**使用git进行版本控制**

本文将介绍一种强大的版本控制工具，git的基本使用。与之前svn工具类似，首先给出一些常见的使用需求，然后以这些需求为中心，来展开git的学习过程。由于我也是在学习当中所以其中不准确之处可以通过后面的联系方式来交流，谢谢。^\_^

主要内容：

简介

基本概念

常用命令

具体实践

其它

**简介**

**=========**

本文将介绍一种强大的版本控制工具，git的基本使用。与之前svn工具类似，本文首先给出一些常见的使用需求，然后以这些需求为中心，来展开git的学习过程。

带着问题学习：

从svn过渡过来的用户初次使用git，会遇到许多的问题，这里以习惯了svn用户的角度，来考虑学习使用git需要解决的一些问题：

\*如何查看库路径信息？

\*如何合并和创建分支？

\*如何恢复到指定的版本？

\*如何查看指定版本的信息？

\*如何理解分布式存储？

\*什么是索引？什么是工作树？

\*如何查看log详细信息？

这些问题也可作为学习之后的练习，我们可以尝试自己来解决一下^\_^。

**基本概念**

**=========**

这部分内容，大部分来自git的man手册，更为具体的信息可以参见"man gittutorial"的命令输出。

svn和git都是用于进行代码版本控制的工具。但是两者有许多不同，这里只对两个重要的地方：分布式管理、是否可更改历史，进行说明，随着以后的学习，我们会发现两者之间更多的不同之处。

使用过svn的用户可知svn有两个比较重要的特点：（1）在svn中一个项目已经提交的历史无法被修改。（2）svn服务器通过其中的某个统一的代码库集中管理某个项目代码，其他部分的代码都是这个服务器中库的工作拷贝，提交时都提交到这个统一的服务器代码库中，不同的svn代码库之间不能相互同步不同的代码修改（当然这并不是绝对的，我们可以通过特殊方法修改svn的历史，也可以让svn实现“分布”管理）。

而对于git，至少在这两点上与svn不同：（1）在git中，我们可以任意修改已经提交的内容的历史信息。（2）git中的服务器可以不只一个，服务器中同一项目的的代码库也可以不止是一个。因为我们可以将一台机器器上面的git代码库拷贝到另外一台机器上，然后提交的时候只往那个库中提交，而这些库之间还可以相互同步它们之间不同的修改部分。

下面是学习git时需要了解的一些重要内容，其中有些涉及到的内容，在后面的部分会详细提及。

**1、git中的"status"和"diff"以及"add"和"commit"**

git中，每次提交修改文件之前要add，然后再commit。git的add命令和许多版本控制系统不一样，不是增加新增的文件，而是以内容为单位，将最新修改的内容，添加到一个临时的索引中，等待commit。

实践发现，如果修改一个文件，然后diff查看修改，然后add，然后再修改，然后在commit，那么只把第一次修改提交了。commit的时候，建议遵守这样的格式：开始是简短的一行描述（不超过50字符），然后根着一个空行，最后是详细的描述。这样，就便于有些软件将开始的简短行作为email的题目，后面的详细描述作为email的信息内容，完成自动化转换。

另外，如果修改了一个文件然后没有"git add"，那么可以用"git diff"查看修改的内容，而"git status"无法看到修改的内容；当"git add"这个文件之后，用"git diff"就看不到这个内容了，用"git status"就可以看到。例如：

lv-k@quietheart:~/temp/temp/git\_t$ echo hello5 >>hello

lv-k@quietheart:~/temp/temp/git\_t$ git diff

diff --git a/hello b/hello

index 0863c71..bba0e39 100644

--- a/hello

+++ b/hello

@@ -3,3 +3,4 @@ hello1

hello2

hello3

hello4

+hello5 <===这里说明在"git add hello"之前，对hello文件进行了修改。

lv-k@quietheart:~/temp/temp/git\_t$ git status

# On branch master

# Changes not staged for commit: <===这里说明修改没有被添加（stage）,并且提示用"git add"。

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: hello <===这里说明修改了hello文件

#

no changes added to commit (use "git add" and/or "git commit -a") <===说明不会提交修改，可用"git add"或者"git commit -a"来提交修改。

lv-k@quietheart:~/temp/temp/git\_t$ git add hello

lv-k@quietheart:~/temp/temp/git\_t$ git diff <===添加之后，没有任何输出。

lv-k@quietheart:~/temp/temp/git\_t$ git status

# On branch master

# Changes to be committed: <===提示将会提交修改的内容，并且提示用"git reset"来取消stage。

# (use "git reset HEAD <file>..." to unstage)

#

# modified: hello

#

lv-k@quietheart:~/temp/temp/git\_t$ echo hello6 >>hello <===提交之前又修改了

lv-k@quietheart:~/temp/temp/git\_t$ git diff

diff --git a/hello b/hello

index bba0e39..34e8bc5 100644

--- a/hello

+++ b/hello

@@ -4,3 +4,4 @@ hello2

hello3

hello4

hello5

+hello6 <===显示最新没有add的修改

lv-k@quietheart:~/temp/temp/git\_t$ git status

# On branch master

# Changes to be committed: <===显示提交时会提交一部分修改

# (use "git reset HEAD <file>..." to unstage)

#

# modified: hello

#

# Changes not staged for commit: <===显示提交时也会有一部分修改不会提交上去。

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: hello

#

lv-k@quietheart:~/temp/temp/git\_t$ git add hello

lv-k@quietheart:~/temp/temp/git\_t$ git diff

lv-k@quietheart:~/temp/temp/git\_t$ git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: hello

#

lv-k@quietheart:~/temp/temp/git\_t$ git reset hello <===撤消"git add"的动作

Unstaged changes after reset:

M hello

lv-k@quietheart:~/temp/temp/git\_t$ git diff

diff --git a/hello b/hello

index 0863c71..34e8bc5 100644

--- a/hello

+++ b/hello

@@ -3,3 +3,5 @@ hello1

hello2

hello3

hello4

+hello5

+hello6

lv-k@quietheart:~/temp/temp/git\_t$ git status

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: hello

#

no changes added to commit (use "git add" and/or "git commit -a")

lv-k@quietheart:~/temp/temp/git\_t$ git add hello <===再次"git add"

lv-k@quietheart:~/temp/temp/git\_t$ git diff

lv-k@quietheart:~/temp/temp/git\_t$ git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: hello

#

对于以上内容中出现的命令，我们可以在后续的学习中看到。

**2、pull、push和fetch**

git中的pull子命令做两个工作：从远处库的分支fetch相应的修改，然后将修改合并到当前分支。如果发生冲突，那么也会进行fetch但是会拒绝对冲突的文件进行合并（这时需要解决冲突然后再次pull???）。

git-fetch用于从另一个reposoitory下载objects和refs，但是并不提交到本地库。其中<repository>表示远端的仓库路径。其中<refspec>的标准格式应该为<src>:<dst>，<src>表示源的分支，如果<dst>不为空，则表示本地的分支；如果为空，则使用当前分支。

而push和pull相反，是向远端发送本地提交的修改，当然需要让远端设置成可以接受本地提交的修改。具体参见后面的实践。

**3、常用常量和范围表示**

有一些常用常量可以表示特定的版本，它们是：

HEAD：表示最近一次的commit。

MERGE\_HEAD：如果是merge产生的commit，那么它表示除HEAD之外的另一个父母分支。

FETCH\_HEAD：使用git-fetch获得的object和ref的信息都存储在这里，这些信息是为日后git-merge准备的。

HEAD^：表示HEAD父母的信息

HEAD^^：表示HEAD父母的父母的信息

HEAD~4：表示HEAD上溯四代的信息

HEAD^1：表示HEAD的第一个父母的信息（和HEAD^相同）

HEAD^2：表示HEAD的第二个父母的信息

COMMIT\_EDITMSG：最后一次commit时的提交信息。

当我们表示两个特定版本之间的内容的时候，一般有如下表示：

使用".."表示两个提交版本之间的内容：

例如：“$ git log v2.5..v2.6”，表示查看v2.5到v2.6之间的log信息；再如：“HEAD..FETCH\_HEAD”，表示显示抓取的开始之后的信息不显示HEAD开始之后的信息(也就是显示的时候不显示自HEAD以来本地库的信息，而是只显示远端的信息)；还有使用"$ git log v2.5.."表示自v2.5之后提交的内容。

使用"..."表示包含更多的范围：

例如“HEAD...FETCH\_HEAD”，可以用来查看两者都修改了什么，是两者分别修改的而不是同时修改的。

**常用命令**

**=========**

这里给出常用的基本命令。

\*获取git的子命令的帮助信息的方式：

$man git-log

$git help log

这里，子命令用"git log"或"git log --graph"之类的log子命令进行举例。

\*为git添加自己的名字以及邮箱信息：

$ git config --global user.name "Your Name Comes Here"

$ git config --global user.email you@yourdomain.example.com

另外，通过当前目录的".git/config"，来查看类似"svn info"输出的url相关的信息等等。

\*\*将一个目录添加到git控制中

过程如下：

\*初始化目录：

$ tar xzf project.tar.gz

$ cd project

$ git init

\*将当前目录下所有文件的快照(snap)加入git

$ git add .

这样会告诉git将当前目录下所有的修改的文件，以及以前不在git下面的文件等的当前状态，添加到git的一个临时"stage area"中(但是还没有被保存下来)，git称它为索引(index)。

\*将索引(index)的内容保存到git库中：

$git commit

这样会提示你输入日志信息，之后就将当前的目录所有最新内容存放到git中管理了。另外需要注意的是，"commit"有一个"--amend"选项，可以给修改版本的信息，例如注释信息，具体参见man手册。

\*\*

\*\*修改文件并提交

\*将新增或者修改等变化的文件添加到git的索引(index)中：

$ git add file1 file2 file3

这里，添加之后，后面的commit命令将会把此次添加的内容都提交上去(由此可见git的"add"添加的不仅仅是文件，它是以内容为单位的，添加的是当前的修改)。

\*提交之前，查看修改状态：

$git diff --cached

或者

$git status

这里，diff如果没有--cached选项，信息更多（有则只显示没有"add"的内容），status显示的使概要的简洁信息。

\*提交修改：

$ git commit

这样，会提示你输入日志信息，然后更新git库的版本。

其实，每次提交修改文件之前都要用"add"将修改的内容添加进去，然后才能commit，如果合并两步，可以这样：

$git commit -a

这样会自动将修改的文件（不包含新增的文件）添加到索引，并且提交。

注意，提交时候的日志信息，最好用一个简单行开始，然后一个空行，因为许多工具将其日志第一行自动转换作为邮件标题，剩下内容作为邮件内容。

\*\*

\*\*查看历史信息

$git log

或者$git log -p

或者$git log --stat --summary

或者$git log --graph

这里，-p查看完整历史消息并且重定向到"less"之类的pager程序查看，--stat --summary也会重定向到pager但是信息比较简要，而只有"git log"则更简要并且不会将输出重定向到pager程序中。使用"git log --graph"可以用文本图形的方式显示版本之间的关系。

\*\*

\*\*分支管理

\*创建分支：

$ git branch experimental

\*查看分支：

$ git branch

这样，会列出所有分支，并且当前分支前面用'\*'标识出来。

\*切换分支：

$ git checkout experimental

这样，会切换到experimental分支，默认分支是'master'分支，它是被自动创建的。

\*合并分支：

$ git merge experimental

这里，如果正常则合并成功，如果冲突，那么冲突文件中会有冲突标记。合并之后可以提交。使用diff可以查看冲突。

\*删除分支：

$ git branch -d experimental

这里分支要确保被合并到上层才行，如果不考虑合并状态，那么使用-D代替-d。

\*\*

\*\*使用git合作开发

假设用户A有一个git库，地址为：/home/alice/project

B和A同为一台机器，其用户主目录为/home/bob。

\*B在A库的基础上面工作：

bob$ git clone /home/alice/project myrepo

(edit files)

bob$ git commit -a

(repeat as necessary)

这里，首先B从A拷贝一份库到本地，然后在本地进行修改，并且提交，这个时候，提交的内容都是提交到B自己的库中。

\*A合并B的修改：

alice$ cd /home/alice/project

alice$ git pull /home/bob/myrepo master

这里，当A想要合并B的内容的时候，可以通过pull实现；如果A已经在本地有过一些修改，那么可能会出现并处理一些合并的冲突问题。其实，"pull=fetch+merge"，就是先获取远端修改的内容，然后合并到本地。

\*只查看B的修改，但是不合并：

alice$ git fetch /home/bob/myrepo master

alice$ git log -p HEAD..FETCH\_HEAD

这里，fetch只是将修改信息抓取下来，并不进行合并，这样就能够在合并之前看到远端（抓取端）进行了哪些修改HEAD是最新的一次commit，FETCH\_HEAD使抓取端的信息。HEAD..FETCH\_HEAD表示显示抓取的开始之后的信息不显示HEAD开始之后的信息(也就是显示的时候不显示自HEAD以来本地库的信息，而是只显示远端的信息).而用HEAD...FETCH\_HEAD可以用来查看两者都修改了什么，是两者分别修改的而不是同时修改的。

\*A设置fetch时使用的B的远端位置的简写：

alice$ git remote add bob /home/bob/myrepo

如果A经常和B进行交互，那么可以使用remote命令，这样以后A便可以通过这个简写来获取B的修改而不用输入完整名称了。

\*A使用remote设置的B的简写来获取B的修改并合并：

alice$ git fetch bob

alice$ git merge bob/master

这里，也可以用如下命令来完成合并：

alice$ git pull . remotes/bob/master

\*B获取A最新修改的内容：

bob$ git pull

这里，B不需要给出A的库的位置，因为B是来自A的一个拷贝，A的位置信息在拷贝的时候被记录到库的配置中了，这个位置信息在B使用没有参数的pull的时候会被自动地应用。使用如下命令可以查看B的配置中的这个url信息：

bob$ git config --get remote.origin.url

/home/alice/project

\*查看B来自A的原始master拷贝：

bob$ git branch -r

origin/master

拷贝B的时候，git会自动将一份原始的A的master分支拷贝到"origin/master"中，用这个命令可以看到。

\*与A在不同主机上的B拷贝A的库到本地：

bob$ git clone alice.org:/home/alice/project myrepo

这里，可以使用特定的本地协议例如“rsync”或者“http”，可以查看git-pull的信息来获取更多帮助。另外，查看git-push(1)和gitcvs-migration(7)，可以知道git还可以设置成像cvs那样集中控制版本的模式。

由上面的过程我们可以知道，git版本控制工具中分布式管理的含义。与svn不同的是，在svn中，只能有一个svn服务器集中存放版本控制库，其他的客户端全部都是这个版本库的工作拷贝，提交的时候全部都提交到这个集中的svn库中或者库的某个分支中。git分步式管理的意思就是可以将一个由git创建的版本控制库拷贝到另外的地方，所有这个git库的拷贝都可以各自作为一个“版本控制中心”，管理自己工作拷贝以及本地分支的提交，同时这些库之间还可以相互合并其修改到另外的库的特定分支中。

\*\*

\*\*指定版本操作

这里将git中的版本号假设为特定的commit的id。

\*查看所有历史信息：

$git log

这样会显示所有的历史信息，包括"commit"的id，使用"git log --graph"会以文本图形的方式直观地显示出每个历史版本之间的合并关系等。其实，我们可以将历史看成一系列的"commit"。

\*查看特定id的历史信息：

$git show c82a22c39cbc32576f64f5c6b3f24b99ea8149c7

或

$git show c82a22c39c

这样，我们可以看到指定"commit"的id为"c82a22c39cbc32576f64f5c6b3f24b99ea8149c7"的历史信息。这里，只要能够保证前缀唯一，也可指定部分id的前缀作为相应历史的标识。

\*查看当前分支当前的历史状态：

$git show HEAD

这里，HEAD表示当前分支最新版本。关于查看时候指定的类似"HEAD"的常量大致如下：

HEAD：表示最近一次的commit。

MERGE\_HEAD：如果是merge产生的commit，那么它表示除HEAD之外的另一个父母分支。

FETCH\_HEAD：使用git-fetch获得的object和ref的信息都存储在这里，这些信息是为日后git-merge准备的。

HEAD^：表示HEAD父母的信息

HEAD^^：表示HEAD父母的父母的信息

HEAD~4：表示HEAD上溯四代的信息

HEAD^1：表示HEAD的第一个父母的信息（和HEAD^相同）

HEAD^2：表示HEAD的第二个父母的信息

COMMIT\_EDITMSG：最后一次commit时的提交信息。

\*自定义"commit"的id名称：

$ git tag v2.5 1b2e1d63ff

这样定义之后，就可以使用"v2.5"这样比较容易理解的方式来引用"1b2e1d63ff"这种看起来没有意义的字符串，如果想要和别人共享这个定义的名称，需要创建一个"tag"对象，并且给它"sign"，具体参见"git help tag"。

\*比较两个不同的版本：

$git diff HEAD bc32576f64f5c6b3f24b99ea8149c7

这里，比较"HEAD"和"bc32576f64f5c6b3f24b99ea8149c7"之间的区别，也可以使用自己定义的名称。

\*回退到指定版本，清除该版本之后所有的信息：

$ git reset --hard HEAD^

这样将会把当前工作分支的内容回退到"HEAD^"版本。注意，这个命令不仅会把当前的修改给移除，而且还会把自"HEAD^"以后的所有"commit"给删除（HEAD^版本本身保留为当前的HEAD），如果当前的分支是唯一的分支的话，那么运行这个命令之后，之前"HEAD^"以后的修改将会完全丢失（当然可以通过"pull"再将之前的HEAD拉取回来，但是自己没有提交的本地修改是无法找回来的了）；另外如果在一个公共的分支上面执行这个命令将会导致其他开发者pull的时候都进行一遍这个清除的工作，如果不加--hard选项，那么可能当前HEAD基础上有修改的情况导致这个命令不会成功。

\*回退到指定版本，并作为新版本提交上去（保留该版本之后所有的信息）：

$ git revert HEAD^

这样只会把内容回退为HEAD^之前的版本（即HEAD^^），再将这样的内容做为新的HEAD提交上去，原来的HEAD变成HEAD^（即在库中保留HEAD^之后的版本的提交，而不像"reset"那样完全清除）。这里，运行之后，会打开一个编辑窗口让你编辑提交的log信息，退出就直接提交了，或者"revert"命令也有不编辑提交log信息的选项。

\*使用checkout的“回退”：

$ git checkout v2.5

这样，会将当前v2.5版本重新检出，效果是使用v2.5做为当前的HEAD了，但是，原来的HEAD对应的commit id还存在于库中，可以使用那个id用"checkout"重新检回。另外，如果检出v2.5之后做了修改并且提交到库中，那么会以v2.5为基础重新生成一个版本，并且提交上去，这时候可以再用原来那个HEAD检查出，但是就“隐藏”了这个v2.5之后提交的新HEAD相关的一系列修改版本，这时候一般git会提醒你为此次提交创建一个新分支，并且会给出提示命令。实践发现，如果修改了没有提交，那么checkout会有问题，就是修改的内容并到了checkout的版本中，具体需要实践。

\*从指定版本搜索某个字符串：

$ git grep "hello" v2.5

这样将会从"v2.5"这个自定义的commit名字的版本中的所有文件搜索"hello"字符串。如果没有指定"v2.5"类似的commit的号码的话，将会从当前路径搜索这个"hello"字符串。

\*介于v2.5和v2.6之间的commit信息：

$ git log v2.5..v2.6

\*介于v2.5之后的commit信息：

$ git log v2.5..

\*两周以前的commit信息：

$ git log --since="2 weeks ago" # commits from the last 2 weeks

\*自Makefile修改之后的commit信息：

$ git log v2.5.. Makefile # commits since v2.5 which modify

\*两个没有前后关系的commit之间的信息：

$git log v2.5 s2.5

这里，v2.5和s2.5是两个没有直系关系的版本，这样会只显示关于s2.5的提交信息。

\*指定某个版本的文件并进行操作：

$ git diff v2.5:Makefile HEAD:Makefile.in

或

$ git show v2.5:Makefile

这里，我们运行操作文件的命令的时候，可以给文件名称添加一个commit的ID，以标识对那个版本的文件进行操作。

\*\*

**具体实践**

**=========**

下面用一个具体的实践展示使用git的简单过程。若有更多内容，再做更新。

**1、简单命令**

\*查看帮助：

$ git

具体如下：

lv-k@quietheart:~/temp/git\_test$ git

usage: git [--version] [--exec-path[=<path>]] [--html-path]

[-p|--paginate|--no-pager] [--no-replace-objects]

[--bare] [--git-dir=<path>] [--work-tree=<path>]

[-c name=value] [--help]

<command> [<args>]

The most commonly used git commands are:

add Add file contents to the index

bisect Find by binary search the change that introduced a bug

branch List, create, or delete branches

checkout Checkout a branch or paths to the working tree

clone Clone a repository into a new directory

commit Record changes to the repository

diff Show changes between commits, commit and working tree, etc

fetch Download objects and refs from another repository

grep Print lines matching a pattern

init Create an empty git repository or reinitialize an existing one

log Show commit logs

merge Join two or more development histories together

mv Move or rename a file, a directory, or a symlink

pull Fetch from and merge with another repository or a local branch

push Update remote refs along with associated objects

rebase Forward-port local commits to the updated upstream head

reset Reset current HEAD to the specified state

rm Remove files from the working tree and from the index

show Show various types of objects

status Show the working tree status

tag Create, list, delete or verify a tag object signed with GPG

See 'git help <command>' for more information on a specific command.

这里，通过这个命令我们可以看见git最常用的一些命令。

\*创建一个空白的版本库：

$ git init

具体如下：

lv-k@quietheart:~/temp/git\_test$ git init

Initialized empty Git repository in /home/lv-k/temp/git\_test/.git/

lv-k@quietheart:~/temp/git\_test$ ls -a

. .. .git

\*向版本库中导入数据：

$ git add example hello mydir

$ git commit -m 'initial commit contents'

实际上，可以使用"git add ."将当前目录加入。具体过程如下：

（1）添加文件

lv-k@quietheart:~/temp/git\_test$ echo "hello world" >hello

lv-k@quietheart:~/temp/git\_test$ echo "my example" >example

lv-k@quietheart:~/temp/git\_test$ mkdir mydir

lv-k@quietheart:~/temp/git\_test$ cd mydir/

lv-k@quietheart:~/temp/git\_test/mydir$ echo "dir contents" >dir\_content

lv-k@quietheart:~/temp/git\_test/mydir$ cd ..

lv-k@quietheart:~/temp/git\_test$ ls

example hello mydir

lv-k@quietheart:~/temp/git\_test$ tree

.

├── example

├── hello

└── mydir

└── dir\_content

1 directory, 3 files

（2）将文件添加到git库

lv-k@quietheart:~/temp/git\_test$ git add example hello mydir

lv-k@quietheart:~/temp/git\_test$ git status

# On branch master

#

# Initial commit

#

# Changes to be committed:

# (use "git rm --cached <file>..." to unstage)

#

# new file: example

# new file: hello

# new file: mydir/dir\_content

#

lv-k@quietheart:~/temp/git\_test$ git commit -m 'initial commit contents'

[master (root-commit) 098e14c] initial commit contents

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

3 files changed, 3 insertions(+), 0 deletions(-)

create mode 100644 example

create mode 100644 hello

create mode 100644 mydir/dir\_content

（3）查看日志

lv-k@quietheart:~/temp/git\_test$ git log

commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test$ tree .git/

.git/

├── branches

├── COMMIT\_EDITMSG

├── config

├── description

├── HEAD

├── hooks

│ ├── applypatch-msg.sample

│ ├── commit-msg.sample

│ ├── post-commit.sample

│ ├── post-receive.sample

│ ├── post-update.sample

│ ├── pre-applypatch.sample

│ ├── pre-commit.sample

│ ├── prepare-commit-msg.sample

│ ├── pre-rebase.sample

│ └── update.sample

├── index

├── info

│ └── exclude

├── logs

│ ├── HEAD

│ └── refs

│ └── heads

│ └── master

├── objects

│ ├── 09

│ │ └── 8e14c671f8cef3507a8b43fba7b386892ed5bc

│ ├── 31

│ │ └── 96f9d3ac40f9662f5398567d49de3e2c7dad7d

│ ├── 3b

│ │ └── 18e512dba79e4c8300dd08aeb37f8e728b8dad

│ ├── 6e

│ │ └── 93434ddde9f6fa8729896e2b52fb3ef9280b5a

│ ├── 70

│ │ └── 4d3223540b8924fb8c4bd565db016c4e601633

│ ├── cc

│ │ └── 1ef173226e0a0a130dad14e0108ec8b62eeb93

│ ├── info

│ └── pack

└── refs

├── heads

│ └── master

└── tags

18 directories, 25 files

\*修改并查看修改：

#git diff

具体如下：

lv-k@quietheart:~/temp/git\_test$ echo 'hello world' >>hello

lv-k@quietheart:~/temp/git\_test$ git diff

diff --git a/hello b/hello

index 3b18e51..10bda51 100644

--- a/hello

+++ b/hello

@@ -1 +1,2 @@

hello world

+hello world

\*提交修改：

$ git commit -a -m 'add modify'

这里，提交不能只用"commit"，应该先"add"再"commit"，其实"add"命令添加的不是文件而是新变化的内容，如果将两步合并为一个命令，可以使用像这里带有"-a"选项的"commit"，将修改的内容添加到索引缓存并且提交。

具体过程如下：

（1）只使用"commit"提交，无法成功：

lv-k@quietheart:~/temp/git\_test$ git commit -m 'modify'

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: hello

#

no changes added to commit (use "git add" and/or "git commit -a")

提交之后，查看提交的日志如下：

lv-k@quietheart:~/temp/git\_test$ git log

commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test$ git diff

diff --git a/hello b/hello

index 3b18e51..10bda51 100644

--- a/hello

+++ b/hello

@@ -1 +1,2 @@

hello world

+hello world

由此可见，只是"git commit"无法成功提交，因为git中的概念是首先将修改的文件"add"，然后再commit，所以如果不想用add命令，那么就用-a的commit。

（2）使用带有"-a"选项的"commit"，提交成功。

根据前面的提示，再次尝试如下：

lv-k@quietheart:~/temp/git\_test$ git commit -a -m 'add modify'

[master 07ee8b6] add modify

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test$ git diff

lv-k@quietheart:~/temp/git\_test$ git log

commit 07ee8b68e90a470aec9a5194079e54332093fc70

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:29:00 2012 +0800

add modify

commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

如此可见，提交成功。

(尝试使用"git add hello"之后，用"git diff"没有输出了，但是"git status"有修改的输出，使用"git reset"可以恢复成没有"git add hello"的状态)。

**2、关于分支**

\*查看当前分支：

$ git branch

具体如下：

lv-k@quietheart:~/temp/git\_test$ git branch

\* master

这里，可知当前处于主分支"master"，当前分支用'\*'和高亮标记出来。

\*创建分支：

$ git branch quietheart

具体如下：

lv-k@quietheart:~/temp/git\_test$ git branch quietheart

lv-k@quietheart:~/temp/git\_test$ git branch

\* master

quietheart

这里，创建了一个'quietheart'分支，但是创建之后并没有切换过去。

\*切换分支：

$ git checkout quietheart

具体如下：

lv-k@quietheart:~/temp/git\_test$ git checkout quietheart

Switched to branch 'quietheart'

lv-k@quietheart:~/temp/git\_test$ git branch

master

\* quietheart

这里，使用'checkout'命令切换到了指定的分支。

\*分别在分支中进行修改：

lv-k@quietheart:~/temp/git\_test$ git checkout master

lv-k@quietheart:~/temp/git\_test$ echo 'change in master' >>hello

lv-k@quietheart:~/temp/git\_test$ git commit -m 'change in master' -a

lv-k@quietheart:~/temp/git\_test$ git checkout quietheart

lv-k@quietheart:~/temp/git\_test$ echo 'changed in quietheart' >>example

lv-k@quietheart:~/temp/git\_test$ git commit -m 'change in quietheart' -a

具体如下：

lv-k@quietheart:~/temp/git\_test$ git checkout master

Switched to branch 'master'

lv-k@quietheart:~/temp/git\_test$ git branch

\* master

quietheart

lv-k@quietheart:~/temp/git\_test$ ls

example hello mydir

lv-k@quietheart:~/temp/git\_test$ echo 'change in master' >>hello

lv-k@quietheart:~/temp/git\_test$ git diff

diff --git a/hello b/hello

index 1b11f8b..60cacc0 100644

--- a/hello

+++ b/hello

@@ -1,3 +1,4 @@

hello world

hello world

hello world

+change in master

lv-k@quietheart:~/temp/git\_test$ git commit -m 'change in master' -i

fatal: No paths with --include/--only does not make sense.

lv-k@quietheart:~/temp/git\_test$ git commit -m 'change in master' -a

[master d8f32e0] change in master

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test$ git diff

lv-k@quietheart:~/temp/git\_test$ git log

commit d8f32e0e0b16e6dc499d5295a8a13ec3938664f8

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:54:10 2012 +0800

change in master

commit fa4abf6455aa01cf7d74810b96e2279983287fc8

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:41:51 2012 +0800

add modify2

commit 07ee8b68e90a470aec9a5194079e54332093fc70

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:29:00 2012 +0800

add modify

commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

这里，在执行本例子之前提交过一次，而那次提交没有在本文档记录所以所以多了一条记录'add modify2'。

lv-k@quietheart:~/temp/git\_test$ git checkout quietheart

Switched to branch 'quietheart'

lv-k@quietheart:~/temp/git\_test$ git log

commit fa4abf6455aa01cf7d74810b96e2279983287fc8

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:41:51 2012 +0800

add modify2

commit 07ee8b68e90a470aec9a5194079e54332093fc70

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:29:00 2012 +0800

add modify

commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test$ echo 'changed in quietheart' >>example

lv-k@quietheart:~/temp/git\_test$ git diff

diff --git a/example b/example

index 6e93434..0706323 100644

--- a/example

+++ b/example

@@ -1 +1,2 @@

my example

+changed in quietheart

lv-k@quietheart:~/temp/git\_test$ git commit -m 'change in quietheart' -a

[quietheart 8a95074] change in quietheart

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test$ git log

commit 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 18:00:10 2012 +0800

change in quietheart

commit fa4abf6455aa01cf7d74810b96e2279983287fc8

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:41:51 2012 +0800

add modify2

commit 07ee8b68e90a470aec9a5194079e54332093fc70

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:29:00 2012 +0800

add modify

commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

\*合并某分支到当前分支最新：

$ git checkout master

$ git merge quietheart

具体过程如下：

lv-k@quietheart:~/temp/git\_test$ git branch

master

\* quietheart

lv-k@quietheart:~/temp/git\_test$ git checkout master

Switched to branch 'master'

lv-k@quietheart:~/temp/git\_test$ git branch

\* master

quietheart

lv-k@quietheart:~/temp/git\_test$ git merge quietheart

Merge made by recursive.

example | 1 +

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test$ git diff

lv-k@quietheart:~/temp/git\_test$ git log

commit 83548b1a87668a5dbd606c6bcefdc984970ffb40

Merge: d8f32e0 8a95074

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 18:09:51 2012 +0800

Merge branch 'quietheart'

commit 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 18:00:10 2012 +0800

change in quietheart

commit d8f32e0e0b16e6dc499d5295a8a13ec3938664f8

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:54:10 2012 +0800

change in master

commit fa4abf6455aa01cf7d74810b96e2279983287fc8

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:41:51 2012 +0800

add modify2

commit 07ee8b68e90a470aec9a5194079e54332093fc70

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 17:29:00 2012 +0800

add modify

commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

这样，就将刚才在quietheart分支上面的从master继承过来之后的所有修改合并到了master主分支上面来了，同时提交分支quietheart时候的日志信息也合并进来了。

**3、使用clone、pull、push实践多人合作**

\*拷贝两份远端库到本地：

$ git clone /home/lv-k/temp/git\_test git\_test\_clone

具体如下：

lv-k@quietheart:~/temp$ git clone /home/lv-k/temp/git\_test git\_test\_clone

Cloning into git\_test\_clone...

done.

lv-k@quietheart:~/temp$ git clone /home/lv-k/temp/git\_test\_clone/ /home/lv-k/temp/git\_test\_clone1

Cloning into /home/lv-k/temp/git\_test\_clone1...

done.

这里，其实拷贝的是当前库所在的分支，通过"git branch"查看可知，得到的git\_test\_clone其实只有master分支，因为拷贝的时候，git\_test库所在的当前分支是master。

\*在一个拷贝上面修改并提交：

$ cd git\_test\_clone1/

$ echo "hello after clone" >>hello

$ git commit -a -m 'commin in clone'

具体如下：

lv-k@quietheart:~/temp$ cd git\_test\_clone1/

lv-k@quietheart:~/temp/git\_test\_clone1$ echo "hello after clone" >>hello

lv-k@quietheart:~/temp/git\_test\_clone1$ git commit -a -m 'commin in clone'

[master bf44b20] commin in clone

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 1 insertions(+), 0 deletions(-)

这里，修改之后，本地库便和远端的原始库发生了变化。

\*将远端拷贝修改的内容拉取过来：

$ cd git\_test\_clone/

$ git pull /home/lv-k/temp/git\_test\_clone1/

具体如下：

lv-k@quietheart:~/temp$ cd git\_test\_clone/

lv-k@quietheart:~/temp/git\_test\_clone$ git pull /home/lv-k/temp/git\_test\_clone1/

remote: Counting objects: 5, done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0)

Unpacking objects: 100% (3/3), done.

From /home/lv-k/temp/git\_test\_clone1

\* branch HEAD -> FETCH\_HEAD

Updating 83548b1..bf44b20

Fast-forward

hello | 1 +

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test\_clone$ cat hello

hello world

hello world

hello world

change in master

hello after clone

lv-k@quietheart:~/temp/git\_test\_clone$ git log |head -n 10

commit bf44b2085fe716b1fdb4c04b0dabbca60c451f39

Author: lv-k <lv-k@quietheart.(none)>

Date: Wed Jun 13 16:28:32 2012 +0800

commin in clone

commit 83548b1a87668a5dbd606c6bcefdc984970ffb40

Merge: d8f32e0 8a95074

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 18:09:51 2012 +0800

从这里我们可以看到，拉取过来的内容被自动地提交上去了，其实拉取内容的时候应该指明是哪个分支。实际如果没有参数则从clone时候的来源来pull，这里可以通过.git/config中查看来源。

\*将本地库拷贝的修改推送到远程库中：

$ cd git\_test\_clone1/

$ echo "test push" >>hello

$ cd ../git\_test\_clone1/

$ vim .git/config

###编辑内容，添加如下###

[receive]

denyCurrentBranch = ignore

###编辑内容，添加如上###

$ cd ../git\_test\_clone1

$ git push /home/lv-k/temp/git\_test\_clone

具体如下：

lv-k@quietheart:~/temp$ cd git\_test\_clone1/

lv-k@quietheart:~/temp/git\_test\_clone1$ echo "test push" >>hello

lv-k@quietheart:~/temp/git\_test\_clone1$ git push /home/lv-k/temp/git\_test\_clone

Everything up-to-date

lv-k@quietheart:~/temp/git\_test\_clone1$ cat /home/lv-k/temp/git\_test\_clone

cat: /home/lv-k/temp/git\_test\_clone: Is a directory

lv-k@quietheart:~/temp/git\_test\_clone1$ cat /home/lv-k/temp/git\_test\_clone/hello

hello world

hello world

hello world

change in master

hello after clone

lv-k@quietheart:~/temp/git\_test\_clone1$ git commit -a -m 'to be pushed'

[master c478225] to be pushed

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test\_clone1$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 2 commits.

#

nothing to commit (working directory clean)

lv-k@quietheart:~/temp/git\_test\_clone1$ git pull

From /home/lv-k/temp/git\_test\_clone

83548b1..bf44b20 master -> origin/master

Already up-to-date.

lv-k@quietheart:~/temp/git\_test\_clone1$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 1 commit.

#

nothing to commit (working directory clean)

lv-k@quietheart:~/temp/git\_test\_clone1$ git push

Counting objects: 5, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 339 bytes, done.

Total 3 (delta 1), reused 0 (delta 0)

Unpacking objects: 100% (3/3), done.

remote: error: refusing to update checked out branch: refs/heads/master

remote: error: By default, updating the current branch in a non-bare repository

remote: error: is denied, because it will make the index and work tree inconsistent

remote: error: with what you pushed, and will require 'git reset --hard' to match

remote: error: the work tree to HEAD.

remote: error:

remote: error: You can set 'receive.denyCurrentBranch' configuration variable to

remote: error: 'ignore' or 'warn' in the remote repository to allow pushing into

remote: error: its current branch; however, this is not recommended unless you

remote: error: arranged to update its work tree to match what you pushed in some

remote: error: other way.

remote: error:

remote: error: To squelch this message and still keep the default behaviour, set

remote: error: 'receive.denyCurrentBranch' configuration variable to 'refuse'.

To /home/lv-k/temp/git\_test\_clone/

! [remote rejected] master -> master (branch is currently checked out)

error: failed to push some refs to '/home/lv-k/temp/git\_test\_clone/'

lv-k@quietheart:~/temp$ cd ../git\_test\_clone/

lv-k@quietheart:~/temp/git\_test\_clone$ git config --bool core.bare true

lv-k@quietheart:~/temp$ cd ../git\_test\_clone1/

lv-k@quietheart:~/temp/git\_test\_clone1$ git push

Counting objects: 5, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 339 bytes, done.

Total 3 (delta 1), reused 0 (delta 0)

Unpacking objects: 100% (3/3), done.

To /home/lv-k/temp/git\_test\_clone/

bf44b20..c478225 master -> master

lv-k@quietheart:~/temp/git\_test\_clone1$ cat ../git\_test\_clone/hello

hello world

hello world

hello world

change in master

hello after clone

lv-k@quietheart:~/temp$ cd git\_test\_clone/

lv-k@quietheart:~/temp/git\_test\_clone$ git reset --hard

fatal: This operation must be run in a work tree

lv-k@quietheart:~/temp/git\_test\_clone$ git config --bool core.bare false

lv-k@quietheart:~/temp/git\_test\_clone$ git reset --hard

HEAD is now at c478225 to be pushed

lv-k@quietheart:~/temp/git\_test\_clone$ cat hello

hello world

hello world

hello world

change in master

hello after clone

test push

lv-k@quietheart:~/temp/git\_test\_clone$ vim .git/config

添加如下：

[receive]

denyCurrentBranch = ignore

lv-k@quietheart:~/temp$ cd git\_test\_clone1

lv-k@quietheart:~/temp/git\_test\_clone1$ echo "push2" >>hello

lv-k@quietheart:~/temp/git\_test\_clone1$ git commit -a -m 'push2'

[master 8b45c2f] push2

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test\_clone1$ git push

Counting objects: 5, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 332 bytes, done.

Total 3 (delta 1), reused 0 (delta 0)

Unpacking objects: 100% (3/3), done.

To /home/lv-k/temp/git\_test\_clone/

c478225..8b45c2f master -> master

lv-k@quietheart:~/temp$ cd git\_test\_clone

lv-k@quietheart:~/temp/git\_test\_clone$ ls

example hello mydir

lv-k@quietheart:~/temp/git\_test\_clone$ cat hello

hello world

hello world

hello world

change in master

hello after clone

test push

lv-k@quietheart:~/temp/git\_test\_clone$ git reset --hard

HEAD is now at 8b45c2f push2

lv-k@quietheart:~/temp/git\_test\_clone$ cat hello

hello world

hello world

hello world

change in master

hello after clone

test push

push2

这里我们可见，本地修改之后，直接使用"git push"无法提交，还需要远端用"git config --bool core.bare true"配置好之后，才能push，push之后，如果远端也在同样分支上面，并不能立即反应，需要将"core.bare"设置为"false"然后再"git reset --hard"。

根据网上查询的资料，push无法进行有如下解决方法：

方法1：

git config --bool core.bare true

方法2：

修改.git/config添加如下代码：

[receive]

denyCurrentBranch = ignore

我们使用第1个方法，比较麻烦，使用第二个方法，就不用再从服务端重新设置"core.bare"了。

另外，参考资料还提到，最好用git -bare init初始化库，而不是git init，本例子因为使用git init初始化，所以这样。

\*\*

\*版本回退的实践

这里分别对reset,revert,checkout三种方式进行一下实践具体过程如下：

lv-k@quietheart:~/temp$ git clone git\_test git\_test\_clone2

Cloning into git\_test\_clone2...

done.

lv-k@quietheart:~/temp$ cd git\_test\_clone2/

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 18:00:10 2012 +0800

|

| change in quietheart

|

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test\_clone2$ git reset --hard fa4abf6455aa01cf7d74810b96e2279983287fc8

HEAD is now at fa4abf6 add modify2

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test\_clone2$ git pull

Updating fa4abf6..8a95074

Fast-forward

example | 1 +

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 18:00:10 2012 +0800

|

| change in quietheart

|

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test\_clone2$ git revert fa4abf6455aa01cf7d74810b96e2279983287fc8

...自动打开编辑器，编辑log，退出编辑器（不用保存）...

[quietheart d707529] Revert "add modify2"

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 0 insertions(+), 1 deletions(-)

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit d707529ef6ed1b2bca493d22d367e56751ae3f3d

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Jun 14 16:24:51 2012 +0800

|

| Revert "add modify2"

|

| This reverts commit fa4abf6455aa01cf7d74810b96e2279983287fc8.

|

\* commit 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 18:00:10 2012 +0800

|

| change in quietheart

|

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test\_clone2$ git diff d707529ef6ed1b2bca493d22d367e56751ae3f3d:hello 07ee8b68e90a470aec9a5194079e54332093fc70:hello

lv-k@quietheart:~/temp/git\_test\_clone2$ git reset --hard fa4abf6455aa01cf7d74810b96e2279983287fc8

HEAD is now at fa4abf6 add modify2

lv-k@quietheart:~/temp/git\_test\_clone2$ git pull

Updating fa4abf6..8a95074

Fast-forward

example | 1 +

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 18:00:10 2012 +0800

|

| change in quietheart

|

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test\_clone2$ git checkout fa4abf6455aa01cf7d74810b96e2279983287fc8

Note: checking out 'fa4abf6455aa01cf7d74810b96e2279983287fc8'.

You are in 'detached HEAD' state. You can look around, make experimental

changes and commit them, and you can discard any commits you make in this

state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may

do so (now or later) by using -b with the checkout command again. Example:

git checkout -b new\_branch\_name

HEAD is now at fa4abf6... add modify2

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test\_clone2$ echo "add content" >>hello

lv-k@quietheart:~/temp/git\_test\_clone2$ git commit -a -m 'add'

[detached HEAD 9bd6059] add

Committer: lv-k <lv-k@quietheart.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 files changed, 1 insertions(+), 0 deletions(-)

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit 9bd605958283f6b33264af3ec6c9a0faa07feb55

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Jun 14 16:28:51 2012 +0800

|

| add

|

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

lv-k@quietheart:~/temp/git\_test\_clone2$ git checkout 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

Warning: you are leaving 1 commit behind, not connected to

any of your branches:

9bd6059 add

If you want to keep it by creating a new branch, this may be a good time

to do so with:

git branch new\_branch\_name 9bd605958283f6b33264af3ec6c9a0faa07feb55

HEAD is now at 8a95074... change in quietheart

lv-k@quietheart:~/temp/git\_test\_clone2$ git log --graph

\* commit 8a95074a30e32a4df3cdf50648fce2d6a0a73a97

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 18:00:10 2012 +0800

|

| change in quietheart

|

\* commit fa4abf6455aa01cf7d74810b96e2279983287fc8

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:41:51 2012 +0800

|

| add modify2

|

\* commit 07ee8b68e90a470aec9a5194079e54332093fc70

| Author: lv-k <lv-k@quietheart.(none)>

| Date: Thu Feb 9 17:29:00 2012 +0800

|

| add modify

|

\* commit 098e14c671f8cef3507a8b43fba7b386892ed5bc

Author: lv-k <lv-k@quietheart.(none)>

Date: Thu Feb 9 16:59:04 2012 +0800

initial commit contents

这里，根据前面的运行可知，reset可以将包括注释以及以前的提交全部清除，实现根本的回退；revert只是将指定版本的内容做为新的HEAD重新提交（类似打反向补丁），并且不会清除已有原来的提交；checkout是检出一个版本，已有的版本信息仍然保存在库中，但是在检出的旧版本上提交会导致新的分支。

???没有解决的问题：

使用git checkout "commit id"可以提取以前的版本，并且也可以回到以前之后的版本，但是如何回到以前的版本之后，知道以后有什么版本？

**参考资料:**

http://www.cnblogs.com/abeen/archive/2010/06/17/1759496.html

http://stackoverflow.com/questions/2816369/git-push-error-remote-rejected-master-master-branch-is-currently-checked-ou

man gittutorial

**作者：QuietHeart**

**Email：quiet\_heart000@126.com**

**日期：2012年6月14日**