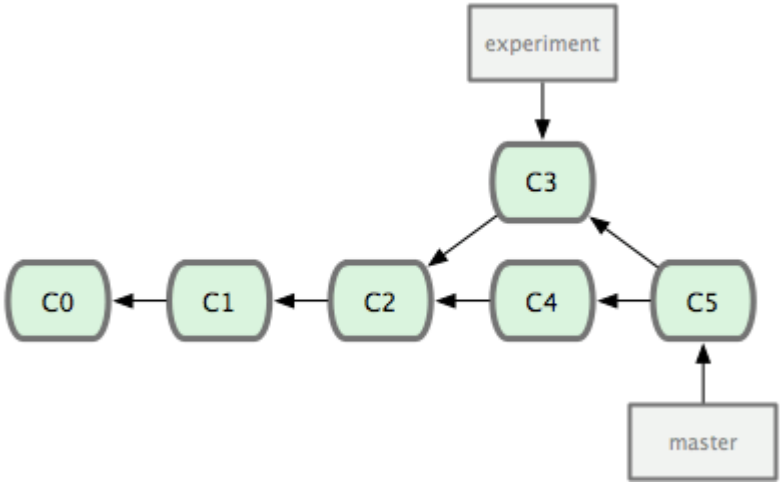
Používejte větve!

- Je to **velmi** rychlé
  - V praxi: přepnutí do 4 roky staré verze za cca minutu

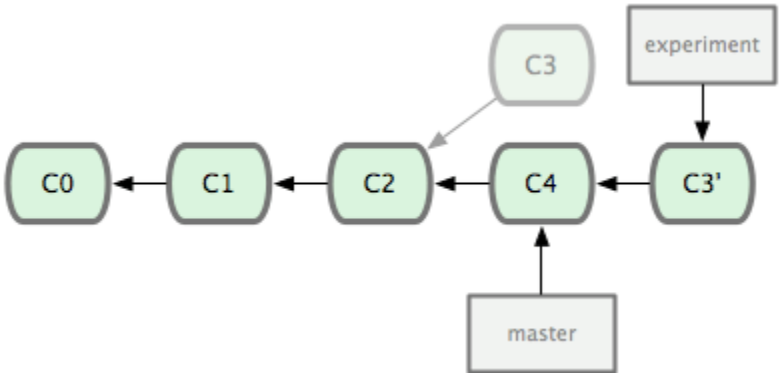
- Lze dělat více věcí najednou
  - pracuji ve větvi **development**
  - přijde urgentní požadavek – založím větev **issue123**
  - chci optimalizovat, ale nevím, jak to dopadne – založím větev **xyz-optimization**
  - našla se chyba ve staré verzi – ze staré verze založím větev **issue456**
    - git branch issue456 **v0.8**

git rebase (přeskládání)

- Tak trochu jiné slučování
- Aplikování změn v jedné větvi na vrcholu jiné
- Oproti slučování je výsledkem čistší historie
- Interaktivní režim – lze použít pro úpravu historie

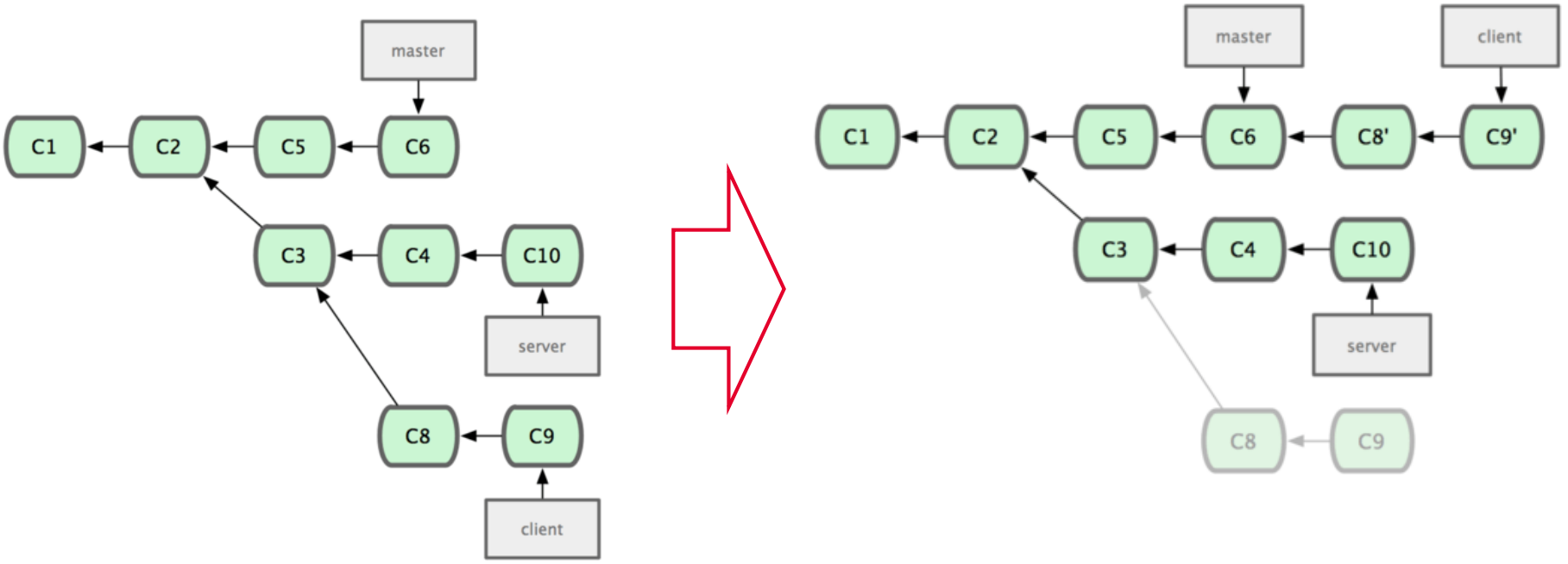


Slučování (merge)



Přeskládání (rebase)

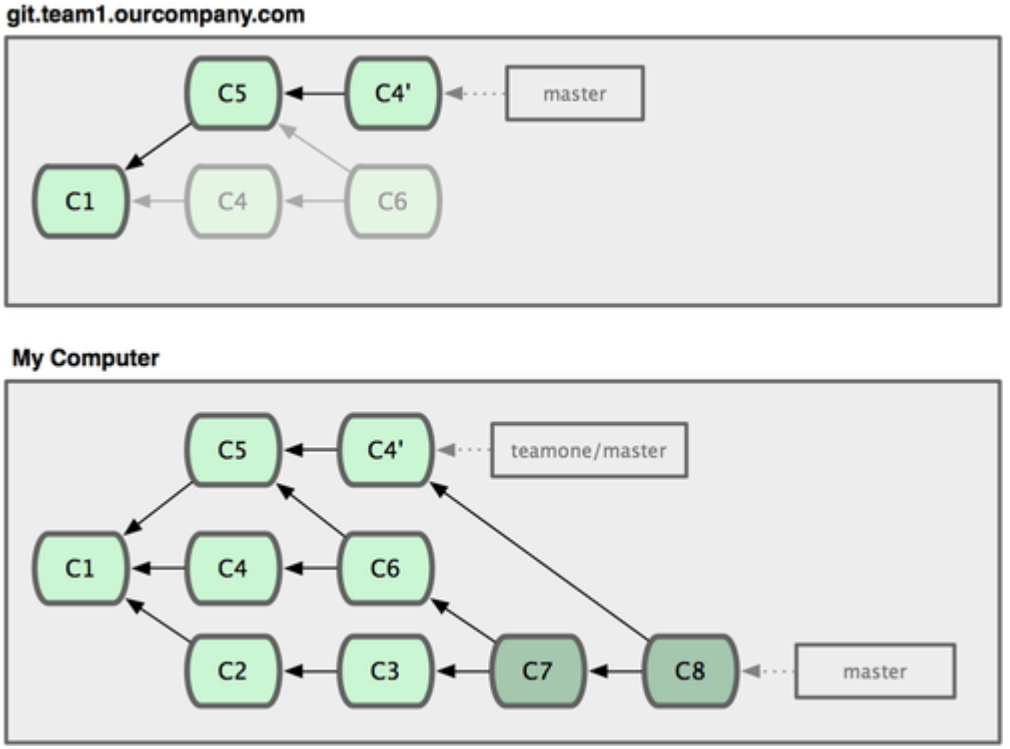
```
$ git rebase --onto master server client
```



Neprovádějte přeskládání u revizí, které jste odeslali do veřejného repozitáře.

Udělá to nepořádek v historii a kolegové vás nebudou mít rádi. Fakt!

Přeskládání používejte zejména pro pročištění historie před odesláním do veřejného repozitáře.



git checkout <kam>

Umí přepnout prakticky kamkoliv

- git checkout **branch**
- git checkout **a8d621** (vznikne stav *detached head*)
- git checkout **origin/master** (také *detached head)*

Ale také umí vrátit zpět změny

- git checkout **main.c**
- git checkout **subdir**
- git checkout **.**

git reset --soft <revize>

- návrat aktivní větve na danou revizi
- rozdíly revizí se objeví ve vyčkávacím prostoru

git reset [--mixed] <revize>

- --soft + vyprázdní vyčkávací prostor (ale změny zůstanou zachovány)

git reset --hard <revize>

- --mixed + zruší všechny změny **v pracovním stromu**
- **nebezpečné**, obtížně se vrací zpět

Druhé nejdůležitější!

VZDÁLENÉ REPOZITÁŘE

Vzdálené repozitáře

- Bez nich se spolupracuje velmi obtížně :)
- Pro připomenutí: neexistuje centrální repozitář, každý má úplnou kopii celého projektu a jeho historie
- Konvence: hlavní vzdálený repozitář je **origin**
  - git clone takto pojmenuje zdrojový repozitář
- Vzdálených repozitářů může být více

Připojení vzdáleného repozitáře:

- git clone **<zdroj>**
- git remote add **origin <zdroj>**

Vzdálené repozitáře

Zdroj může být:

ssh

- git@github.com:example/example.git
- ssh://example.com/git/example.git

http

- https://example.com/example.git

složka na disku

- /opt/repos/example.git

protokol gitu

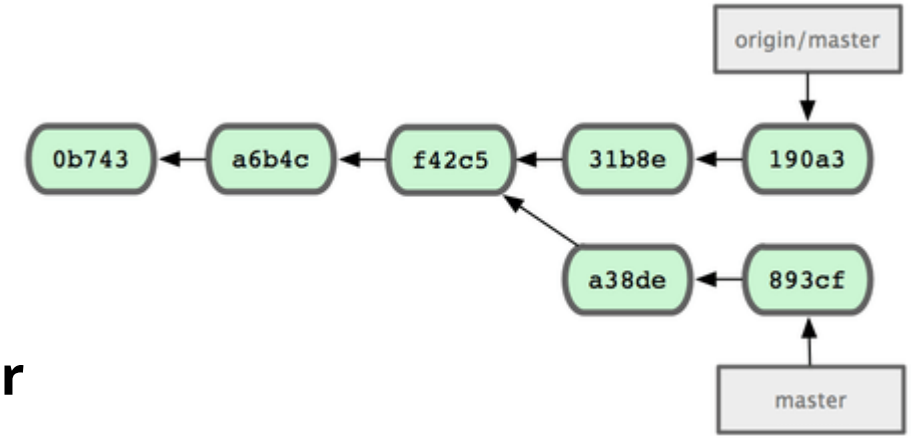
- git://example.com/example.git

Vzdálené repozitáře

- Stažení aktualizací z repozitáře:
  - git fetch **origin**

- Větve ze vzdáleného repozitáře jsou odděleně!
  - **origin** obsahuje větev **master**
  - **vývojář** má větve **master** a **origin/master**
  - je na vývojáři, aby si větve sloučil
    - git merge vytvoří merge commit
    - git rebase vytvoří jednodušší historii

- Zkratka:
  git pull **origin master**
  1. git fetch **origin**
  2. git merge **origin master**



- Odeslání změn do vzdáleného repozitáře:
  - git push **origin master**

- Je třeba mít zaktualizované lokální větve, jinak skončí chybou (non-fast-forward updates were rejected)
  - git pull origin master

- Tagy se odesílají nezávisle na větvích:
  - git push **origin v0.8**
  - git push **origin --tags**

- Fetch automaticky nevytváří lokální větve

- **origin** obsahuje **master** a **issue123**
- **vývojář** má větve **master** a **origin/master**

- Po stažení změn (git fetch origin) má **vývojář** větve **master**, **origin/master** a **origin/issue123**

- Než začne pracovat, musí vytvořit lokální větev:
  - git checkout **-b issue123 origin/issue123**
  - **issue123** sleduje **origin/issue123**

Tracking branch

- Lokální větev může sledovat nějakou vzdálenou
  - typicky se jmenují obě stejně (branch a origin/branch)
  - „branch is tracking origin/branch“

- Pull a push pak fungují i bez parametrů
- Sledování se nastaví automaticky:
  - git clone (master sleduje origin/master)
  - git checkout -b vetev origin/branch
  - git checkout --track origin/branch
- Lze nastavit dodatečně:
  - git branch -u origin/branch

Hostingy

- github.com
  - nejznámější
  - pro studenty privátní repozitáře zdarma
    - https://education.github.com/
- bitbucket.com
  - privátní repozitáře zdarma pro až 5 uživatelů
- gitlab.com
  - open-source (GitLab Community Edition)
  - nedávno nechtíc smazali kus ostrých dat
- merlin.fit.vutbr.cz, ivs.fit.vutbr.cz

- V podstatě sociální síť

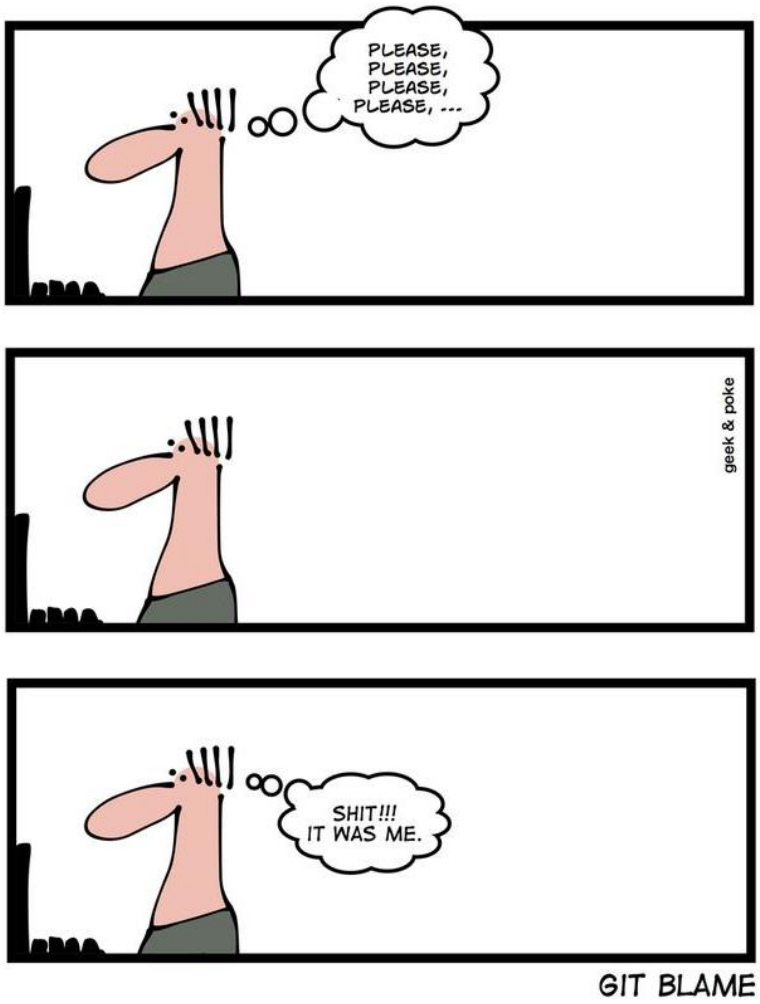
- Repozitáře
  - i přímá editace z prohlížeče
- Issue tracking
- Pull requests
- Wiki
- GitHub Projects
- Celý ekosystém externích služeb

- GitHub Pages – hostovaný web spravovaný přes git
- Gist – jednoduché sdílení souborů (~ pastebin)

DALŠÍ POTENCIÁLNĚ UŽITEČNÉ FUNKCE GITU

- Pokud chceme zjistit autora změny

git blame <soubor>
git blame <soubor> -L 40,45
git blame <soubor> -L 40,+5
git blame <soubor> -L /regexp/
git blame <soubor> -L :funkce



- Skrýš
- Před přepnutím jinam, pokud nechci udělat commit

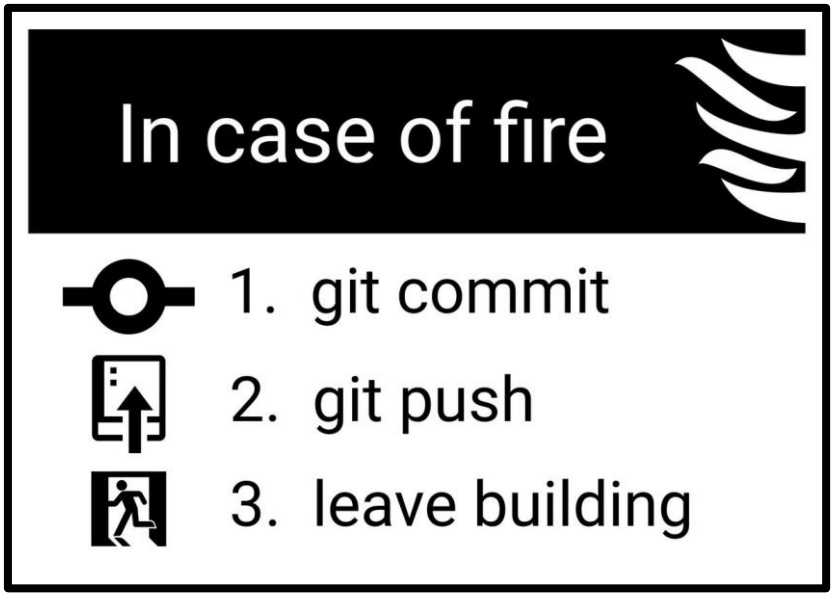
- **git stash**
- **git stash pop**
- **git stash apply**
- **git stash list**
- **git stash branch**
  - vytvoření větve ze skrýše
  - commit, kde byla skrýš vytvořena + obsah skrýše

Telegraficky

- git cherry-pick
  - Vyzobávání commitů do aktuální větve
- Submoduly
  - Repozitář v repozitáři
- „Storno“ commitu
  - git revert <commit>

- A když se něco pokazí?
  - http://ohshitgit.com/
  - https://stackoverflow.com/
  - rm --rf, git clone (výjimečně)

- Zvolte si hosting dle vaší chuti
- Dohodněte si workflow
  - jak budete organizovat větve
  - kdo bude zodpovědný za integraci
  - jak budete psát poznámky – jazyk, styl…



Možná přijde i kouzelník...

PAVEL DEDÍK

iwiglasz@fit.vutbr.cz