Git

Ľubomír Prda

IT4Innovations



IT4Innovations#
národní01#$%@&0
superpočítačové
centrum$@00&1@&

lubomir.prda@vsb.cz

support@it4i.cz

PRACE

VCS – Version Control System

Versioning - creation and management of multiple releases of a product, all of which have the same general function but are improved, upgraded or customized.

- We need VCS for
  - History tracking. Who? When? What?
  - Fast roll-back to previously working version
  - Natural back-up
  - Merging changes from large number of contributors
  - Environment management (testing, stable, LTS)
  - Hot-fix deployment

VCS – Version Control System

Revision (commit) – base unit of versioning. Singular logical result of work

Version – set of logically aggregated commits based on some key (weekly version, beta version, free version)

- A good commit is
  - Bug fix
  - New feature
  - Functionality, that cannot be logically split
  - Change, that might be rolled back

Git - the stupid content tracker

- Git
  - 2005 – Linus Torvalds
  - Terminal based VCS
  - Focus on non-linear development, speed and huge projects
  - Designed for development of Linux kernel
  - Can version anything
  - Garbage accumulated until collected
  - Uses well established protocols for security (ssh, https)
  - Objects identified as SHA-1 hashes of its content. In case of commits also all of its history.
  - 3 data types: BLOB, TREE, COMMIT (bd9dbf5aae1a3862dd1526723246b20206e5fc37)

Setting it up

- # <apt|yum> install git

- # export GIT_AUTHOR_NAME="Name Surname"
- # export GIT_AUTHOR_EMAIL=superdud@email.com
- # export GIT_COMMITTER_NAME="$GIT_AUTHOR_NAME"
- # export GIT_COMMITTER_EMAIL="$GIT_AUTHOR_EMAIL"

- # git config --global color.ui auto
- # git config --global core.editor vim
  - # git config --global core.editor “gedit -w”

- Settings are stored in $HOME/.gitconfig
  - allow environment forwarding or use tools like sshrc if you commit from a shared machine

Let’s begin

- # git init / git clone <URL>
- # git status // use every time, you do not know what to do

- # git add <PATH> // select files to take snapshot from
  - # git add <PATH> --patch // select individual lines

- # git diff [--cached] // see, what you are commiting

- # git commit // create the snapshot
  - # git commit –am “Commit message”

Branching

Branch - named pointer to a revision. Name HEAD is reserved and pointing to a revision/branch, that is currently worked on

- # git checkout -b <NAME> // create new branch named NAME
- # git checkout <NAME> // switch current branch to NAME
- # git checkout <NAME> <PATH> // switch just one file to look like the one in branch/revision NAME

- # git checkout = I want my files to look, like they look in branch/revision NAME

- # git branch [-v] // list branches
- # git merge <NAME> // merge branch into current one

Collision and history

Collision – happens when 2 people try to edit the same part of a document/code. Git does not know, how to merge them and lets the merging user decide, what to do. Just use git status

- # git log // just see, what has been done
  - # git log --oneline --decorate
  - # git log --oneline --graph
  - # git log <PATH>
  - # git log --grep <PATTERN>

- # git show [NAME] // see changes done to files
- # git blame [PATH] // see who changed a file, line by line

Rewind, shuffle, edit

WARNING!! Following commands are destructive. To be used only on private branches. Changing the history changes all commits up to the latest including their SHA1 identifier

- # git reset <NAME> // rewind current branch back to NAME
  - # git reset --hard <NAME>
  - # git reset --soft <NAME>

- # git rebase <NAME> // append all changes done in this branch to another branch
- # git rebase --interactive <NAME> // interactively edit all commits done after NAME

Remote repository

- # git remote add <NAME> <URL> // add remote repository info to local repository
  - Default remote repository is called origin
  - Cloning remote repository automatically adds it as origin

- # git fetch // get updates from remote repository
- # git pull // fetch and merge remote changes into local repository
  - # git pull --rebase // replace local repository with remote version and append local changes to the end
- # git push [--force] // upload local changes to remote repository

Thank you for your attention!

Ľubomír Prda

IT4Innovations



IT4Innovations#
národní01#$%@&0
superpočítačové
centrum$@00&1@&

PRACE