Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

reCommers

Metodika pre Git

Tím: 03 - reCommers
Vedúci tímu: Ing. Ivan Srba, PhD.
Vyhotovil: Lukáš Manduch
Dátum poslednej zmeny: 09. 05. 2018
Verzia: 1.1

1. Úvod

Táto metodika rozpracováva spôsob, akým sa pracuje s verziovacím systémom v projekte. Verziovanie je realizované pomocou systému Git. V zimnom semestri sme pracovali s gitovým repozitárom v TFS, začiatkom letného semestra bola realizovaná migrácia na Github.

2. Git worflow

2.1. Začiatok práce

Po začatí práce na tasku sa pozriem, či náhodou už neexistuje branch pre user story do ktorého patrí tento konkrétny task.

- Ak existuje, tak sa nastavím na tento branch a mozem zacat pracovat.
- Ak neexistuje, u seba lokálne vytvorím branch ktorý bude mať názov vo forme *čislo_user_story/meno-user-story* a začnem pracovať

2.2. Vývoj

Počas práce si možem vytvárať commity ako spôsob istej zálohy, keď sa napríklad prepínam medzi rôznymi vetvami. Na tento účel môžem použiť aj stash.

Po tom čo som prácu dokončil skontrolujem aké commity som vytvoril, či každý predstavuje nejakú ucelenú časť a či majú rozumný popis. Ak nie použijem napríklad squash, aby som ich spojil do väčších častí a vytvoril tak rozumnejšie celky. Taktiež si overím či môj kód prechádza všetkými testmi.

Znova skontrolujem či náhodou už takýto branch neexistuje a ak nie tak môžem spraviť push na server. Ak už branch existuje, najskôr si tento branch stiahnem, a urobím rebase na tento branch. Až následne zmeny pushnem.

2.3. Code review

Po tom čo sú moje zmeny na serveri, počkám, kým zbehnú testy cez systém Travis. Až keď všetky testy budú mať pozitívny výsledok, vytvorím cez webové rozhranie *pull request* na *dev* vetvu a pridám vhodných reviewerov. Následne čakám na dokončenie review. Ak majú nejaké výhrady ku kódu, znova postupujem rovnako ako na začiatku a keď mám zmeny hotové, urobím push na server. Tento cyklus sa opakuje až dokým nie sú moje zmeny schválené a mergnuté.

2.4. Branching

Existujú tri dôležité vetvy a to master, dev a staging. Každá z týchto vetiev obsahuje len kód, ktorý prešiel cez code review a teda sa dá považovať za funkčný.

2.4.1. Master

Obsahuje kód, ktorý je nasadený reálne v prevádzke, takže do tejto vetvy nikto len tak nemerguje.

2.4.2. Staging

Obsahuje kód, ktorý je v testovacej fáze, ktorý je určený do reálnej prevádzky, ale ešte tam nie je.

2.4.3. Dev

Obsahuje kód, ktorý prešiel cez code review a testami.

2.5. Commit message

Začína modulom, v ktorom bola zmena vykonaná a krátkym popisom toho čo bolo zmenene/pridane. V ďaľšich riadkoch treba opísať a ako som to spravil a prečo to robím. Na koniec pridame id tasku z tfska - možno nam to bude nanič, ale je to dobra prax.

Pri pisani spravy treba dbat aj na to aby som neprekročil 70 - 75 znakov. Ak to budete robiť štandardne cez konzolu a cez vim, tak ten začne svietiť na červeno, alebo to aj sam enterovať.

Vsetky commit-y treba pisat po anglicky.

2.5.1. Štruktúra

```
Module/File: Co som pridal

Detailnejsi opis zmeny. Co konkretne som pridal alebo
odobral.

Preco zmenu robim, alebo aky je sucasny stav

123456
```

2.5.2. Príklad

```
tracing/samples: Fix creation and deletion of simple_thread_fn creation

Commit 7496946 ("tracing: Add samples of DECLARE_EVENT_CLASS() and
DEFINE_EVENT()") added template examples for all the events. It created a
DEFINE_EVENT_FN() example which reused the foo_bar_reg and foo_bar_unreg
functions.

Enabling both the TRACE_EVENT_FN() and DEFINE_EVENT_FN() example trace
events caused the foo_bar_reg to be called twice, creating the test thread
twice. The foo_bar_unreg would remove it only once, even if it was called
multiple times, leaving a thread existing when the module is unloaded,
causing an oops.

Add a ref count and allow foo_bar_reg() and foo_bar_unreg() be called by
multiple trace events.
```

2.6. Git cheatsheet

2.6.1. Create branch

Pred tým ako začnem pracovať na svojej úlohe, musím vytvoriť vetvu, pre túto úlohu a prepnúť sa na ňu. To spravím nasledovne

```
git checkout -b meno_novej_vetvy
```

2.6.2. Add files

Nie všetky súbory, ktoré mám rozpracované, musím automaticky pushnut na server (ak mám napríklad súbor s tagmi, ten nemá čo robiť v online repozitári), tak je potrebné použiť príkaz git add

```
git add meno_prveho_suboru menu_druheho
```

2.6.3. Commit files

Po tom, čo som takto pridal potrebné súbory vytvorím commit.

```
git commit
```

2.6.4. Push

Následne keď mám všetku prácu hotovú (môžem mať jeden alebo viacero commitov), urobím push

```
git push origin meno_aktualnej_branch
```