Tento dokument slouží jako rychlá příručka k ovládání verzovacího systému Git. Po jejím přečtení by mělo být jasné, jak se systémem pracovat.

Contents

1 Základní práce s repozitářem

Všechny operace se provádí v rámci repozitářů, tj. složek, ve kterých je — mimo samotný obsah — speciální složka `.git`, která obsahuje informace o repozitáři a jeho nastavení. Repozitář může obsahovat i další speciální soubory, o kterých je zmíněno v sekci <todo>.

1.1 init

Repozitář můžeme inicializovat v libovolné složce příkazem `git init`, jenž vytvoří dříve zmíněnou složku `.git`. Tato složka není běžně vidět, protože začíná tečkou, což v rámci Unixového systému znamená skrytá složka či soubor. Tím je repozitář vytvořen a připraven ke správě souboru.

1.2 clone

Nechceme-li inicializovat lokální repozitár, ale raději chceme stáhnout nějaký existující repozitář ze sítě, použijeme příkaz `clone`. Ten vytvoří lokální kopii vzdáleného repozitáře z dané adresy.

```
git clone https://git.microlab.space/em/timer
```

Zároveň můžeme specifikovat složku, do které chceme repozitář klonovat.

```
git clone https://git.microlab.space/em/timer nova-slozka
```

1.3 status

Stav repozitáře lze vypsat příkazem `status`. Ten vypíše, zda proběhly od poslední verze repozitáře nějaké změny, nebo zda jsou nějaké změny připraveny k uložení.

```
git status
```

1.4 log

Všechna historie v systému Git je řazena do takzvaných *commitů*. Každý *commit* je v podstatě bodem v historii a ke každému z nich se lze vracet a jinak s nimi pracovat.

Historii těchto commitů lze vypsat příkazem `log`. Ten otevře program, který nám dovolí se v historii posouvat. Tento prohlížeč lze zavřít zkratkou `q`.

```
git log
```

1.5 pull

Pokud na vzdáleném repozitáři nastanou změny, je možné je stáhnout do lokální kopie příkazem `pull`. Ten je od příkazu `clone` liší tím, že porovná stavy repozitářů a stáhne pouze změny, zatímco `clone` stahuje celý repozitář.

```
git pull
```

2 Vytváření commitů a nahrání změn

Proces vytvoření nového bodu v historii — commitu — má několik kroků, kterými je třeba projít. Následující obrázek tento postup znázorňuje na poštovním balíku, který v reálném světě prochází podobným procesem[1].

Představíme-li si, že commit je poštovní balík, ve kterém chceme něco (změny) odeslat, musíme je nejdříve poskládat do krabice, poté krabici zalepit, něco na krabici napsat (ne adresu, ale popis obsahu) a nakonec balík odeslat. Adresu má Git uloženou právě v dříve zmíněné složce `.git`.

Figure 1: Proces nahrání lokálních změn do repozitáře

2.1 add

Připraví vybrané soubory ke commitu. Dokud se neprovede commit, lze soubory přidávat a odstraňovat.

```
git add soubor1 soubor2 soubor3
```

Nechceme-li vypisovat všechny soubory, můžeme specifikovat všechny soubory (které byly změněny) tečkou.

```
git add .
```

[1] S tím rozdílem, že v Gitu se balík neztratí, jako tomu bývá v případě České pošty.

2.2 commit

Commit v podstatě *zabalí* všechny připravené soubory a vytvoří nový stav, ke kterému se v budoucnu lze vrátit. Změny provedené v commitu lze měnit, ale po odeslání na server to může být obtížné, a proto je dobré dbát na to, že commit je vytvořen správně.

Příkaz pro commit otevře editor, ve kterém se na první řádek napíše stručný popis změn s maximálním počtem 72 znaků. Na další řádky lze změny popsat do detailu, ovšem řádky by se měly zalamovat po nejvíce 72 znacích. Editor *vim* může zalamování automatizovat pomocí `:set tw=72`.

```
git commit
```

Nechceme-li pracovat v editoru, máme možnost popis zadat rovnou přepínačem `-m`.

```
git commit -m "Kratký popis změn"
```

Pomocí příkazu `commit` můžeme podvádět a přeskočit předchozí krok `add`, a to tak, že při commitu použijeme navíc přepínač `-a`, který je v podstatě náhradou za `git add .`.

```
git commit -am "Kratký popis změn"
```

2.3 push

Nahraje všechny lokálně vytvořené commity na vzdálený repozitář. K tomu je potřeba mít práva po push na vzdáleném repozitáři a správné přihlačovací údaje.

```
git push
```

Pokud Git začne křičet, že vzdálený repozitář má v sobě změny, které nemáte u sebe, je potřeba je nejdříve stáhnout námi známým příkazem `pull`. Je také dobré použít přepínač `--rebase`, který se pokusí všechny změny ze vzdáleného repozitáře zakomponovat do těch našich bez dalších nutných manuálních zásahů.

```
git pull --rebase
git push
```

Při pullování může nastat konflikt, o kterém se pojednává v následující sekci.

3 Konflikty

Dojde-li k situaci, kdy dva uživatelé pracovali na stejném souboru a jeden nahrál změny do hlavního repozitáře, musí druhý uživatel před nahráním vlastních změn vyřešit potenciální konflikty.

Konflikt nastane v případě, že oba uživatelé upravili stejný řádek, nebo provedli podobné změny. Druhý uživatel musí tedy ručně (s pomocí gitu) zvolit, které řádky chce zachovat a které nahradit

K vyřešení konfliktu staří otevřít příslušný soubor, ve kterém konflikt vznikl, a upravit ho tak, jak chceme aby vypadal. Git do souboru přidá znaky, kterými ukazuje, kde konflikt vznikl. Do souboru vepíše obě verze obsahu, tedy tu cizí a tu naši. Obecně platí, že stačí jednu z verzí odmazat a pokračovat dál.

```
Toto je text ve vzdáleném repozitáři.
Je rozdělen na řákdy.
<<<<<<< HEAD
Mám rád zeleninu.
=======
Mám rád ovoce.
>>>>>>> 0343280bc3f754455fb1d5ffc6931c1ae258b62b
```

V příkladu vidíme, že první uživatel změnil řádek 3 na *Mám rád ovoce* a změny nahrál do repozitáře. Druhý uživatel mezitím změnil stejný řádek na *Mám rád zeleninu* a při pokusu o push nastal konflit. V souboru se ukáže, kde konflikt začíná a kde končí. Je teď na uživateli, jak soubor upraví. Můžeme tedy smazat nepotřebné řádky a ponechat například řádek se zeleninou, nebo řádek změnit úplně.

```
Toto je text ve vzdáleném repozitáři.
Je rozdělen na řákdy.
Mám rád zeleninu.
```

Po uložení je opět nutné provést celý proces `add`, `commit` a `push`.

```
git commit -am "Mám raději zeleninu"
git push
```

4 Větve

Pro výběr větve, se kterou chceme pracovat, použijeme příkaz `git checkout <branch>`, ve kterém musíme zvolit správný název větve. Chceme-li větev vytvořit, můžeme příkaz rozšířit o přepínač `-b`, tj. `git checkout -b <branch>`. Mazání větví provádíme přepínačem `-d`.

Existující větve lze vypsat příkazem `git branch -a`. Aktivní větev je označena hvězdičkou.

Spojení větví se provádí příkazem `git merge <branch>`, který spojí specifikovanou větev s aktivní větví vybranou příkazem `checkout`.