GIT

Version Control System

Nikol Ježková

Verze

2.45.8

Ceny 2018

Pivo	25 Kč
Mléko	19 Kč

8.0.20

Ceny 2020

Pivo	15 Kč
Mléko	25 Kč

Sémantické verzování

https://semver.org/lang/cs/

v0.1.1	x	0.1.1
v1.0.0	x	1.0.0
v2.45.8	x	2.45.8

v2.45.8, 2.45.8

1. **MAJOR - když nastala změna, která není zpětně kompatibilní s ostatními (API)**
2. **MINOR - když se přidá funkcionalita se zachováním zpětné kompatibility**
3. **PATCH - když se opravila chyba a zůstala kompatibilita**

https://git-scm.com/, https://www.root.cz/knihy/pro-git/

https://git-scm.com/docs/git-init

```
// vytvoří repozitář v daném adresáři
$ git init
```


9.2.2018 12:45
cenik.doc
9.2.2018 11:45
cenik.doc
adresar.doc
1.1.2018 18:45
adresar.doc
1.1.2018 17:45
adresar.doc

Vytvoříme soubor

jmena.txt

https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

```
// přidá provedené změny
$ git add <soubor>
$ git add jmena.txt
```

$ git commit

https://git-scm.com/docs/git-commit

```
// vytvoří commit ze souborů, které
   jsme přidali pomocí git add
$ git commit -m "text zprávy"
```

$ git log

https://git-scm.com/book/en/v2/Git-Basics-Tagging

// zobrazí historii commitů

$ git log

```
commit 90ecf76a11fe73a22710 (HEAD -> master)
Author: roxtri <roxtri@email.cz>
Date:   Thu Mar 22 12:23:34 2018 +0100

    Text zpravy
```

Staging area

https://git-scm.com/docs/git-status

```
$ git status
```

Upravíme soubor

```
// třeba do něj napíšeme své jméno
```

jmena.txt

https://git-scm.com/docs/git-status

$ git status

```
On branch master

No commits yet

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

  jmena.txt
```

https://git-scm.com/docs/git-add

```
// přidá soubor do staging area
$ git add <soubor>
```

https://git-scm.com/docs/git-status

$ git status

```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

  new file:   jmena.txt
```

$ git commit

https://git-scm.com/docs/git-commit

```
// vytvoří commit z přidaných souborů
$ git commit -m "text zprávy"
```

Jak přidat více souborů do staging area jedním příkazem?

upravíme jmena.txt
vytvoříme nový soubor file.txt

https://git-scm.com/docs/git-add

```
// přidá všechny změněné soubory do
   staging area

$ git add .
```

https://git-scm.com/docs/git-status

```
$ git status
```

$ git commit

https://git-scm.com/docs/git-commit

```
// vytvoří commit ze souborů, které
   jsme přidali pomocí git add.
$ git commit -m "text zprávy"
```

Jak se přepnonout do jiného commitu?

$ git log

https://git-scm.com/book/en/v2/Git-Basics-Tagging

```
// zobrazí historii commitů
$ git log
```

$ git checkout

https://git-scm.com/docs/git-checkout

```
// přepne do commitu
$ git checkout <hash>
```

Jak otagovat commit?

Označit za významný v historii…

Vytvořit verzi…

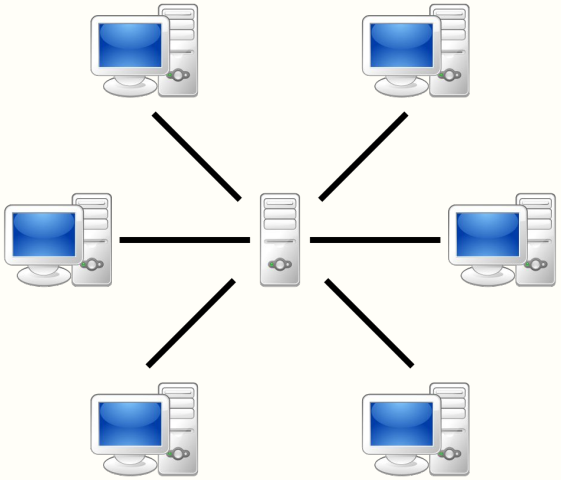
$ git tag

https://git-scm.com/book/en/v2/Git-Basics-Tagging

```
// zobrazí všechny tagy
$ git tag
// vytvoří nový tag nazvaný v1.0.0
$ git tag v1.0.0
// zobrazí všechny tagy
$ git tag
```

Klient - server

https://cs.wikipedia.org/wiki/Klient-server



- https://bitbucket.org
- https://github.com/

$ git clone

https://git-scm.com/docs/git-clone

```
// zkopíruje repozitář ze serveru
$ git clone <repozitář>
```

https://git-scm.com/book/en/v2/Git-Basics-Tagging

```
// zobrazí historii commitů ve větvi
$ git log
```

V repozitáři na serveru přibyly změny (nové commity)

Chci si změny stáhnout k sobě, abych s nimi mohl pracovat.

$ git pull

https://git-scm.com/docs/git-pull

```
// stáhne změny ze serveru a
   zaintegruje je do mé větve
```

$ git pull

https://git-scm.com/book/en/v2/Git-Basics-Tagging

```
// zobrazí historii commitů ve větvi
$ git log
```

Chci udělat změny a dát je na server

https://cs.wikipedia.org/wiki/Klient-server

vytvořím soubor nazvaný dle svého username na GitGub

roxtri.txt

Vytvořím commit

```
// přidám všechny změny do staging area
$ git add .

// vytvoří commit
$ git commit -m "text zprávy"
```

$ git push

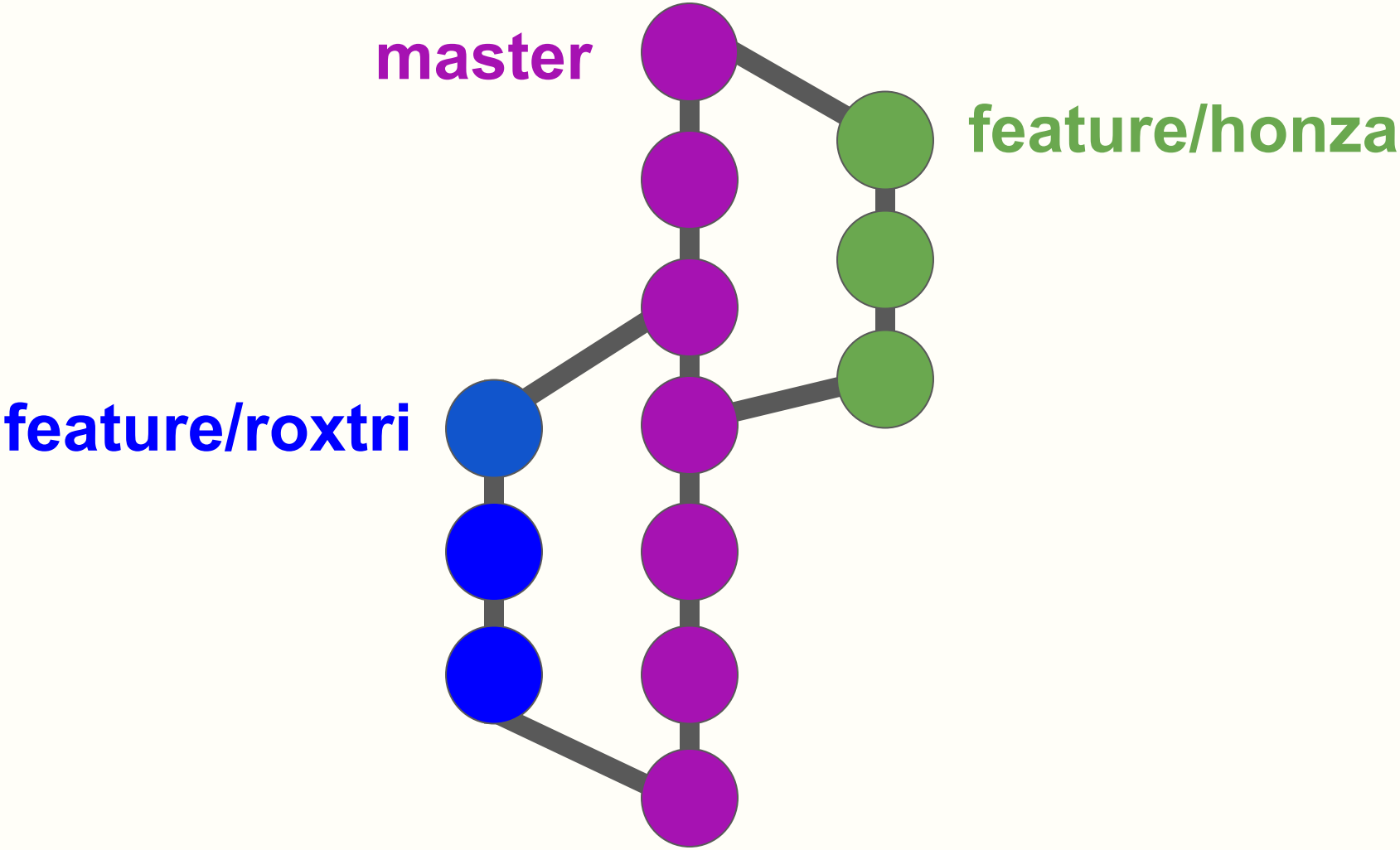
https://git-scm.com/docs/git-pull

```
// odešle mé změny na server
$ git push origin master
```

Jak izolovat svou práci od ostatních?

https://cs.wikipedia.org/wiki/Klient-server

větve


master
feature/honza
feature/roxtri

Master

Je hlavní vývojová větev

Ta, která bývá nejčastěji na produkci.

Je vytvořená defaultně při vytvoření repozitáře

$ git branch

https://git-scm.com/docs/git-branch

```
// zobrazí všechny lokální větve
$ git branch
```

$ git branch

https://git-scm.com/docs/git-branch

```
// vytvoří větev
$ git branch <nazev>
```

$ git checkout

https://git-scm.com/docs/git-checkout

```
// přepne do větve
$ git checkout <vetev>
```

Nebo jednodušeji jedním příkazem

```
// vytvoří větev a přepne do ní ukazatel
$ git checkout -b <nazev>

    stejné jako tyto příkazy

// vytvoří větev
$ git branch <nazev>
// přepne do větve
$ git checkout <nazev>
```

Vytvoříme si vlastní větev

```
// vytvoříme větev feature/username
$ git checkout -b
<feature/roxtri>
```

$ git branch

https://git-scm.com/docs/git-branch

```
// zobrazí všechny lokální větve
$ git branch
```

Mazání větví

```
// nemůžeme mazat větev, ve které jsme
$ git checkout master
// vytvoříme větev feature/username
$ git branch -D
<feature/roxtri>
```

$ git branch

https://git-scm.com/docs/git-branch

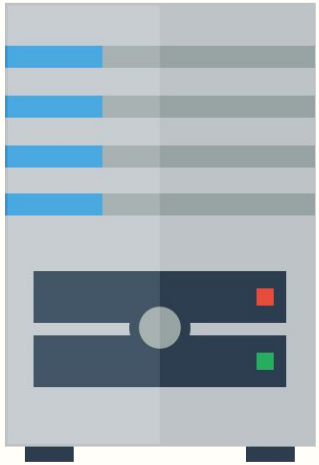
```
// zobrazí všechny lokální větve
$ git branch
```

Vytvoříme si vlastní větev

```
// vytvoříme větev feature/username
$ git checkout -b
<feature/roxtri>
```

Posíláme změny na Git server

```
// odešle vetev na git server
$ git push origin <vetev>
```



Github
Bitbucket
Gitlab

$ git push

https://git-scm.com/docs/git-pull

```
// odešle mé změny na server
$ git push origin
  feature/roxtri
```

$ git branch

https://git-scm.com/docs/git-branch

```
// zobrazí všechny remote větve
$ git branch -a
```

Mazání větve ze serveru

```
// odstraní větev ze serveru
$ git push origin :
<feature/roxtri>
```

$ git branch

https://git-scm.com/docs/git-branch

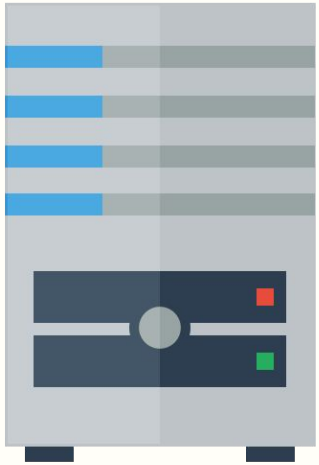
```
// zobrazí všechny remote větve
$ git branch -a
```

Odešleme větev na server

```
// odelš větev na server
$ git push origin
<feature/roxtri>
```

Stahujeme změny z Git server

Na server někdo poslal nové úpravy.
Abych s nimi mohl pracovat u sebe na počítači,
musím je stáhnout ze serveru.



Github
Bitbucket
Gitlab

$ git fetch

https://git-scm.com/docs/git-fetch

```
// stáhne změny ze serveru
// aktualizuje origin u mne na pc
// ovšem změny stále nevidím …?
$ git fetch
```

$ git merge

https://git-scm.com/docs/git-merge

```
// spojí změny ze serveru s větví,
// ve které jsem
$ git merge
```

Nebo jednodušeji jedním příkazem

zjednodušení! $ git pull

https://git-scm.com/docs/git-merge

- místo dvou příkazů **fetch** a **merge** mohu použít
- příkaz **pull**

```
// nahradí příkazy fetch a merge
// stáhne novinky ze serveru a
// aktualizuje mou větev, ve které jsem
  $ git pull
```

Slučování větví

Chci svou větev sloučit do jiné větve

Svou větev **feature/username**
do větve **master**

Chci svou větev zaintegrovat do master

https://cs.wikipedia.org/wiki/Klient-server

```
// spojím mou větev s masterem
$ git merge <nazev> --no-ff
```

$ git log

https://git-scm.com/book/en/v2/Git-Basics-Tagging

```
// zobrazí historii commitů
$ git log
```

Chci vytvořit novou verzi, tag

Každý commit můžeme označit jako důležitý - otagovat

Vytvořím nový tag

https://git-scm.com/book/en/v2/Git-Basics-Tagging

```
// zobrazí všechny tagy
$ git tag
// vytvoří nový tag nazvaný v1.0.0
$ git tag v1.0.0
// zobrazí všechny tagy
$ git tag
```

Odešlu tag na server

https://git-scm.com/book/en/v2/Git-Basics-Tagging

```
$ git push origin --tags
```

Konflikty

Konfikt

Vznikne při merge nebo rebase, pokud je upraven stejný soubor v nových commitech obou větvích

Řešení konfiktu

Abychom mohli větve sloučit, musíme konflikt vyřešit.

Tzn, vybereme, jaká varianta je správná.

Vytvoříme konflikt

Z větve master vytvořím větev conflict a vytvořím commit, ve kterém upravím soubor jmena.txt

Vytvoříme větev conflict + commit

```
// vytvořím větev conflict
$ git checkout -b conflict
// upravím soubor jmena.txt
// vytvořím commit
$ git add .
$ git commit -m "text zprávy"
```

Ve větvi master
někdo přidal commit,
ve kterém upravil
soubor jmena.txt

Chci sloučit do větve master větev conflict

```
// do větve master součí větev conflict
$ git merge conflict --no-ff
```

```
Auto-merging jmeno.txt
CONFLICT (content): Merge conflict in jmeno.txt
Automatic merge failed; fix conflicts and then commit the
result.
```

Rozhodneme, co v souboru ponechat

```
// otevřu soubor jmena.txt
<<<<<<< HEAD
Michael Jackson
=======
Nikol Ježková
>>>>>>> conflict
```

V tomto případě asi ponechámobě varianty, protože obě chci v souboru ponechat

```
// upravím soubor jmena.txt
```

Michael Jackson

Nikol Ježková

V tomto případě asi ponechámobě varianty, protože obě chci v souboru ponechat

```
// upravím soubor jmena.txt
```

Michael Jackson

Nikol Ježková

Vytvořím commit

```
// přidám všechny změny do staging area
$ git add.

// vytvoří commit
$ git commit -m "Fic conflict"
```

Merge

https://git-scm.com/docs/git-merge

fast forward x non fast forward

Merge non fast forward

https://git-scm.com/docs/git-merge

Použit dosud ve všech příkladech

Používá se při slučování větví

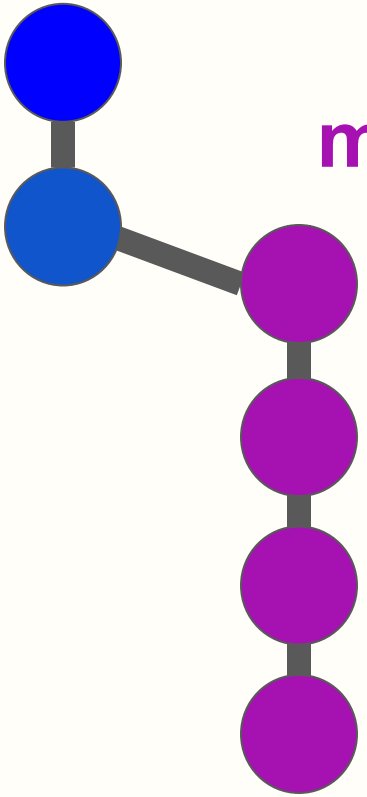
```
// spojím mou větev s masterem
$ git merge <nazev> --no-ff
```

vytvoří merge commit

to chceme kvůli přehlednosti

feature/roxtri

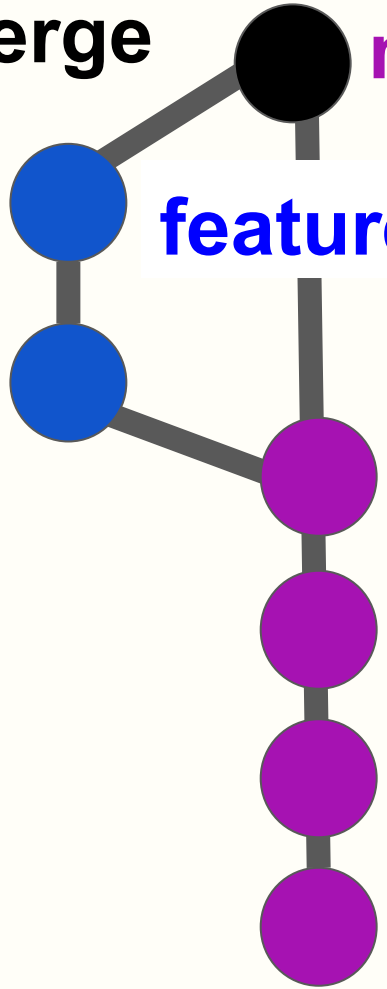
master



nový merge commit

master

feature/roxtri



Merge fast forward

https://git-scm.com/docs/git-merge

Přeskládá commity větve, ve které je ukazatel nad commity jiné větve

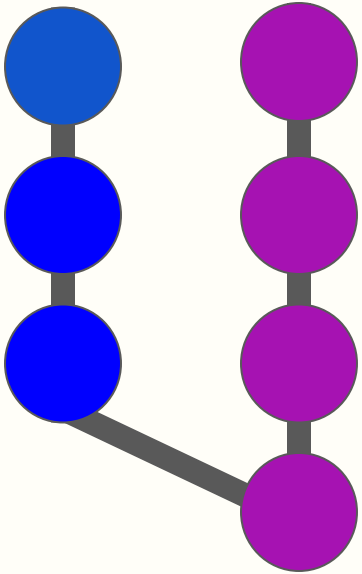
Používá se k získání nových změn

Nevytvoří merge commit

```
$ git merge <nazev>
```

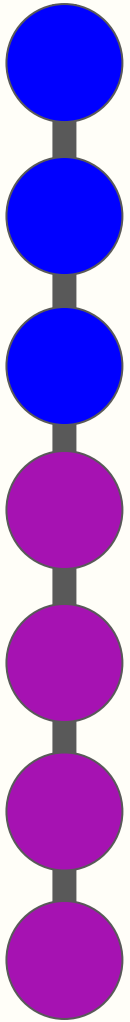
feature/roxtri

master



feature/roxtri

master



Rebase

https://cs.wikipedia.org/wiki/Klient-server

Přeskládá konflikty větve, ve které je ukazatel nad jinou větev

Pozor - mění historii commitů (hash)

Používat, jen pokud s větví nepracují ostatní

Rebase

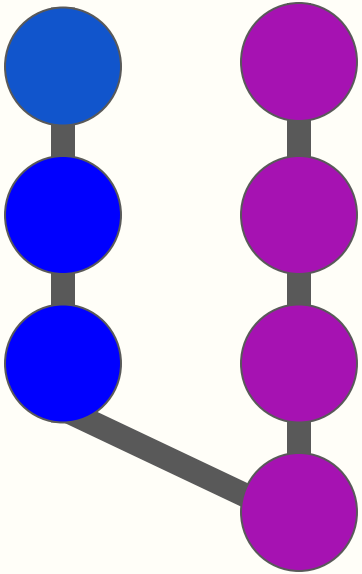
https://git-scm.com/docs/git-rebase

Přeskládá konflikty větve, ve které je ukazatel nad jinou větev

```
// přeskládá commity aktuální větve
// nad master
$ git rebase master
```

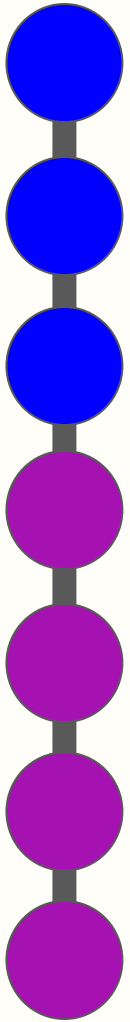
feature/roxtri

master



feature/roxtri

master



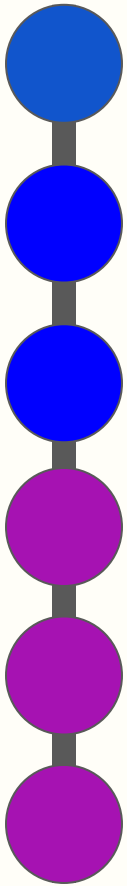
rebase vs. merge

Rebase a fast forward merge se liší pouze tím, že:

Rebase mění historii commitů (hash)

Merge fast forward nemění historii commitů

feature/roxtri



master

Předcházení konfliktů

Svou vývojovou větev často aktualizovat ad hlavní vývojovou větev

Jak na to?

Rebase nebo fast forward merge

Proč ne merge non fast forward

Vytváří merge commity a nemáme přehlednou historii - graf

Merge commity chceme jen pokud integrujeme větve - např, do masteru

Nějaké další tipy

Git workflow

https://datasift.github.io/gitflow/IntroducingGitFlow.html

Git stash

odložit si bokem práci, co nechci commitnout

https://git-scm.com/book/en/v1/Git-Tools-Stashing

Git cherry-pick

Vycucnout si commit z historie

https://git-scm.com/docs/git-cherry-pick

Změny v historii

https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History

commit --amend

https://www.atlassian.com/git/tutorials/rewriting-history

Jak zobrazit název větve v terminalu na osx?

```
sudo open -a TextEdit ~/.bash_profile
// vložit a restartovat terminál
parse_git_branch() {
  git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/*
\(.*\)/ (\1)/'
}

export PS1="\u@\h
\W\[\033[32m\]\$(parse_git_branch)\[\033[00m\] $ "
```

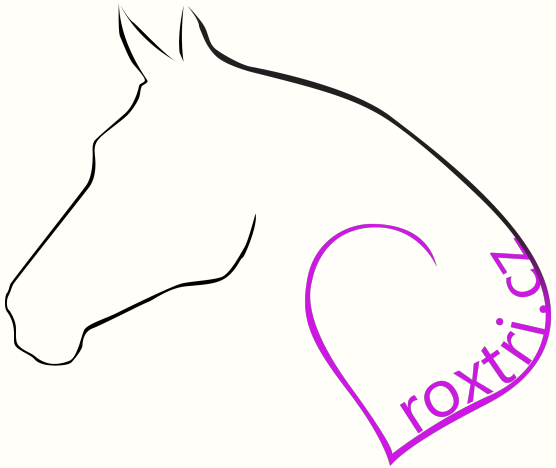
SSH kíče

Github

https://help.github.com/articles/connecting-to-github-with-ssh/

Bitbucket

https://confluence.atlassian.com/bitbucket/set-up-an-ssh-key-728138079.html



@roxtri_cz

roxtri@email.cz