

Učebnice Gitu a GitHubu

k videím na YouTube

David Šetek

Odkaz na učebnici

<https://bit.ly/hsb-git>



Seznam videí na kanále YouTube

David Šetek - Hackni svou budoucnost

https://www.youtube.com/playlist?list=PLQ8x_VWW6AkuVs1oyWth3IXA4D4jxD_7F

Úvod, Git, GitHub, základní práce v Gitu - 1. část

1. Co je to Git

Video: https://youtu.be/_qSB6jN4A3s

Odkaz na prezentaci v PDF:

https://drive.google.com/file/d/13e_0GT3J9suFkgdIdSLgMxrae6e8IAoc/view?usp=sharing

2. Rozdíl mezi Gitem a GitHubem

Video: <https://youtu.be/mHQGuQseZFE>

Odkaz na prezentaci v PDF:

<https://drive.google.com/file/d/1fBCTOVv5yoAIW-uD8R-5SyS9qIDBR4MO/view?usp=sharing>

3. Kdo Git a GitHub používá

Video: <https://youtu.be/WD92wrsiwsI>

Odkaz na prezentaci v PDF:

<https://drive.google.com/file/d/1EhZh-lruz9x6-cD9KXUdaByze21rBKV2/view?usp=sharing>

4. Instalujeme git

Video: <https://youtu.be/Px4SzsEUjzI>

Git stáhnete na této adrese - vpravo uvidíte nápis Download for Windows nebo Download for Mac: <https://git-scm.com/>

Nebo zde je stránka s instalacemi a jen si vyberete tu svou: <https://git-scm.com/downloads>

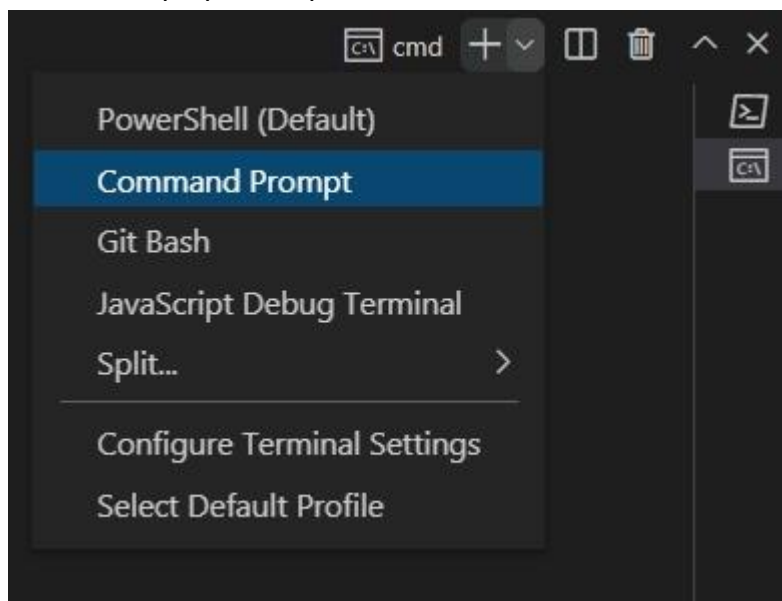
Ve VS code dejte:

git --version

a pokud se ukázala jakákoliv verze Gitu, tak je vše v pořádku. Mělo by to vypadat nějak takto:

```
C:\Users\David\Desktop\GitGitHub>git --version  
git version 2.33.1.windows.1
```

Jaký terminál používat? Pokud by cokoliv nefungovalo v terminálu ve VS code, tak si můžete ve VS code přepnout v pravém horním rohu terminál na Command Prompt a zkusit ten.



Nebo společně s Gitem by se vám měl nainstalovat Git Bash. Jen ho vyhledáte ve svém počítači a spustíte.



5. Terminál - základní příkazy 1

Video: <https://youtu.be/k1UndHXyvLA>

Příkazy ve videu:

ls
ls -a
ls -l
cd
pwd

Tabulka příkazů:

https://docs.google.com/spreadsheets/d/122QFr0kYAKiWndrw2n_rHbDe2HaxGWOvSqxbbXTRzw/edit?usp=sharing

Seznam příkazů na Macu:

<https://www.makeuseof.com/tag/mac-terminal-commands-cheat-sheet/>

Pokud by vám na Macu nefungoval příkaz clear, tak zkuste místo toho Ctrl + L

6. Terminál - základní příkazy 2

Video: <https://youtu.be/18QSbN8KZMA>

Příkazy ve videu:

touch
mkdir

7. Terminál - základní příkazy 3

Video: <https://youtu.be/9uHs9LwoFPw>

Příkazy ve videu:

rm

rm -rf

8. Trénujeme příkazy

Video: <https://youtu.be/jl9RapjqV1I>

Zadání

Co máme udělat

1. Navigujte se na plochu
2. Zde vytvořte složku s názvem Prikazy
3. Uvnitř složky vytvořte 3 další složky (slozka1, slozka2 a slozka3)
4. Uvnitř slozka2 vytvořte dva soubory test1.txt a test2.txt
5. Oba dva soubory test1.txt a test2.txt smažte
6. Smažte celou složku Prikazy

9. Zakládáme uživatelské jméno a email

Video: <https://youtu.be/xSittyBMq98>

```
git config --global user.name "David Šetek"
```

```
git config user.name
```

```
git config --global user.email david.setek40@gmail.com
```

```
git config user.email
```

```
C:\Users\David\Desktop\GitGitHub>git --version
git version 2.33.1.windows.1

C:\Users\David\Desktop\GitGitHub>git config --global user.name "David Šetek"

C:\Users\David\Desktop\GitGitHub>git config user.name
David Šetek

C:\Users\David\Desktop\GitGitHub>git config --global user.email david.setek40@gmail.com

C:\Users\David\Desktop\GitGitHub>git config user.email
david.setek40@gmail.com
```

10. Zakládáme repository pomocí git init a git status

Video: <https://youtu.be/VnEztRwGtOU>

git status

fatal: not a git repository (or any of the parent directories): .git

git init

Initialized empty Git repository in C:/Users/David/Desktop/GitGitHub/.git/

git status

On branch master

No commits yet...

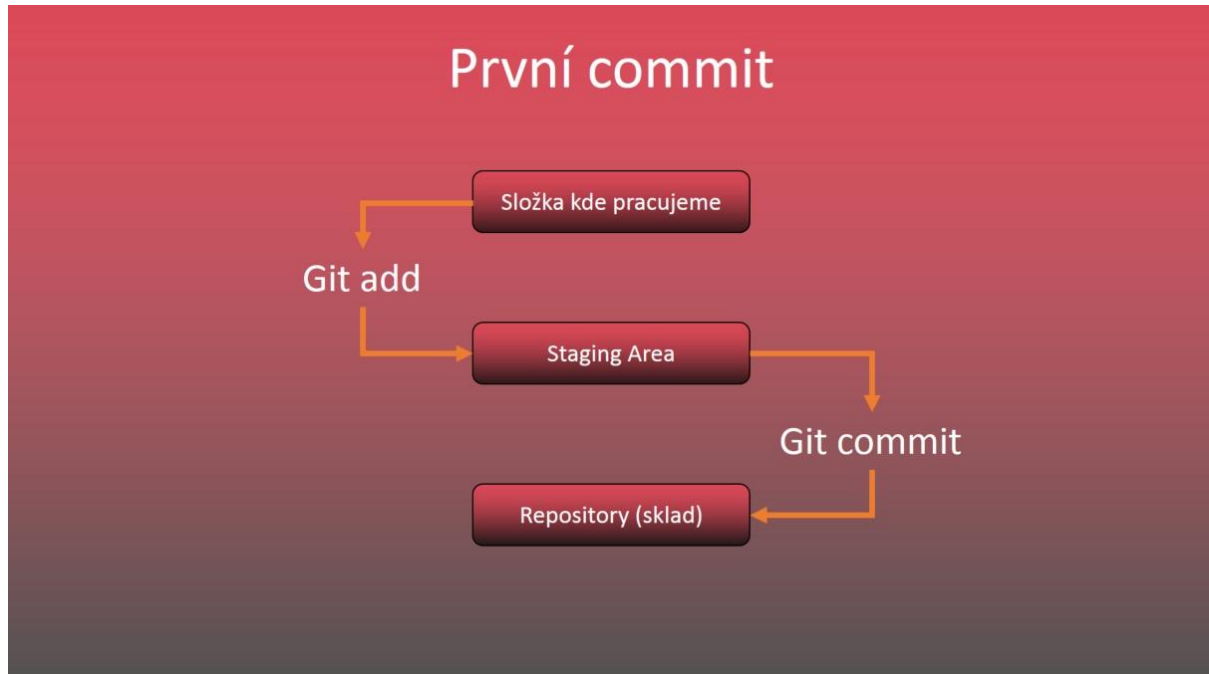
Smazání souboru .git

rm -rf .git

11. První commit a staging area, vypsání commitů (git add, git commit, git log)

Video: <https://youtu.be/IWq0AEmBp5M>

Jak funguje odesílání nových souborů a změn do **Staging area** a poté do **Repository**



Přidání do Staging area

git add index.html

Commitnutí ze Staging area do Repository (skladu)

git commit -m "Vytvořena základní struktura HTML"

Ujištění se, že nemáme co posílat do Staging area a do Repository (nemusíte dávat)

git status

Vypíšeme si naše commity, ke kterým se pak v budoucnu můžeme vracet

git log

POZOR - Pokud by se vám po příkazu **git commit -m "vaše zpráva"** neobjevil řádek pro zadání dalšího příkazu (např. **git status** nebo cokoliv jiného), tak zmáčkněte **klávesu Q** a ono vás to pustí k zadání dalšího příkazu.

12. Procvičuje git add, git commit a git log

Video: <https://youtu.be/rMUerXkjMbk>

13. Kratší výpis díky git log --oneline a jak funguje git add s tečkou, zobrazení jednoho commitu

Video: https://youtu.be/gKclNp6yL_Q

Přidání všeho ze Stage area (za add musí být tečka)

git add .

Vypsání commitů každý jen na jeden řádek

git log --oneline

```
C:\Users\David\Desktop\GitGitHub>git log --oneline
42c26e3 (HEAD -> master) Přidán odstavec do 0 nás a pozadí hlavičky
c4e92ea Vytvoření style.css, základní styly a jejich napojení
7dd68f2 Vytvořena sekce 0 nás
eb2a8c2 Změna názvu firmy
f0fd59f Vytvořen HTML kód hlavičky
1745602 Vytvořena základní struktura HTML
```

Zjištění informací o jednom konkrétním commitu

git show eb2a8c2

```
C:\Users\David\Desktop\GitGitHub>git show eb2a8c2
commit eb2a8c24f8074f5fda13097a2843d4d9c249f73b
Author: David Šetek <david.setek40@gmail.com>
Date: Sat Oct 23 07:18:49 2021 +0200
```

Změna názvu firmy

```
diff --git a/index.html b/index.html
index f57a360..ad147dc 100644
--- a/index.html
+++ b/index.html
@@ -8,7 +8,7 @@
 </head>
 <body>
   <header>
-    <h1>David's design</h1>
+    <h1>David's</h1>
   </header>
 </body>
 </html>
\ No newline at end of file
```

14. Přepínáme mezi commity (cestujeme v čase)

Video: <https://youtu.be/wiAGT4GYdfU>

Toto je můj současný stav commitů (vy můžete mít jiný)

git log --oneline

```
42c26e3 (HEAD -> master) Přidán odstavec do O nás a pozadí hlavičky
c4e92ea Vytvoření style.css, základní styly a jejich napojení
7dd68f2 Vytvořena sekce O nás
eb2a8c2 Změna názvu firmy
f0fd59f Vytvořen HTML kód hlavičky
1745602 Vytvořena základní struktura HTML
```

Přepnutí do libovolného commitu (jako příklad si vyberu úplně první commit z předešlého obrázku):

git checkout 1745602

Přepnutí zpět do posledního commitu (přepne mě zpět na veškerou historii mých commitů):

git checkout master

15. Velké společné opakování

Video: <https://youtu.be/ASJur8ezWWU>

Příkazy použité v procvičovacím videu:

git init

git status

git add .

git commit -m "text vaší zprávy"

git log

git log --oneline

git checkout 5594b32

clear

git checkout master

16. Vim editor a jak ho změnit na VS code, Sublime Text atd.

Video: <https://youtu.be/9gjsldsX9Ps>

Při použití git commit bez dalších možností se commit otevře v přednastaveném editoru **git commit**

Zde najdete příkazy pro různé typy editorů:

<https://git-scm.com/book/en/v2/Appendix-C%3A-Git-Commands-Setup-and-Config>

Vim editor

Spuštění pomocí tohto příkazu:

vim test1.txt

Pokud chcete do editoru zapisovat, tak stiskněte na klávesnici klávesu **i** (pokud nefunguje, tak ještě dejte klávesu ESC a pak teprve i)

Pokud chcete z editoru vim vyskočit, tak dejte klávesu ESC na klávesnici a poté zapište **:wq**

Tímto příkazem vypíšete obsah souboru

cat test1.txt

Pro otevření souboru ve VS code použijte tento příkaz

code test1.txt

17. Přidání dalších věcí do již proběhnutého commitu

Video: https://youtu.be/mMC_AAsD8E8

Tímto příkazem přidáte vše ve Stage area do posledního commitu

git commit --amend

18. Gitignore - soubory, kterých si git nemá všímat

Video: <https://youtu.be/b9yhZWzrVY>

Některé soubory nechceme commitovat a dávat k dispozici veřejnosti

Vytvoříme soubor .gitignore

touch .gitignore

a do něj napíšeme seznam souborů a složek, které nechceme commitovat. Např.:

```
secrets.txt
```

Pokud by to byla složka, tak musíme názevSložky/ (i s lomítkem)

Musíme ale .gitignore commitnout, ale to, co je uvnitř, tak se bude ignorovat a když dáme git status, tak i když jsme obsah secrets.txt změnili, tak se nám ke commitování nenabízí.

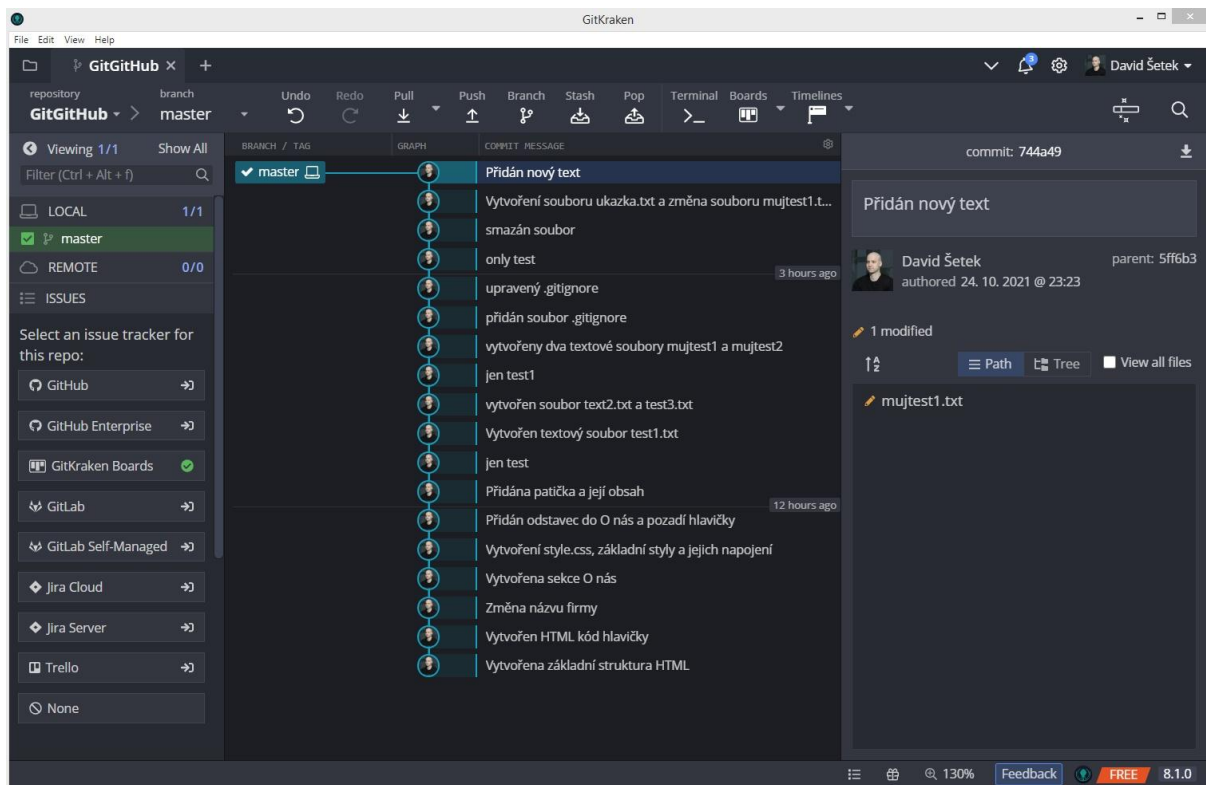
19. Grafické uživatelské prostředí - GUI (GitKraken)

Video: <https://youtu.be/SFHTPTCcv3A>

Zde si stáhněte GitKraken: <https://www.gitkraken.com/>

Zde GitKraken pro všechny platformy: <https://www.gitkraken.com/download>

Takto vypadá prostředí GitKrakena

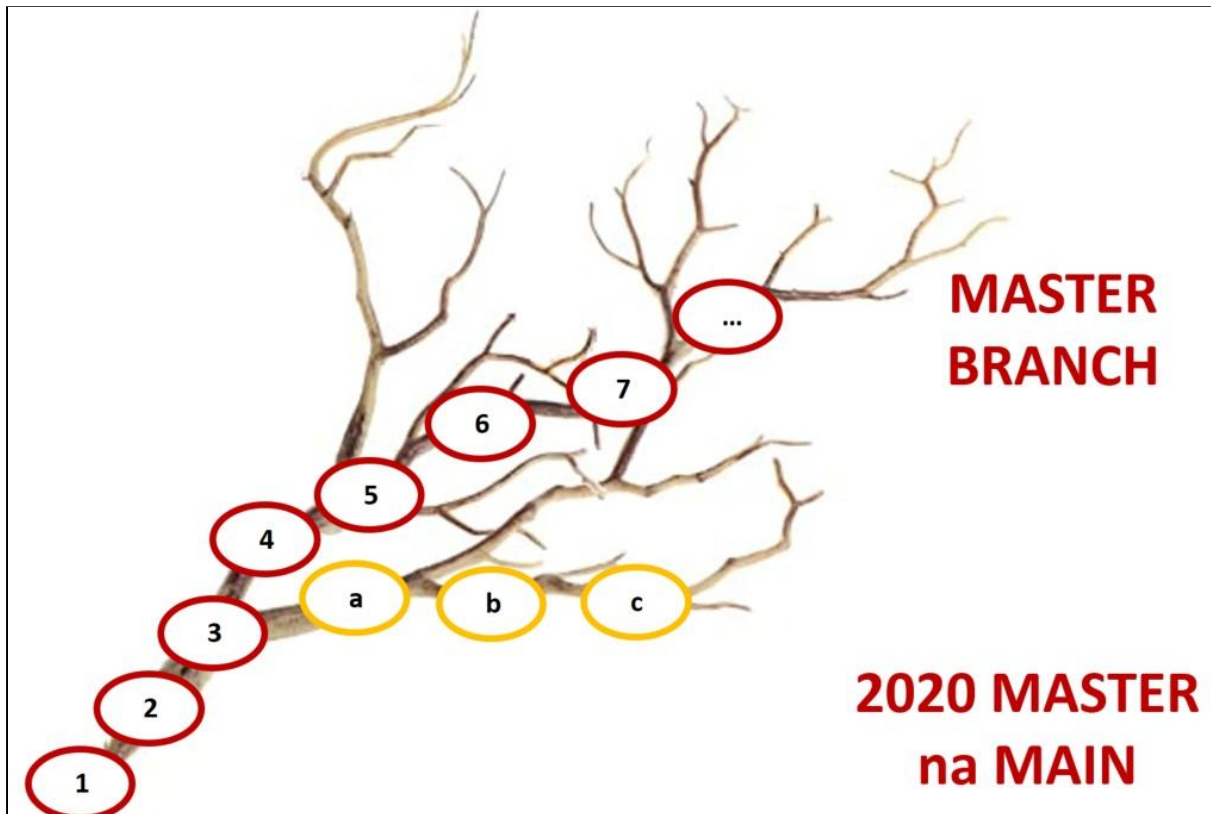


Větve (branches) - 2. část

20. Branches - co je to, seznam větví (git branch)

Video: <https://youtu.be/rSpDJvD-jLk>

Co to jsou větve:



Výpis větví (zde je výsledkem výpisu jen to, že máme větev master)

git branch

* master

21. Branches - vytvoření nové větve, přepínání mezi větvemi (git switch)

Video: <https://youtu.be/4rCGnEIA8Gw>

Vytvoření nové větve s názvem oprava-chyby-formular

git branch oprava-chyby-formular

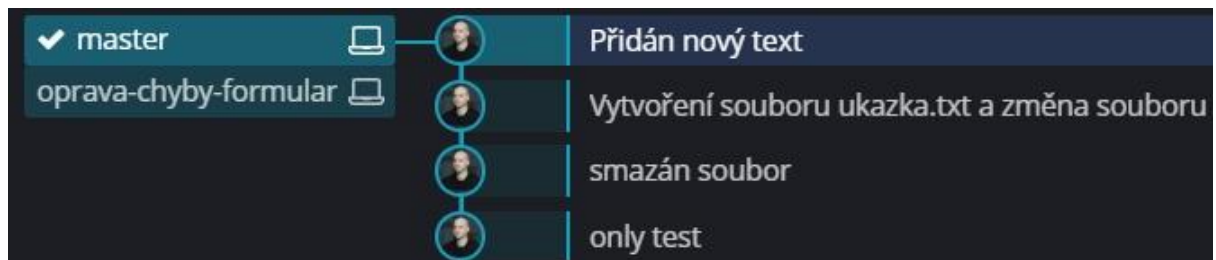
Přepnutí do větve oprava-chyby-formular

git switch oprava-chyby-formular

Přepnutí zpět do větve master

git switch master

Jak to vypadá v GitKraken



22. Branches - Komplexní procvičování: píšeme knihu o Harry Potterovi

Video: <https://youtu.be/z8vyJzWz-y8>

1. Vytvořte novou složku s názvem Harry Potter
2. Aktivujte git pomocí git init
3. Uvnitř vytvořte jeden textový soubor s názvem first-chapter.txt
4. Proveďte commit a okomentujte
5. Vypište si do terminálu seznam commitů a použijte verzi bez i s --oneline
6. Do souboru first-chapter.txt zkopírujte libovolný text o Harry Potterovi z této stránky https://cs.wikipedia.org/wiki/Harry_Potter
7. Proveďte commit a okomentujte
8. Vypište si do terminálu seznam commitů a použijte verzi bez i s --oneline
9. Vytvořte soubory second-chapter.txt a third-chapter.txt
10. Pouze second-chapter.txt pošlete do commitu a okomentujte "vytvořeny dva nové textové soubory second-chapter a third-chapter"
11. Nyní si vzpomenete, že jste zapomněli přidat do commitu i third-chapter.txt. Chybu napravte (použijte příkaz --amend na správném místě)
12. Vypište si do terminálu seznam commitů a použijte pouze verzi s --oneline
13. Do second-chapter.txt i third-chapter.txt přidejte libovolný text o Harry Potterovi z této stránky https://cs.wikipedia.org/wiki/Harry_Potter
14. Odešlete commitem nejedle second-chapter.txt a okomentujte
15. Dalším commitem odešlete third-chapter.txt a okomentujte
16. Vypište si do terminálu seznam commitů a použijte pouze verzi s --oneline
17. Zjistěte informace o druhém commitu (použijte příkaz git show + označení commitu)
18. Přepněte se do prvního commitu (použijte příkaz git checkout + označení commitu)
19. Přepněte se zpět do posledního commitu (použijte příkaz git checkout master)
20. Vytvořte novou větev s názvem new-character-snape
21. Vypište seznam větví
22. Přepněte se do větve new-character-snape
23. V této větvi vytvořte nový soubor s názvem snape.txt
24. Proveďte commit a okomentujte
25. Vypište si do terminálu seznam commitů a použijte pouze verzi s --oneline
26. Do snape.txt vložte kus textu z této stránky: https://cs.wikipedia.org/wiki/Severus_Snape
27. Proveďte commit a okomentujte
28. Vypište si do terminálu seznam commitů a použijte pouze verzi s --oneline
29. Do snape.txt vložte další kus textu z této stránky https://cs.wikipedia.org/wiki/Severus_Snape
30. Proveďte commit a okomentujte
31. Přepněte se do master větve
32. Vypište seznam větví a zkontrolujte, že jste ve větvi master
33. Vytvořte soubor fourth-chapter.txt
34. Proveďte commit a okomentujte
35. Vložte do fourth-chapter.txt libovolný text o Harry Potterovi z této stránky https://cs.wikipedia.org/wiki/Harry_Potter
36. Proveďte commit, ale pouze příkazem git commit

37. Do otevřeného souboru doplňte komentář a soubor zavřete
38. Vypište si do terminálu seznam commitů a použijte pouze verzi s --oneline
39. Pokud používáte GitKraken, tak se podívejte, jak vaše práce vypadá v tomto grafickém prostředí

23. Branches - co je to head

Video: <https://youtu.be/WjG8iy0foY8>

Prezentace ke stažení v PDF zde:

<https://drive.google.com/file/d/1jGUlFKif0A89lZc3Och5G4CzFKBmWpTC/view?usp=sharing>

24. Branches - praktická ukázka head

Video: https://youtu.be/7S_WHeNz6-w

Příkazy použité ve videu:

```
git init
git status
touch prvni.txt
git add .
git commit -m "prvni"
git log
touch druhy.txt

git branch
git branch bug
git switch bug
git switch master
```

25. Branches - další příkazy k větvím a jak větve fungují

Video: <https://youtu.be/KAlco4r6eVs>

Pokud chceme vytvořit novou větev a rovnou do ní být přepnutí, tak použijeme tento příkaz:
git switch -c libovolnyNazevVetve

Kromě příkazu switch se do již existující větve můžeme přepnout pomocí historicky staršího příkazu: **git checkout new1**

26. Branches - smazání větve a přejmenování větve

Video: <https://youtu.be/HSNS1nRAY-Q>

Smazání větve

Příkaz pro smazání větve

git branch -d nazevVetve

POZOR - pokud by příkaz vyhazoval text v podobě error: *The branch 'test' is not fully merged. If you are sure you want to delete it, run 'git branch -D test'*. Tak použijte příkaz s velkým D:

git branch -D nazevVetve

Přejmenování větve

POZOR - Pokud chcete nějakou větev přejmenovat, tak se do ní nejdříve musíte přepnout pomocí:

git switch nazevVetve

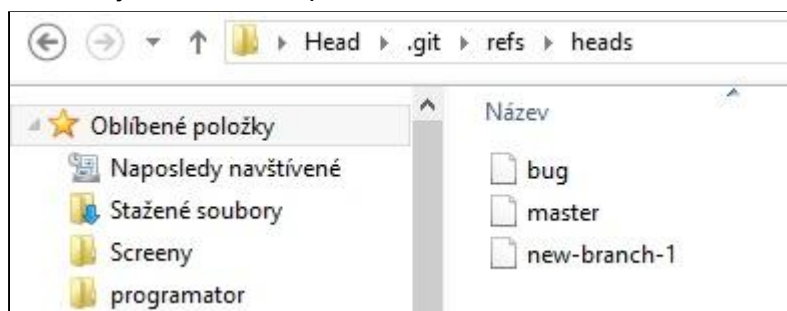
Poté můžete použít příkaz pro přejmenování:

git branch -m "novyNazevVetve"

27. Branches - ukládání Head a Branches

Video: <https://youtu.be/1VvrSNNNHVY>

Na jaké commity Head odkazuje, tak to nejadete ve složce .git -> refs -> head. V každém souboru je uveden kód posledního commitu v dané větvi.



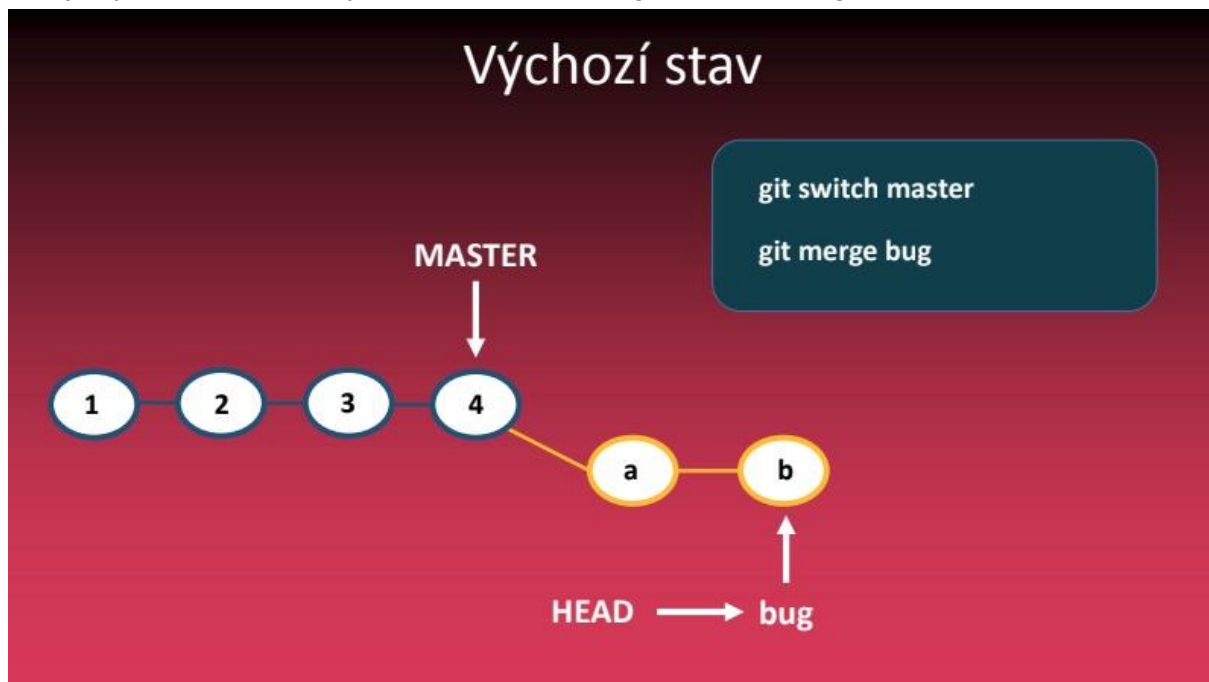
28. Merging branches - co je to merge a fast-forward merge

Video: <https://youtu.be/fzrtLSuVWwC>

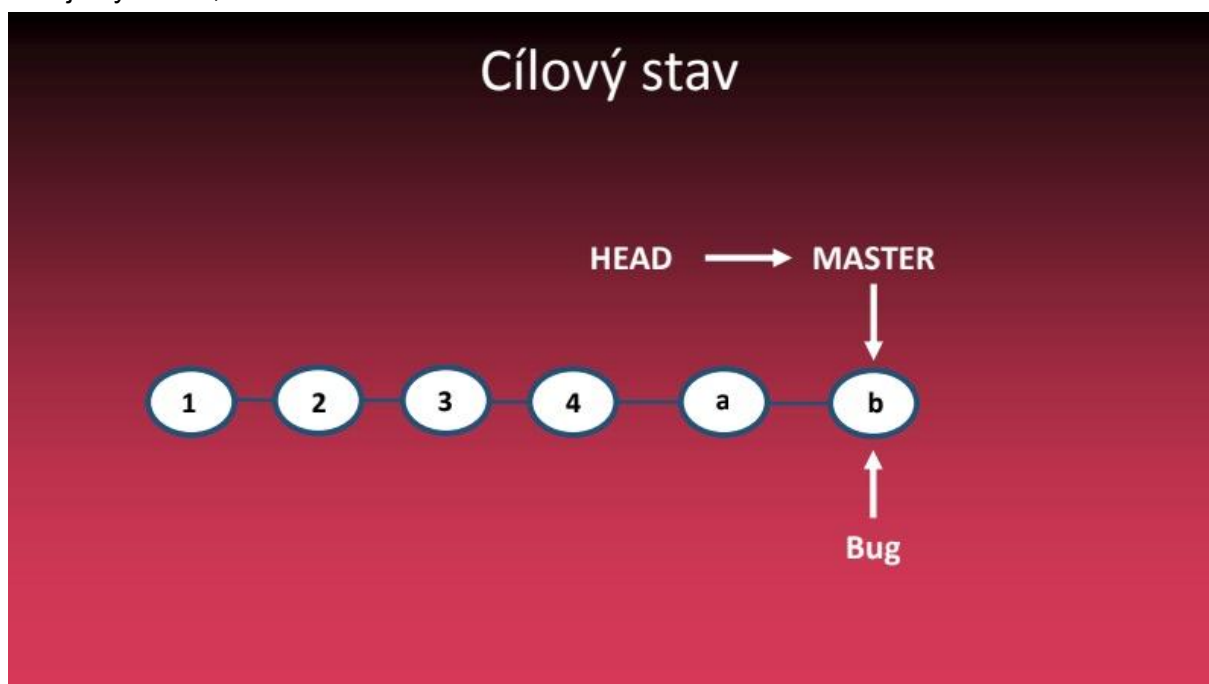
Prezentace ke stažení v PDF zde:

<https://drive.google.com/file/d/13WbDU1u6cGgPomawo6gQtIQIOIRSDgWw/view?usp=sharing>

Toto je výchozí situace, kdy chceme sloučit (mergovat) větev bug do větve master



Toto je výsledek, kterého se snažíme docílit:



Dva hlavní příkazy:

git switch master

git merge bug

29. Merging branches - fast-forward merge v praxi

Video: <https://youtu.be/esS3tmQ-4mA>

Dva hlavní příkazy použité ve videu:

git switch master

git merge bug

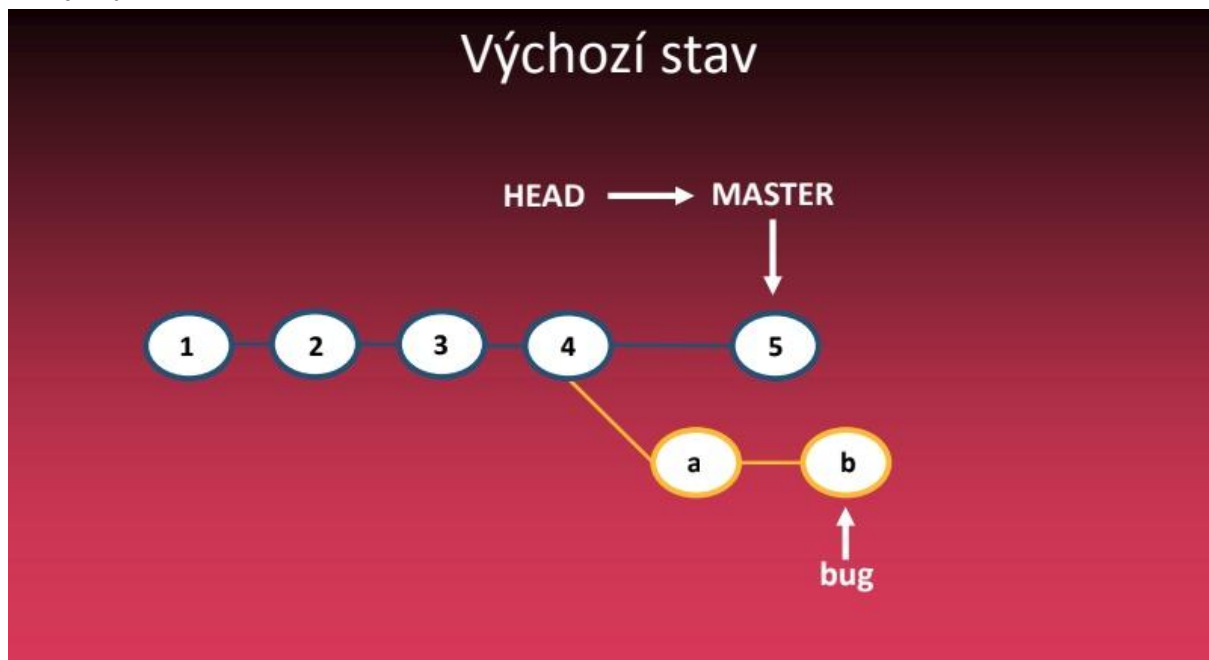
30. Merging branches - non fast-forward merge

Video: <https://youtu.be/xOn36WtOVLs>

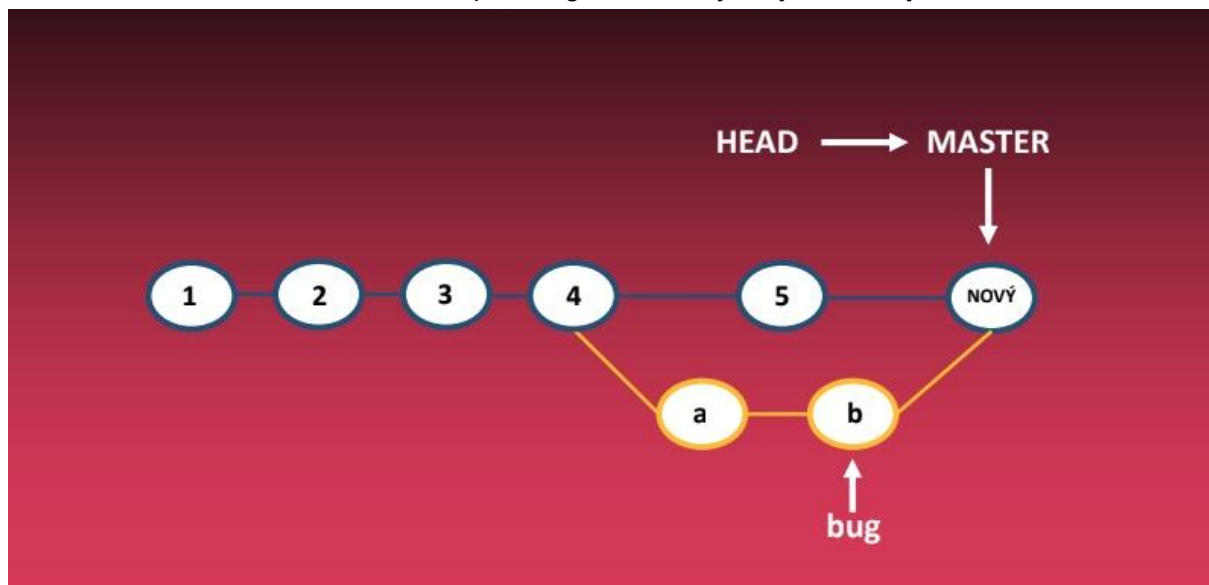
Prezentace ke stažení v PDF zde:

<https://drive.google.com/file/d/1QxJD4mEJJ2C22raZomwarmCQxWsLsynB/view?usp=sharing>

Toto je výchozí stav



Do tohoto stavu se chceme dostat - při mergování se nyní vytvoří nový commit



Dva hlavní příkazy použité ve videu:

git switch master

git merge bug

31. Merging branches - změna popisu posledního commitu

Video: <https://youtu.be/ZAC3IV0m01w>

Zde si jen ukážeme, jak změnit text posledního commitu. Již jsme to dělali u videa číslo 17..
Takže zde jen zopakujeme.

Jeden hlavní příkaz použitý ve videu:

git commit --amend

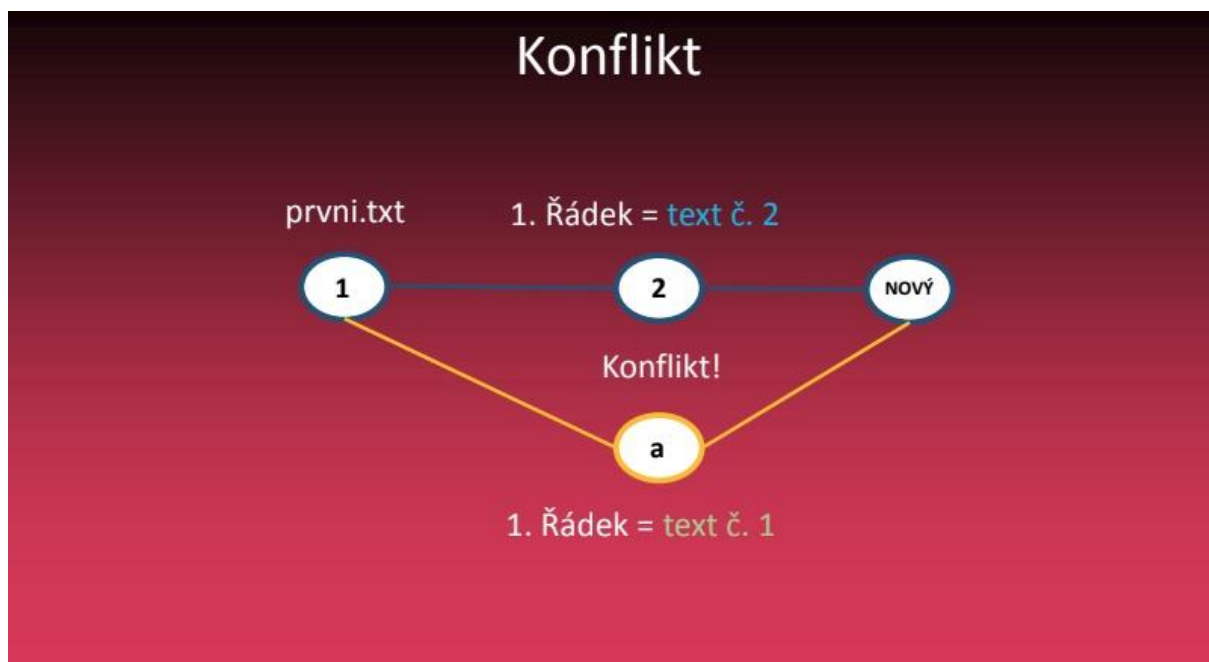
32. Merging branches - konflikt při mergování

Video: <https://youtu.be/8ZkOiCna0Fc>

Prezentace ke stažení v PDF zde:

https://drive.google.com/file/d/1GQmlh-GgSFL4YoPTaR_E-rQCm2zNu7zw/view?usp=sharing

Jak vzniká konflikt:



Dva hlavní příkazy použité ve videu:

git switch master

git merge bug

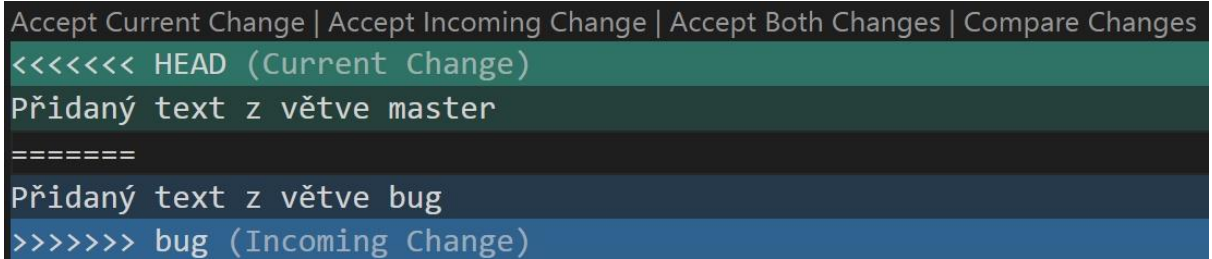
33. Merging branches - Řešení konfliktů ve VS code a smazání posledního commitu

Video: <https://youtu.be/SmedLHy6xIQ>

Příkaz na vymazání posledního commitu

git reset --hard HEAD^

Ve VS code můžeme použít možnosti nahoře v souboru - Accept Current Change, Accept Incoming Change atd.



```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
Přidaný text z větve master
====
Přidaný text z větve bug
>>>>>> bug (Incoming Change)
```

34. Merging branches - Komplexní procvičování: stránky s dark modem

Video: <https://youtu.be/JW-N6Tds0jA>

1. vytvořte ve svém počítači složku s názvem Merge - procvičování
2. otevřete ji ve svém editoru (VS code ...)
3. vytvořte git repository pomocí terminálu (git init)
4. přesvědčte se, že git funguje (git status)
5. v terminálu vytvořte dva soubory index.html a style.css
6. vytvořte commit s textem "vytvořen index.html a style.css"
7. index.html naplňte běžnou HTML strukturou a propojte se styly
8. vytvořte commit s textem "HTML struktura a napojení stylů v index.html"
9. vylogujte si dva provedené commity
10. vložte do body nadpis H1, dva podnadpisy H2 s libovolnými texty, pod každý podnadpis odstavec P s lorem ipsum. Podívejte se na stránku, jak vypadá v prohlížeči
11. vytvořte commit s textem "vložen obsah do body - H1, H2, P"
12. vytvořte novou větev s názvem darkmode a přepněte se do ní
13. otevřete style.css pomocí terminálu (pokud je nemáte ještě otevřené) a nastavte pozadí stránky na black a barvu textu na bílou
14. vytvořte commit s textem "vytvořena black verze stránek"
15. vylogujte si commity (měly by být 4)
16. přepněte se zpět na master a podívejte se na stránku
17. přepněte se zpět na větev darkmode a podívejte se na stránku
18. pokud používáte GitKraken, tak se podívejte, jak commity vypadají
19. předpokládejme, že se vám black verze líbí, tak proveďte fast-forward merge
20. zobrazte si vše v GitKrakenovi, pokud ho používáte

35. Git diff - k čemu to je

Video: <https://youtu.be/5CRPRZnSo-o>

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Document</title>
</head>
<body>
  <h1>Můj nadpis</h1>
</body>
</html>
```

Style.css

```
h1 {
  color: red;
}
```

Nyní změníme na:

```
h1 {
  color: blue;
}
```

Dáme v terminálu příkaz

git diff

Výsledek v terminálu:

```
diff --git a/style.css b/style.css
index 4717ad4..c2140b8 100644
--- a/style.css
+++ b/style.css
@@ -1,3 +1,3 @@
 h1 {
-  color: red;
+  color: blue;
 }
\ No newline at end of file
```

36. Git diff - rozbor výpisu a co se z něj dozvíme

Video: <https://youtu.be/xk6QD0GpEXU>

Výsledek v terminálu:

```
diff --git a/style.css b/style.css
index 4717ad4..c2140b8 100644
--- a/style.css
+++ b/style.css
@@ -1,3 +1,3 @@
h1 {
- color: red;
+ color: blue;
}
\ No newline at end of file
```

Rozbor

`diff --git a/style.css b/style.css` (použili jsme příkaz diff a původní style.css je označeno písmenem "a" a nové style.css je označené písmenem "b")

`index 4717ad4..c2140b8 100644` (jen metadata - data navázané na soubor - nemusíme řešit)

`--- a/style.css`
`+++ b/style.css`
(původní style.css má přiřazený symbol mínus a nový style.css má přiřazený symbol plus)

`@@ -1,3 +1,3 @@` (ze starého style.css, které má značku mínus, se začalo na prvním řádku a vzali se celkem 3 řádky; z nového style.css, které má značku plus, se začalo na prvním řádku a vzali se 3 řádky)

```
h1 {
- color: red;
+ color: blue;
}
```

červené bylo odebráno ze starého style.css (proto je před tím mínus - označení pro staré style.css)

zelené bylo přidáno do nového style.css (proto je před tím plus - označení pro nové style.css)

`\ No newline at end of file`

Na konci souboru není enter, který by nás hodil na další řádku. Znamená to, že za závorkou `}` jsme neudělali enter. Nic víc. Není to nic důležitého

37. Git diff - použití u unstaged changes

Video: https://youtu.be/b_cxawtbZJk

Git diff jsme zatím používali jen u unstaged changes. To jsou změny, které nejsou ve staging area ani commitlé v repository. Jsou na tomto obrázku v tom prvním boxu:

