

Systemy 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

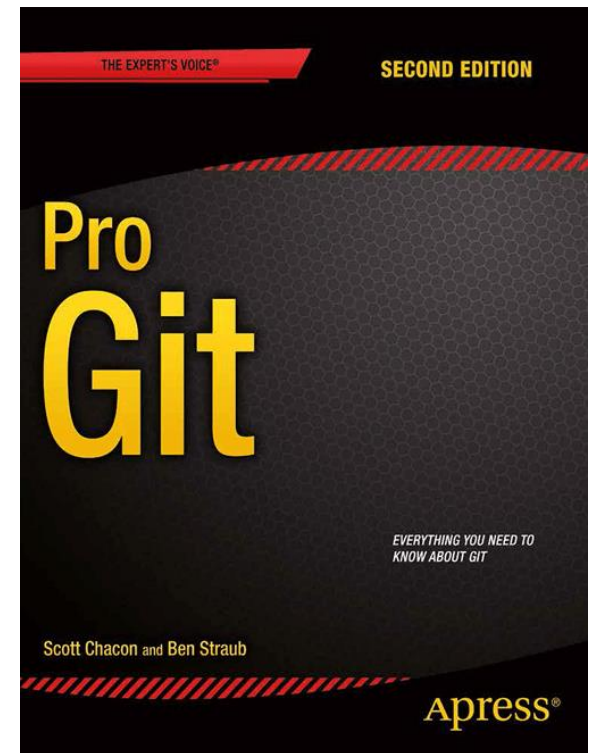
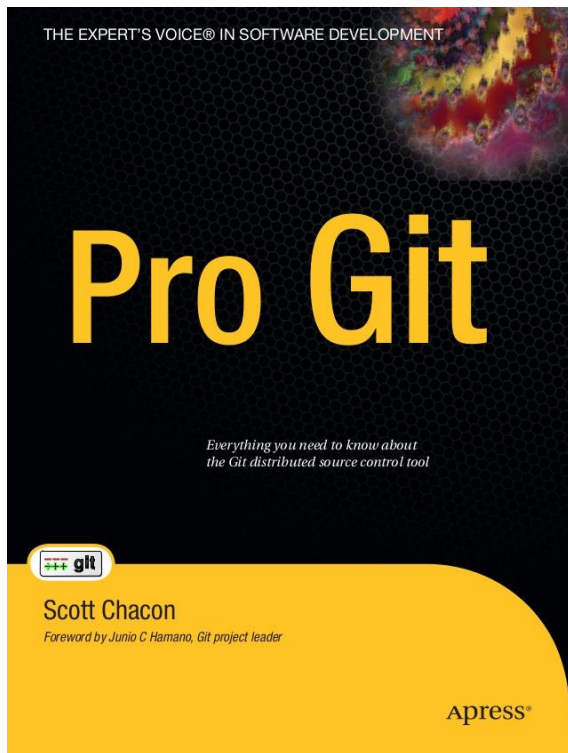


27. 2. 2018

Praktické aspekty vývoje software (IVS)

- 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/book/cs/v2> (neúplný překlad)

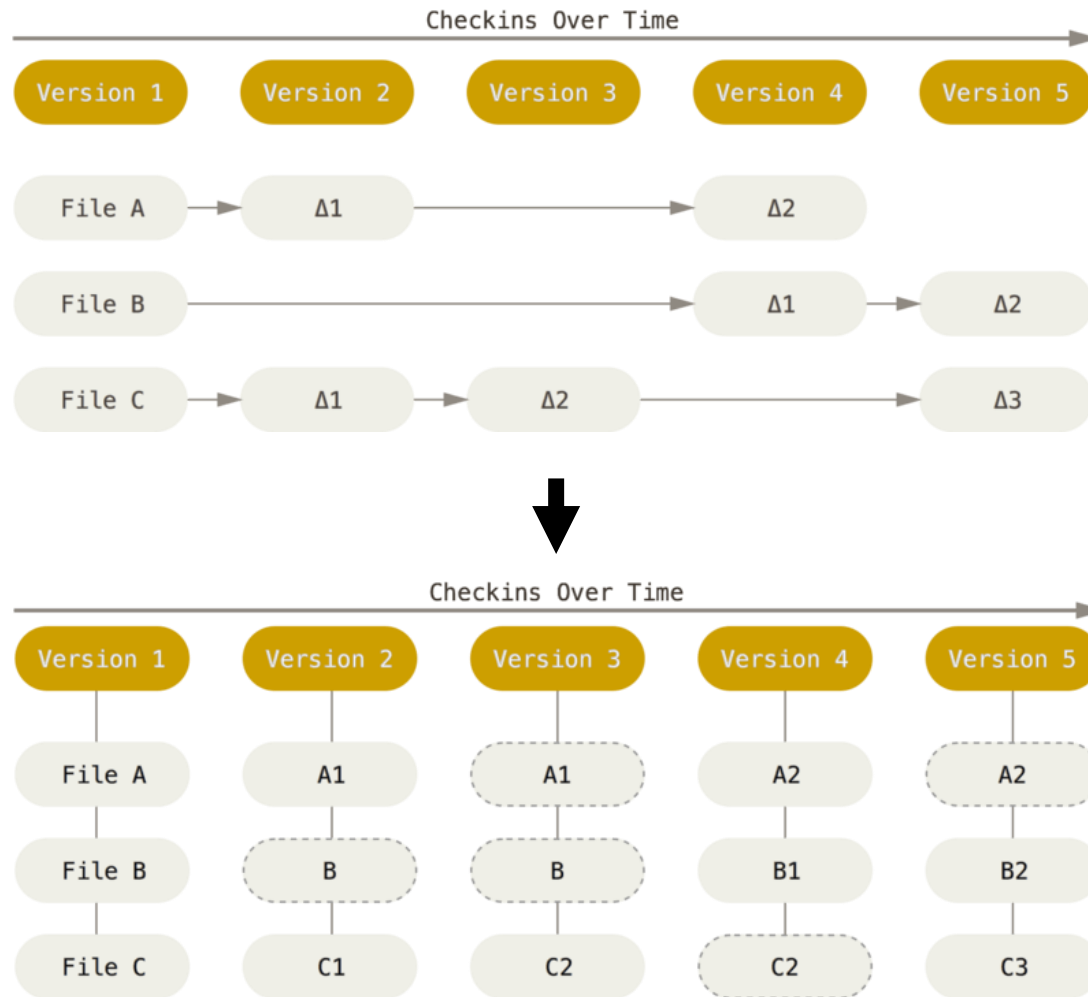


- <https://git-scm.com/>
- yum install git
- apt-get install git

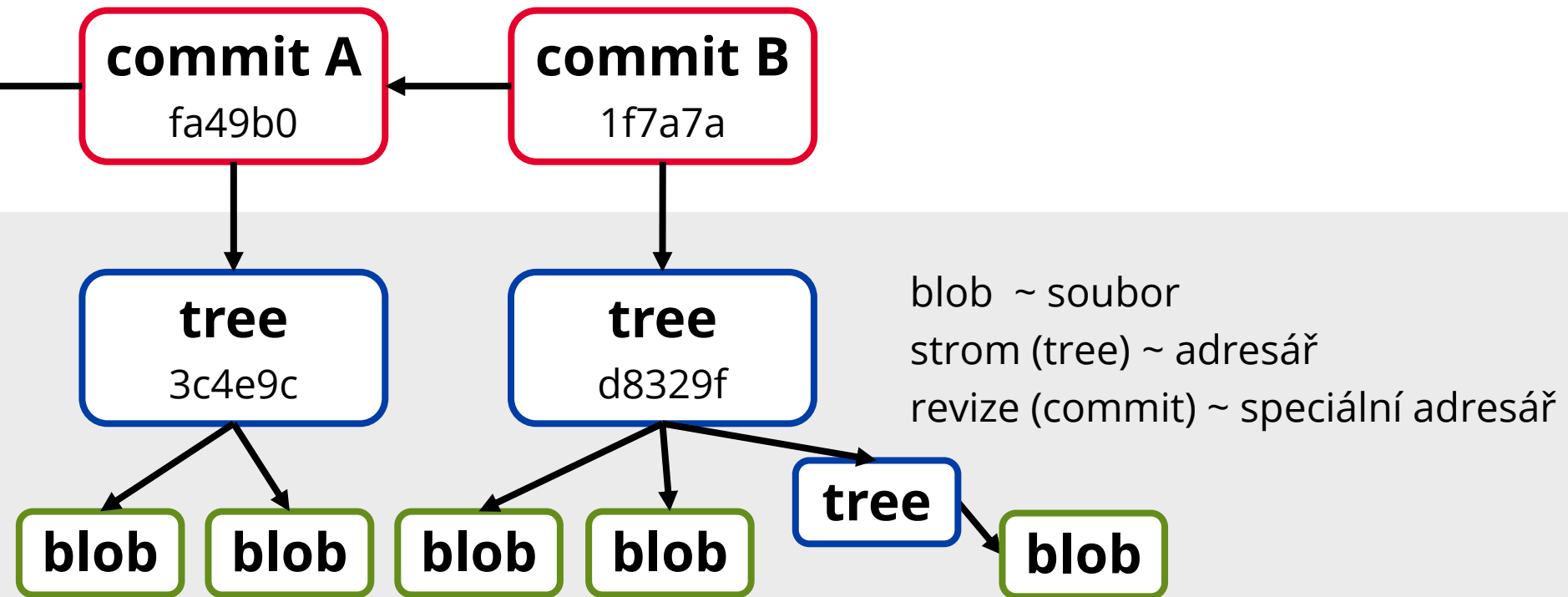


- Distribuovaný verzovací systém
- Vše lokálně, lze pracovat i ze zaoceánského parníku
- Neexistuje centrální repozitář, každý vývojář má úplnou kopii (a tedy i zálohu)
 - velké soubory → Git Large File Storage
- Ignoruje prázdné složky!
- Nepoužívá inkrementy, ale snapshoty

Inkrementy vs snapshoty:



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



- Konfigurace
 - **git config**
- 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]
- **git help [příkaz]**

git config <key> <value>

- konfigurace repozitáře, má přednost před globální

git config --global <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 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 895740c342fdbbc03ce26b6417051ab6f2188a6d0
```

```
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

- 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 gui: jen základní funkcionality

gitk: jen procházení historie

gitg:

- málo funkcí, ale přehledná historie a commit

git cola

- přehledné, automaticky obnovuje status, pokročilé hledání (např. commity podle změněného souboru)

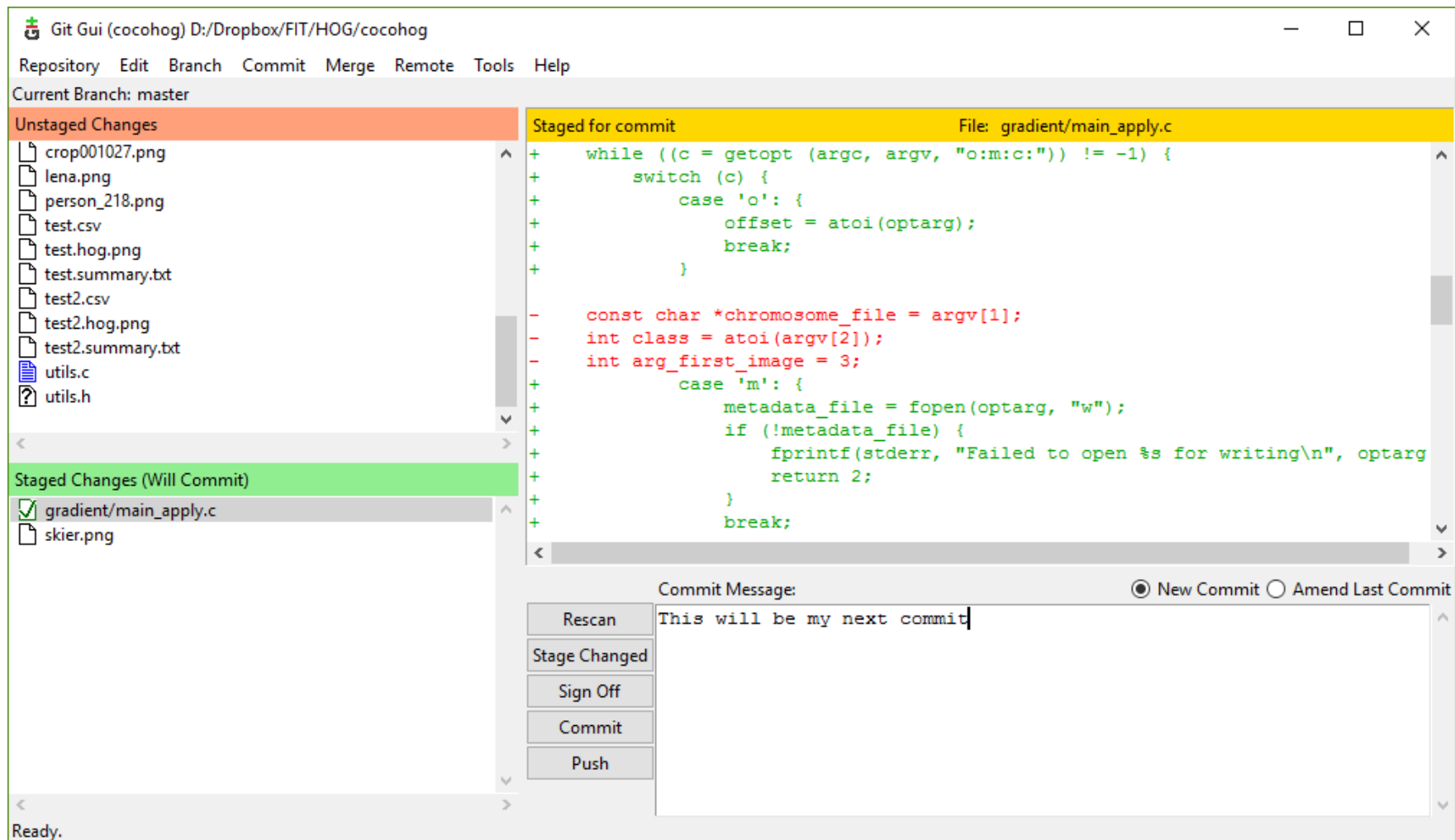
TortoiseGIT

- integrované do Průzkumníka ve Windows

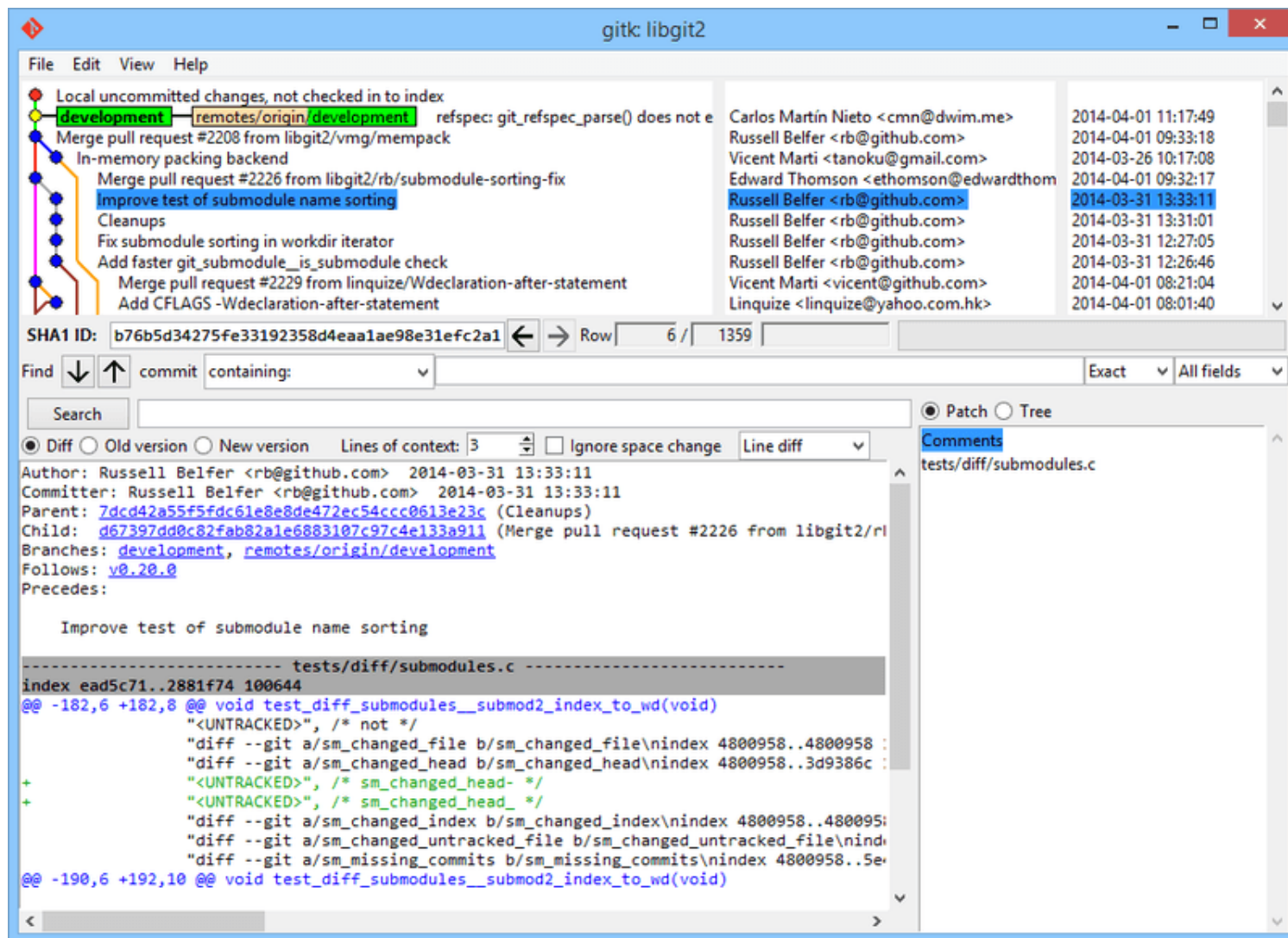
GitKraken

- zdarma s omezeními

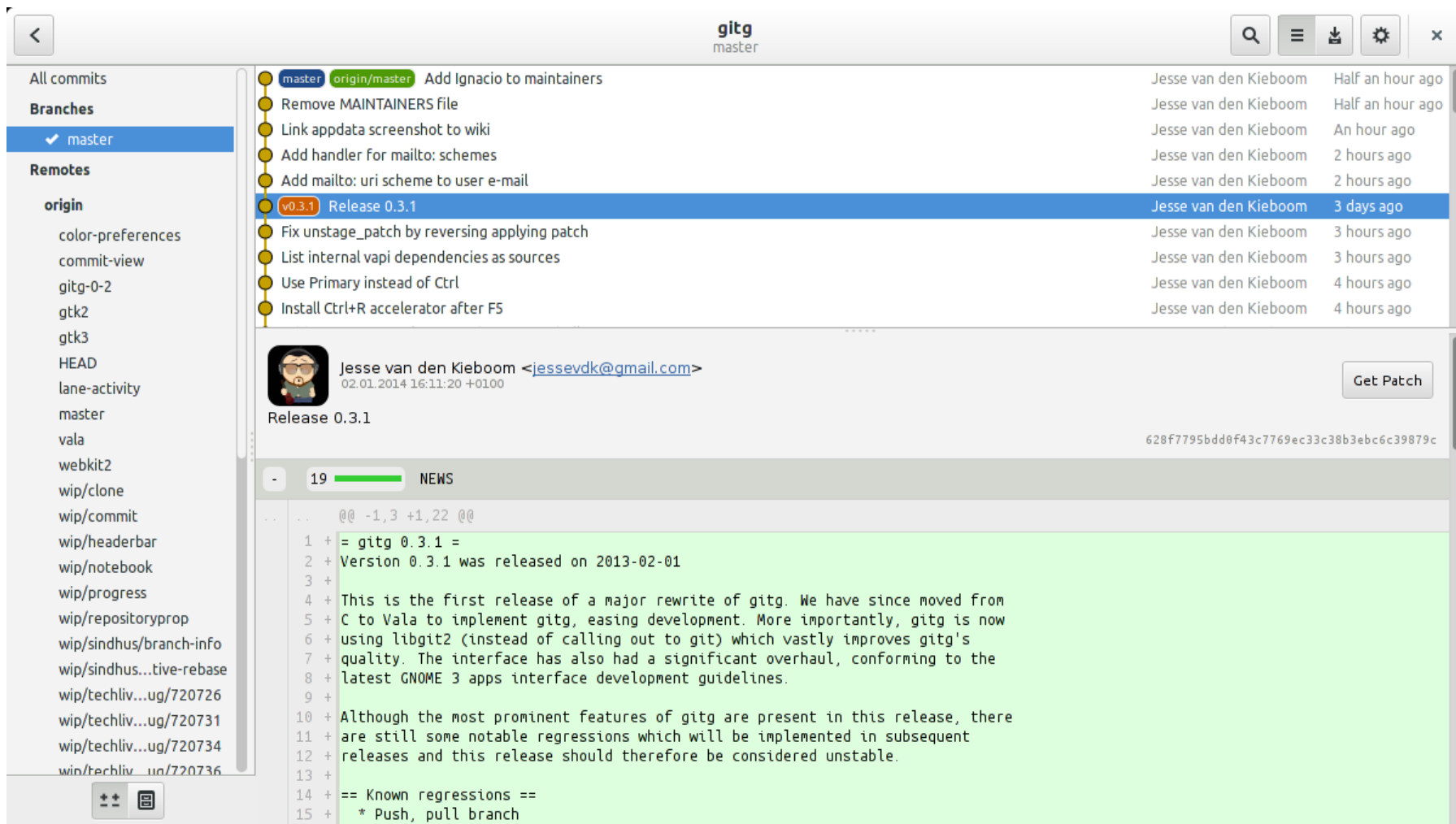
git gui



gitk



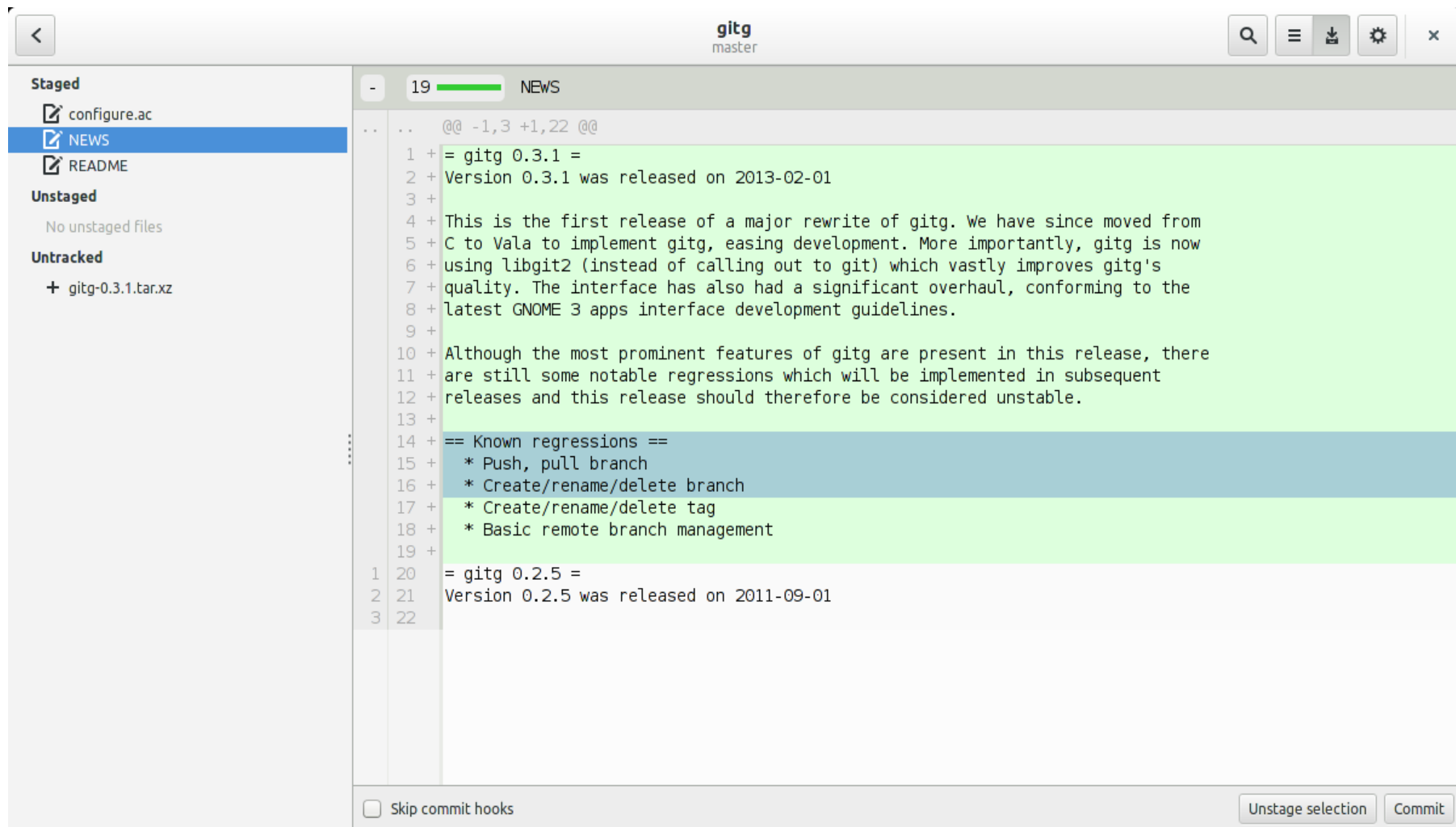
gitg



The screenshot shows the gitg application window. The title bar includes a search icon, a menu icon, a download icon, a settings icon, and a close icon. The main window is divided into three panes:

- Left Pane:** Contains a sidebar with sections:
 - All commits:** A list of commit hashes.
 - Branches:** A list of branches, with 'master' selected and highlighted in blue.
 - Remotes:** A list of remote repositories, including 'origin' and 'HEAD'.
- Center Pane:** Displays a list of commits. The selected commit is 'v0.3.1 Release 0.3.1' by Jesse van den Kieboom, dated 02.01.2014 16:11:20 +0100. The commit message is 'Release 0.3.1'. Below the commit list, there is a section for the selected commit, showing the commit hash '628f7795bdd0f43c7769ec33c38b3ebc6c39879c' and a 'Get Patch' button.
- Right Pane:** Displays the commit message for the selected commit, which is 'Release 0.3.1'. The message includes a header '== gitg 0.3.1 ==' and a body describing the release, mentioning the move from C to Vala and the use of libgit2.

gitg



1. Nastavím si git
 - hlavně jméno a email
2. Založím lokální repozitář
 - `mkdir mujprogram && cd mujprogram`
 - **git init**
3. Vytvořím první revizi (commit)
 - **git commit** -m "initial commit"
4. Tvořím
 - `vim main.c / emacs main.c / subl main.c`
 - **git add / git rm / git mv**
5. 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`

- Nastavte si své jméno a e-mail
- 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

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

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/
```

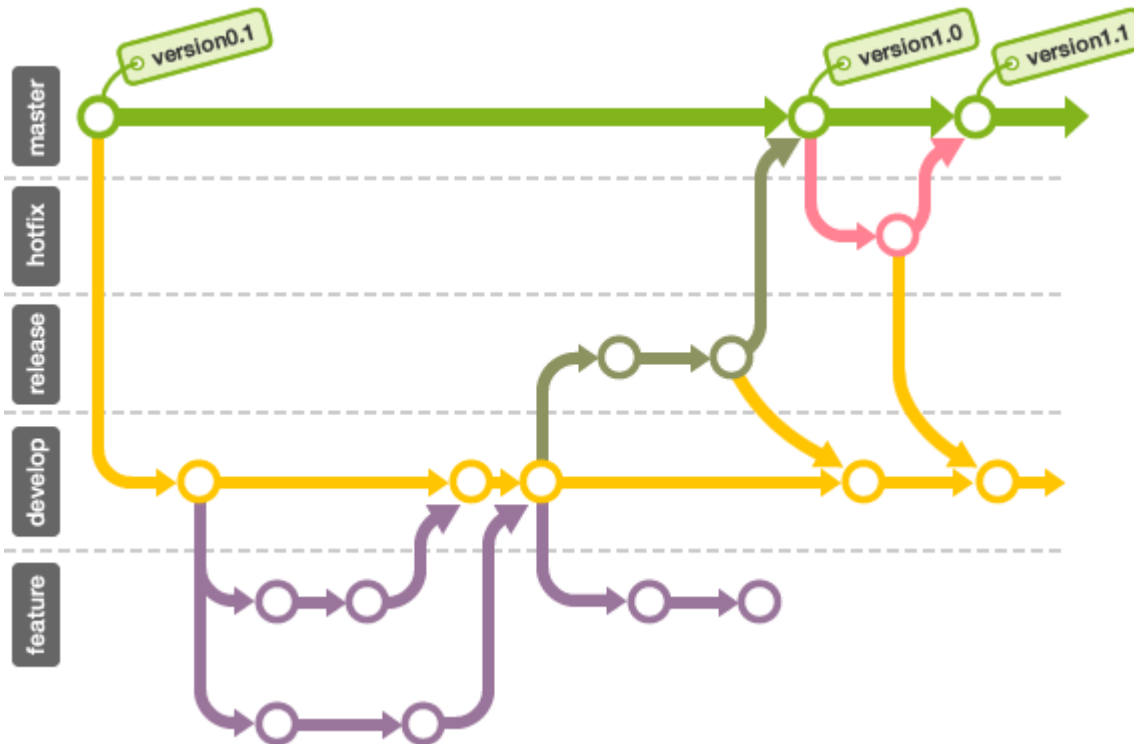
```
# ignoruj doc/notes.txt, ale ne doc/server/arch.txt  
doc/*.txt
```

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

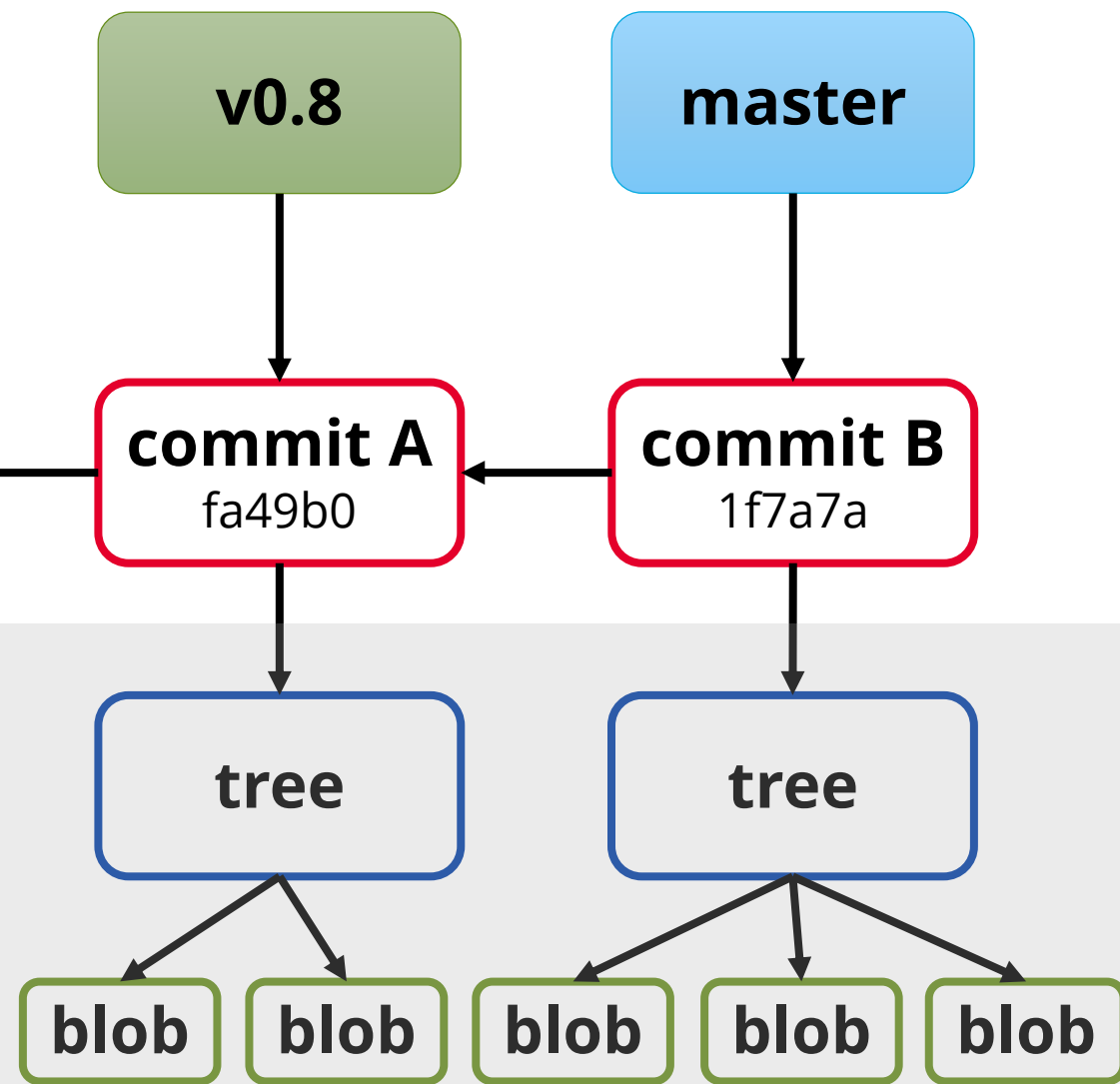
To nejdůležitější!

VĚTVENÍ A TAGY

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



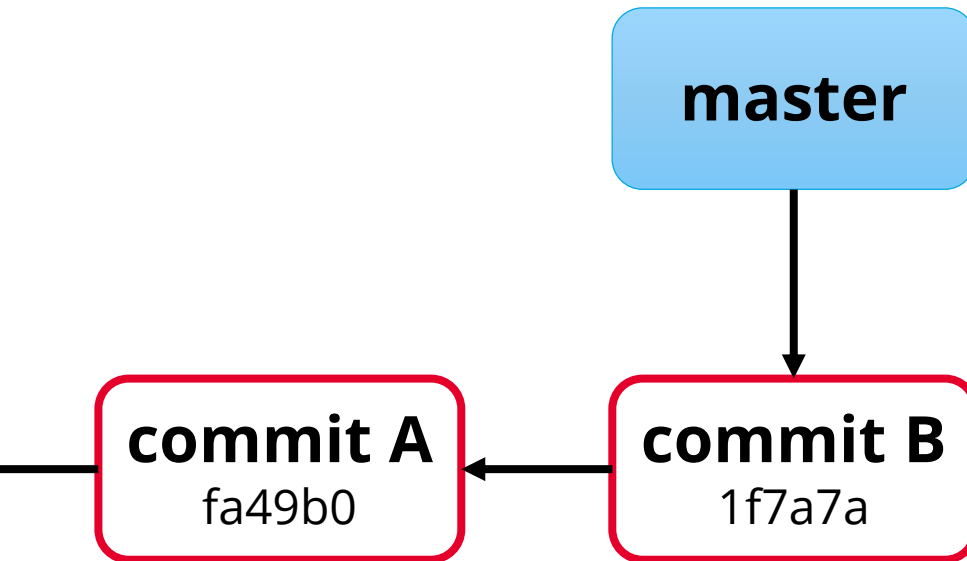
V gitu **velmi** rychlé!



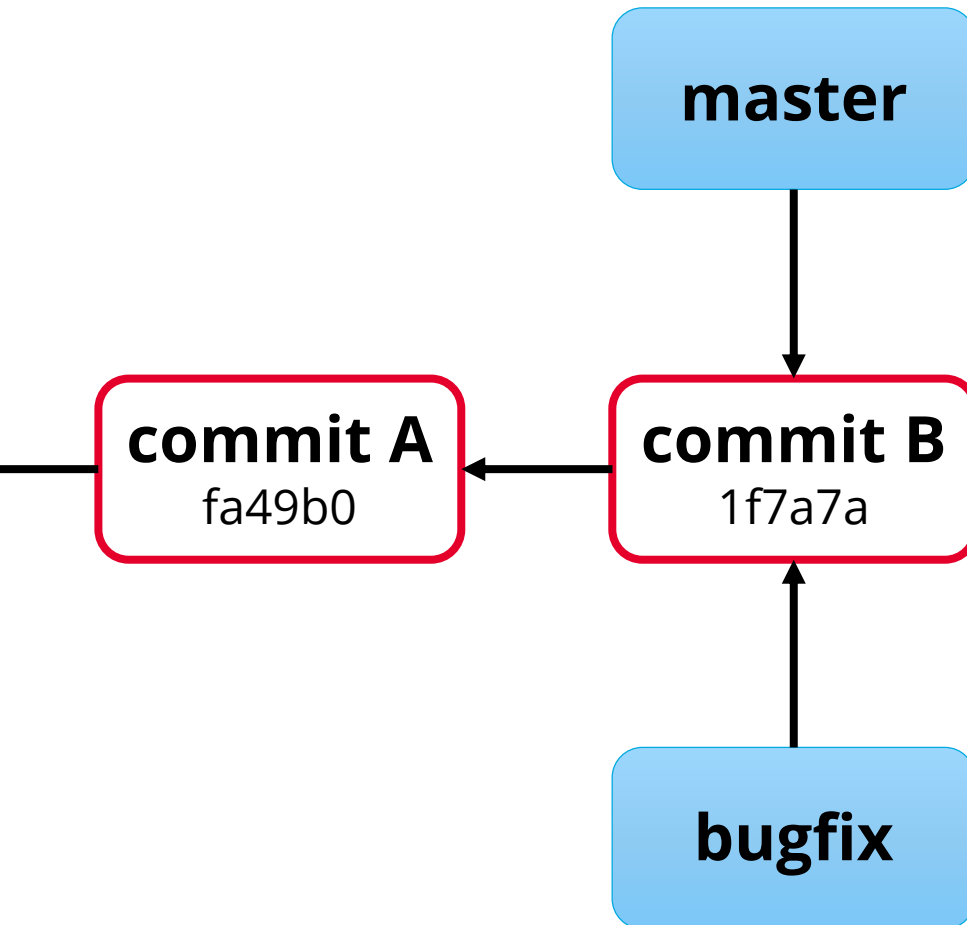
Větve a tagy jsou jen ukazatele na konkrétní revizi

- **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

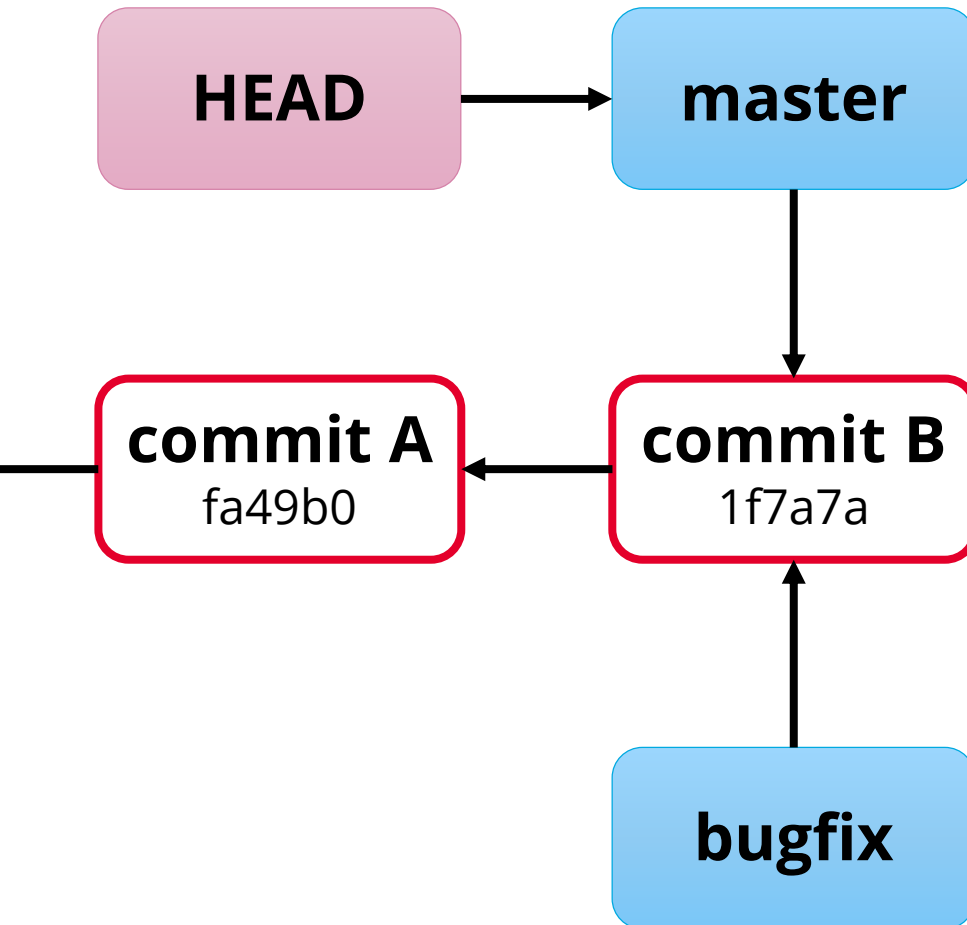
```
$ git branch bugfix
```



```
$ git branch bugfix
```

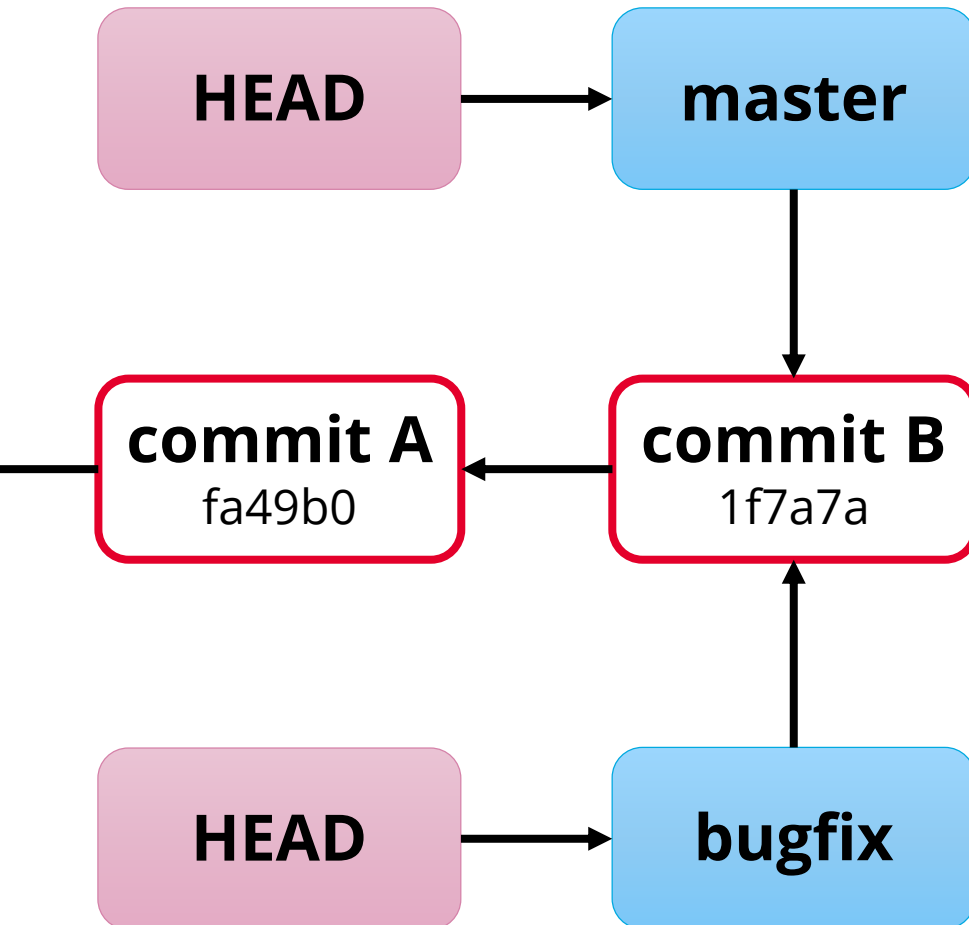


Jak ale git ví,
která větev je aktivní?



Ukazatel HEAD

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

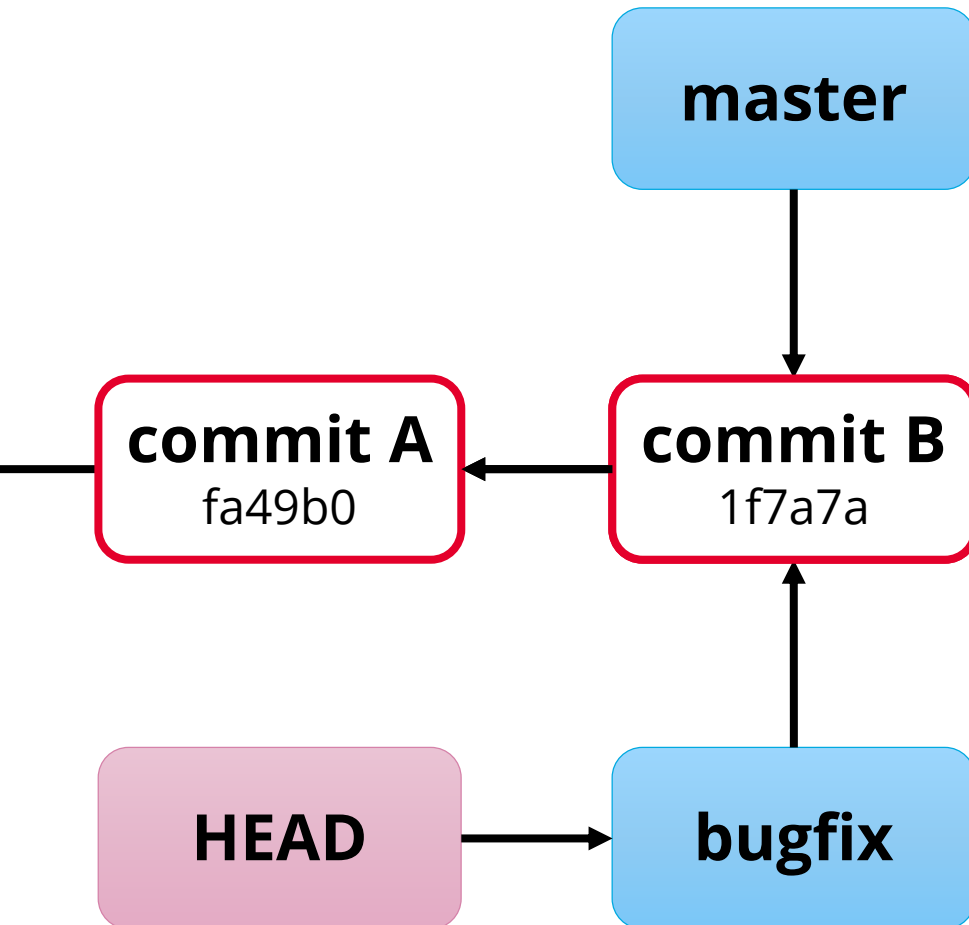


```
$ git checkout bugfix
```

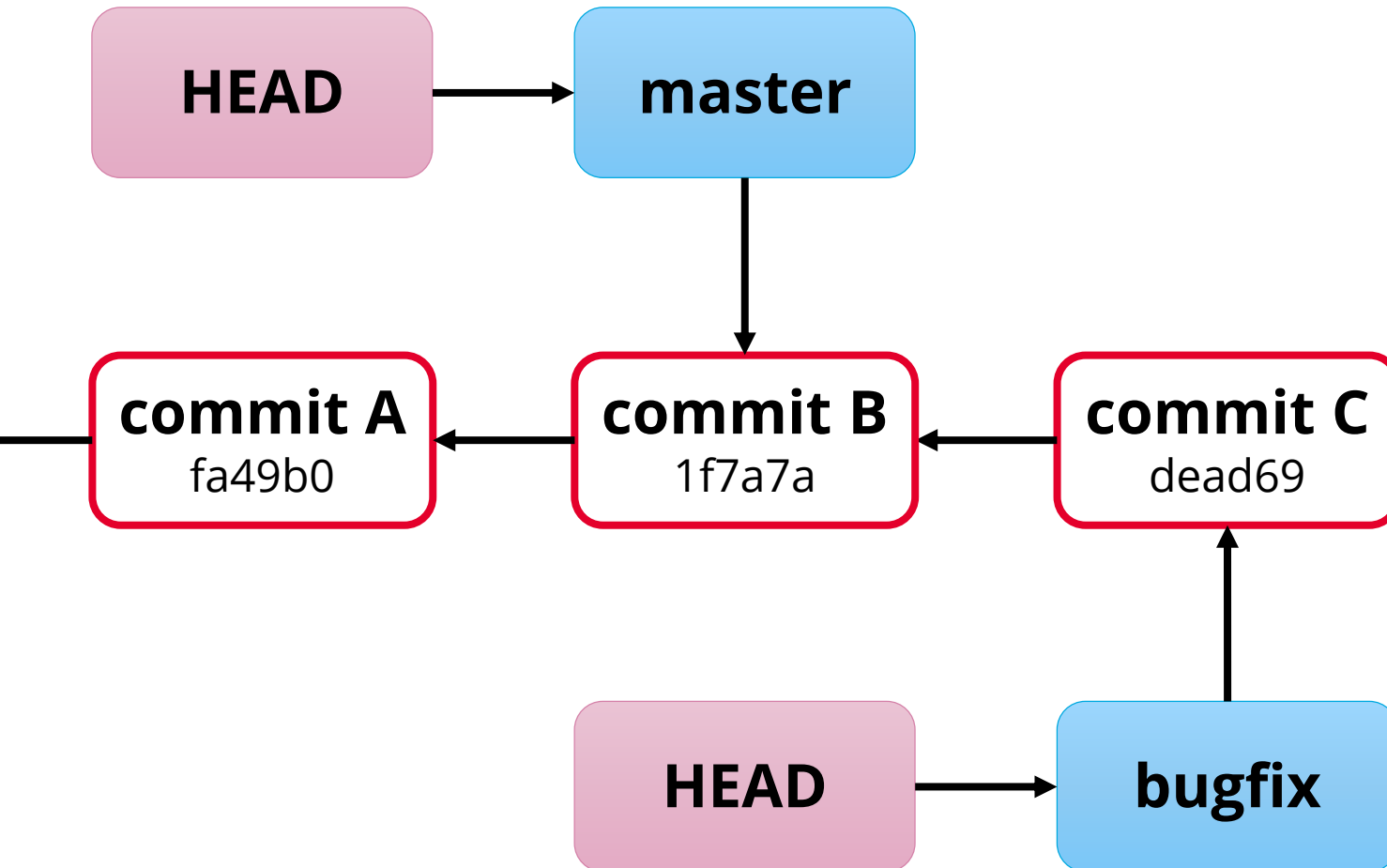
Vytvoření a přepnutí
větve v jednom kroku:

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

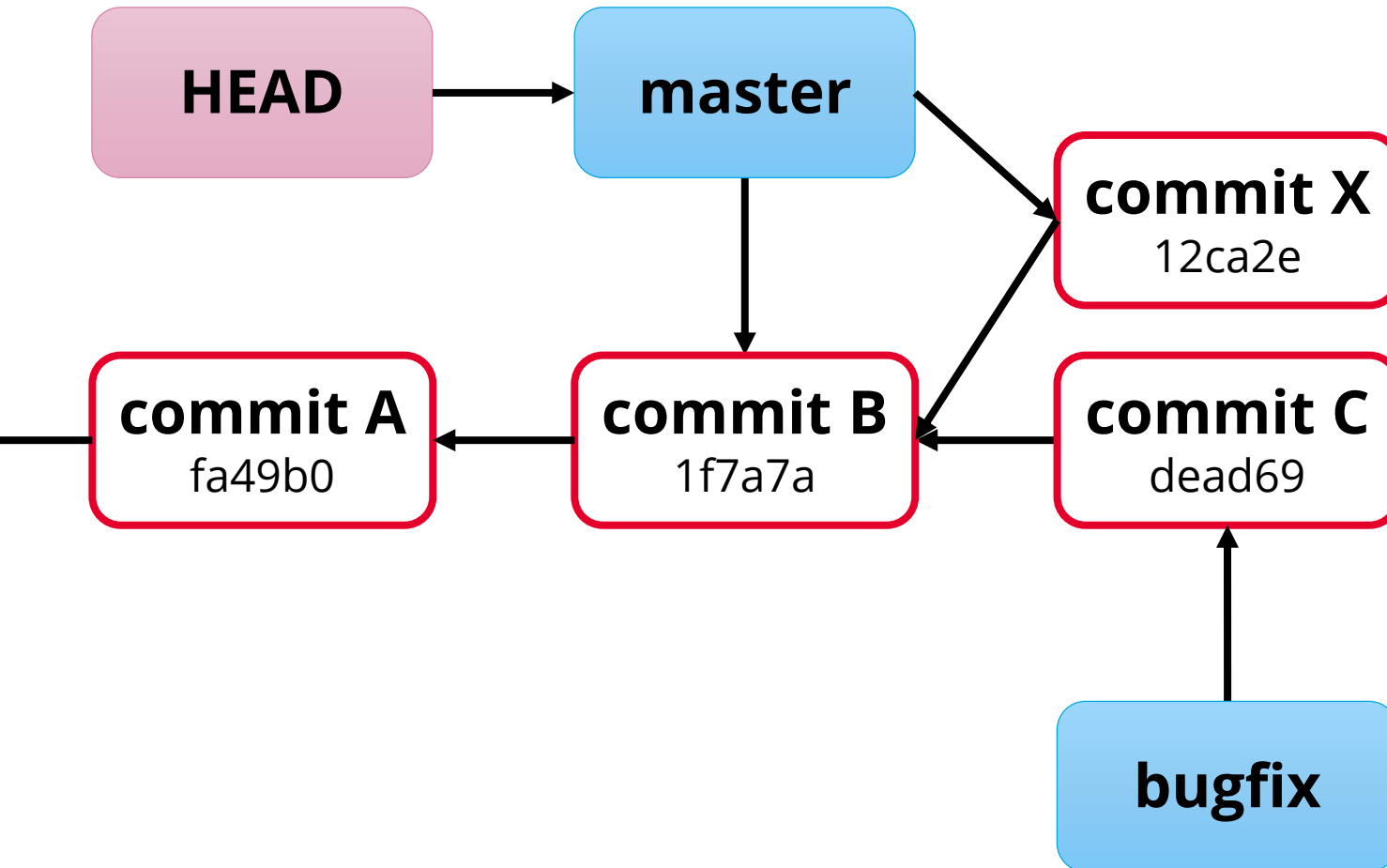
```
$ git commit -m "C"
```



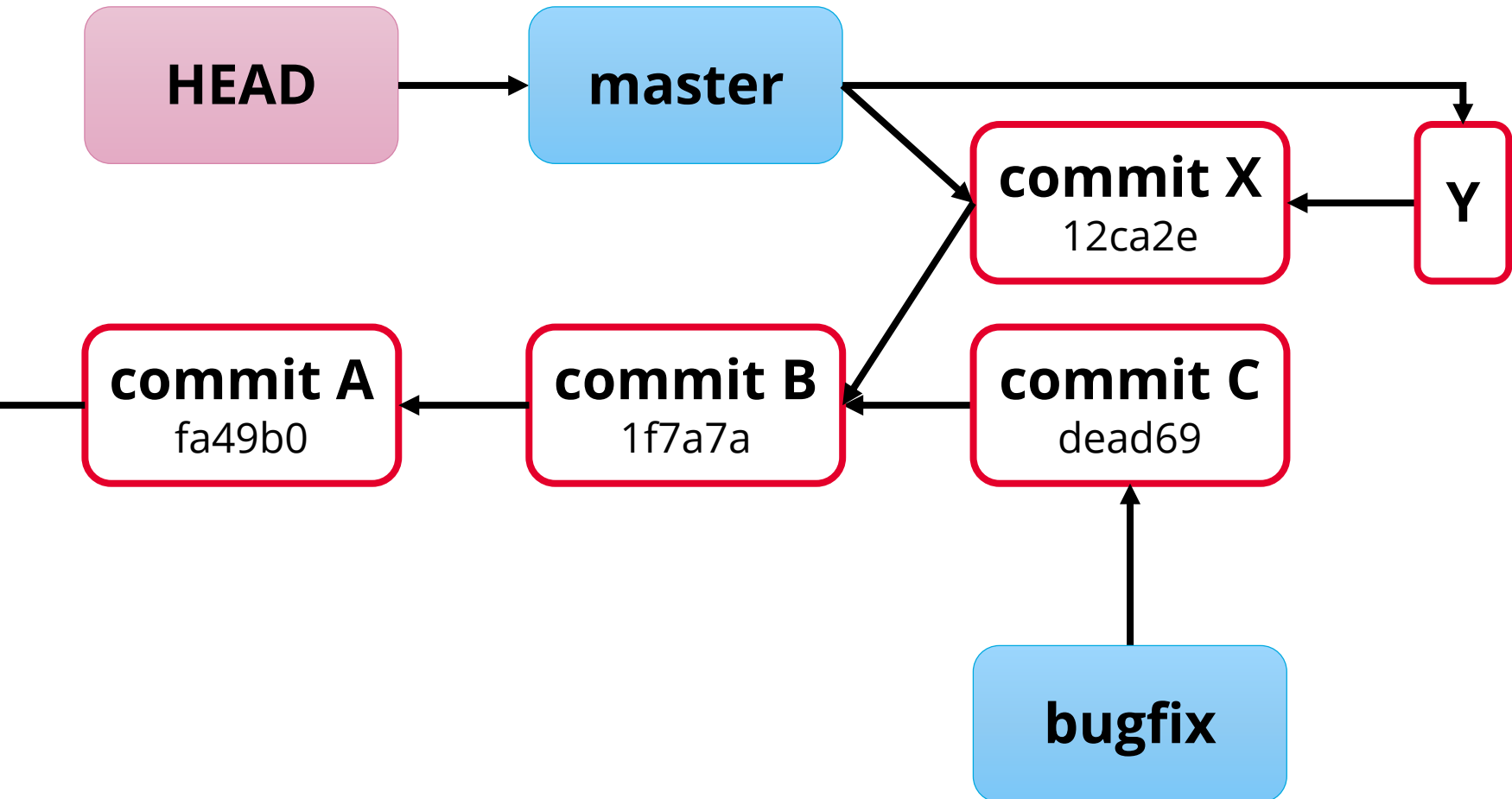

```
$ git checkout master
```



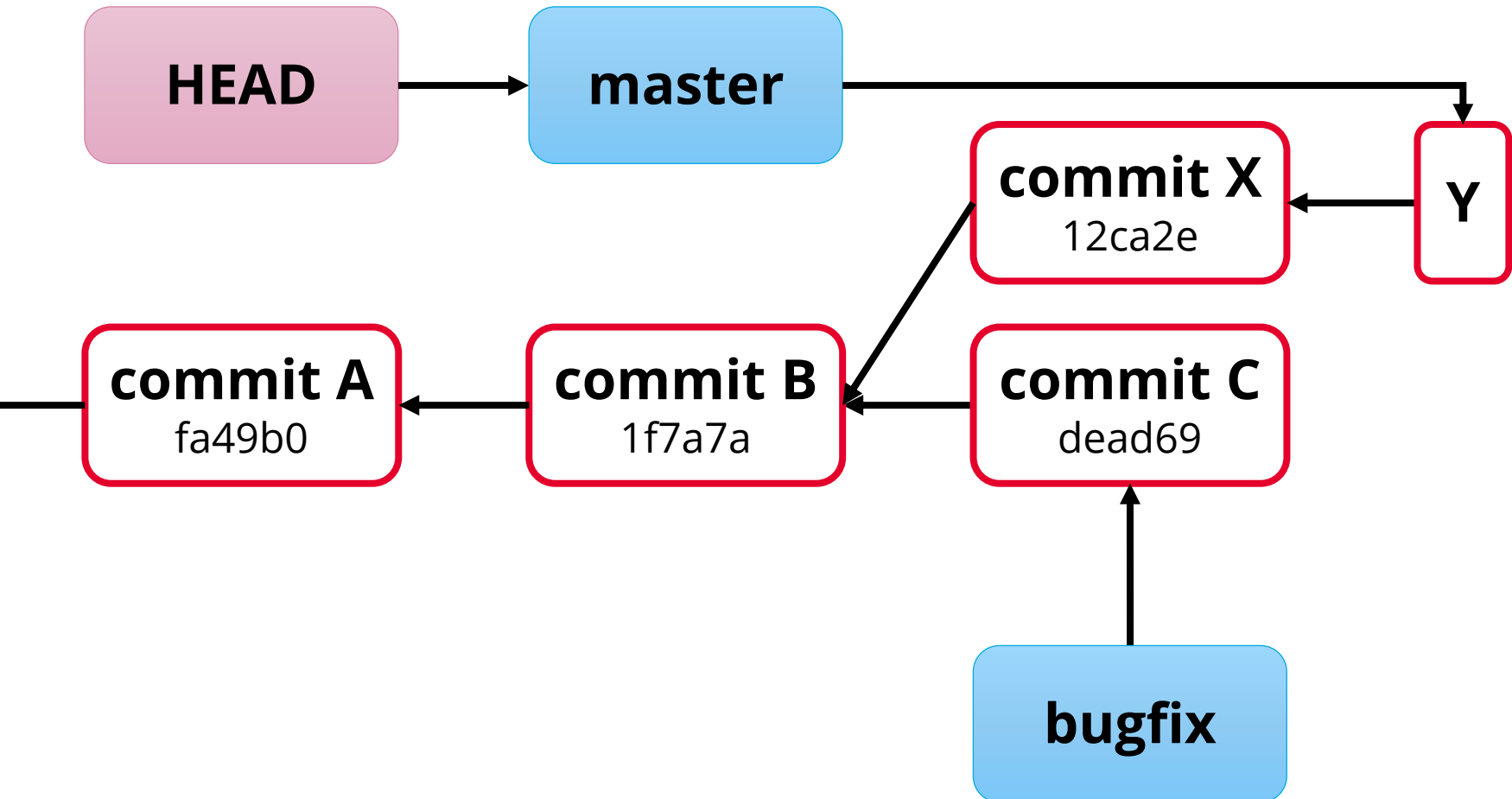
```
$ git commit -m "X"
```



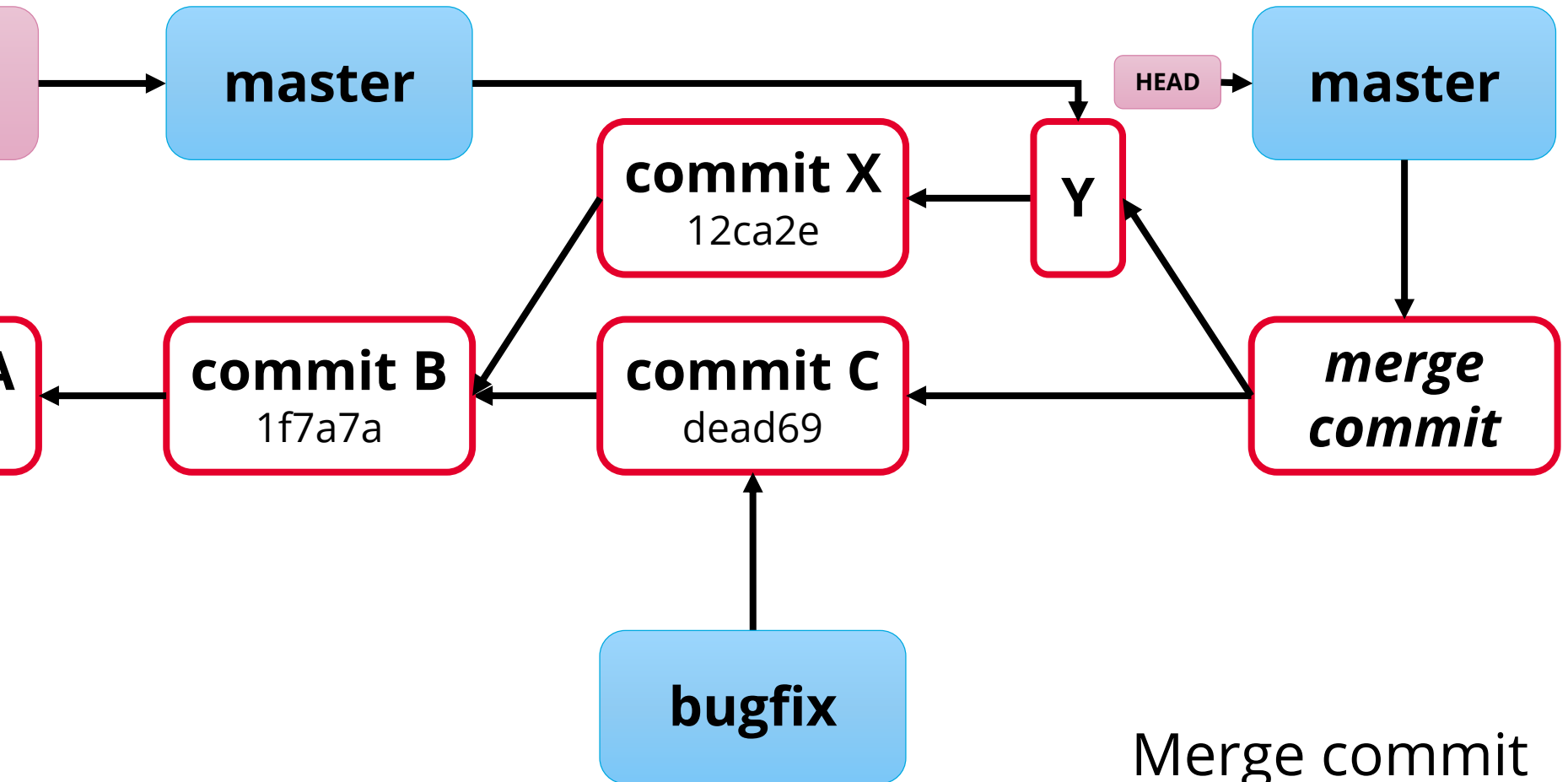
```
$ git commit -m "Y"
```



```
$ git commit -m "Y"
```

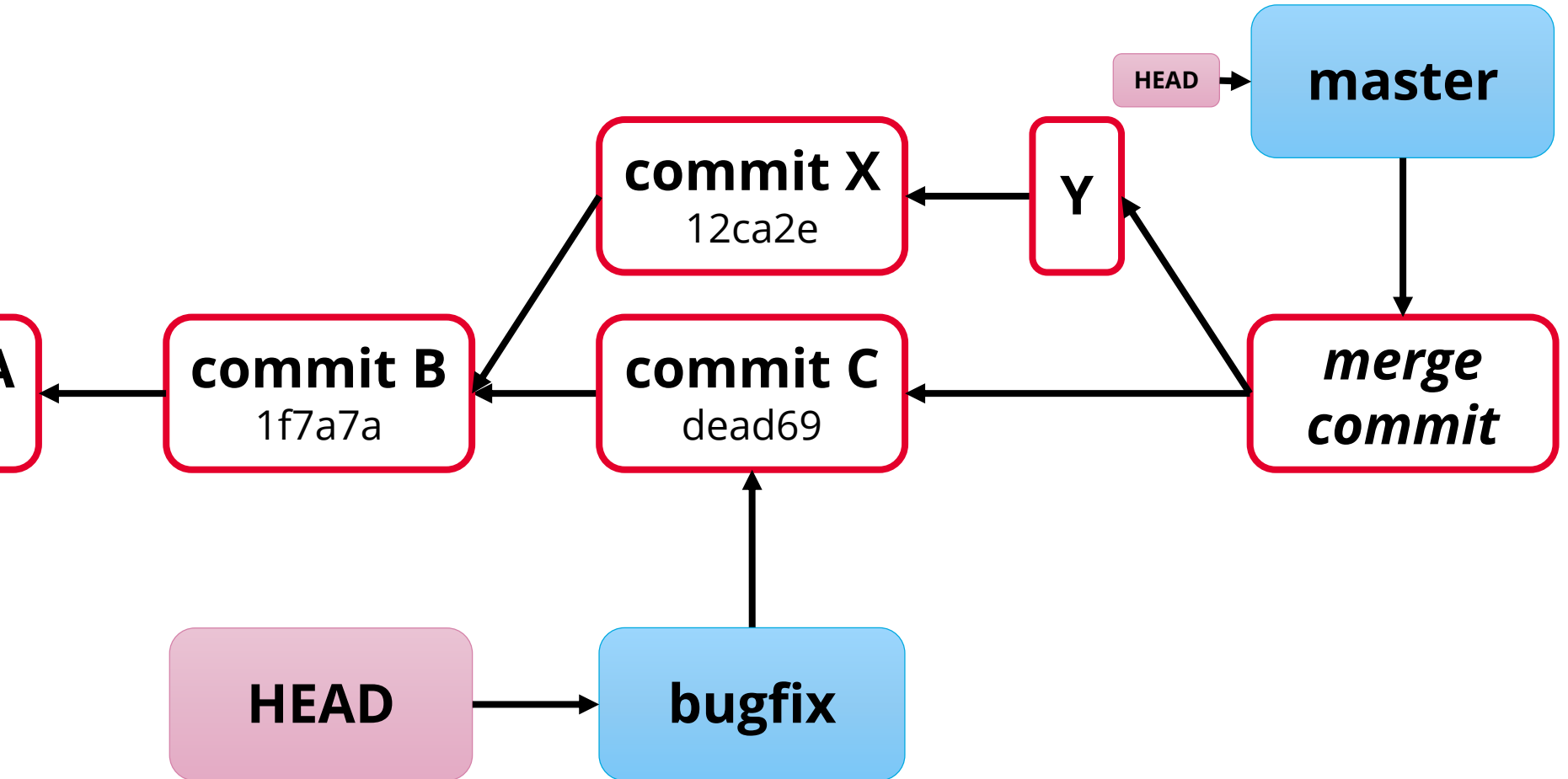


\$ git merge bugfix

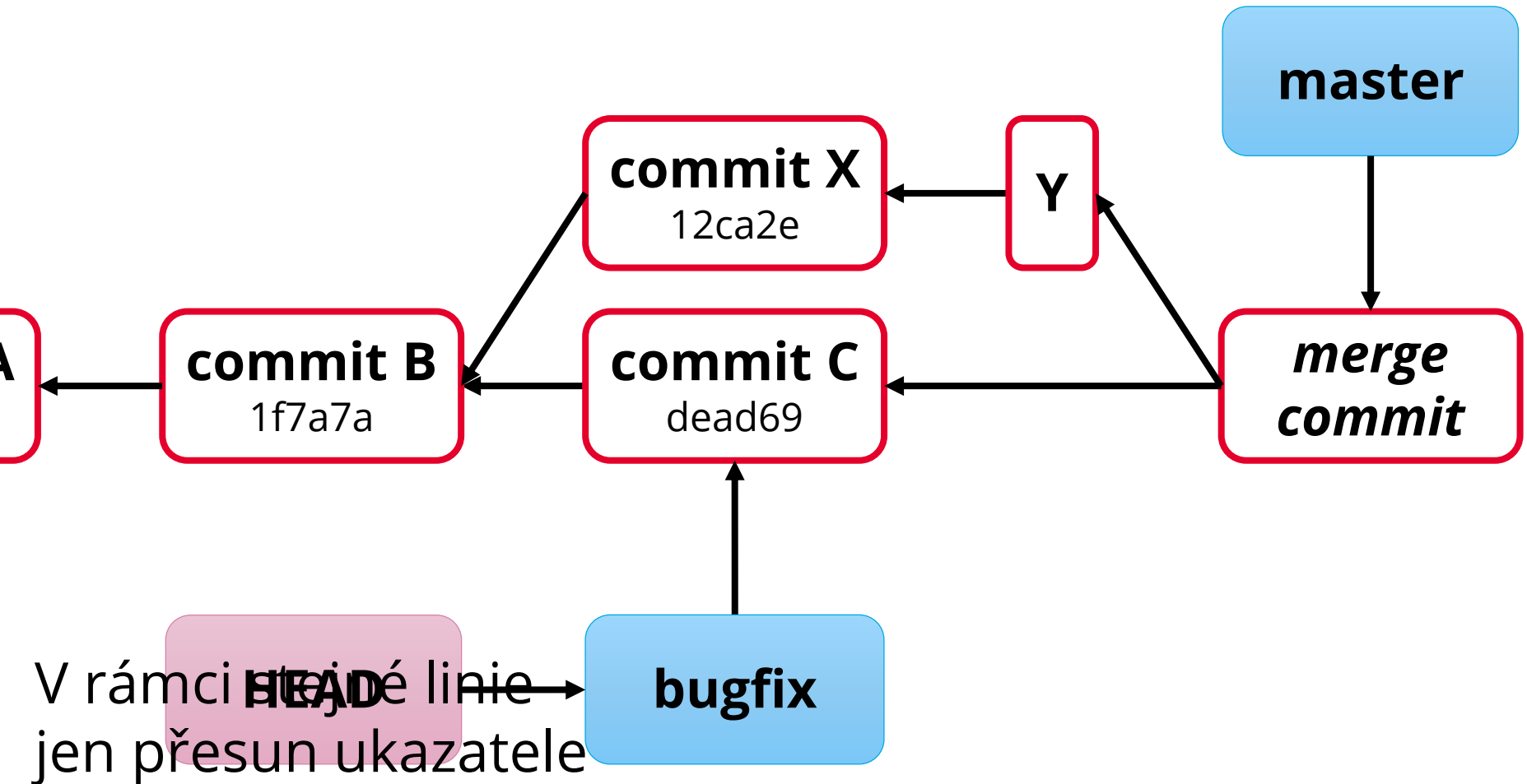


Merge commit
má dva rodiče

```
$ git checkout bugfix
```



```
$ git merge master
```



Fast-forward

Fast-forward

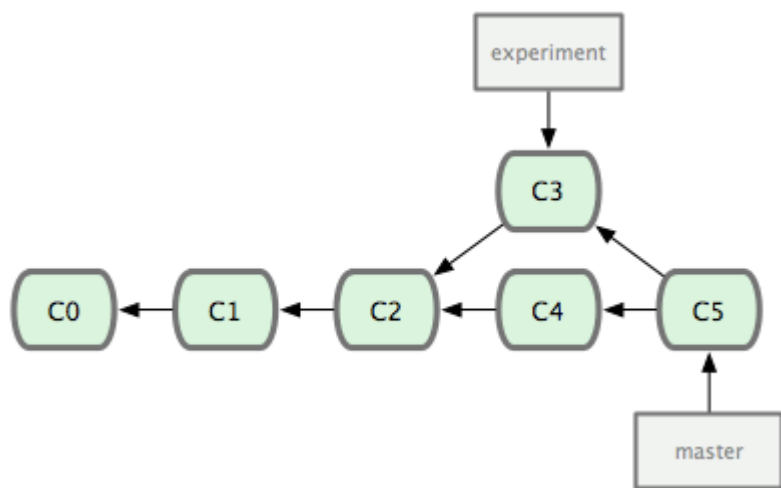
- slučované větve jsou následovníky linii
- jen se přesune ukazatel

Merge commit

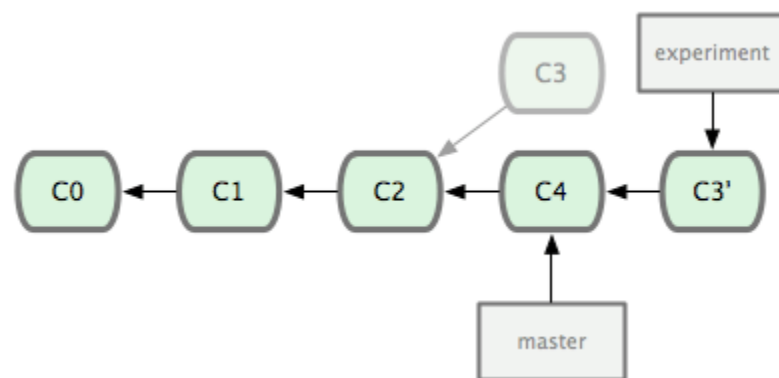
- slučované větve nejsou následovníky
- je třeba nalézt společného předka
- mohou vzniknout konflikty

- 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**

- 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

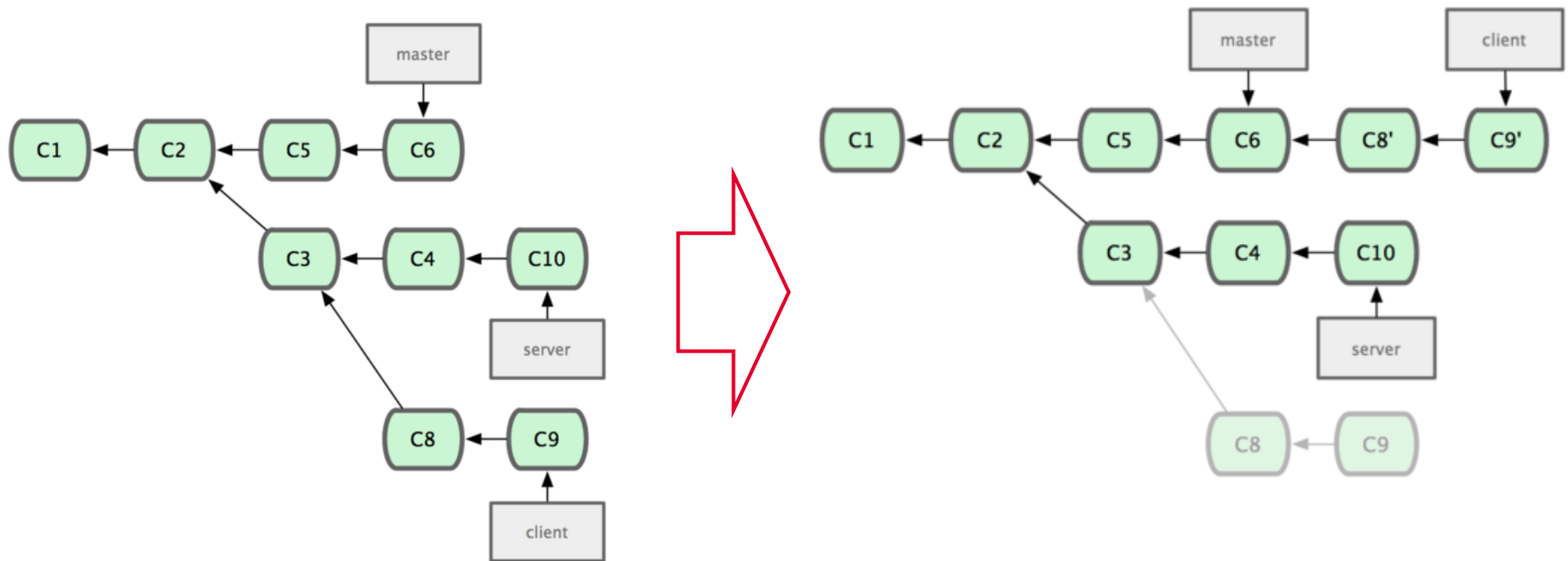


Slučování (merge)



Přeskládání (rebase)

\$ git rebase --onto master server client

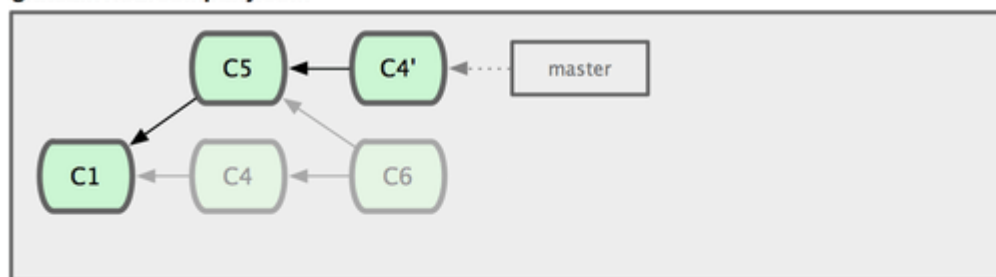


Neprovádějte přeskládání u revizí, které jste odeslali do veřejného repozitáře.

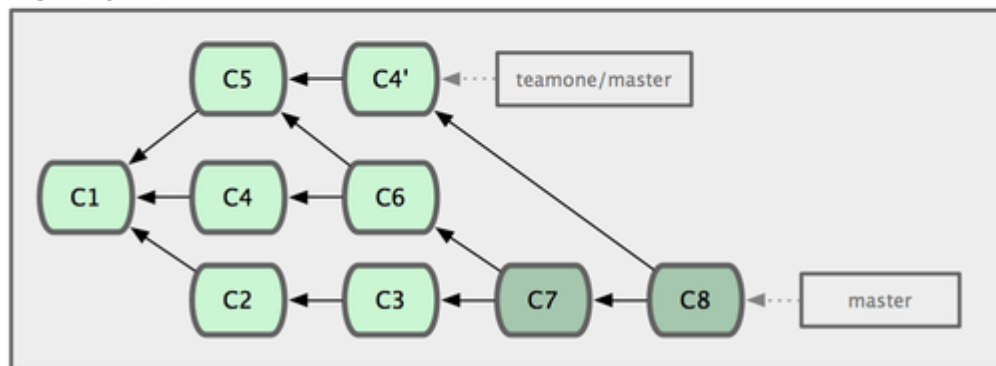
Udělá to nepořádek v historii a kolegové vás nebudou mít rádi. Fakt!

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

git.team1.ourcompany.com



My Computer



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 . (**nebezpečné**)

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

- 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>`

Zdroj může být:

ssh

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

http

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

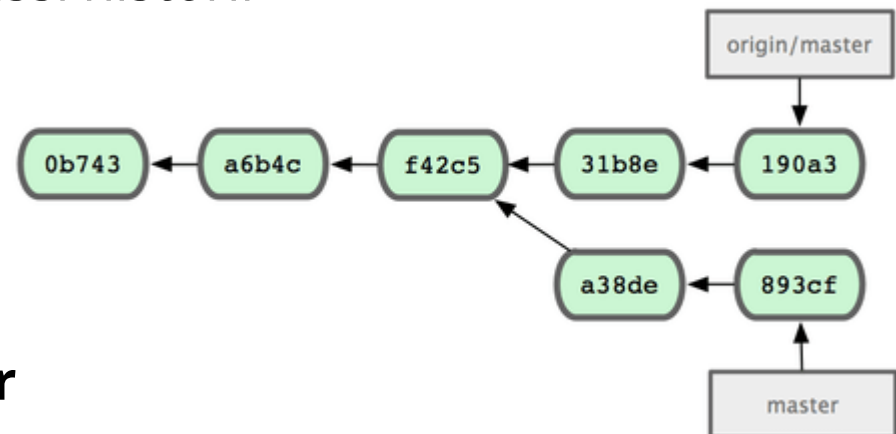
složka na disku

- `/opt/repos/example.git`

protokol gitu

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

- 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**

- 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**

- 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
 - <https://merlin.fit.vutbr.cz/repo/>

- 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

git config [--global] alias.jmeno prikaz

- vytvoří alias **git jmeno**, který vykoná příkaz **git prikaz**

```
git config --global alias.s status
```

```
$ git s
```

```
On branch master
```

```
Your branch is up to date with 'origin/master'.
```

```
nothing to commit, working tree clean
```

git config [--global] alias.jmeno prikaz

- vytvoří alias **git jmeno**, který vykoná příkaz **git prikaz**

```
git config --global alias.last 'log -1 HEAD'
```

```
$ git last
```

```
commit 66938dae3329c7aebe598c2246a8e6af90d04646
```

```
Author: Josh Goebel <dreamer3@example.com>
```

```
Date: Tue Aug 26 19:48:51 2008 +0800
```

```
test for current head
```

```
Signed-off-by: Scott Chacon <schacon@example.com>
```

git config [--global] alias.jmeno !externi-prikaz

- vykřičník znamená externí příkaz (spuštěný v shellu)
- vykřičník má v shellu zvláštní význam, je třeba použít jednoduché uvozovky (zápis \! není spolehlivý)

```
git config --global alias.vis '!gitk --all'
```

```
git config --global alias.la '!git config -l | grep  
alias | cut -c 7-'
```

```
$ git vis (spustí gitk se všemi větvemi)
```

```
$ git la (vypíše všechny aliasy)
```

Alias **git lg**: grafický barevný výpis historie do konzole

```
git config --global alias.lg "log --all --color --graph --pretty=format:'%Cred%h%Creset
%Cgreen(%ad)%Creset -%C(yellow)%d%Creset %s %C(bold
blue)<%an>%Creset %Cgreen(%cr)%Creset' --abbrev-commit
--date=short"
```

```
* 92e5ec4 (2018-02-22) - (HEAD -> master, tag: v5.1.3, origin/master, origin/HEAD) Merge pull request #974 from
|\
* 75120b9 (2018-02-22) - fix: Update coffeescript to the latest of the 1.x.x series <Honza Javorek> (3 days ago)
|/
* c94d287 (2018-02-22) - (tag: v5.1.2) Merge pull request #973 from ProdigyAI/fix-html-and-markdown-reporters <H
|\
* cb2d8d3 (2018-02-22) - fix(reporters): Use correct context after faulty CoffeeScript conversion <Stephan Bönne
* | 5cd3529 (2018-02-22) - (tag: v5.1.1) Merge pull request #943 from polc/patch-1 <Honza Javorek> (3 days ago)
|\ \
* | e6a19d6 (2018-01-07) - fix: Update coffee-script to coffeescript <Paul Le Corre> (2 weeks ago)
* | | d721f18 (2018-02-22) - Merge pull request #971 from apiaryio/honzajavorek/issue-tpl <Honza Javorek> (3 day
|\ \ \
* | | 86ce138 (2018-02-21) - docs: Improve the issue template with hints <Honza Javorek> (4 days ago)
* | | | 5241b51 (2018-02-22) - Merge pull request #970 from apiaryio/honzajavorek/pr-tpl <Honza Javorek> (3 days
|\ \ \ \
|/ \ \ \
* | | | cc93963 (2018-02-21) - docs: Improve the PR template with a hint <Honza Javorek> (4 days ago)
|/ \ \
* | | | 78185d2 (2018-02-22) - Merge pull request #972 from apiaryio/michalholasek/upgrade-request <Michal Holasek
|\ \ \ \
```

- 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

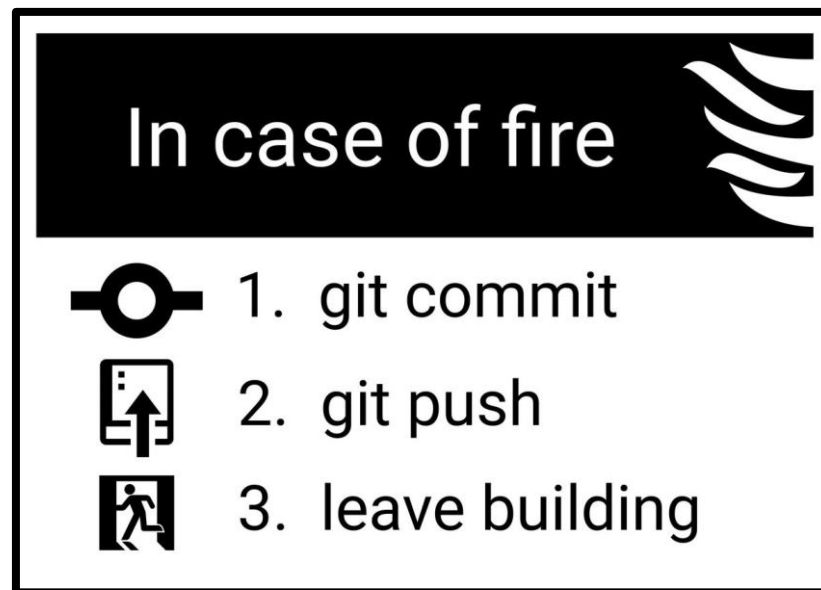


GIT BLAME

- 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

- 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