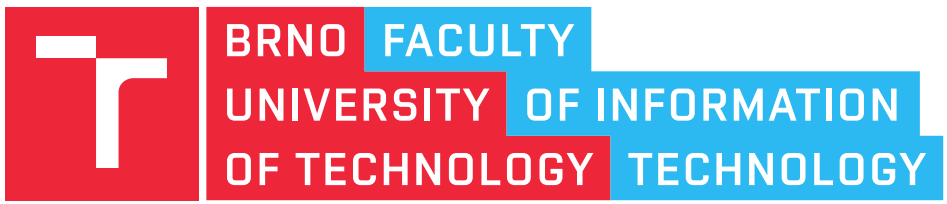
GIT

Jaroslav Dytrych

Fakulta informačních technologií Vysokého učení technického v Brně
Božetěchova 1/2. 612 66 Brno - Královo Pole
dytrych@fit.vutbr.cz

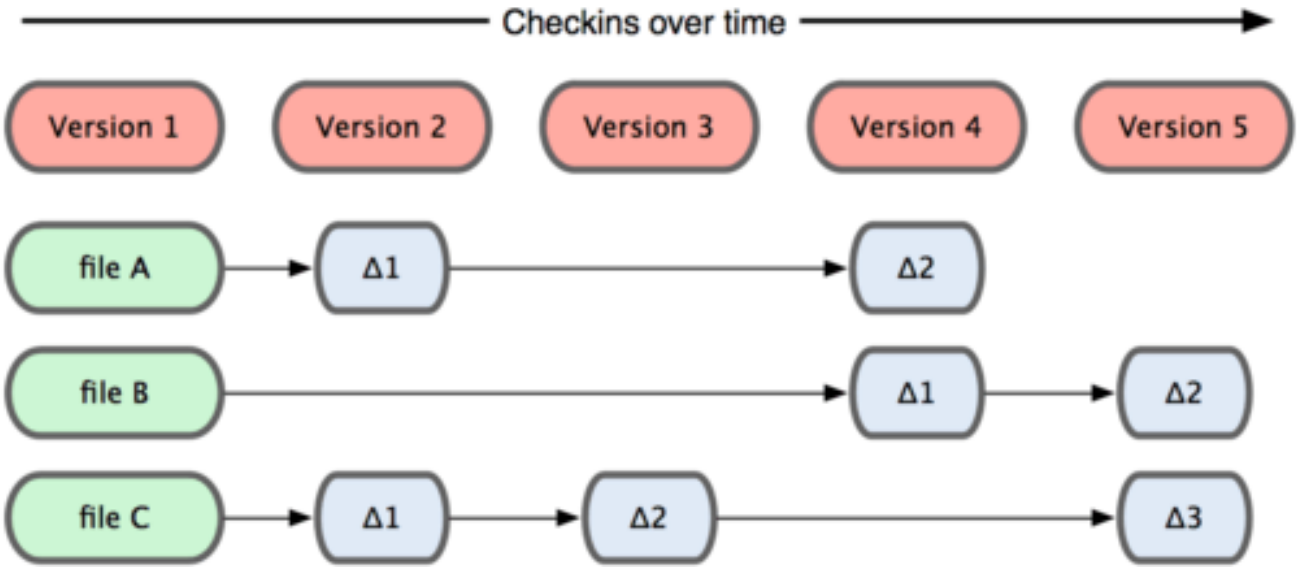


25. února 2020

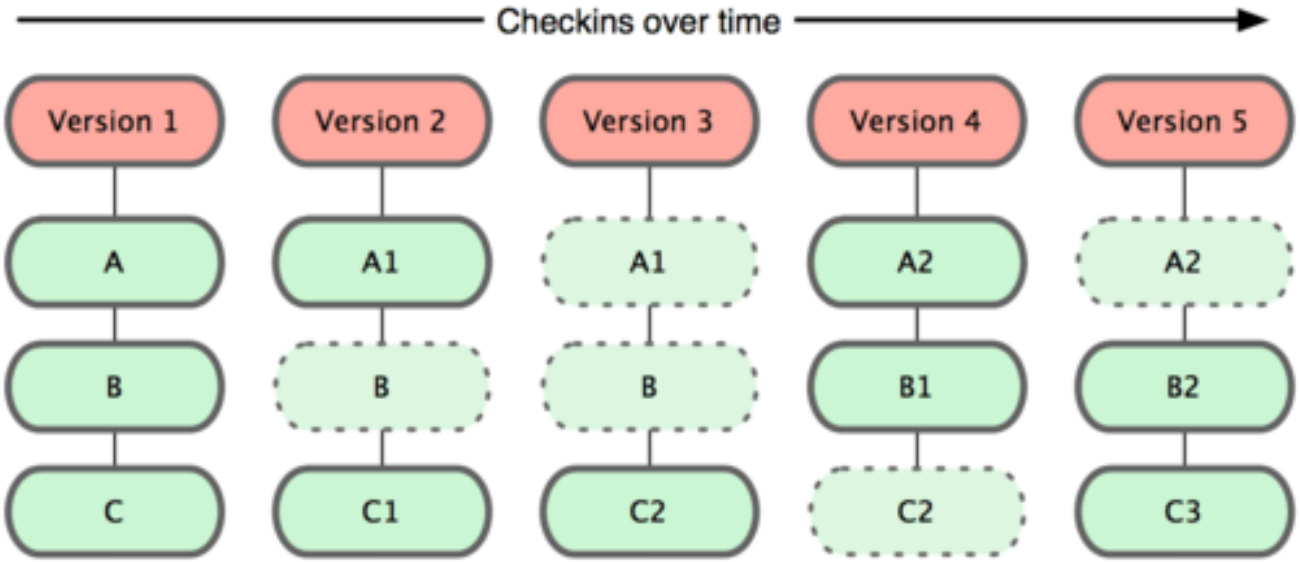
- Distribuovaný verzovací systém
- Neexistuje centrální repozitář, každý vývojář má úplnou kopii (a tedy i zálohu)
- Pro velké soubory mohou vývojáři použít GIT Large File Storage `https://git-lfs.github.com/`
- Submoduly pro zanoření projektů `https://git-scm.com/book/cs/v2/Git-Tools-Submodules`
- Ignoruje prázdné složky
- Místo inkrementů používá snapshoty

Inkrementy vs. Snapshoty

- Inkrementy:



- Snapshoty:



- Lokální (jeden repozitář):

```
git config <key> <value>
```

- Globální (v domovském adresáři pro všechny repozitáře):

```
git config --global <key> <value>
```

- Důležitá nastavení:

```
git config --global user.name "John Doe"
git config --global user.email "john.doe@example.com"
git config --global color.ui auto
```

Vytvoření repozitáře

- Nový repozitář

  `git init`

  - vytvoří lokální repozitář s pracovním stromem v aktuálním adresáři
  - metadata v adresáři `.git`
- Nový repozitář sdílený na serveru

  `git init --bare`

  - vytvoří repozitář bez pracovního stromu
  - metadata přímo v adresáři repozitáře
- Klon existujícího repozitáře

  `git clone <...>`

  - naklonuje vzdálený repozítář

git status

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

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

Stav pracovního stromu

`git diff`

- změny, které 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 --oneline

- Každá revize je ucelený balík změn řešící jednu věc (oprava chyby, nová funkce, . . .).
  - Raději více menších (vyhněte se „Code bomb“ či „Commit bomb“).
- Commit je jedna revize.
- Identifikován hashem (SHA-1)
  - **bf2ca58**1680417f466fbedf35f1a4aa1c4c6c5e9
- Obsahuje zprávu, autora, datum a čas.
- Častá konvence pro zprávy:
  - první řádek = stručné shrnutí
    - je viditelné ve všech výpisech historie
  - druhý řádek je prázdný
  - ostatní řádky = podrobnější informace, např.:
    - proč byla změna provedena
    - podrobnosti řešení
    - vedlejší efekty

Základní pravidla pro zprávy

- Zprávy k revizím musí být smysluplné:
  - **Správně**:
    - Přidán dialog s nápovědou
    - Opraven výpočet faktoriálu
  - **Špatně**:
    - Updated main.c
    - Add files via upload
    - fixes
  - **Určitě NE**:
    - Vulgarismy!

Umístění změn do vyčkávacího prostoru

`git add <soubor/složka>`

- Přidání souborů/složek do vyčkávacího prostoru.

`git mv <zdroj> <cíl>`

- Přejmenování/přesun složky či souboru.
- Pouze `mv` způsobí smazání a vytvoření – ztrátu historie.

`git rm <soubor/složka>`

- Odstranění z disku a označení, že má být odstraněno z repozitáře.

`git rm --cached <soubor/složka>`

- Pouze označení, že má být odstraněno z repozitáře.

`git commit [-m message]`

- Pokud není zadána 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, včetně těch mimo vyčkávací prostor
- pozor – vyhnout se verzování lokálních nastavení pro testování apod.

`git commit <file> [-m message]`

- pouze vybraný soubor

git gui

- pouze základní funkcionalita

gitk

- pouze procházení historie

gitg:

- málo funkcí, ale přehledná historie a commit

git cola

- přehledné, automaticky obnovuje status, pokročilé hledání (hledání commitů dle změněného souboru apod.)

TortoiseGIT

- integrované do Průzkumníka ve Windows

GitKraken

- zdarma s omezeními
- Pro studenty zdarma v rámci GitHub Student Developer Pack

1. Nastavení GITu
   - hlavně jméno a e-mail

```
git config --global user.name "John Doe"
git config --global user.email "jd@example.com"
```

2. Vytvoření lokálního repozitáře
   - `mkdir myprogram && cd myprogram`
   - **`git init`**
3. Přidání všech smysluplných nesledovaných souborů – pokud žádné neexistují, vytvořit `README` a `.gitignore`
   - **`git add`** `<soubor>`
4. Vytvoření první revize (commit)
   - **`git commit`** `-m "Initial commit"`
5. Vytvoření vlastního programu
   - `vim main.c / emacs main.c / subl main.c`
   - **`git add / git rm / git mv`**
6. Vytvoření další revize (commit)
   - **`git commit`** `-m "add main.c"`

Co by repozitář měl obsahovat

- konfigurace nástrojů
  - statická analýza (linter)
  - testovací framework
  - překladač (např. `Makefile`)
  - `...`
- `README`
- `.editorconfig`
  - nastavení konců řádků a odsazení
- `.gitignore`
  - seznam ignorovaných adresářů a souborů

Co by repozitář neměl obsahovat

Ne vše je vhodné verzovat, například:

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

- seznam složek a souborů, které bude GIT ignorovat
- https://github.com/github/gitignore

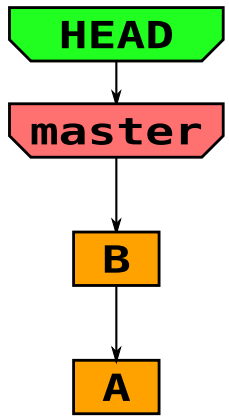
```
# ignoruj soubory .a
*.a

# s vyjímkou lib.a
!lib.a

# ignoruj soubory TODO v tomto adresáři, ale ne v podadresářích
/TODO

# ignoruj vše v adresáři build/
build/

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

# ignoruj všechny soubory pdf v adresáři doc/ a jeho podradresářích
doc/**/*.pdf
```

- Název projektu
- Autoři
- Stručný popis funkce
- Licence
- Instalační instrukce
- Jak přispívat (hlavně u open source)
- Cokoliv dalšího, co dává smysl . . .

- Příklady viz
  `https://github.com/matiassingers/awesome-readme`

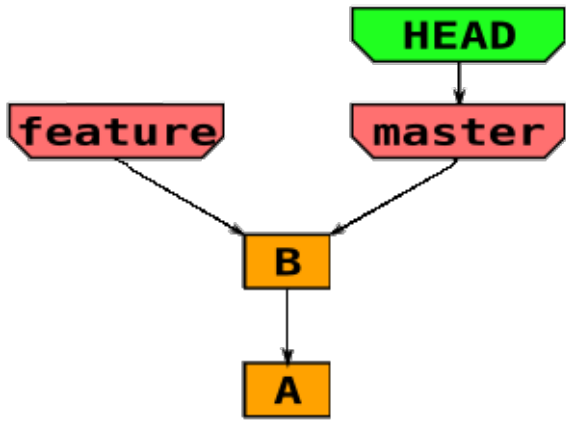
Větve v GITu

Aktuální repozitář



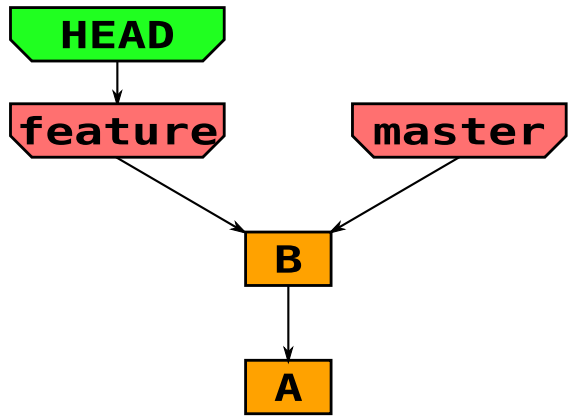
Vytvoření větve pro funkci (feature):

```
git branch feature
```



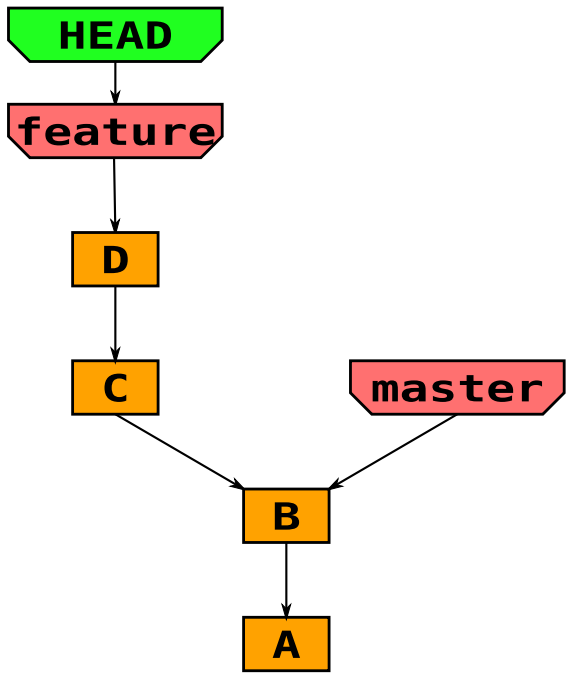
Přepnutí na větev feature:

```
git checkout feature
```



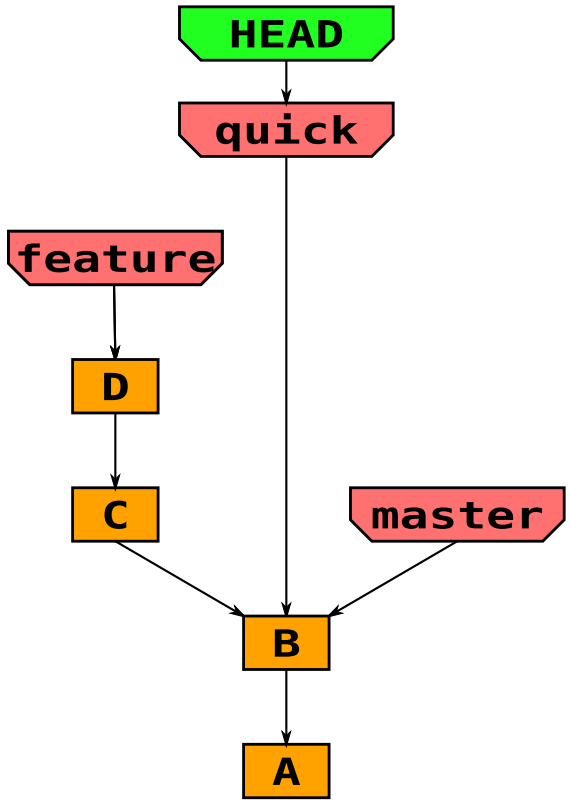
Vývoj funkce (feature).

```
git add ...
git commit -m ...
```



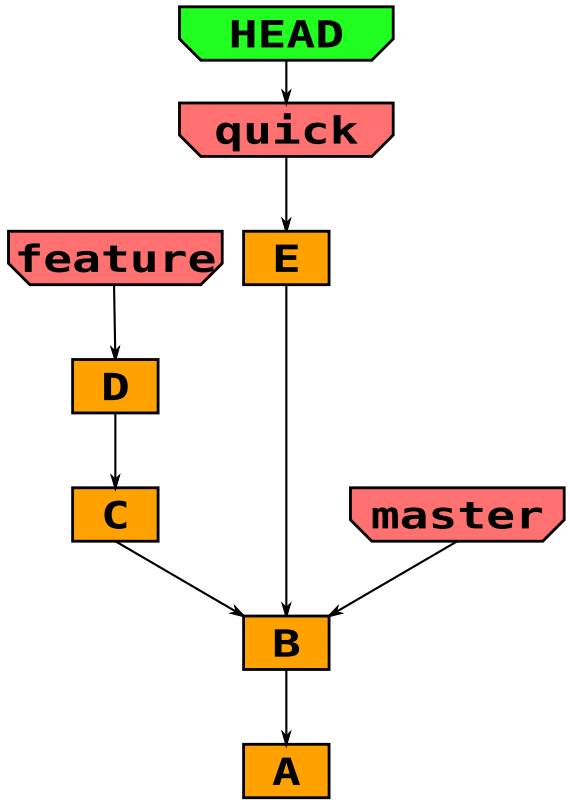
Byla odhalena chyba,
která brzdí vývoj,
a tak se hned opraví.
Vytvoření nové větve:

```
git checkout -b quick
```



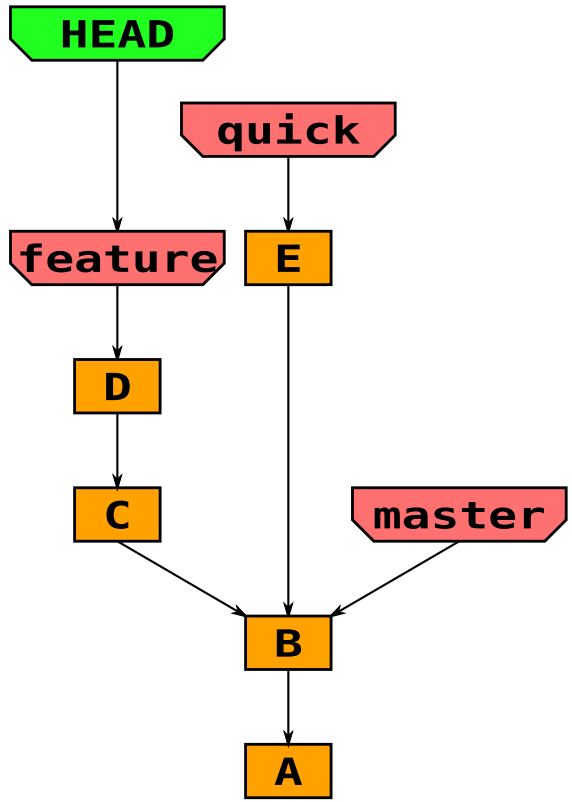
Rychlá oprava chyby.

```
git add ...
git commit -m ...
```



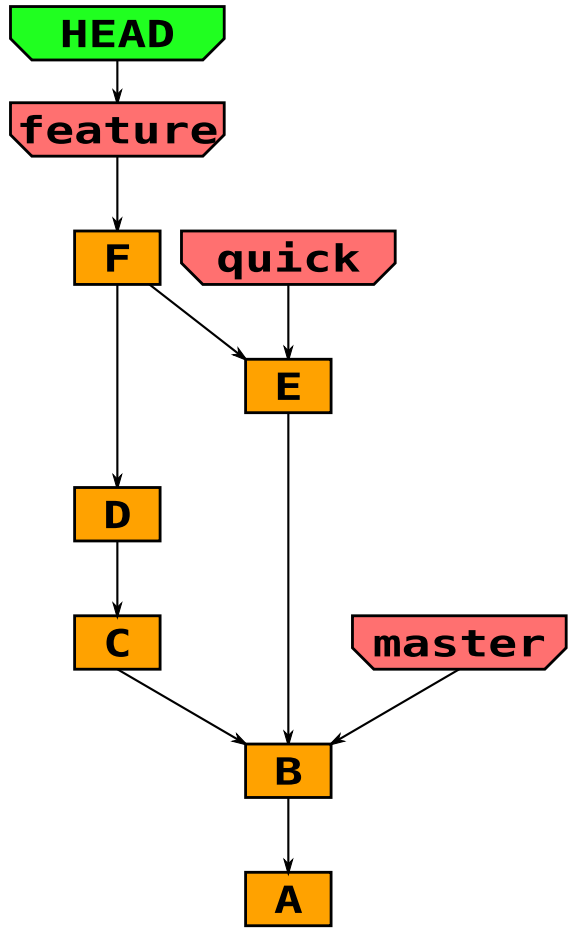
Zpět k vývoji funkce (feature):

```
git checkout feature
```



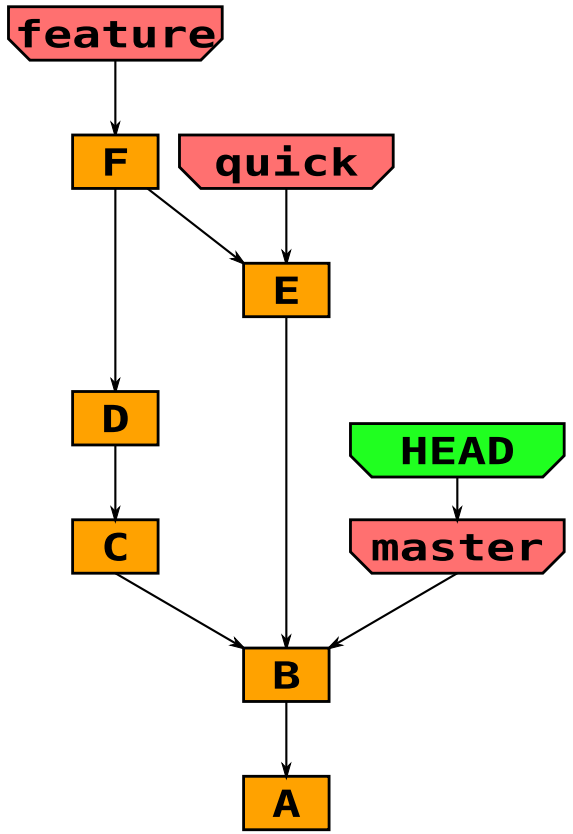
Sloučení opravy:

```
git merge quick
```



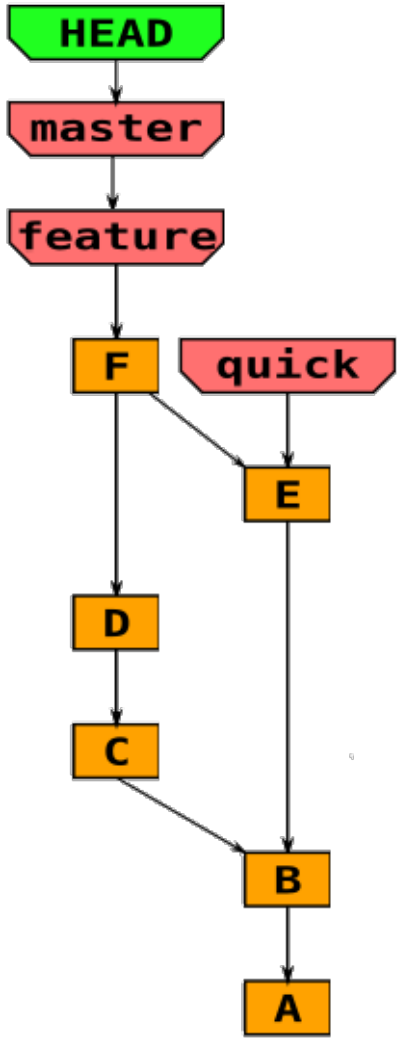
Funkce je hotová
a otestovaná.
Nyní může být integrována.
Přepnutí na hlavní větev:

```
git checkout master
```



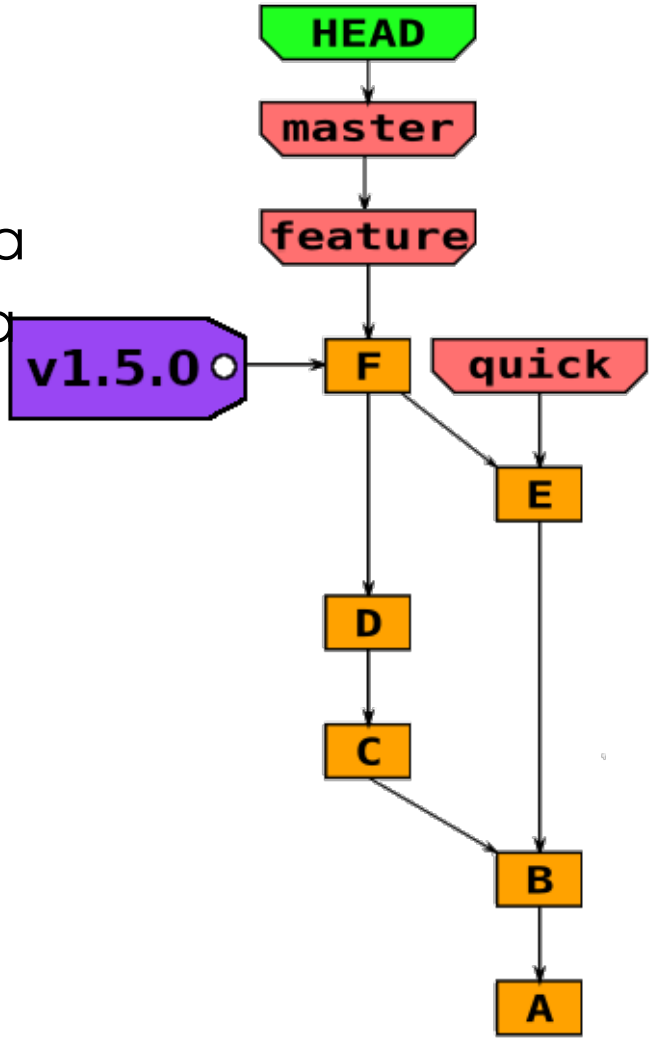
A nyní může být funkce sloučena:

```
git merge feature
```



Také může být vytvořena nová verze a zveřejněna pro zákazníky.
Označíme ji tagem:

```
git tag v1.5.0
```



Dva způsoby slučování

Fast-forward

- slučované větve jsou následovníky v linii
- jde o prostý přesun ukazatele

Merge commit

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

Checkout

`git checkout <destination>`

Umožňuje přepnout prakticky kamkoliv

- `git checkout` **`branch`**
- `git checkout` **`a8d621`** (vytvoří stav *detached head*)
- `git checkout` **`origin/master`** (rovněž *detached head*)

Může také vrátit zpět změny

- `git checkout` **`main.c`**
- `git checkout` **`subdir`**
- `git checkout .` (**nebezpečné** – zahodí všechny změny)

`git reset --soft <revision>`

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

`git reset [--mixed] <revision>`

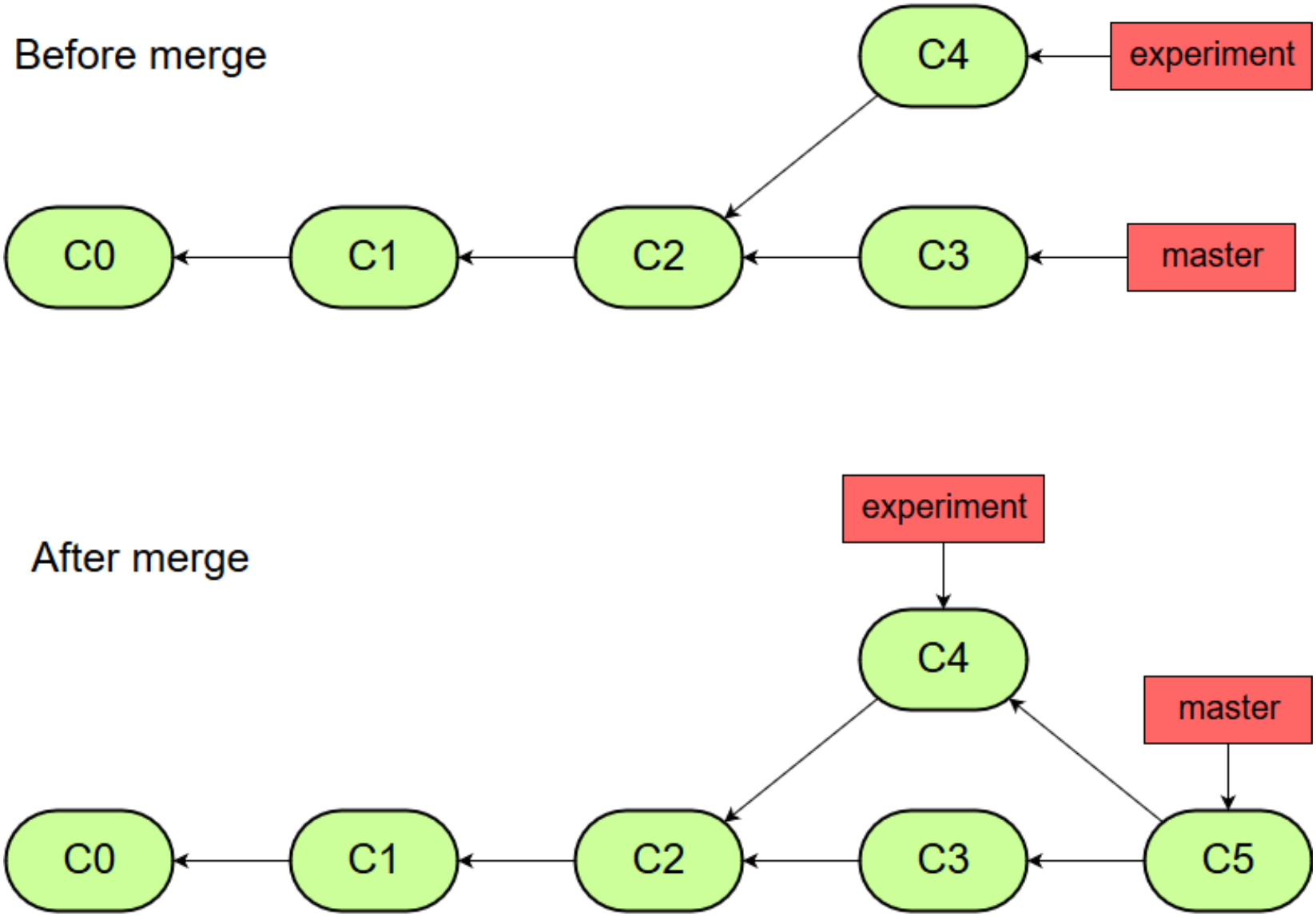
- --soft + vyprázdnění vyčkávacího prostoru (změny však budou zachovány)

`git reset --hard <revision>`

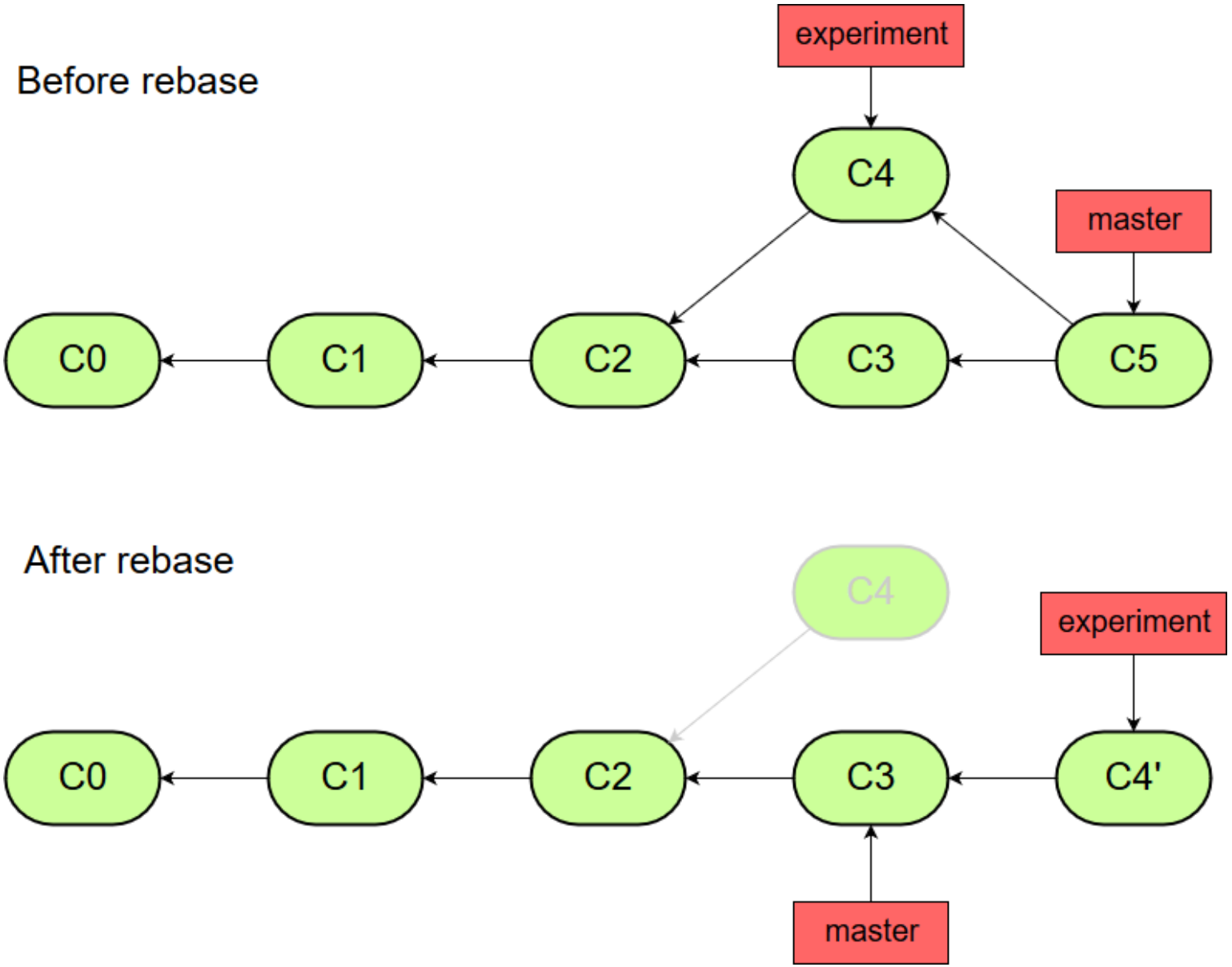
- --mixed + **zruší všechny změny v pracovním stromě**
- **nebezpečné**, složitě se vrací zpět

- příkaz **merge** kombinuje změny ze dvou větví
- příkaz **rebase** bere změny z commitů jedné větve a aplikuje je na ukazatel HEAD druhé větve.
- Výsledkem je čistší historie, ale je to nebezpečné.
  - Jsou zde commity vytvořené automaticky na základě výsledků porovnání větví.
  - V některých případech může být výsledek špatný/nefunkční.

Příklad sloučení



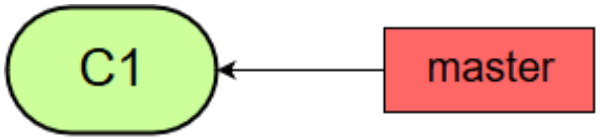
Příklad přeskládání



Přeskládání

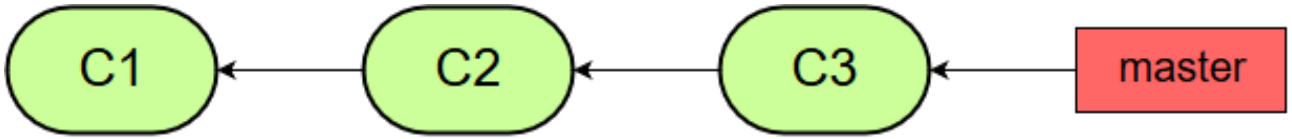
- Neprovádějte přeskládání u revizí, které jste odeslali do veřejného repozitáře.
  - Udělá to pouze nepořádek v historii a kolegové z toho nebudou moc nadšeni.
- Přeskládání by se mělo používat hlavně pro pročištění lokální historie před odesláním do veřejného repozitáře.

Vývojář A si naklonuje repozitář a začne pracovat

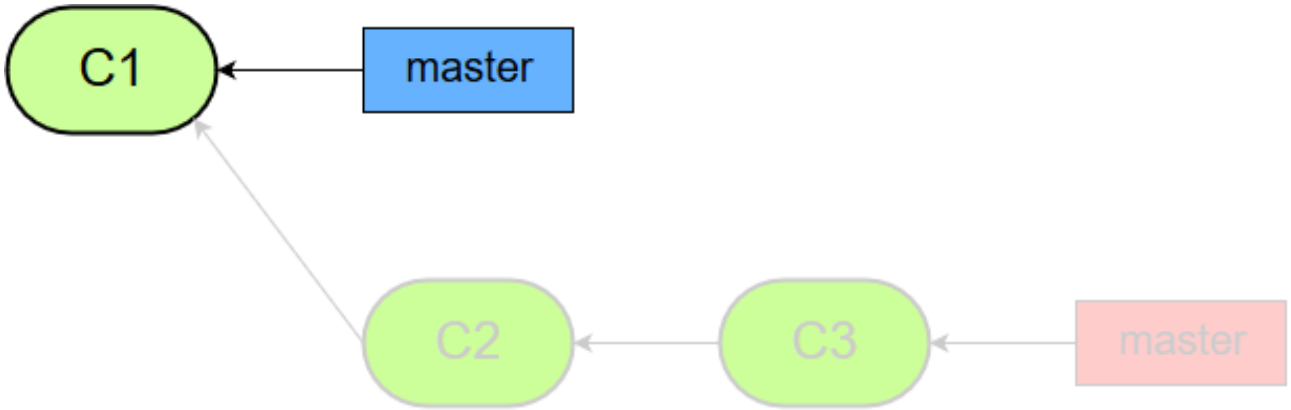


Vývojář A vytvoří nějaké commity . . .

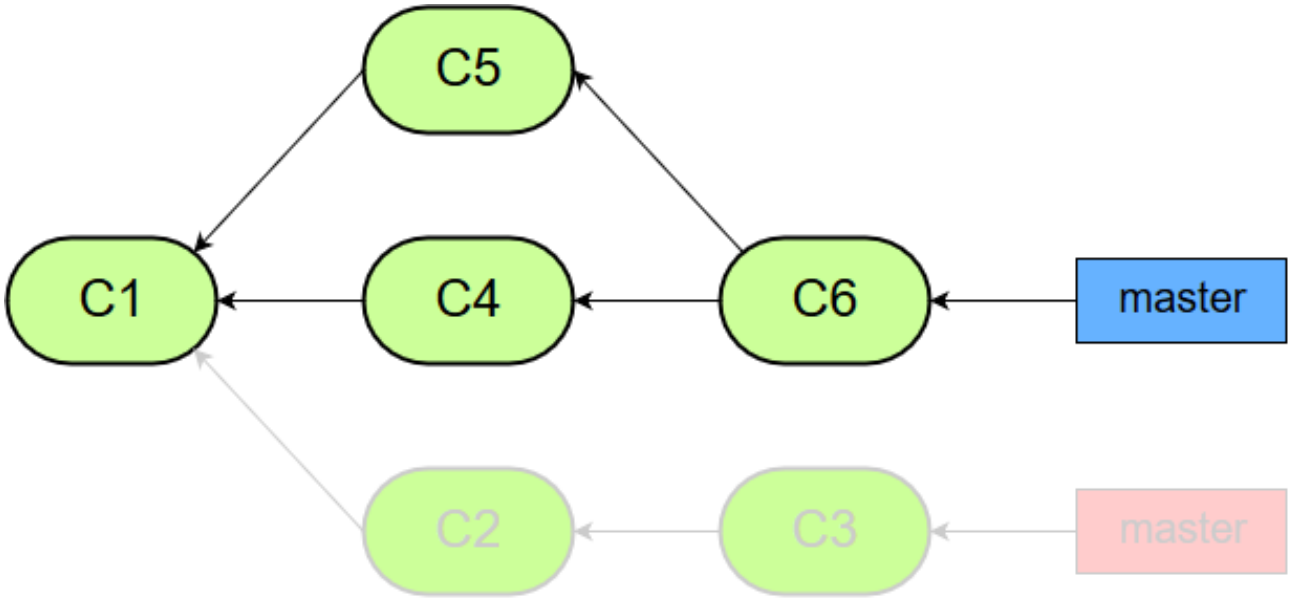
Vývojář A vytvoří další commit



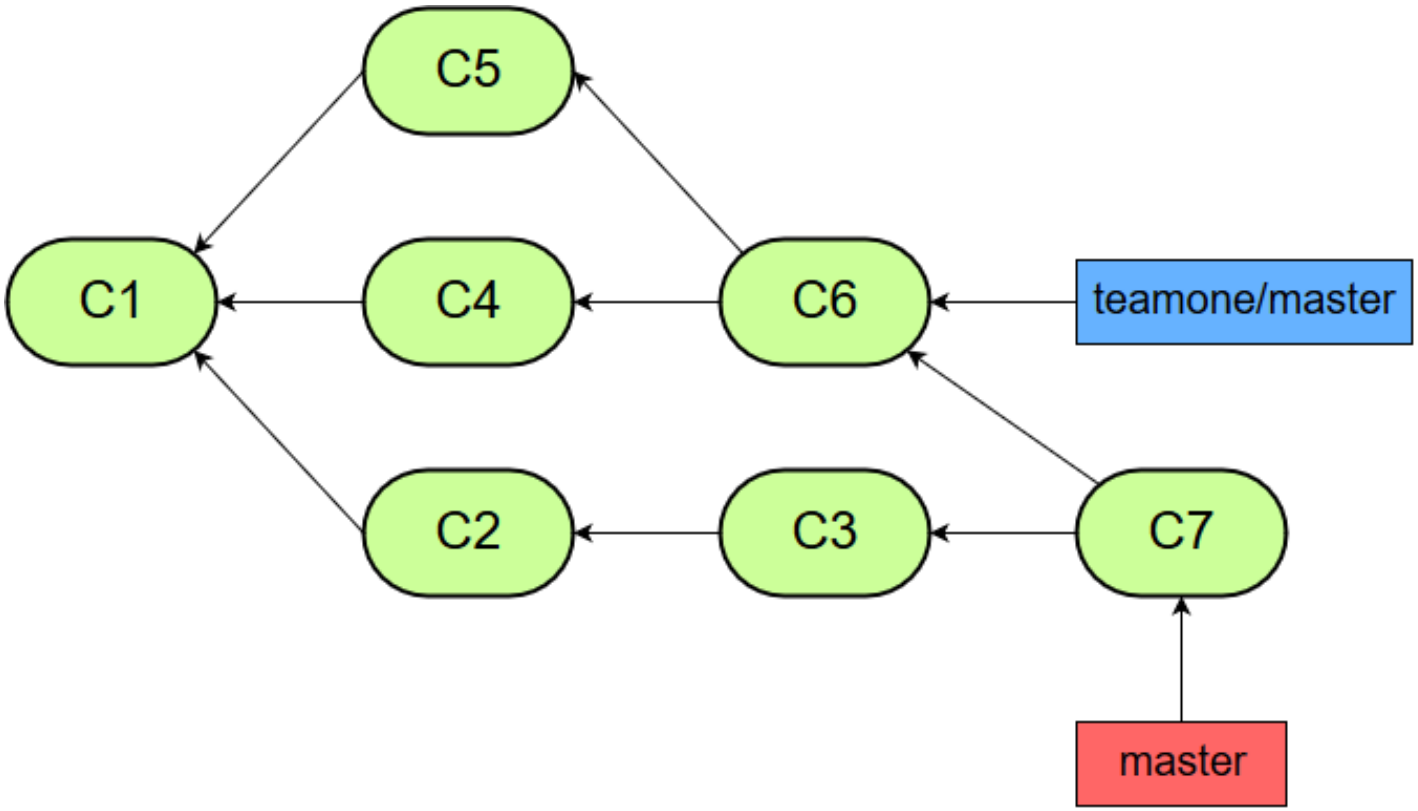
Vývojář B si naklonuje repozitář a začne pracovat . . .



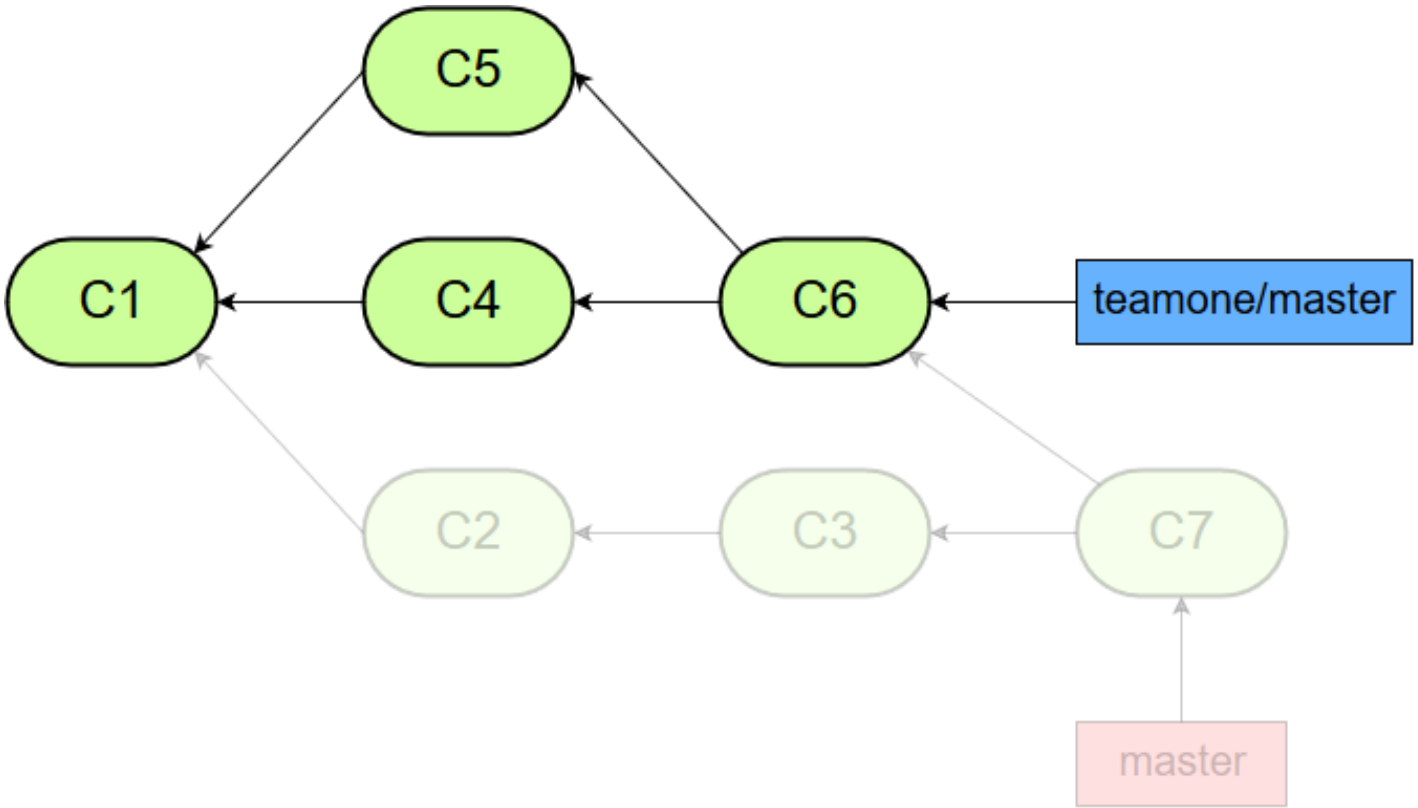
Vývojář B pracuje s větvemi a odešle změny do sdíleného repozitáře



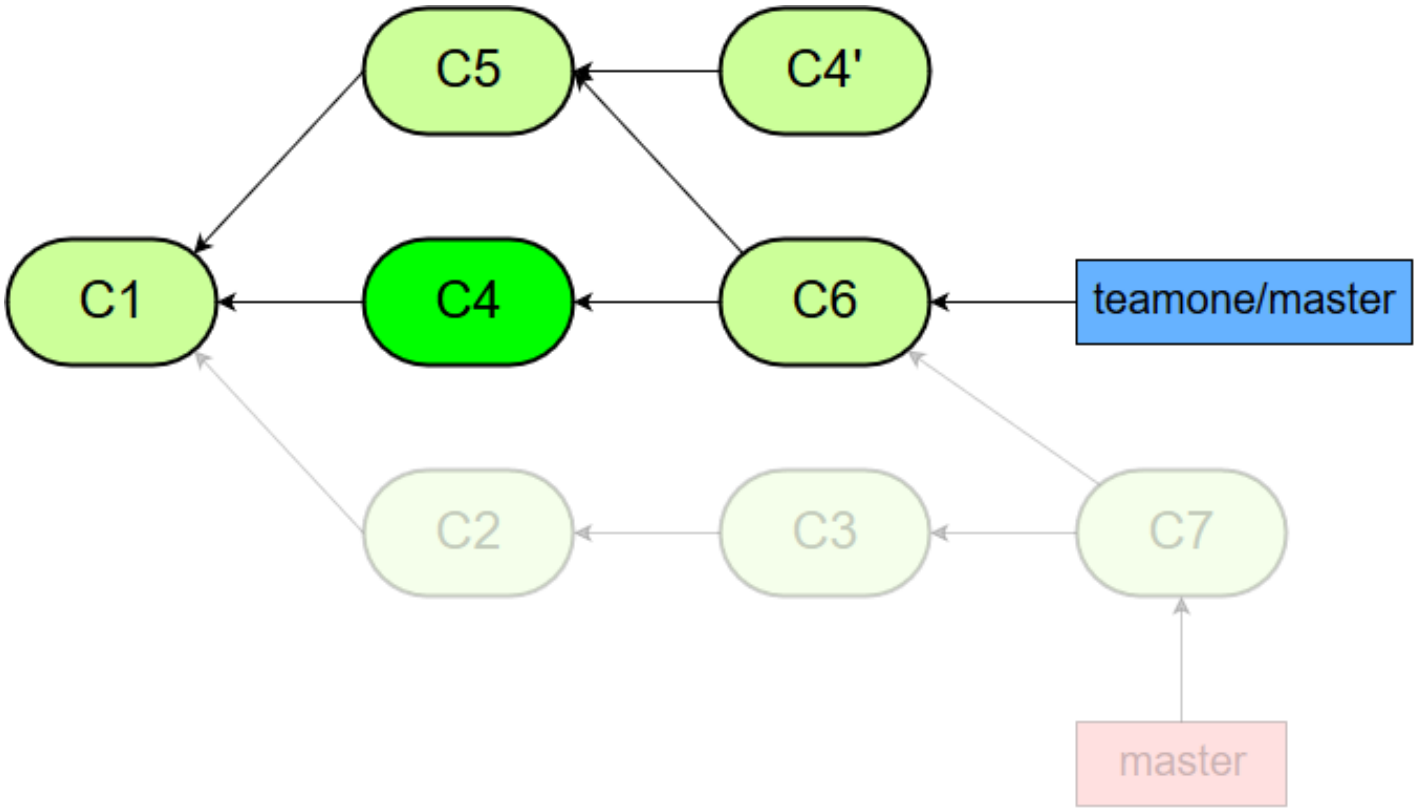
Vývojář A si zaktualizuje hlavní větev



Vraťme se k pohledu vývojáře B

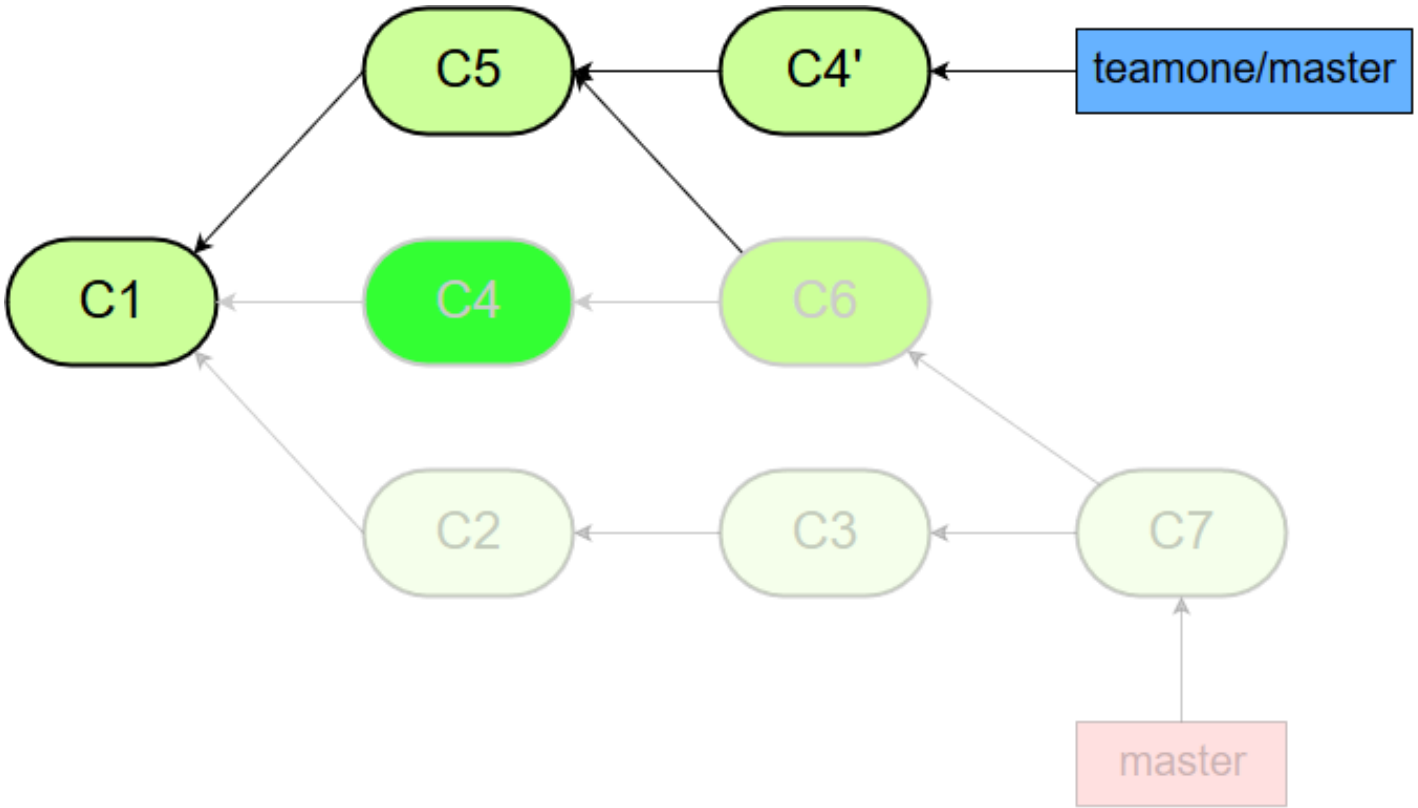


Vývojář B provede přeskládání a commity C4 a C6 již neexistují

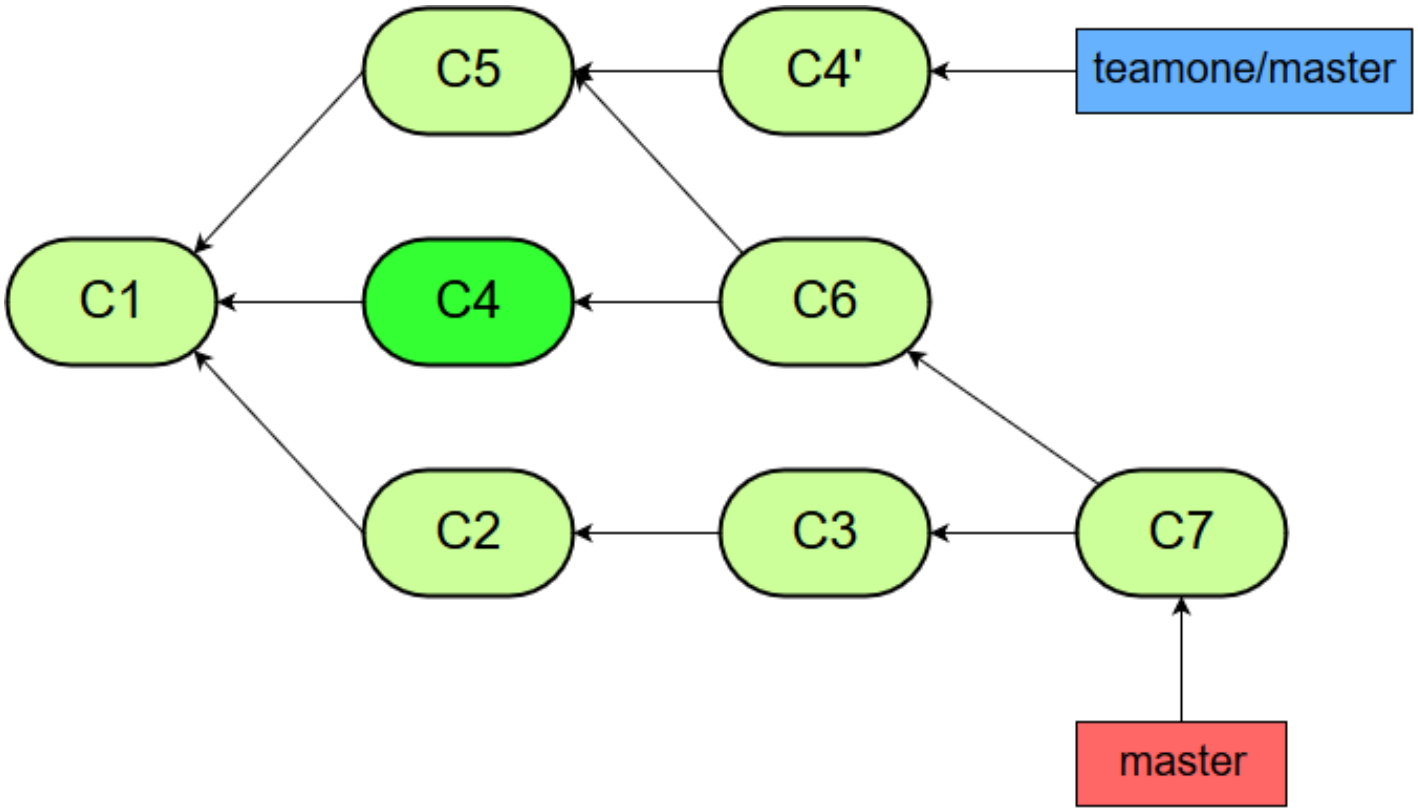


Problém s přeskládáním

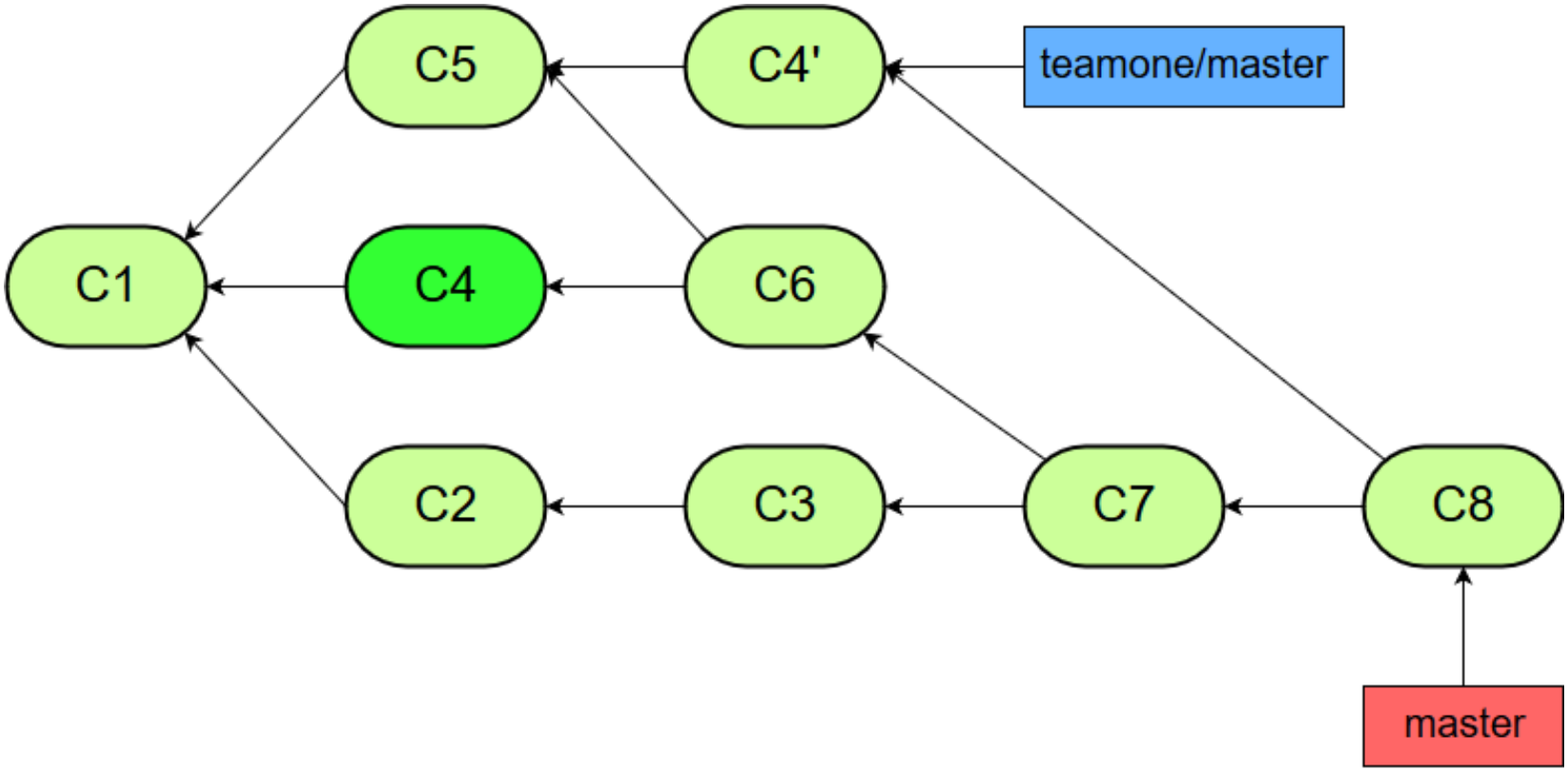
Vývojář B odešle změny na vzdálený repozitář



Vývojář A však tyto commity stále má . . .



. . . a pokud vývojář A sloučí změny s hlavní větví, commit C4 bude mít duplikovaný



Vzdálené repozitáře

- Nutné pro práci v týmu.
  - Bez nich by měl každý programátor vlastní kopii.
- Může existovat více vzdálených repozitářů.
- Hlavní repozitář se nazývá **origin**.
- Když se klonuje vzdálený repozitář, je automaticky nastaven jako **origin**.
- Vzdálený repozitář může být také přidán později:

```
git remote add origin <zdroj>
```

Zdrojem může být:

ssh

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

http(s)

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

adresář na disku

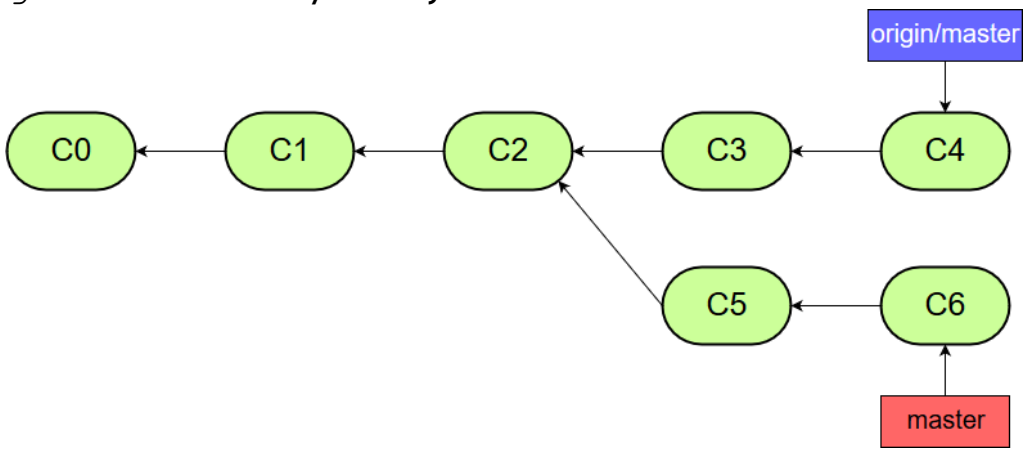
- /opt/repos/example.git

protokol GITu

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

Vzdálené repozitáře

- Stažení aktualizací z repozitáře:
  - `git fetch` **`origin`**
- Větve ze vzdáleného repozitáře jsou oddělené!
  - **origin** obsahuje větev **master**
  - **vývojář** má větve **master** a **origin/master**
  - je na vývojáři, aby si větve sloučil
    - `git merge` vytvoří slučovací 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 aktualizované lokální větve, jinak vyvolá chybu – nutno aktualizovat pomocí `git pull`:
  - `git pull origin master`
- Tagy se odesílají nezávisle na větvích:
  - `git push` **`origin v0.8`**
  - `git push` **`origin --tags`**

- Fetch stáhne a aktualizuje objekty a odkazy z jiného repozitáře.
- Fetch nevytváří lokální větve automaticky.

- **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**

Sledující větev (Tracking branch)

- Lokální větev může sledovat nějakou vzdálenou větev
  - typicky se jmenují obě stejně (branch a origin/branch)
  - „branch sleduje origin/branch“
- `pull` a `push` pak fungují i bez parametrů
- Sledování se nastaví automaticky:
  - `git clone` (master sleduje origin/master)
  - `git checkout -b branch` **`origin/branch`**
  - `git checkout --track origin/branch`
- Dodatečně lze nastavit:
  - `git branch -u origin/master`

1. Nastavení GITu
   - hlavně jméno a e-mail

     ```
     git config --global user.name "John Doe"
     git config --global user.email "jd@example.com"
     ```
2. Vytvoření centrálního repozitáře (pouze první vývojář)
   - `ssh dytrych@ivs.fit.vutbr.cz`
   - `cd /<sdílený adresář>`
   - **`git init --bare`**
3. Vytvoření lokálního klonu
   - **`git clone`** `ssh://dytrych@ivs.fit.vutbr.cz/<sdílený adresář>`
4. Vytvoření `README` a `.gitignore` a přidání do repozitáře (pouze první vývojář)
   - **`git add`** `<soubor>`
5. Vytvoření první revize (pouze první vývojář)
   - **`git commit`** `-m "První commit"`
6. Odeslání první revize (commit) (pouze první vývojář)
   - **`git push`** `origin master`

7. Vytvoření nové větve
   - `git branch myFeature`
   - `git checkout myFeature`
8. Práce na vlastní funkci (vlastní program)
   - `vim main.c / emacs main.c / subl main.c`
   - `git add / git rm / git mv`
9. Vytvoření další revize (commit)
   - `git commit -m "add main.c"`
10. Odeslání vlastní větve
    - `git push origin myFeature`
11. Integrace
    - `git pull origin master`
    - `git checkout master`
    - `git merge myFeature`
    - `git pull origin master`
    - `git push origin master`

Odeslání lokálního repozitáře do prázdného vzdáleného

- **`git remote add`** `origin ssh://dytrych@ivs.fit.vutbr.cz/<sdílený adresář>`
- **`git push`** `-u origin master`
  - `-u` přidá upsteam (sledovací) odkaz

Hostingy

- `https://github.com/`
  - Pro studenty privátní repozitáře zdarma `https://education.github.com/pack`
  - Project management, isue tracking, pull requests, wiki, GitHub Pages (web spravovaný přes GIT), GitHub Gist (sdílení souborů – obdoba Pastebin), ...
  - V podstatě sociální síť
- `https://bitbucket.org/`
  - Privátní repozitáře zdarma až pro 5 uživatelů.
  - Integrace s Trello / Jira pro issue tracking, ...
- `https://about.gitlab.com/`
  - Open-source (GitLab Community Edition)
  - Jednou nechtěně smazali část ostrých dat.
- `https://merlin.fit.vutbr.cz/repo/`
  - Na 1 semestr, pak se smaže!
  - Nelze dodatečně přidávat spolupracovníky.
- `ivs.fit.vutbr.cz`
  - `/ivs-proj2/<složka týmu>`
  - Pouze pro tento předmět, pak se zaarchivuje.

Další užitečné funkce

`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'.
```

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

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

- Vykřičník značí 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'`
  - spustí gitk se všemi větvemi
- `git config --global alias.la '!git config -l | grep alias | cut -c 7-'`
  - vypíše všechny aliasy

```
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"
```

- Grafický barevný výpis historie do konzoly.

Zjištění autora změny

`git blame <soubor>`

- Ke každému řádku souboru vypíše informace o posledním commitu, který jej změnil.

`git blame <soubor> -L 40,45`

`git blame <soubor> -L 40,+5`

`git blame <soubor> -L /regexp/`

`git blame <soubor> -L :funkce`

- `-L` umožňuje omezit rozsah.

- Skrýš (Stash) se využívá před přepnutím do jiné větve, pokud nechceme udělat commit (uložíme změny do skrýše a po návratu aplikujeme).

`git stash`

- Uloží změny v pracovním stromě do zásobníku (1 sada změn v zásobníku).

`git stash pop`

- Vyjme sadu změn ze zásobníku a aplikuje na pracovní strom.

`git stash apply`

- Aplikuje sadu změn z vrcholu zásobníku na pracovní strom, ale ponechá ji v zásobníku.

`git stash apply <n>`

- Aplikuje sadu změn z dané pozice v zásobníku na pracovní strom, ale ponechá ji v zásobníku.

`git stash drop <n>`

- Zahodí sadu změn z dané pozice v zásobníku.
- Bez uvedení pozice zahodí sadu z vrcholu zásobníku – potřebné např. po chybě při aplikaci změn (konfliktu).

`git stash list`

- Vypíše obsah zásobníku.

```
stash@{0}: gitg auto stash ...
stash@{1}: WIP on master ...
```

(Work In Progress)

`git stash branch`

- Vytvoří novou větev ze sady změn na vrcholu zásobníku.
- Commit, kde byla sada změn vytvořena, + obsah sady změn.

`git cherry-pick <commit>`

- Aplikace vybraného commitu na aktuální větev).
- S parametrem `--edit` se dotáže na novou zprávu k commitu.
- S parametrem `--no-commit` nevytvoří commit, ale pouze aplikuje změny na pracovní strom.

`git revert <commit>`

- Vytvoří commit, který revertuje změny provedené jiným commitem.
- Provádí se s čistým pracovním stromem.
- S parametrem `--no-commit` nevytvoří commit, ale pouze aplikuje změny na pracovní strom.

`git help`

- Nápověda

`git help <příkaz>`

Jak začít?

- Zvolte si hosting
- Dohodněte si konvence a workflow
  - Jak budete pojmenovávat větve
  - Jak budete organizovat větve
  - V jakém jazyce budou commit messages
  - Kdo založí repozitář
  - Kdo bude integrovat
  - ...
- Nemáte-li zkušenosti, vyzkoušejte si GIT na testovacím repozitáři.

Odkazy

- `https://git-scm.com/docs/`
- `https://git-scm.com/book/cs/v2` (částečný překlad)
- `https://git-scm.com/book/en/v2`
- `https://www.vogella.com/tutorials/Git/article.html`
- Když se něco pokazí: `https://ohshitgit.com/`

Něco navíc . . .

- Skript `git.sh` v adresáři `sftp://dytrych@ivs.fit.vutbr.cz:22/ivs/`
- Synchronizace všech zadaných repozitářů. Pro každý:
  - `stash` všech změn,
  - `fetch` ze všech vzdálených repozitářů,
  - `checkout` každé změněné větve,
  - `pull` ze všech vzdálených repozitářů do každé změněné větve,
  - `push` každé změněné větve do všech vzdálených repozitářů,
  - `stash pop`
- Vytvoří všechny chybějící lokální větve.
- Když se něco pokazí, zastaví se a lze manuálně vyřešit problém a pokračovat nebo skončit.
- Přijdu do práce, spustím, pracuji, . . . spustím, jdu domů, spustím, . . .
- Nedostatky:
  - Nepodporuje submoduly.
  - Nedořešená práce s tagy.

- Přihlašování na server bez zadávání hesla.
- Vygeneruje se privátní a veřejný klíč. Privátní máte u sebe, veřejný umístíte na server (do souboru `authorized_keys` ve složce `.ssh` ve svém domovském adresáři) nebo do konfigurace na hostingu jako je GitHub.
- Postup:
  - `ssh-keygen`
  - Zadáte-li heslo, bude požadováno pro použití klíče (bezpečnější, lze využít `ssh-agent` a zadávat jej pouze 1x za sezení – viz `http://rabexc.org/posts/using-ssh-agent` a `https://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch02_05.htm`).
  - Umístíme veřejný klíč na server: `ssh-copy-id <server>` nebo `cd /home/login/.ssh/ && ssh login@ivs.fit.vutbr.cz "mkdir -p /home/login/.ssh" && scp id_rsa.pub login@ivs.fit.vutbr.cz:/home/login/.ssh/authorized_keys`
- Nemáte-li rádi konzoli, GitKraken umí vygenerovat klíč a pak jej pomocí `https://winscp.net/` nakopírujete na správné místo.

Děkuji za pozornost!

Pokročilý GIT od Red Hat bude 14. 4. 2020.