Version Control Systems
(and an introduction to git)

Steven Fernandez <steve@lonetwin.net>
Jun 2009

What is a Version Control System ?

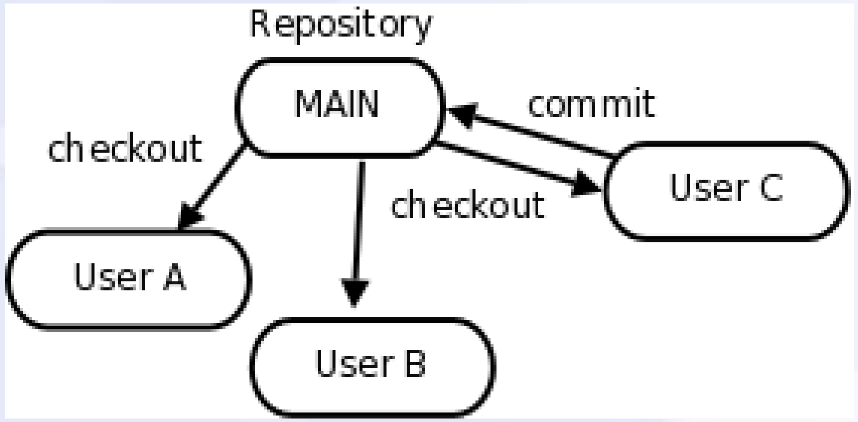
- Editing documents is an iterative activity
- Editing source code is an iterative, team activity
- A Version Control System (VCS) is a tool which helps us control these activities by keeping track of changes using some manner of data versioning

Why do you need a VCS ?

- Helps keep track of changes
  - Multiple people typically work on the same code base, simultaneously
  - Even with a single coder, the ability to track changes is very useful (for instance, to trace changes that introduced bugs)
- Makes it easier to share changes, try out different ideas, revert to an earlier state ...etc

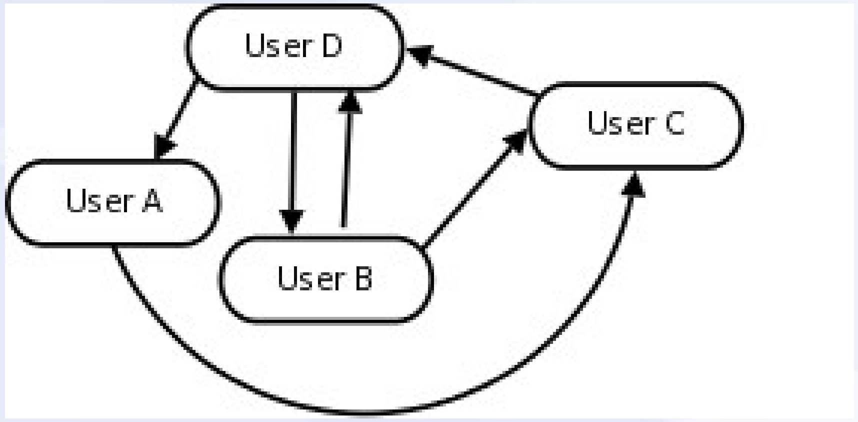
Types of VCS - Simple

- Single *central* repository
- All changes tracked in one place, everyone syncs with this main repository
- Different branches of development tracked centrally and users work on the copy of a branch
- Committing changes or changing the branch involves an update/sync with the server
- Examples: cvs, svn



Types of VCS - Distributed

- De-centralized repository
- Changes tracked in individual repositories. Anyone can sync with any repository
- Different branches of development tracked locally
- Committing changes or changing branches an *inexpensive and fast* process
- Examples: git, mercurial, bazaar



Introduction to GIT

- Designed for distributed development
  - Every repository is server as well as client
  - Every repository has ability to track history of changes, create branches and sync with any other repository
- Tracks content rather than files
- Is especially suitable for large projects due to it's speed and it's *cheap branching*

Getting started with git

- Installation:
  Fedora: $ yum install git-core
  Debian: $ apt-get install git-core
  OpenSuse: $ yast install git-core
  Mac: $ port install git-core
  Windows: http://code.google.com/p/msysgit/

  Source Code: http://kernel.org/pub/software/scm/git/

- Git command line structure

```
$ git [action] <arguments>
      OR
$ git-action <arguments>
```

Using git – A new project

- Getting help

```
$ git help <command>
```

- Create a repository

```
$ mkdir myproject
$ cd myproject
$ git init
```

- Add files to the repository

```
...<create coolapp.c and coolapp.h>...
$ git add coolapp.c coolapp.h
```

- Commit changes to the repository

```
$ git commit -m “Created my cool new app”
```

Using git – Basic Operations

```
$ vi coolapp.c                  # Modify your code
$ vi coolapp_client.c           # Create a new file
$ git add coolapp_client.c      # Tell git about your new file
$ git status                    # Get a summary of changes since your
                                # last commit
$ git commit -a -m “Included coolapp client API and created client”
$ git log                       # show commit logs
```

Using git – clone and pull

- git clone

```
$ git clone git://git.kernel.org/pub/scm/.../linux-2.6 my2.6
$ cd my2.6
<make some changes>
$ git commit -m “Explain the changes”   # commit to your repo.
$ git format-patch origin       # Prepare a patch for submission
        OR
$ <do nothing !>                # Anyone can git-pull or git-clone
                                # from you
```

- git pull

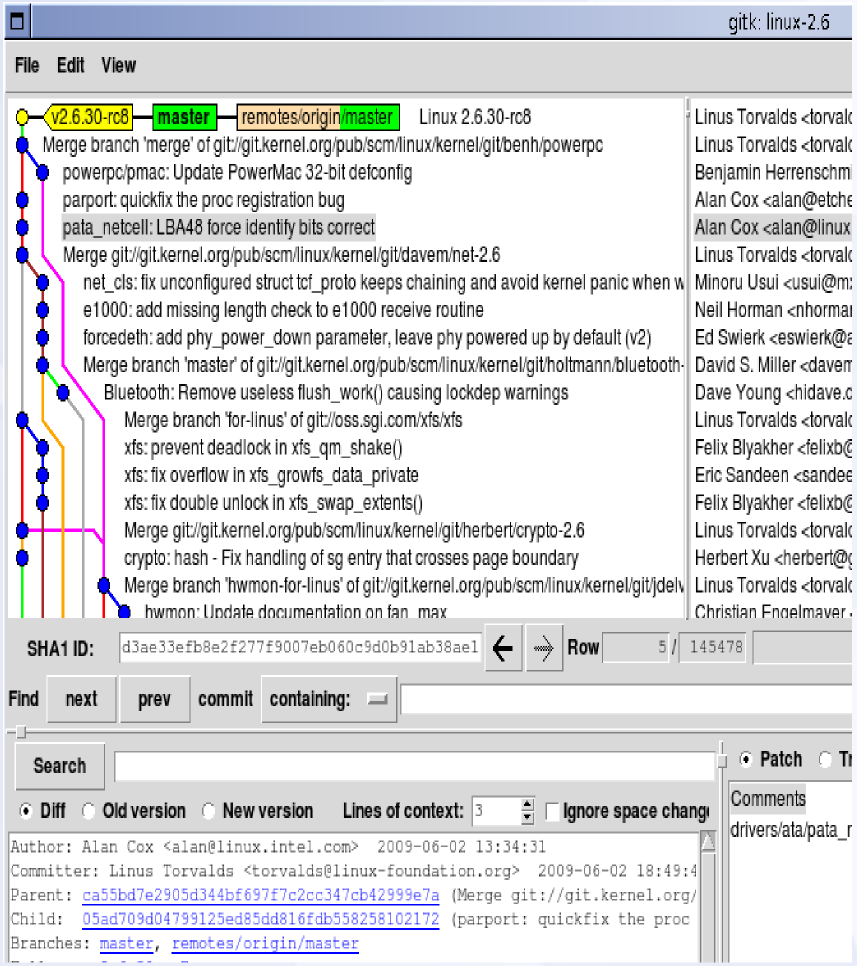
```
$ git pull <repository>         # Pull changes from <repository> and
                                # merge into local git repository
```

Using git – Branches

```
$ git branch new_feature                 # create a branch

$ git branch                             # list all the existing branches

$ git checkout new_feature               # switch current branch
<make changes>
$ git commit -a -m “describe changes”

$ git log master..                       # see log since branching

$ git checkout -b exprerimental          # create a branch and switch to it
<make changes>
$ git commit -a -m “describe changes”
$ git log master..experimental           # see log since branching from
                                         # master branch

$ git log new_feature..experimental      # see log since branching from
                                         # new_feature branch
```

Using git – Tracking changes

```
$ gitk

$ git log –pretty=oneline foo

$ git diff 6e4291..f404d2e foo
$ git diff HEAD^..HEAD

$ git blame -L 40,60 foo
$ git blame  v2.6.18.. -- foo

$ git tag v1.1

$ git revert  6e4291
$ git reset --hard
```



```
v2.6.30-rc8  master  remotes/origin/master  Linux 2.6.30-rc8                                    Linus Torvalds <torvalc
Merge branch 'merge' of git://git.kernel.org/pub/scm/linux/kernel/git/benh/powerpc              Linus Torvalds <torvalc
  powerpc/pmac: Update PowerMac 32-bit defconfig                                                Benjamin Herrenschmi
  parport: quickfix the proc registration bug                                                   Alan Cox <alan@etche
  pata_netcell: LBA48 force identify bits correct                                               Alan Cox <alan@linux
  Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net-2.6                             Linus Torvalds <torvalc
    net_cls: fix unconfigured struct tcf_proto keeps chaining and avoid kernel panic when w     Minoru Usui <usui@m
    e1000: add missing length check to e1000 receive routine                                    Neil Horman <nhormai
    forcedeth: add phy_power_down parameter, leave phy powered up by default (v2)               Ed Swierk <eswierk@a
    Merge branch 'master' of git://git.kernel.org/pub/scm/linux/kernel/git/holtmann/bluetooth-  David S. Miller <davem
      Bluetooth: Remove useless flush_work() causing lockdep warnings                           Dave Young <hidave.c
        Merge branch 'for-linus' of git://oss.sgi.com/xfs/xfs                                   Linus Torvalds <torvalc
        xfs: prevent deadlock in xfs_qm_shake()                                                 Felix Blyakher <felixb@
        xfs: fix overflow in xfs_growfs_data_private                                            Eric Sandeen <sandee
        xfs: fix double unlock in xfs_swap_extents()                                            Felix Blyakher <felixb@
        Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6                  Linus Torvalds <torvalc
        crypto: hash - Fix handling of sg entry that crosses page boundary                      Herbert Xu <herbert@
        Merge branch 'hwmon-for-linus' of git://git.kernel.org/pub/scm/linux/kernel/git/jdelv   Linus Torvalds <torvalc
          hwmon: Update documentation on fan_max                                                Christian Engelmayer

SHA1 ID: d3ae33efb8e2f277f9007eb060c9d0b91ab38ae1   Row 5 / 145478

(•) Diff  ( ) Old version  ( ) New version   Lines of context: 3   [ ] Ignore space chang     (•) Patch ( ) Tr

Author: Alan Cox <alan@linux.intel.com>  2009-06-02 13:34:31                                  Comments
Committer: Linus Torvalds <torvalds@linux-foundation.org>  2009-06-02 18:49:4                 drivers/ata/pata_r
Parent: ca55bd7e2905d344bf697f7c2cc347cb42999e7a (Merge git://git.kernel.org/
Child:  05ad709d04799125ed85dd816fdb558258102172 (parport: quickfix the proc
Branches: master, remotes/origin/master
```

Resources

- Version Control: http://en.wikipedia.org/wiki/Version_control
- Essay and comparisons of different version control systems: http://www.dwheeler.com/essays/scm.html
- Introduction to git: http://kernel.org/pub/software/scm/git/docs/v1.2.6/tutorial.html
- Linux Torvalds on git: http://www.youtube.com/watch?v=4XpnKHJAok8
- Free hosting for projects using git:
  - http://repo.or.cz
  - http://github.com