

# Realm Linux Training: The Git DVCS

Jack Neely  
linux@help.ncsu.edu

Campus Linux Services  
Office of Information Technology  
North Carolina State University

Thursday, July 7, 2011

Copyright © 2011 NC State University

Git Presentation Source:

Copyright © 2008 Karel Zak

Copyright © 2008 Tomas Janousek

Copyright © 2008 Florian Festi

Copyright © 2008 Bart Trojanowski

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

Git Presentation Original Source: <http://kzak.fedorapeople.org/>

Source Code: <git://git.linux.ncsu.edu/bcfg2-training.git>

Please download this presentation and follow along.

`http://go.ncsu.edu/bcfg2-git`



# Table of Contents

- 1 Learning Git
- 2 Conclusions

# Section Outline

## 1 Learning Git

- Introduction to Git
- Workflow
- Git Commands Interface
- Implementation
- Naming Revisions
- Getting started
- Branches
- Real Life with Git
- Handling Patches

# What is Git?

*"I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'." – Linus Torvalds*

- Fast Distributed Version Control System
- Unusually Rich Command Set
- Provides Both High-Level Operations and Full Access to Internals
- Originally Created by Linus Torvalds for Kernel Development
- Design was Inspired by BitKeeper and Monotone
- GNU General Public License, Version 2

# Basic features

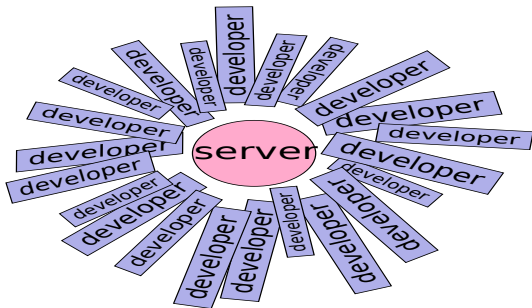
- Distributed Development Model
- Support for Non-Linear Development (Branching and Merging)
- Ready for Large Projects (Very Good Performance)
- Repositories Can be Easily Published: `git://`, `ssh://`, `http://`, `rsync://`, ...
- Cryptographic Authentication of History (GPG-Signed Tags)
- Internally Objects are Addressed by Content (SHA-1) – not filenames
- Local Branches are Local Only (Off-Line Work)

# Section Outline

## 1 Learning Git

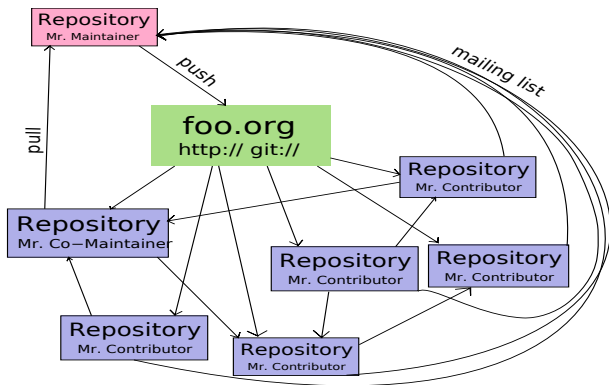
- Introduction to Git
- **Workflow**
- Git Commands Interface
- Implementation
- Naming Revisions
- Getting started
- Branches
- Real Life with Git
- Handling Patches

# Centralized Model



- Extra Policy for Write Access
- VCS Is Not Development Tool, But Source Code Archive Only
- Every Change Has Impact To All Developers
- No Private Branches
- Needs Connection to Server for Most Operations

# Distributed Model



- Maintainer Has Full Control Over Primary (His) Repository
- Support for Non-Linear Development
- Repositories Can Be Easily Published

# Git Improves Your Work Manners and Habits

- Branching and Merging are Cheap
  - You Can Prototype
  - You Can Collaborate with Other Developers on Incomplete and Unstable Stuff
  - You Can Easily (e.g. Every Day) Rebase Your Changes to New Upstream Code
  - Merge (Rebase) Often Minimizes Conflicts Between Your Patches and Upstream
- Small Patch Is the Best Patch (Patch Per Feature/Change)
  - Reviewers Hate Huge Patches
  - Well Separated Feature or Change Is Easy to Revert
  - Per Item Commit Messages
- Much Less Dependent On Your Patches Going in Upstream
- Manage Patches - Not Just Store Them

# Workflow

Changes go through several stages before ending up in their final destination.

**Working Copy** or Working Directory – Current Checkout – Editing Happens Here

**Index** AKA “Cache” AKA “Staging area” – Changes That Are Selected to Be Committed

**Commit** Now Packaged Up With a Commit Message and Part of the History

Working with other developers and repositories.

**Push or Pull** Push the Commits Over to Public Repository or Request Someone Pull a Topic Branch From Your Repository

**Origin** Track the Upstream Repository’s Changes

# Section Outline

- 1 Learning Git
  - Introduction to Git
  - Workflow
  - **Git Commands Interface**
  - Implementation
  - Naming Revisions
  - Getting started
  - Branches
  - Real Life with Git
  - Handling Patches

# Syntax

- `git <commandname> [options]`
- `git-<commandname> [options]`
- `git help <commandname>`
- `man git-<commandname>`

## High Level Commands: Porcelain

```
$ git commit -a -s -m "cool change"
```

## Low Level Commands: Plumbing

```
$ git rev-list --pretty=oneline v2.13..
```

# Basic Commands (Local)

`git init` Creates an Empty Repository At `./`.`git`

`git add` Adds File Contents To the Index

`git rm` Removes a File

`git mv` Move a File

`git status` Shows the Working Tree Status

`git commit` Records Changes To the Repository

`git log` Shows Commit Log

`git show` Shows Commit (or Another Object)

## Basic Commands (Remote)

`git clone` Make a Local Clone of a Remote Repository

`git fetch` Get New Changes From External Repository (Origin)

`git pull` Fetches and Merges Updates Into Working Tree

`git push` Write New Changes To External Repository

`git format-patch` Exports a Change as a Patch

`git send-email` Sends Patch(s)

`git am` Applies a Series of Patches From a Mailbox

# More Local Commands

`git branch` Create/Show/Modify/Delete Branches

`git tag` Create/Remove Symbolic Tags

`git checkout` Switch Work Directory To Another Branch/Commit

`git merge` Merge Two or More Branches

`git rebase` Changes Starting Point of a Branch

`git cherry-pick` Copy Specific Change From Another Branch

`git reset` Set Back a Branch HEAD

`git bisect` Find the Breaking Patch

`git stash` Save/Restore Current Work Directory Changes

`git gc` Compact and Garbage Collect Repository

`git reflog` Show/Expire the Reference Log

# Section Outline

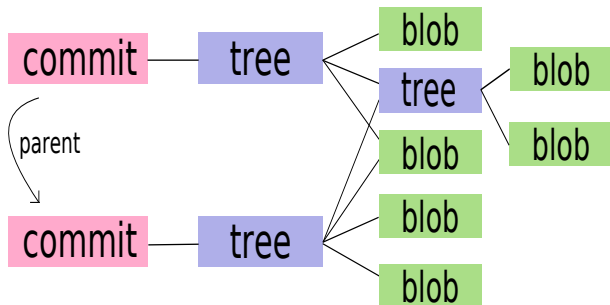
- 1 Learning Git
  - Introduction to Git
  - Workflow
  - Git Commands Interface
  - **Implementation**
  - Naming Revisions
  - Getting started
  - Branches
  - Real Life with Git
  - Handling Patches

# Internal Objects

All objects are content-addressable by SHA-1.

- commit** Refers to a “Tree” and “Parent” (Connection Into the Project History) and Contains the Commit Message
  - tree** Represents the State of a Single Directory (List of “Blob” Objects and Subtrees)
  - blob** Contains File Data Without any Other Structure

# Internal Objects



# References

- Tag
  - Contains SHA-1 Sum of a Commit
  - May Contain an Explaining Message
  - Can Be GPG-Signed
  - Stays Fixed
  - `.git/refs/tags`
- Branch
  - SHA-1 Sum of a Commit
  - “Leaf” of the History “Tree”
  - Follows the Commits to That Branch
  - `.git/refs/heads`
- Remote Branches - `.git/refs/remotes/` and `origin`
- HEAD - the Current Branch
- ORIG\_HEAD - HEAD Before the Last Reset

# Trust

- Everything is Content-Addressed and Based on SHA-1
- Two Trees Are the Same When HEAD SHA-1 Are Equal
- SHA-1 Sum Is Checked to Verify Data Integrity
- Content, History and Commit Messages Can Be Signed by Only GPG-Signing One Tag

```
$ git tag -v v2.13
object 49ef7acdf77066ed05a6c828c261d332c4f54644
type commit
tag v2.13
tagger Karel Zak <kzak@redhat.com> Tue Aug 28 01:01:35 2007 +0200

stable release v2.13
gpg: Signature made Tue 28 Aug 2007 01:01:35 AM CEST using DSA key ID DC06D885
gpg: Good signature from "Karel Zak <kzak@redhat.com>"
```

# Section Outline

## 1 Learning Git

- Introduction to Git
- Workflow
- Git Commands Interface
- Implementation
- **Naming Revisions**
- Getting started
- Branches
- Real Life with Git
- Handling Patches

# Object Reference

**SHA-1** 40 Hexdigit Object Name

**refname** Human Readable Tag or Branch

**commit<sup>n</sup>** *N<sup>th</sup>* Parent (Caret ^ Symbol)

**commit<sup>n</sup>** *N<sup>th</sup>* Generation Grand-Parent of the Named Commit Object, Following Only the First Parent. (Tilde ~ Symbol)

**HEAD** Refers To the Head of the Current Branch

```
rev~3 = rev^^^ = rev^1^1^1
```

```
$ git reset HEAD^
```

# Advanced Object Reference

`:/regex` Commit Whose Commit Message Matches the Regular Expression

`refname@{date}` Value of the Ref at a Prior Point In Time<sup>1</sup>

`refname:path` The File or Directory as Seen From `refname`

## Prior Point in Time Without Reflog

```
$ git checkout \  
$(git rev-list -n1 --before="Mon Dec 31 2007 23:59:59")
```

```
$ git show HEAD^:somefile.c
```

More Information: `git help revisions`

---

<sup>1</sup>Dependent on the reflog

# Ranges

`r1..r2` Commits Reachable From `r2` But Exclude the Ones Reachable From `r1`

`r1...r2` Commits That Are Reachable From Either One of `r1` or `r2` But Not From Both

```
$ git log v2.13..v2.14
```

```
$ git diff 2.0.7..2.0.9
```

# “Tree-ish”

Lots of commands take a tree as an argument. A tree can be referred to in many different ways, by:

- Name for That Tree
- Name of a Commit That Refers to the Tree
- Name of a Branch Whose Head Refers to That Tree

# Section Outline

## 1 Learning Git

- Introduction to Git
- Workflow
- Git Commands Interface
- Implementation
- Naming Revisions
- **Getting started**
- Branches
- Real Life with Git
- Handling Patches

# Configuration

Global Configuration Is In `~/.gitconfig`

```
$ git config --global --list
  user.name=Florian Festi
  user.email=ffesti@redhat.com
```

Repository Configuration Is In `<repo>/.git/config`

```
$ git config --list
```

# Initial Setup

## Required Setup:

```
$ git config --global user.name "Florian Festi"  
$ git config --global user.email ffesti@redhat.com
```

## Using Colors:

```
$ git config --global color.branch auto  
$ git config --global color.diff auto  
$ git config --global color.interactive auto  
$ git config --global color.status auto
```

~/ .gitconfig

## Simple Sample Config:

```
[user]
  name = Jack Neely
  email = jjneely@ncsu.edu
[aliases]
  co = checkout
  st = status
[color]
  branch = auto
  diff = auto
  interactive = auto
  status = auto
```

See `git help config` for more information.

# Create a Repository

- Create a New Repository

```
$ mkdir project
$ cd project
$ git init
Initialized empty Git repository in /path/to/project/.git/
```

- Clone an Existing Remote Repository (The “Origin” Repository)

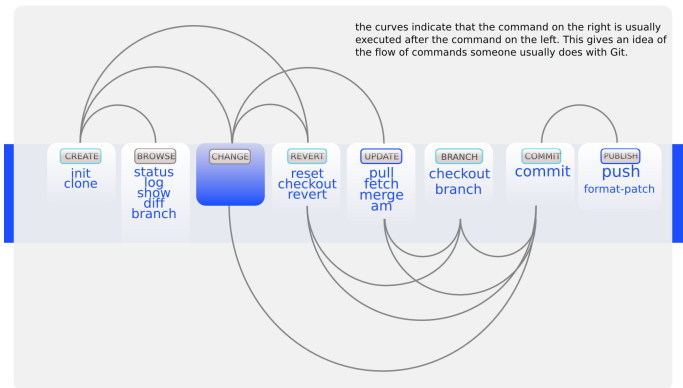
```
$ git clone git://git.linux.ncsu.edu/rlmtools.git
Cloning into rlmtools...
remote: Counting objects: 2173, done.
remote: Compressing objects: 100% (2167/2167), done.
remote: Total 2173 (delta 1553), reused 0 (delta 0)
Receiving objects: 100% (2173/2173), 392.70 KiB, done.
Resolving deltas: 100% (1553/1553), done.
```

# Repository Config File – .git/config

```
[core]
  repositoryformatversion = 0
  filemode = true
  bare = false
  logallrefupdates = true
[remote "origin"]
  fetch = +refs/heads/*:refs/remotes/origin/*
  url = git://git.linux.ncsu.edu/rlmtools.git
[branch "master"]
  remote = origin
  merge = refs/heads/master
```

# Basic Work Flow

## Commands Sequence



Taken from Zack Rusin's Git Cheat Sheet.

<http://zrusin.blogspot.com/2007/09/git-cheat-sheet.html>

See also `git help workflows`

# Visualization

- Visualization Helps When Working with Branches
- Examining History
- <http://git.or.cz/gitwiki/InterfacesFrontendsAndTools>

## History Viewers:

`gitk` Tcl/Tk History Viewer

`qgit` Qt History Viewer, Patch Import/Export

`gitweb` Web Front End (CGI, mod\_perl)

- <http://git.linux.ncsu.edu/git/>

## Commit Tools:

`git gui` Tcl/Tk, Builtin

# Visualization: gitk

The screenshot displays the Git GUI tool `gitk` with the following components:

- Commit History Graph:** A vertical list of commits on the left, each with a colored dot representing a commit. The graph shows a main branch with several branches branching off and merging back. The current commit is highlighted in green.
- Commit Details:** A table on the right showing commit hashes, author names, and timestamps. The current commit is highlighted in green.
- SHA1 ID:** A text field containing the SHA1 hash of the current commit: `e488672nc20720449a7e05cc1104cf69a57b255`.
- Highlight:** A dropdown menu set to "touching paths".
- Search:** A search bar with the text "Search".
- Diff View:** A section showing the diff between the current commit and its parent. It includes the author, committer, parent, child, branch, and follows information. The diff content is shown below.

**Commit Details Table:**

SHA1	Author	Timestamp
e488672nc20720449a7e05cc1104cf69a57b255	Karel Zak <kzak@redhat.com>	2007-09-04 09:45:10
2a457454592ef9544533a524a4649077ef00a8	Karel Zak <kzak@redhat.com>	2007-09-05 11:55:50
38c1f528723d79393cc2dc30764e5f68cc818945c	Pascal Terjan <pterjan@linuxfr.org>	2007-09-04 00:10:16
...	...	...

**Diff View:**

```
Author: Pascal Terjan <pterjan@linuxfr.org> 2007-09-04 00:10:16
Committer: Karel Zak <kzak@redhat.com> 2007-09-05 11:52:28
Parent: 2a457454592ef9544533a524a4649077ef00a8 (rename: add description about option -v to manpage)
Child: 38c1f528723d79393cc2dc30764e5f68cc818945c (:docs: update AUTHORS file)
Branch: master
Follows: v2.13
Precedes:

fdisk: doing useless ioctl when editing an image

When editing a disk image, fdisk wants to ask the kernel to reread the
partition table which is useless and produces an error. A wrong exit
```

# Visualization: qgit

The screenshot shows the QGit application window titled "/home/ffesti/ CVS/yum-ffesti - QGit". The interface includes a menu bar (File, Edit, View, Actions, Help), a toolbar with icons for file operations, and a search field containing the commit hash "6801f08346735d9b3063a0d0421360c2ac7000a6".

The main area is divided into two tabs: "Rev list" and "Patch". The "Rev list" tab is active, showing a commit graph on the left and a list of commits in the center. The commit list has columns for "Graph", "Short Log", and "Author".

Graph	Short Log	Author
	<b>mesu</b> Check only internal checksum or already existing sqliteab files to allow creat... Use a Least Recently Used cache for sqlite packages PRCOs.	Florian Festi <mesu@redhat.com>
	Move most operation to be based on pkgKey instead of pkgId to gain some speed.	Florian Festi <ffesti@redhat.com>
	move creation of sqlite idices to sqlitesack to create them even if the metadatapa...	Florian Festi <ffesti@redhat.com>
	<b>gold</b> Add tsInfo.getUnresolvedMembers() and use if for depsolving	Florian Festi <ffesti@redhat.com>
	Add Obsoletes Ping Pong	Florian Festi <ffesti@redhat.com>
	<b>origin/master</b> Add some erase depsolve tests	James Bowes <jbowes@redhat.com>
	Remove duplication of settestpath contents	James Bowes <jbowes@redhat.com>
	make sure pkgs are added to the tsInfo for depsolving in the proper mode	Seth Vidal <skvidl@redhat.com>
	expose copy_local sensibly to callers	Seth Vidal <skvidl@redhat.com>
	Merge branch 'master' of ssh://login.linux.duke.edu/home/groups/yum/git/yum	James Antill <jantill@redhat.com>
	Fix copy/paste error	Florian Festi <ffesti@redhat.com>
	Packages that are to be removed can't be local packages.	Florian Festi <ffesti@redhat.com>

Below the commit list, there is a section for the selected commit (origin/master):

Author: Florian Festi <ffesti@redhat.com>  
Date: 11/09/2007 05:21:45 PM  
Parent: [Update test case](#)  
Child: [Nothing to commit](#)  
Follows: yum-3-2-7 ([3.2.7 changelog merge](#))

On the right side, there is a file browser showing the file structure:

- yum/\_init\_.py
- yum/depsolve.py (selected)

# Visualization: Gitweb

The screenshot shows a web browser window displaying the Gitweb interface for a repository. The browser's address bar shows the URL `http://git.kernel.org/?p=utils/util-linux-ng/util-linux-ng.git`. The page title is `/pub/scm / utils/util-linux-ng/util-linux-ng.git / summary`. A search bar is present with a dropdown menu set to "commit" and a search input field. Below the search bar, the repository's description and owner information are displayed. The "shortlog" section lists recent commits with their authors, commit messages, and links to commit details, commit diffs, trees, and snapshots.

File Edit View History Bookmarks Tools Help

http://git.kernel.org/?p=utils/util-linux-ng/util-linux-ng.git Google

[/pub/scm](#) / [utils/util-linux-ng/util-linux-ng.git](#) / **summary**

summary | [shortlog](#) | [log](#) | [commit](#) | [commitdiff](#) | [tree](#)  search:

description The util-linux-ng code repository. See <http://kernel.org/~kzak/util-linux-ng/> for more details.  
owner Karel Zak  
last change Wed, 5 Sep 2007 09:55:50 +0000  
URL <git://git.kernel.org/pub/scm/utils/util-linux-ng/util-linux-ng.git>  
<http://www.kernel.org/pub/scm/utils/util-linux-ng/util-linux-ng.git>

**shortlog**

5 days ago	Karel Zak	<b>docs: update AUTHORS file</b> <a href="#">master</a>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
5 days ago	Pascal Terjan	<b>fdisk: doing useless ioctl when editing an image</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
5 days ago	Li Zefan	<b>rename: add description about option -V to manpage</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
5 days ago	Li Zefan	<b>rename: remove useless variable</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
5 days ago	LaMont Jones	<b>build-sys: remove files that are no longer delivered ...</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
5 days ago	Li Zefan	<b>cal: add description about option -V to manpage</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
5 days ago	Norbert Buchmuller	<b>mount: chain of symlinks to fstab causes use of pointer ...</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
7 days ago	Arkadiusz Miskiewicz	<b>setarch: adding groff symlinks to setarch manual page</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
7 days ago	LaMont Jones	<b>mount: improve error message when helper program not ...</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
7 days ago	Karel Zak	<b>docs: fix stable branche name in README.devel</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
8 days ago	Karel Zak	<b>fdisk: fix typo</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
8 days ago	Karel Zak	<b>build-sys: autogen.sh reports versions of autotools now</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
8 days ago	Karel Zak	<b>build-sys: set AC_PREREQ to 2.60, increment version ...</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
8 days ago	A. Costa	<b>flock: typo in man page</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
13 days ago	Karel Zak	<b>build-sys: release++</b> <a href="#">v2.13</a>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>
13 days ago	Karel Zak	<b>docs: update ReleaseNotes</b>	<a href="#">commit</a>   <a href="#">commitdiff</a>   <a href="#">tree</a>   <a href="#">snapshot</a>

Done Adblock

# Browsing Changes

`git log` Shows Commit Logs

`git show` Shows One or More Objects (Blobs, Trees, Tags and Commits)

`git blame` Shows Revision and Author That Last Modified Each Line of a File

`git whatchanged` Shows Logs With Difference Each Commit Introduces

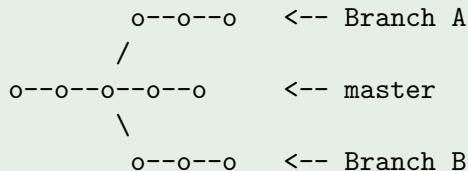
```
$ git log v2.5..                # commits since v2.5
$ git log test..master         # commits reachable from master
                                #   but not test
$ git log --since="2 weeks ago" # commits from the last 2 weeks
$ git log Makefile             # commits which modify Makefile
$ git whatchanged -p 2.0.7..   # Show commits and diffs of changes
                                #   from tag 2.0.7 to HEAD
$ git blame -L 10,15 foo.c     # who modified code between lines
                                #   10 and 15
$ git show c1a47c171b         # shows selected object (commit)
```

# Section Outline

## 1 Learning Git

- Introduction to Git
- Workflow
- Git Commands Interface
- Implementation
- Naming Revisions
- Getting started
- **Branches**
- Real Life with Git
- Handling Patches

# Branches



**branch** A Line of Development

**branch head** A Reference To the Most Recent Commit On a Branch

- Branches Become Remote Branches When Cloning a Repository
- Use `git branch -a` (All) or `git branch -r` (Remote) to See the Remote Branches

# Manipulating Branches

- `git branch` Lists, Creates, or Deletes Branches
- `git checkout <branch>` Makes the Current Branch `<branch>`, Updating the Working Directory
- `git checkout -b <branch>` Creates a New Branch `<branch>` from HEAD and Checks It Out
- `git checkout -b <branch> -t <remote branch>` Create a New Local Branch Tracking a Remote Branch
- `git diff <branch>..<branch>` Unified Diffs Between Branches

```
$ git checkout -b openafs-stable-1_6_x \  
-t origin/openafs-stable-1_6_x
```

# Merge Branch

## Before

```
      A---B---C topic
      /
D---E---F---G master
```

## Command

```
$ git merge topic
```

## After

```
      A---B---C topic
      /           \
D---E---F---G---H master
```

# Rebase Branch

## Before

```
      A---B---C topic
     /
D---E---F---G master
```

## Command

```
$ git rebase master topic
```

## After

```
      A'---B'---C' topic
     /
D---E---F---G master
```

# Merge vs Rebase

- Merge
  - Does a 3-Way Merge (For Simple Cases)
  - Leads to Non-Linear History
  - Merging Several Branches with Each Other looks Messy (But Isn't)
  - Keeps Separate Branches Visible and Usable
  - Use in Public Repositories!
- Rebase
  - Reapplies the Patches On Top
  - Alters History - New Patches with New SHA-1 Sums
  - Breaks Work Based on That Branch
  - Not Suited for Published Repositories
  - Allows Creating the "Perfect Patch" Against Upstream
  - Use for Private Work!

Read `git help merge` and `git help rebase` for details!

# Resolve Conflicts

- Read the Messages
  - `git status`
  - Git Will Tell You What to Do
- Conflicts Are Applied To the Working Directory Only
- Add Resolved Stuff to the Index with `git add`
- For Merges: `git commit`
- For Rebase
  - `git rebase --continue`
  - `git rebase --abort`
  - `git rebase --skip`
- Remember to Commit Early and Often

# Merge Conflict Example

```
$ git merge foo
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.
$ cat file.txt
<<<<<<< HEAD
Goodbye World!
=====
Hello Evil World!
>>>>>>> foo
$ git status
# On branch master
# Unmerged paths:
#   (use "git add/rm <file>..." as appropriate to mark resolution)
#
#       both modified:       file.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
```

# Section Outline

## 1 Learning Git

- Introduction to Git
- Workflow
- Git Commands Interface
- Implementation
- Naming Revisions
- Getting started
- Branches
- **Real Life with Git**
- Handling Patches

# Changes in Project History

- Throw Away Uncommitted Changes: `git reset --hard`
- Deal with Uncommitted Work Later: `git stash`
- Fix the Last Commit: "`git commit --amend`" To Add/Fix Changes and Possibly Alter Commit Message
- Organize Your Own Private Branches
  - "`git cherry-pick`" Select Patch per Patch to Merge Into a Branch
  - "`git rebase -i`" To Freely Reorder Patches
- Deep in Project History
  - "`git rebase`" to Move Around Large Parts of the History
  - "`git revert`" to Commit a Reversed Patch On Top

# Edit 3rd Commit From the HEAD

## 1 Working on Branch Master

```
A--B--C--D--E(master)
```

## 2 Realize You Made a Mistake in Commit B

```
$ git checkout HEAD~3
$ git commit --amend
    .B' (HEAD)
    /
A--B--C--D--E(master)
```

## 3 Bring Back the Other Commits

```
$ git rebase HEAD master
A--B'--C--D--E(master)
```

# Section Outline

## 1 Learning Git

- Introduction to Git
- Workflow
- Git Commands Interface
- Implementation
- Naming Revisions
- Getting started
- Branches
- Real Life with Git
- **Handling Patches**

# Sending a Patch

## Basic Rules:

- One Patch per Email
- Don't Use Stupid Email Clients (e.g. Outlook)
- Don't Use Attachments
- Export Patches with `git format-patch`
- Send Patches with `git send-email`
- Well Formatted Patch is Possible to Apply with `git am`
- Don't Forget to Keep Correct Authorship (e.g. When You Are Not Author of the Patch)
- Use Commit Messages – A Patch Without Comments is Incomplete

# Exporting Patches to Files

```
git format-patch [options] <since|range>
```

- Creates One File per Patch
- Created Patches Are Usable by `git am`

```
$ git format-patch -o ~/ HEAD~5  
/home/kzak/0001-setterm-opened-file-leaving-unclosed.patch  
/home/kzak/0002-sfdisk-opened-files-leaving-unclosed.patch  
/home/kzak/0003-blockdev-fix-opened-file-leaving-unclosed.patch  
/home/kzak/0004-tailf-opened-file-leaving-unclosed.patch  
/home/kzak/0005-tests-use-losetup-s.patch
```

# Send Patches by Email

```
git send-email [options] <file|dir>
```

Takes the patches given on the command line and sends them out in a properly formatted email message(s).

```
$ git send-email --to "God <father@heaven.com>" \  
    ~/0001-make-this-world-better.patch
```

# Patch Description

```
From bfdb8be5c49d8fad825118fb4416ab2a68fc3a16 Mon Sep 17 00:00:00 2001
From: Karel Zak <kzak@redhat.com>
Date: Thu, 25 Oct 2007 12:29:51 +0200
Subject: [PATCH] losetup: canonicalize loopfile name

When setting up a loop device, canonicalize the loop file
name. This simplifies a later identification of loop file names
when querying the loop devices.

Co-Author: Matthias Koenig <mkoenig@suse.de>
Signed-off-by: Matthias Koenig <mkoenig@suse.de>
Signed-off-by: Karel Zak <kzak@redhat.com>
---
 mount/Makefile.am | 4 +--
 mount/lomount.c | 14 ++++++-----
 2 files changed, 14 insertions(+), 4 deletions(-)

diff --git a/mount/Makefile.am b/mount/Makefile.am
index 57a5af8..46bcb5a 100644
--- a/mount/Makefile.am
+++ b/mount/Makefile.am
@@ -25,8 +25,8 @@ umount_LDFLAGS = $(SUID_LDFLAGS) $(AM_LDFLAGS)

swapon_SOURCES = swapon.c swap_constants.h $(utils_common)

-losetup_SOURCES = lomount.c sundries.c xmalloc.c loop.h lomount.h xmalloc.h \
+ sundries.h
+losetup_SOURCES = lomount.c sundries.c xmalloc.c realpath.c \
+ loop.h lomount.h xmalloc.c sundries.h realpath.h
losetup_CPPFLAGS = -DMAIN $(AM_CPPFLAGS)

mount_LDADD = $(LDADD_common)
diff --git a/mount/lomount.c b/mount/lomount.c
index cea3aed..b3c16fb 100644
--- a/mount/lomount.c
+++ b/mount/lomount.c
@@ -24,6 +24,7 @@
#include "nls.h"
#include "sundries.h"
#include "xmalloc.h"
+#include "realpath.h"

#define SIZE(a) (sizeof(a)/sizeof(a[0]))
```

header

commit message

tags

diffstat

patch

# Git References



[Git Website](http://git-scm.com/)

`http://git-scm.com/`



[Git Community Book](http://book.git-scm.com/)

`http://book.git-scm.com/`



[Git Reference](http://gitref.org/)

`http://gitref.org/`



[Git Ready](http://gitready.com/)

`http://gitready.com/`



[Git Hub](http://github.com/)

`http://github.com/`

# Feedback and Questions