



SUNGARD GLOBAL SERVICES

Using Git with Subversion

September 8, 2012

Todd Ricker





Agenda

- » Git overview
- » Basic git usage
- » Branching, merging and rebasing
- » Remote repos – git and subversion
- » Features unique to git
- » Gui tools and other projects



What is Git?

“Git is a free & open source, **distributed version** control system designed to handle everything from small to very large projects with speed and efficiency.

Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server. **Branching and merging are fast** and easy to do.”

Source: <http://git-scm.com/>



A few of my favorite Git features

- » (Almost) any action can be undone
- » Effective branching, sub-branching, and merging
 - *So effective it will change the way you work*
- » Stashing
- » Squashing many local commits into one using interactive rebase



A few more of my favorite Git features

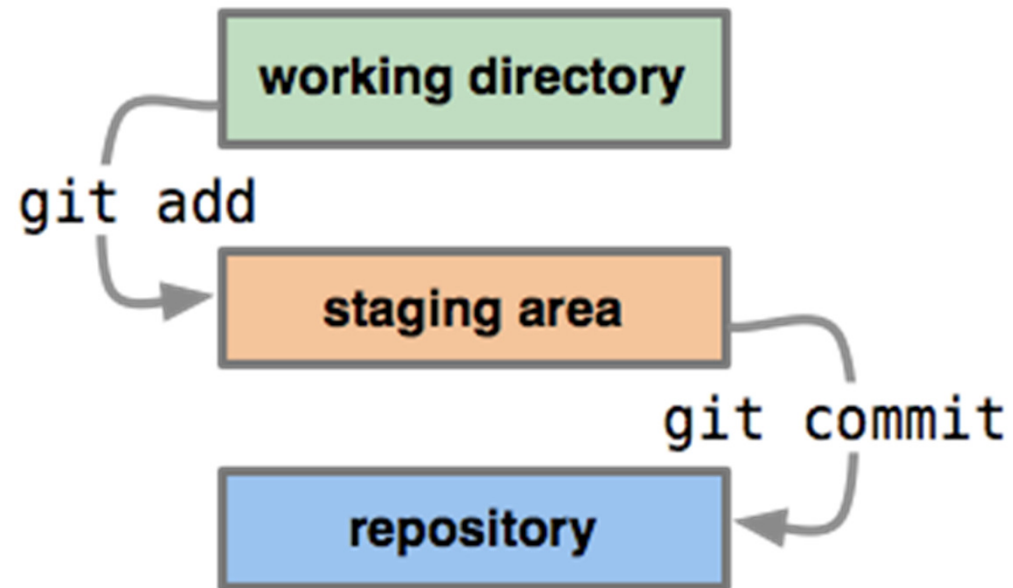
- » Simple enough for even the smallest projects
- » Easily masquerades as a subversion client
- » Tabbed branch name completion CLI
- » My favorite implementation of windows bash shell ships as part of Git for windows



Git: basic usage

- » `mkdir git-hello`
- » `cd git-hello`
- » `touch my-new-file` # create a new empty file
- » `git init` # initialize a new git repo
- » `git add my-new-file` # “stage” the change
- » `git commit -m “added new file”` # commit all staged changes

3 basic Git local states



source: <http://git-scm.com/about/staging-area>



Git: basic usage

» `git log -p` # generate patch, aka view with diff

```
commit 50cf8f48e40b96b3ca35000b9a530166cfcd5131
Author: Todd Ricker <todd.ricker@sungard.com>
Date: Tue Aug 28 12:01:18 2012 -0400

    changed a line

diff --git a/my-new-file b/my-new-file
index ceee30c..77cd400 100644
--- a/my-new-file
+++ b/my-new-file
@@ -1,1 @@
-Aliquam mauris phasellus turpis, elit turpis nunc massa, tempor?
+Ac, magna. Montes purus aliquet elementum, amet mauris amet lacus?
```





~/.gitconfig - basic Setup

- » `$ git config --global user.name "Todd Ricker"`
- » `$ git config --global user.email todd.ricker@sungard.com`
- » `$ git config --global core.editor vim`

~/.gitconfig - basic Setup

```
[user]
  name = Todd Ricker
  email = todd.ricker@sungard.com
[alias]
  br = branch
  ci = commit
  df = diff
  st = status
  sth = stash
  up = pull origin
  who = shortlog -s --
  spull = !git-svn rebase
  spush = !git-svn dcommit
[color]
  ui = auto
  branch = auto
  diff = auto
  status = auto
[merge]
  tool = kdiff3
```





Branching & merging

`git branch defect1234` # creates new branch

`git checkout defect1234`

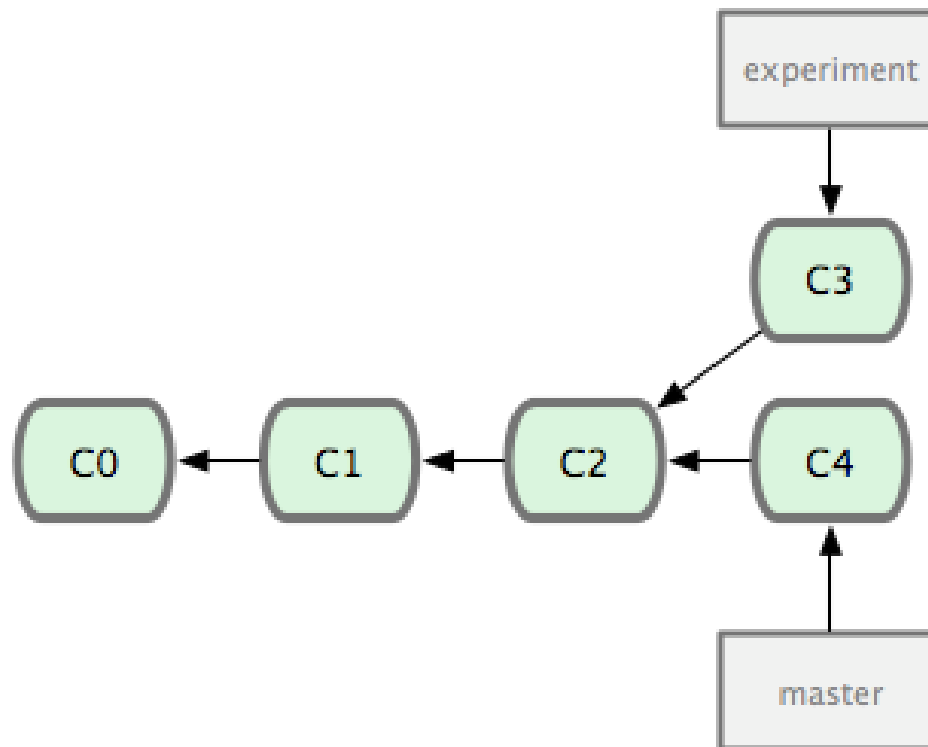
edit files, commit to the branch, and then merge to master

`git checkout master`

`git merge defect1234`

`git -d defect1234` # delete branch

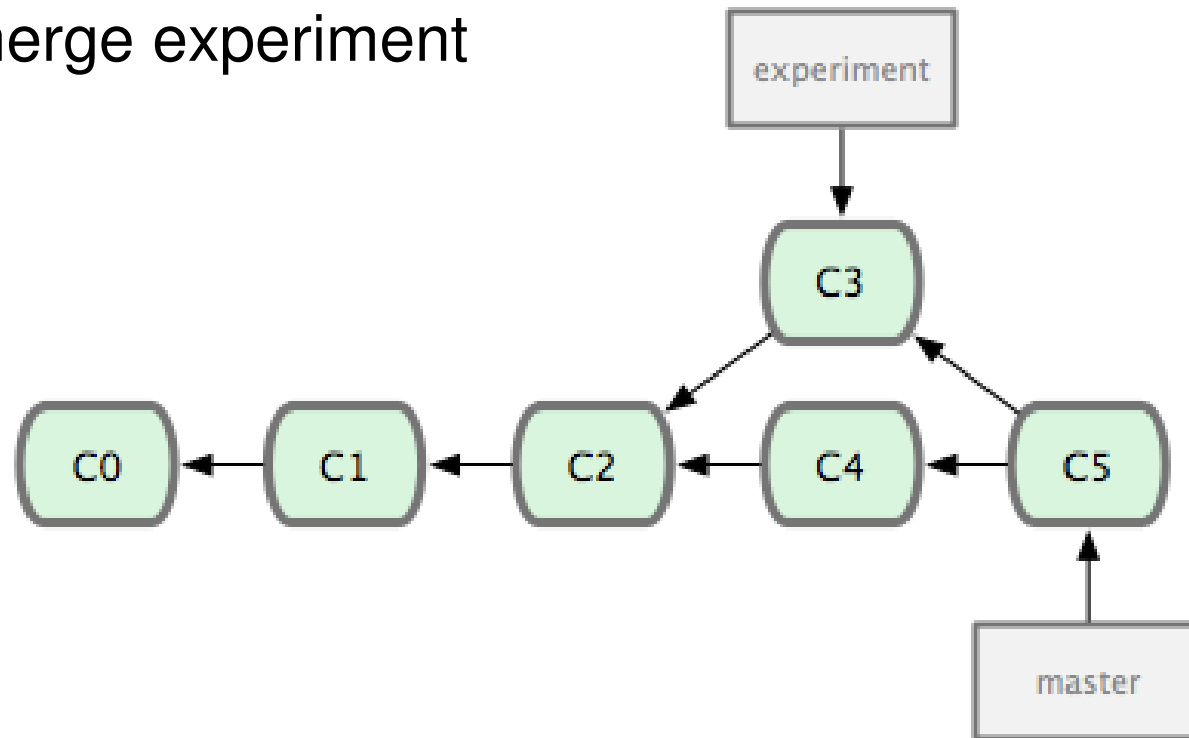
Git merge and rebase



Source: <http://git-scm.com/book/en/Git-Branching-Rebasing>

Git merge

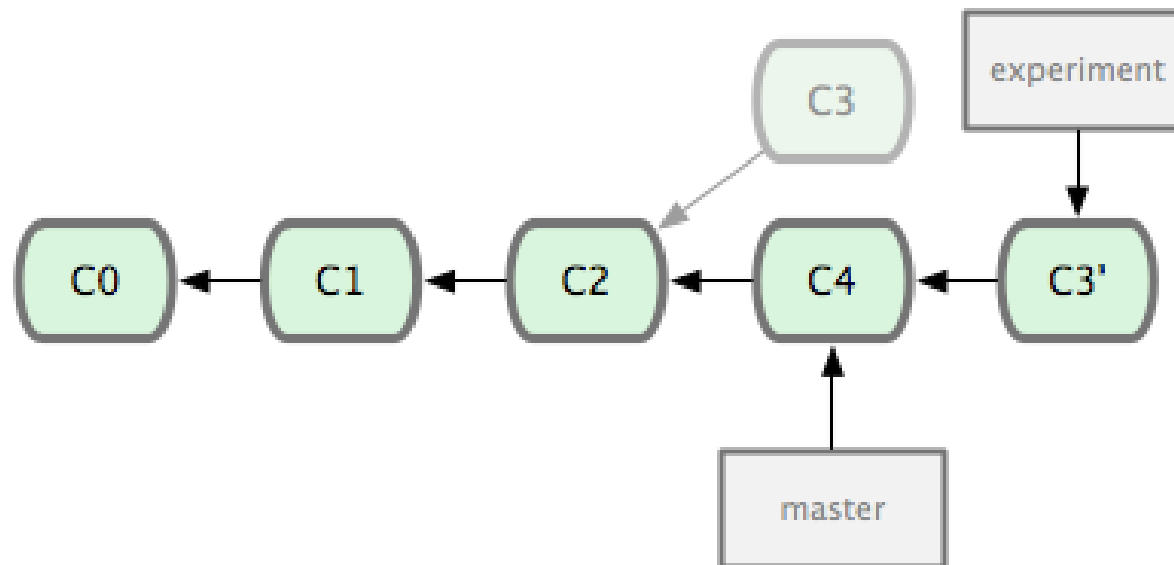
git checkout master
git merge experiment



Source: <http://git-scm.com/book/en/Git-Branching-Rebasing>

Git rebase

- » git checkout experiment
- » git rebase master



Source: <http://git-scm.com/book/en/Git-Branching-Rebasing>



Sub-branching

- » *Good for when you have files locally that need to be overridden, those that you don't want committed but also don't want to ignore.*
 - *eg. local database schema names, file path, etc.*
- » *Create the “prototype” branch*
- » `git branch myoverrides`
- » `git checkout myoverrides`

Make some local edits and commit them to branch.



Sub-branching (continued)

- » *Create your sub-branch, using first branch as a prototype*
 - git branch defect1234 myoverrides
 - *Edit, test, and commit*
 - *Rebase to “subtract” the overridden work*
 - git rebase --onto master myoverrides defect1234

- » *Merge it to master*
 - git checkout master
 - git merge defect1234 (then push)



Merge conflicts!

- » Drops you off a branch
- » Do a git status
- » Fix anything in staging area, manually or using your favorite merge tool
- » `git add`
- » `git rebase --continue`
- » You can bail on the rebase with:
 - `git rebase --abort`



Remote Git repository

- » *The first time only:*
 - `git clone https://github.com/github/gollum`
- » *Hack away, and commit changes locally, then push your changes to the remote repository*
- » `git push`



Remote Git repository

- » *a few days later, get all the most recent updates:*
 - git pull
- » *and repeat!*



Cloning a subversion repository

- » `git svn clone svn://192.168.1.2:80/ProjectFoo`
- » *That brought back too much, clone from a specific subdirectory of SVN instead:*
- » `git svn clone \`
`--trunk trunk/ABC \`
`svn://192.168.1.2:80/ProjectFoo`



Interacting with subversion

- » *Pull the latest Subversion repo changes*
 - git git-svn rebase

- » *Or better yet, use an alias:*
 - git spull



Pulling from subversion

- » *Edit files locally and commit in git, then push your changes back to subversion:*
 - `git git-svn dcommit`

- » *Or better yet, use an alias:*
 - `git spush`



Git with subversion caveats

- » Clone and push only directly to the Subversion server
 - Avoid pull/push/rebasing with other git repos
- » Keep the history you push to subversion repo as linear as possible
 - Use git for its efficient branching & merging, once merged to your master git branch then push to subversion
 - Keep subversion in the dark about the things it's not good at



Squashing local commits


- » `git rebase -i HEAD~3`
 - Where the '3' is the number of commits you want to squash together

```
pick 2675d71 added some text
pick 50cf8f4 changed a line
pick 2409941 fourth commit

# Rebase 484be91..2409941 onto 484be91
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
```


Squashing local commits

- » Choose 's' or 'squash' for some of the commits, and Save



```
pick 2675d71 added some text
s 50cf8f4 changed a line
s 2409941 fourth commit

# Rebase 484be91..2409941 onto 484be91
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```



Squashing local commits

- » Optionally, edit comments and save again

```
# This is a combination of 3 commits.
# The first commit's message is:
added some text

# This is the 2nd commit message:
changed a line

# This is the 3rd commit message:
fourth commit

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# Not currently on any branch.
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   my-new-file
#
~
```

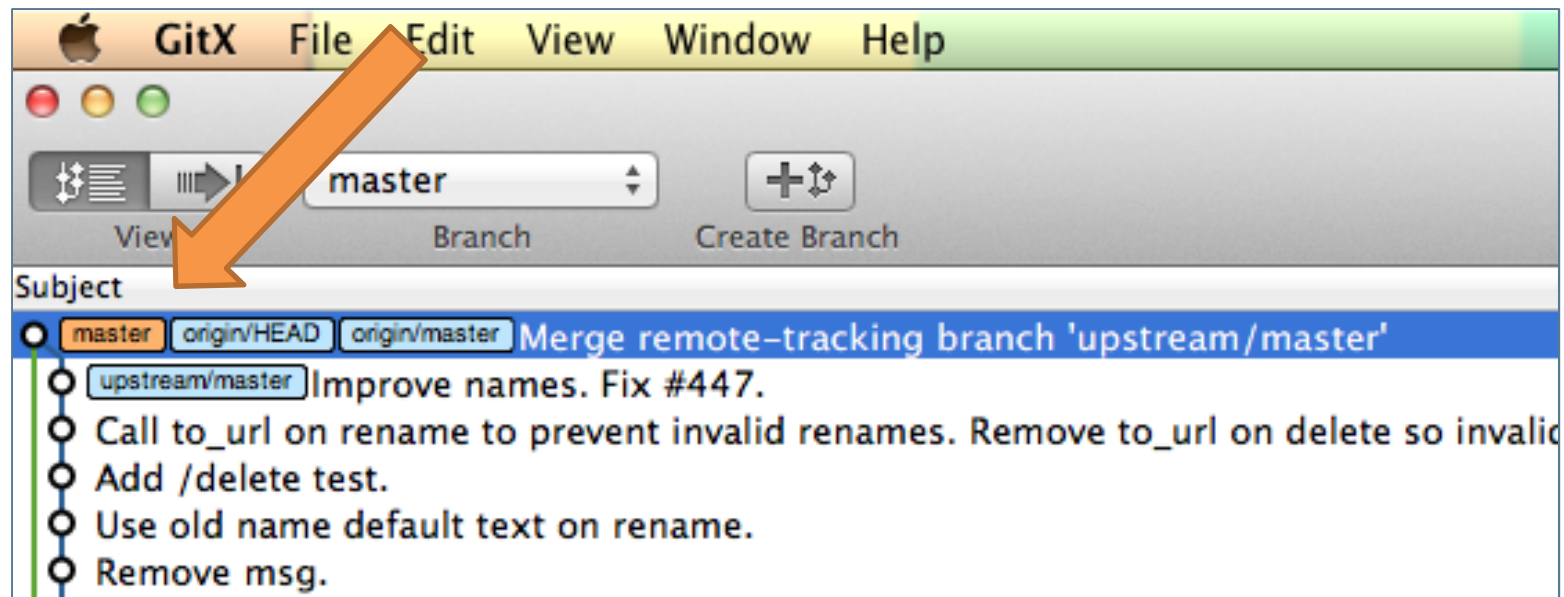


Stashing

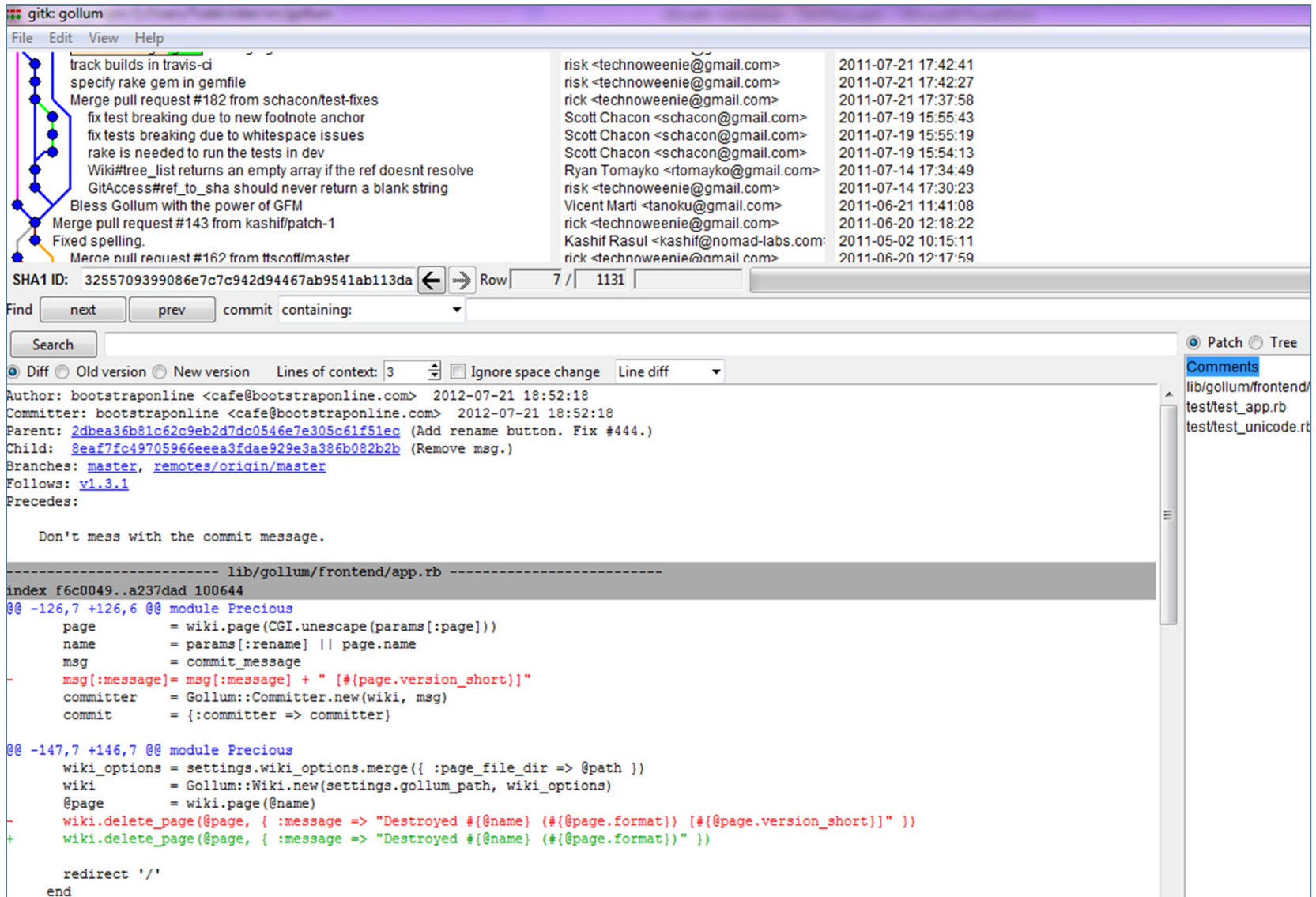
- » Make some edits
- » `git stash save "my stash"`
- » `git stash list`
- » `git stash pop`

```
tricker@hester:~/tmp/git-hello [master] $ git stash list
stash@{0}: On master: my stash
stash@{1}: On master: houston
```

Git GUI – GitX for Mac



Git GUI – Git for Windows



The screenshot shows the Git GUI application window titled "gitk: gollum". The interface includes a menu bar (File, Edit, View, Help), a commit history list on the left, a commit details pane on the right, and a large diff view at the bottom. The commit history list shows a series of commits with their messages and authors. The commit details pane shows the SHA1 ID, author, committer, parent, child, branches, and follows. The diff view shows the changes in the file "lib/gollum/frontend/app.rb", with a search bar and options for diff, old version, new version, lines of context, ignore space change, and line diff. The diff view shows a patch for the file, with changes in the "module Precious" section. The changes include adding a "wiki.delete_page" method and a "redirect" method.

```
gitk: gollum
File Edit View Help

track builds in travis-ci
specify rake gem in gemfile
Merge pull request #182 from schacon/test-fixes
fix test breaking due to new footnote anchor
fix tests breaking due to whitespace issues
rake is needed to run the tests in dev
Wiki#tree_list returns an empty array if the ref doesnt resolve
GitAccess#ref_to_sha should never return a blank string
Bless Gollum with the power of GFM
Merge pull request #143 from kashif/patch-1
Fixed spelling.
Merge null request #162 from tscott/master

SHA1 ID: 3255709399086e7c7c942d94467ab9541ab113da Row 7 / 1131

Find next prev commit containing:

Search

Diff Old version New version Lines of context: 3 Ignore space change Line diff

Author: bootstraponline <cafe@bootstraponline.com> 2012-07-21 18:52:18
Committer: bootstraponline <cafe@bootstraponline.com> 2012-07-21 18:52:18
Parent: 2dbea36b81c62c9eb2d7dc0546e7e305c61f51ec (Add rename button. Fix #444.)
Child: 8eaf7fc49705966eeea3fd9e3a386b082b2b (Remove msg.)
Branches: master, remotes/origin/master
Follows: v1.3.1
Precedes:

Don't mess with the commit message.

----- lib/gollum/frontend/app.rb -----
index f6c0049..a237dad 100644
@@ -126,7 +126,6 @@ module Precious
  page      = wiki.page(CGI.unescape(params[:page]))
  name      = params[:rename] || page.name
  msg       = commit_message
-  msg[:message] = msg[:message] + " [#{@page.version_short}]"
  committer = Gollum::Committer.new(wiki, msg)
  commit    = {:committer => committer}

@@ -147,7 +146,7 @@ module Precious
  wiki_options = settings.wiki_options.merge({ :page_file_dir => @path })
  wiki         = Gollum::Wiki.new(settings.gollum_path, wiki_options)
  @page        = wiki.page(@name)
-  wiki.delete_page(@page, { :message => "Destroyed #{@name} (#{@page.format}) [#{@page.version_short}]" })
+  wiki.delete_page(@page, { :message => "Destroyed #{@name} (#{@page.format})" })

  redirect '/'
end
```

Git GUI – Gigggle for Linux

The screenshot displays the Git GUI interface. The top section shows a commit log with columns for Graph, Short Log, Author, and Date. The bottom section shows a diff view for the file .travis.yml.

Graph	Short Log	Author	Date
	track builds in travis-ci	risk	Jul 21 2011
	specify rake gem in gemfile	risk	Jul 21 2011
	Merge pull request #182 from schacon/test-fixes	rick	Jul 21 2011
	fix test breaking due to new footnote anchor	Scott Chacon	Jul 19 2011
	fix tests breaking due to whitespace issues	Scott Chacon	Jul 19 2011
	rake is needed to run the tests in dev	Scott Chacon	Jul 19 2011
	Wiki#tree_list returns an empty array if the ref doesnt resolve	Ryan Tomayko	Jul 14 2011
	GitAccess#ref_to_sha should never return a blank string	risk	Jul 14 2011
	Bless Gollum with the power of GFM	Vicent Marti	Jun 21 2011
	Merge pull request #143 from kashif/patch-1	rick	Jun 20 2011
	Fixed spelling.	Kashif Rasul	May 02 2011
	Merge pull request #162 from ttscoff/master	rick	Jun 20 2011
	Added spacing to allow has-rightbar and has-footer classes to work	Brett Terpstra	May 23 2011
	Merge pull request #164 from arr2036/patches_1.3.0	rick	Jun 20 2011
	Remove edit button on preview pages	Arran Cudbard-Bell	May 27 2011
	Merge pull request #157 from nealpoole/master	rick	Jun 20 2011
	Fixed broken preview functionality in Gollum.	Neal Poole	May 16 2011
	Merge pull request #169 from JoshCheek/fix_page_file_dir_option	rick	Jun 08 2011
	fix bug where can't edit pages while using page_file_dir	Josh Cheek	Jun 07 2011
	add test that fails when editing pages while using page_file_dir	Josh Cheek	Jun 07 2011

Change 1 of 1
track builds in travis-ci

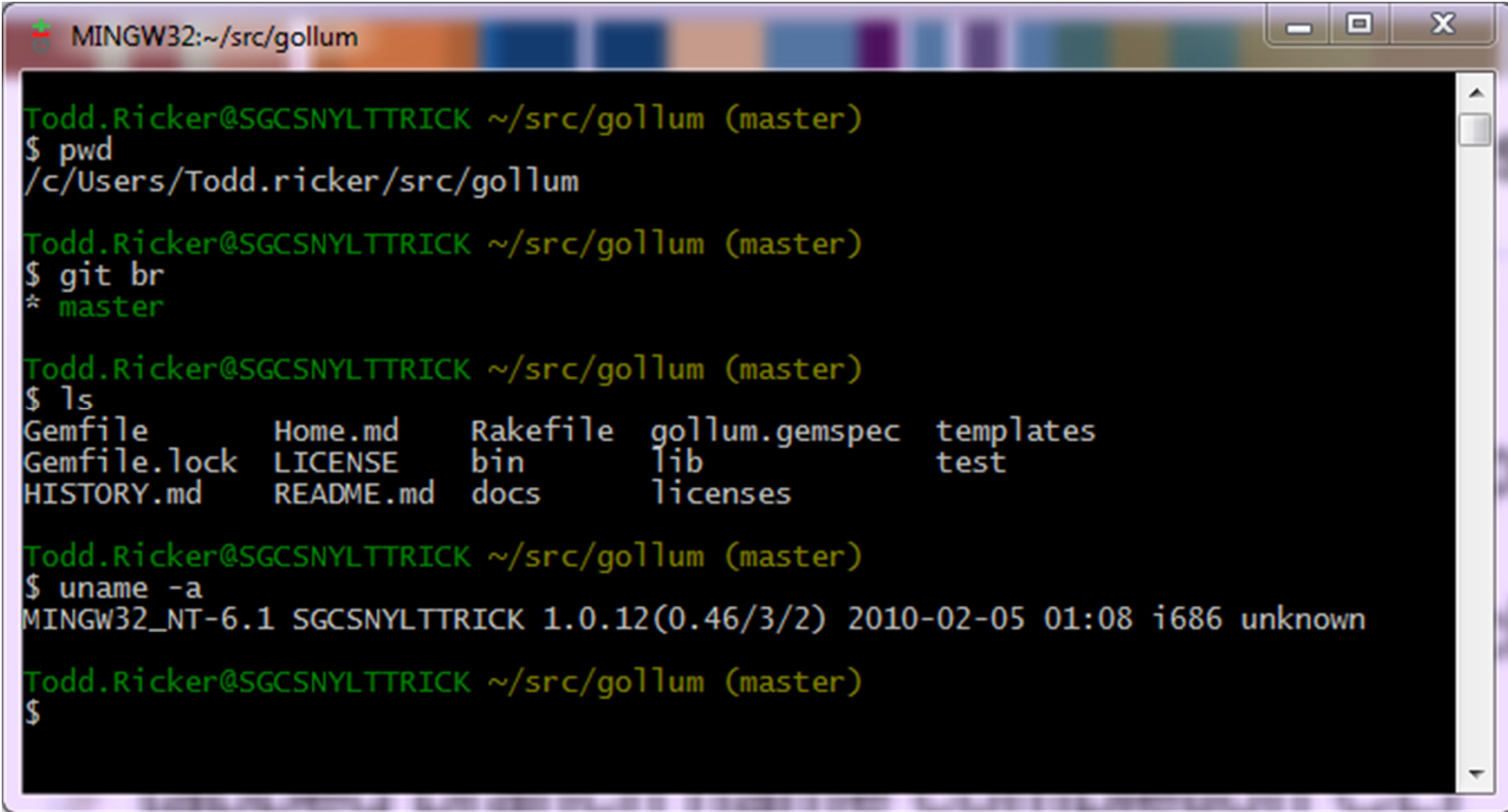
d9b1ea3d5105c9aec2466537b13f62c630e87953
Thu 21 Jul 2011 05:42:41 PM EDT

Changes Details

.travis.yml

```
diff --git a/.travis.yml b/.travis.yml
new file mode 100644
index 0000000..b52778f
--- /dev/null
+++ b/.travis.yml
@@ -0,0 +1,5 @@
+script: "bundle exec rake test"
+rvm:
+  - 1.8.7
+  - 1.9.2
```

Windows Git CLI (and an excellent Bash Shell)



```
MINGW32:~/src/gollum

Todd.Ricker@SGCSNYLTTRICK ~/src/gollum (master)
$ pwd
/c/Users/Todd.ricker/src/gollum

Todd.Ricker@SGCSNYLTTRICK ~/src/gollum (master)
$ git br
* master

Todd.Ricker@SGCSNYLTTRICK ~/src/gollum (master)
$ ls
Gemfile      Home.md      Rakefile     gollum.gemspec  templates
Gemfile.lock LICENSE      bin          lib              test
HISTORY.md   README.md   docs         licenses

Todd.Ricker@SGCSNYLTTRICK ~/src/gollum (master)
$ uname -a
MINGW32_NT-6.1 SGCSNYLTTRICK 1.0.12(0.46/3/2) 2010-02-05 01:08 i686 unknown

Todd.Ricker@SGCSNYLTTRICK ~/src/gollum (master)
$
```



Git as a social network

- » Github – social coding
 - No better way to market yourself as a brilliant developer
- » Github.gist – share code snippets with your friends & colleagues



Other strange and wonderful uses for Git

- » Gollum – a git based wiki
 - Clone your wiki locally and edit with VIM!
- » Gitolite – a git server managed with ... git
- » Gerrit – git code review tool



Other strange and wonderful uses for Git

- » Git as a “database”. Why not? It’s quite good at versioning after all.
- » Embeddable implementations of Git
 - JGit - Java
 - libgit2 – ruby, php, python, obj. c, node, etc.



Next up

- Development Tools for large teams
 - 4:50pm



SUNGARD GLOBAL SERVICES

Contact

Todd Ricker
Senior Manager
SunGard Global Services
340 Madison Ave., 7th Floor
New York, NY 10173

+1 917 727 8633 phone
Todd.ricker@sungard.com

Copyright © 2012 by SunGard Data Systems (or its subsidiaries, "SunGard"). All rights reserved. No parts of this document may be reproduced, transmitted or stored electronically without SunGard's prior written permission.

This document contains SunGard's confidential or proprietary information. By accepting this document, you agree that: (A)(1) if a pre-existing contract containing disclosure and use restrictions exists between your company and SunGard, you and your company will use this information subject to the terms of the pre-existing contract; or (2) if no such pre-existing contract exists, you and your Company agree to protect this information and not reproduce or disclose the information in any way; and (B) SunGard makes no warranties, express or implied, in this document, and SunGard shall not be liable for damages of any kind arising out of use of this document. This document is not a contract of employment, does not guarantee continued employment for any period, and does not alter the at-will status of employment.