Metodika - Práca s GIT

Tím 01

Téma: In memory DB

Členovia: Andrej Fúsek, Nikolas Hamran, Dávid Kubík, Martin Petráš, Alexandra Smolová, Martin Staňo, Veronika Včelková, Jakub Veselý

Vypracoval: Martin Staňo

Náš projekt GPUDB využíva vzdialené Git úložisko (repozitár):

- DropDBase

Umiestnené na Bitbucket cloude poskytovanou firmou Atlassian. Voľba Bitbucket úložiska bola motivovaná dobrou integráciou so službou Jira.
Pre naklonovanie repozitárov treba mať vytvorený Atlassian resp. Bitbucket používateľský účet a byť pridaný ako spolupraconík administrátorom.
Administrátorom je Martin Staňo (martin.stano@icloud.com). Po získaní prístupov stačí v terminály zadať nasledujúce príkazy:

```
> git clone https://ms_fiit@bitbucket.org/ms_fiit/dropdbase.git
```

Kód v oboch repozitároch nasleduje konvenciu GitFlow. Produkčný kód (kód riešenia ktoré, schválil produktový vlastník) sa nachádza vo vetve
master. Vo vetve *develop* sa nachádza merged kód súvisiaci s úlohami riešenými v danom šprinte. Z vetvy *develop* si členovia týmu vytvárajú
feature vetvy v ktorých je rozpracovaný kód súvisiaci s prácou člena týmu na konkrétnej user story resp. tasku. Konvencia pomenovávania *feature*
vetiev je nasledujúca:

```
feature/<jira_issue_id>
# priklad
feature/GPUDB-8
```

Vytvorenie vetvy a prepnutie sa do nej je vykonané spustením príkazu:

```
> git checkout -b <nazov_vetvy>
# priklad
> git checkout -b feature/GPUDB-8
```

Tento príkaz však vytvorí vetvu iba v lokálnej kópií repozitára. Pre nahranie vetvy na vzdialené úložisko je nutné použiť príkaz:

```
> git push origin <nazov_vetvy>
# priklad
> git push origin feature/GPUDB-8
```

Pre zobrazenie všetkých aktívnych vetiev v repozitári použitejeme príkaz. Aktuálna vetva je označena symobolom *.

```
> git branch
# priklad vystupu
- master
* develop
- feature/GPUDB-8
- feature/GPUDB-14
```

Ak v zozname nevidíme vetvu o ktorej sme presvedčení, že existuje tak je možné, že vetva existuje len na vzdialenom úložisku a nie je vytvorená jej kópia
v našom lokálnom repozitáry. Aktualizovať mapovanie vzdialených vetiev na lokálne vetvy je možné vykonať príkazom:

```
> git fetch
```

Prepínanie vetvy na ktorej pracujeme je možné pomocou príkazu:

```
> git checkout <nazov_vetvy>
```

Merge *feature* vetiev do develop je prípustný až po pokrytí kódu jednotkovými testami, po jeho zdokumentovaní a vykonanej code review od iného
člena týmu. Vykonanie mergu *develop* vetvy s konkrétnou *feature* vetvou je vykonané analogicky k príkladu uvedenému nižšie. Na tomto príklade
je vyobrazený merge vetvy *feature/GPUDB-8* do vetvy *develop*.

```
#You are on branch develop
> git merge feature/GPUDB-8
```

Lákavou alternatívou k mergovaniu vetiev je príkaz:

```
> git rebase <nazov_vetvy>
```

Ten dokáže do vetvy itegrovať všetky zmeny z inej vetvy, avšak dokáže značne zneprehladniť históriu commitovania zmien a robí ich revízie a revertovanie obtiažnimi.

Preto je v našom projekte jeho používanie **ZAKÁZANÉ** (prípustné iba vo výnimožných prípadoch s väčšinovým konsenzom týmu). V prípade integrovania zmien z inou vetvou je preto odporúčané sa s danou vetvou mergnúť. Ak nastane scenár že nechceme integrovať do svojej vetvy všetok jej obsah (rozpracovanú funkcionalitu), ale iba konkrétne zmeny je možné využiť príkaz:

```
> git cherry-pick <hash_commitu>
# prikad
> git cherry-pick as88sa99
```

Práve kvôli jednoduchému využitiu príkazu *cherry-pick* je od členov týmu vyžadované aby svoje commity vykonávali čo najčastejšie a atomicky podľa implementovanej funcionality s jej vhodným opisov v správe commitu. Okrem toho je od čelov týmu vyžadované aby kontrolovali, ktoré súbory pridávajú na commit príkazom:

```
> git add <subory>
#POZOR NA
> git add *
```

Príkazom *git add* * pridáme všetky súbory, ktoré boli lokálne zmenené alebo pridané (popr. odstránené). Tie môžu zahŕňť napr. konfiguračné súbory alebo solution/project súbory, ktoré si meniť neželáme. Je nutné vedieť čo komituje. Ktoré súbory majú a nemajú byť comitnuté je možné skontrolovať príkazom:

```
> git status
```

Samotné commitovanie súborou do vetvy prebieha príkazom *git commit* ktorého súčasťou je aj commit správa opisujúca obsah daného commitu.
Tá by mala spĺňať konvenciu uvedenú nižšie v príklade. Hlavne musí obsahovať identifikátor Jira issue kvôli integrácií s týmto systémom. Správa opisujúca commit nesmie byť príliš všeobecná aby bolo z prehladu commitov jasné čo daný commit adresoval.

```
> git commit -m '<jira_issue(s)_id> <Sprava popisujuca commit>'
#priklad
> git commit -m 'GPUDB-8 Added methods to ClassX'
```

Ak si želáme commity na svojej lokálnej vetve uložiť na vzdialené úložisko zavoláme príkaz:

```
> git push
```

Ak sme commit vykonali omylom alebo sme commitli nesprávne súbory možeme commit zvrátiť príkazom:

```
> git revert <hash>
#príklad
> git revert as5asd
```

Avšak tento príkaz je zvráti commit iba lokálne (a túto zmenu zaznemená ako ďalší commit). Pre prenos na vzdialné úložisko je nutné opäť
zavolať príkaz *git push.* Zvrátenie (revert) commit je taktiež možný prikazom:

```
> git rebase -i HEAD~<pocet_commitov_spat_v_case>
#priklad
> git rebase -i HEAD~5
```

Ten nám ukáže posledných *n* commitov a môžeme si vybrať, ktoré chceme ponechať a ktoré zahodiť (v otvorenom editore po spustení príkazu sú napísané
podrobné inštrukcie). Pre prenos zmien na vzdialené úložisko je však po vykonaní *git rebase -i* nutné použiť príkaz:

```
>git push --force
```

Ten dokáže commity označené na zahodenie príkazom *git rebase -i* vymazať z histórie commitov na vzdialenom úložisku.

POZOR!!!

Používanie *git push --force* by malo byť vždy konzultované so zvyškom tímu (minimálne potencionálne dotknutými osobami).

Ak si akoukoľvek operáciou v rámci verziovacieho systému git nie sme istí možeme s históriu commitou pozrieť príkazom:

```
> git log
```

Poprípade si môžme pozrieť celkovú históriu a vizualiáciu štruktúry vetiev v repozitári graficky pomocou príkazu:

```
> gitk --all
```

Nástroj Gitk okrem prehľadu umožňuje aj prezeranie zmien v súboroch v jednotlivých commitch a point-and-click prístup k niektorých z vyššie uvedených
príkazov.