Distribuovaný systém kontroly verzií - Git

Juraj Hreško

ecommerce.cz

29. září 2010

Požiadavky na SCM

- ukladanie dát do archívu (repozitára)
- obnovenie dát z určitej doby
- porovnávanie verzií dát
- anotovanie dát
- riešenie prítupu viacerých vývojárov
- označovanie verzií špecifickým menom
- vetvenie vývoja
- zlučovanie vetví vývoja

Typy verzovacích systémov

- "triviálne"
- centralizované (client-server)
- distribuované

"Triviálne"verzovanie

Použil už snáď každý pri zálohe napr. konfiguračných súborov.

Príklad

```
$ cp config.cfg config.old
c:\>copy config.cfg config.old
```

- vhodné pre jednu dostupnú "zálohu"
- nesystematické pre viac súborov s históriou
- nehovoriac o zdieľaní, vetvení a pod.

Centralizované systémy pre správu verzií

- architektúra typu klient-server
- spoločný centrálny repozitár
- vývoj prebieha v rámci pracovnej kópie
- riešenie súčasného zápisu viacerých programátorov
  - lock/modify/commit
  - modify/merge/commit
- zástupcovia: CVS, SVN, Perforce, TFS

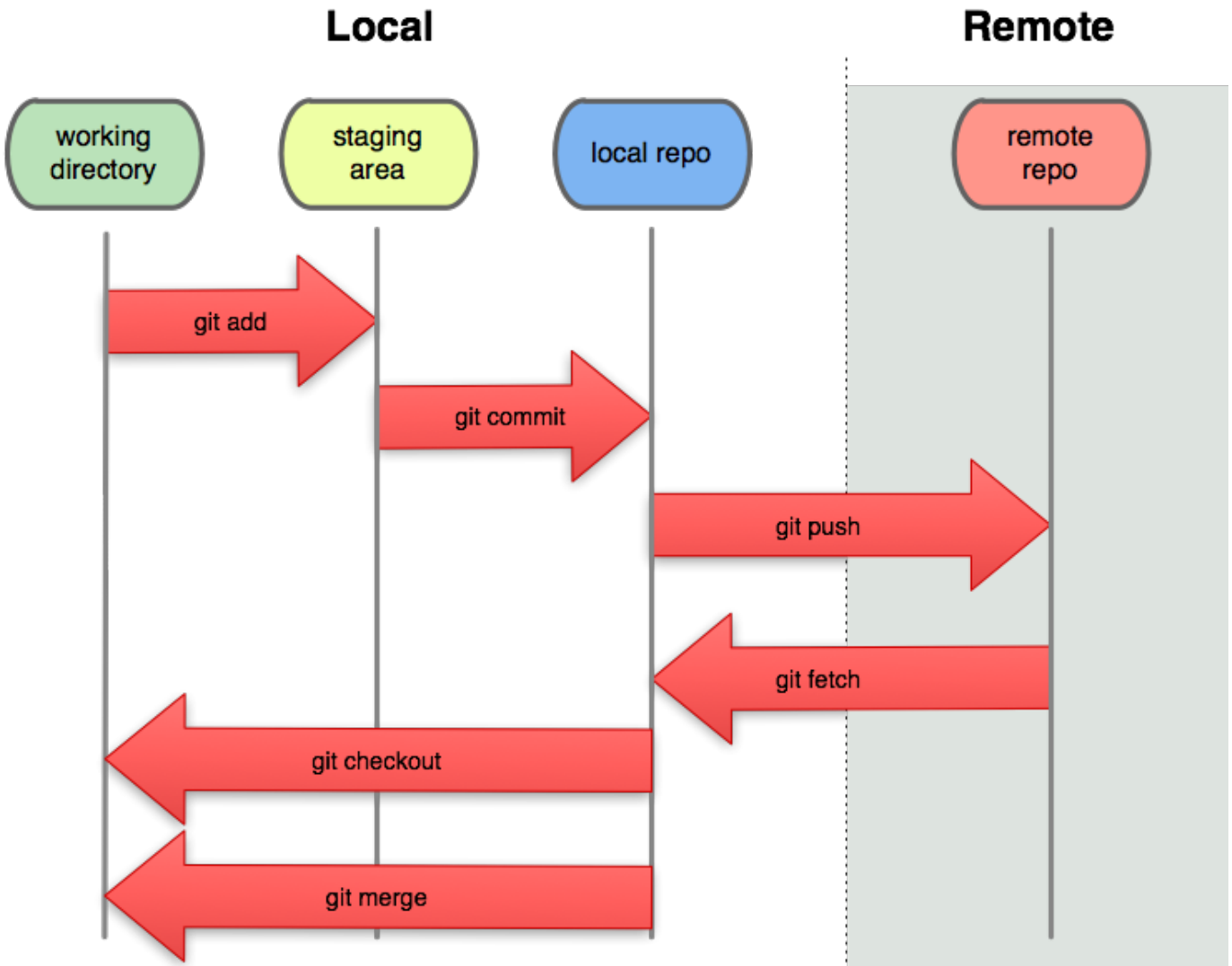
Distribuované systémy pre správu verzií

- neexistuje centrálny repozitár - každý je plnohodnotný
- miesto operácie checkout operácia clone
- repozitár == pracovná kópia
- publikovanie zmien - vystavenie repozitára, push do vzdialenej vetvy, posielanie záplat e-mailom
- zástupcovia: Monotone, Git, Mercurial, Bazaar

Práca s lokálnym repozitárom

- neexistuje "jedno zraniteľné miesto"(single point of failure)
- offline práca s repozitárom (vo vlaku, na chate, pri výpadku serverov)
  - prechádzanie histórie
  - commity
  - vetvenie
- súkromie pri experimentoch
- rýchlosť operácií

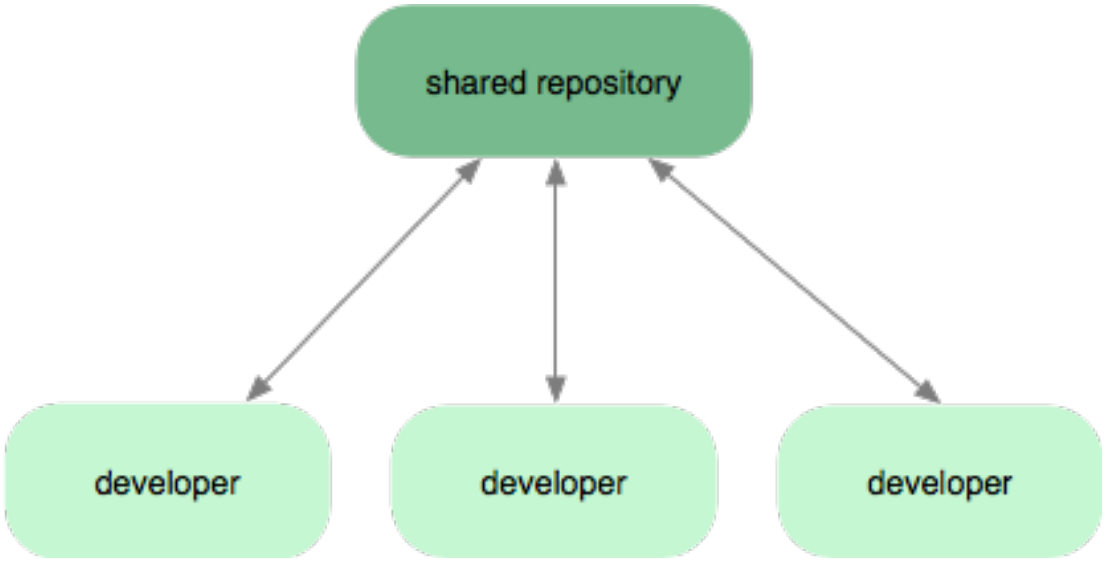
Práca s repozitármi - schéma



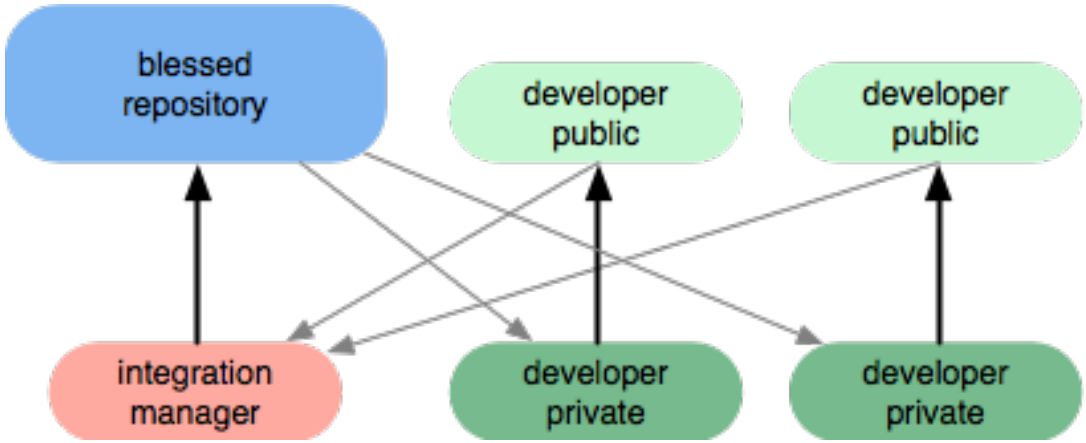
Flexibilný workflow

- je možné pracovať s repozitármi rôznymi spôsobmi, v závislosti od druhu projektu
- iné pre klasický centralizovaný vývoj, vlastné dokumenty, vývoj open source software
- príklady niektorých možných workflow
  - centralizovaný
  - koordinátor
  - generál a pobočníci

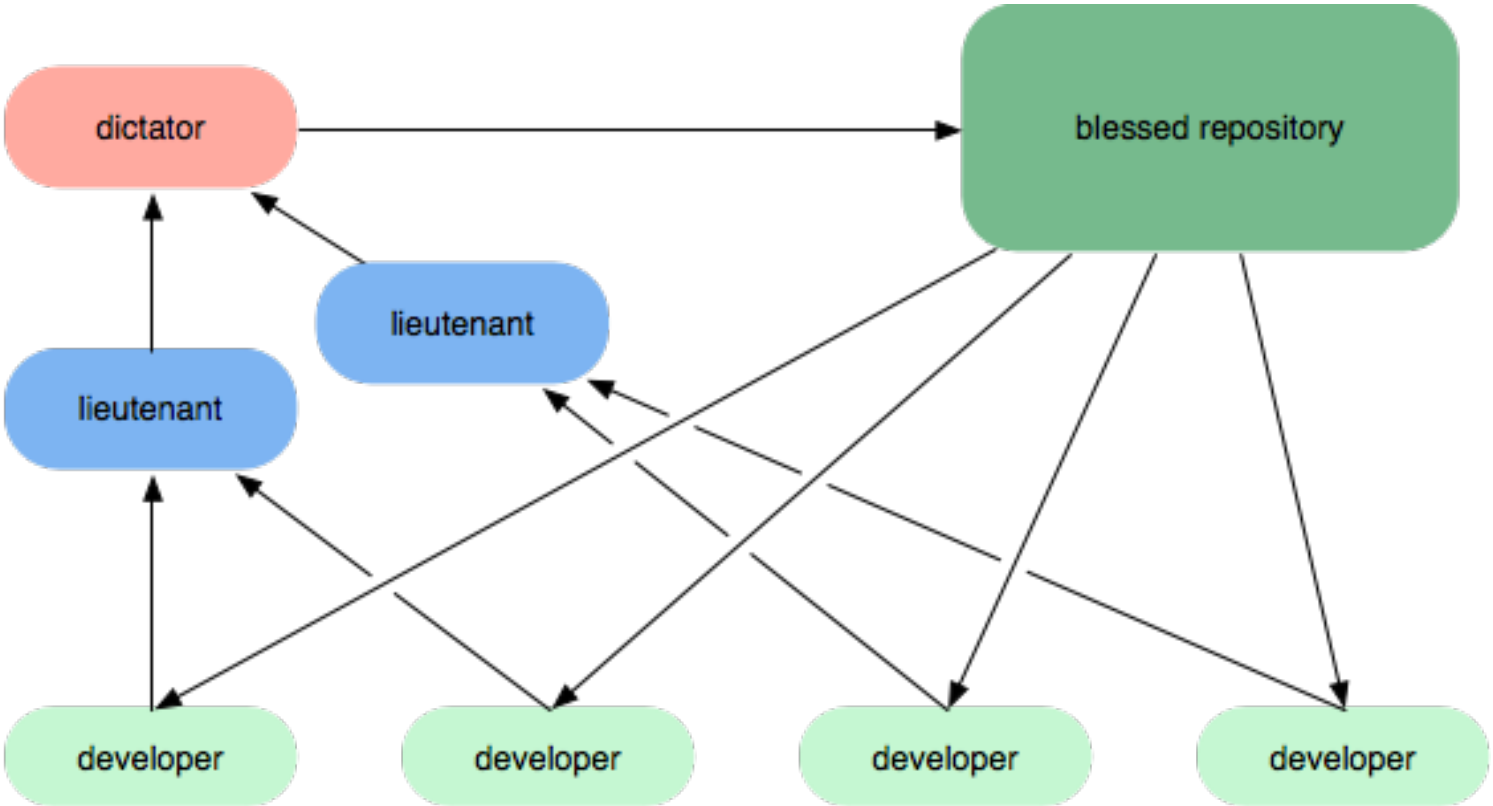
Centralizovaný workflow



Workflow "koordinátor"



Workflow "generál a pobočníci"



Krátko k histórii projektu

git

1. a contemptible person, often a fool

2. a bastard

(Collins English Dictionary)

- vznik v roku 2005, k účelu verzovania projektu linuxového jadra
- hlavné požiadavky
  - Take CVS as an example of what not to do; if in doubt, make the exact opposite decision
  - Support a distributed, BitKeeper-like workflow
  - Very strong safeguards against corruption, either accidental or malicious
  - Very high performance
- spočiatku sada programov v Perl, Bash a C

Súčasnosť projektu

- voľne dostupný open source DSCM software
- stránky projektu http://git-scm.com/
- používaný aj na väčších projektoch
  - Git
  - Linux Kernel
  - Android
  - Ruby on Rails
  - Fedora
  - VLC media player
  - ...
- primárne pre platformu GNU/Linux a ostatné Unix-like systémy - BSD, Solaris, Darwin
- pre MS Windows v Cygwin alebo port mysysgit
- základom je ovládanie cez príkazový riadok, dostupné však aj GUI a pluginy pre IDE (Eclipse, NetBeans, Visual Studio)

Inštalácia

Pre GNU/Linux balíček v repozitári - git-core

Príklad

```
$ yum install git-core
$ apt-get install git-core
```

Pre MS Windows doporučujem kombináciu msysgit + TortoiseGit

- msysgit - posledná verzia Git 1.7.2.3
- pri výbere komponent doporučujem zrušiť Windows Explorer integration (zabezpečí ho TortoiseGit)
- ostatné voľby podľa uváženia, osobne ponechávam východzie
- následne doinštalovať GUI klienta - TortoiseGit
- pri inštalácii je dobré zvoliť totožného SSH klienta ako u Gitu, v mojom prípade u oboch OpenSSH

Veľmi stručný úvod k použitiu (Git) Bash

Git Bash spustíme, z menu Programy/Git/Git Bash. Práca v Bash konzoli je podobná práci s príkazovým riadkom vo Windows/DOS. Niekoľko základných príkazov:

Bash	cmd.exe	Popis
cd	cd	Zmení adresár
cp	copy	Skopíruje súbor(y)
rm	del	Zmaže súbor(y)
mkdir	mkdir	Vytvorí adresár
ls	dir	Vypíše obsah adresáru
cd /c	c:	Zmení adresár na koreň disku C:

Text zo schránky vkladáme pomocou Shift+Insert. Podrobná nápoveda k príkazu - parameter - - help

```
$ mkdir --help
```

Založenie repozitára

V ľubovoľnom existujúcom adresári zadáme príkaz `git init`

```
$ cd /c
$ mkdir git-workshop
$ mkdir testproject
$ cd git-workshop/testproject
$ git init
Initialized empty Git repository in c:/git-workshop/testproj
```

Po tomto kroku môžeme začať verzovať ľubovoľné súbory a adresáre, ktoré sa nachádzajú v adresári `testrepo`.

Pred prvým commitom

Po inštalácii Gitu je dobré nastaviť do konfigurácie správne údaje o autorovi.

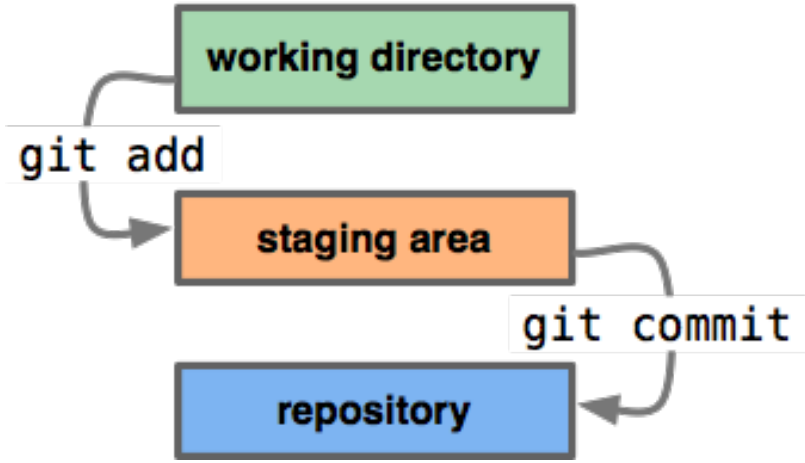
```
$ git config --global user.name "Michal Muthsam"
$ git config --global user.email muthsam@ecommerce.cz
```

Ďalej si pripravíme nejaký súbor, ktorý budeme v ďalšom kroku verzovať. Vytvoríme napríklad textový súbor ahoj.txt a vložíme do neho nejaký text. Toto môžeme spraviť v Notepade (prípadne inom editore), alebo priamo z konzole.

```
$ cd /c/testproject
$ touch ahoj.txt
$ echo Nazdar svet! > ahoj.txt
```

Náš prvý commit

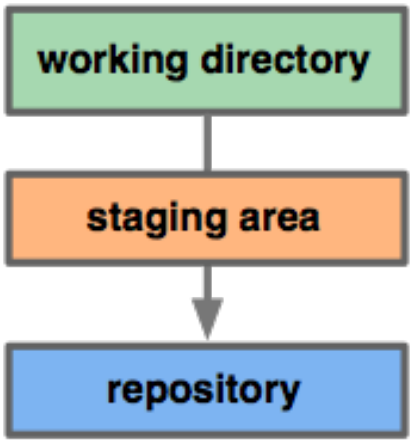
Máme pripravený súbor a radi by sme ho pridali do repozitára. Najskôr označíme zmeny, ktoré chceme zahrnúť do commitu, v tomto prípade súbor ahoj.txt, pomocou príkazu git add.
Potom použijeme git commit, s parametrom -m, ktorým pridáme popis commitu.




```
$ git add ahoj.txt
$ git commit -m "Prvy commit do repa"
[master (root-commit) deb7c00] Prvy commit do repa
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 ahoj.txt
```

Commit "po starom"

Samozrejme niekedy by sme uvítali automatický commit všetkých zmien, bez nutnosti ich označovania. K tomu to slúži parameter -a u príkazu git commit.



`git commit -a`

```
$ echo Hello world! >> ahoj.txt
$ git commit -a -m "Pridany anglicky pozdrav"
[master 141d9c3] Pridany anglicky pozdrav
 1 files changed, 1 insertions(+), 0 deletions(-)
```

Vytvorenie verejného repozitára

Git umožňuje pripojenie na vzdialený repozitár prostredníctvom HTTP, SSH, vlastného GIT protokolu, prípadne priamo cez súborový systém.
Pre ilustráciu práce so vzdialeným repozitárom si vytvoríme takzvaný "bare"repozitár. Tento neobsahuje pracovnú kópiu a používa sa pre publikovanie zmien pre "okolitý svet". Niekedy sa používa aj označenie public repository. Pre jednoduchšie rozpoznanie je názov takéhoto repozitára doplnený príponou .git .

```
$ cd /c/git-workshop
$ mkdir public_repos
$ cd public_repos
$ git clone --bare ../testproject
Cloning into bare repository testproject.git...
done.
```

Pripojenie na existujúci repozitár

Git umožňuje pripojenie na vzdialený repozitár prostredníctvom HTTP, SSH, vlastného GIT protokolu, prípadne priamo cez súborový systém.
Aby sme nemuseli zadávať pri operáciách celú cestu k nášmu public repozitáru, pridáme si ho medzi tzv. remote.

```
$ cd /c/git-workshop/testproject
$ git remote add origin ../public_repos/testproject.git/
$ git remote
origin
```

Odteraz sa na náš public repozitár môžeme odkazovať pod menom "origin". Podobne je možné pridať viacero ďalších aliasov pre vzdialené repozitáre. Všetky nastavené si zobrazíme príkazom git remote.

Odosielanie a sťahovanie zmien

Pre operácie so vzdialeným repozitárom slúžia (okrem iných) príkazy git push a git pull.
Na ukážku zmenu v repozitári testproject vypropagujeme do public repozitára testproject.git (origin).

```
$ cd /c/git-workshop/testproject
$ sed -i 's/Nazdar/Ahojl/g' ahoj.txt
$ git commit -a -m "zamena nazdar za ahoj"
[master 483f007] zamena nazdar za ahoj
 1 files changed, 1 insertions(+), 1 deletions(-)
$ git push origin master
Counting objects: 5, done.
...
To ../public_repos/testproject.git/
    141d9c3..483f007  mastes -> master
```