学习笔记系文章，是我本人的实践实录，也是领悟书生教程网（http://www.656463.com）最重要的资源。如果你有兴趣，诚心邀请你一起打造最靠谱、最实用的教程！让上不起培训班、不想上培训班、有理想从事 IT 行业的人们提供一站式的学习和提高的平台。

huangyineng

笔记名称	git 学习笔记
作者	huangyineng
教程地址	http://www.656463.com
个人博客	http://www.naxsu.com
版本	0.1
创建日期	2012-8-15
最后修改日期	2012-8-15

概述

Git 是一个版本控制工具。与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持，使源代码的发布和交流极其方便。 Git 的速度很快，这对于诸如 Linux kernel 这样的大项目来说自然很重要。 Git 最为出色的是它的合并跟踪（merge tracing）能力。

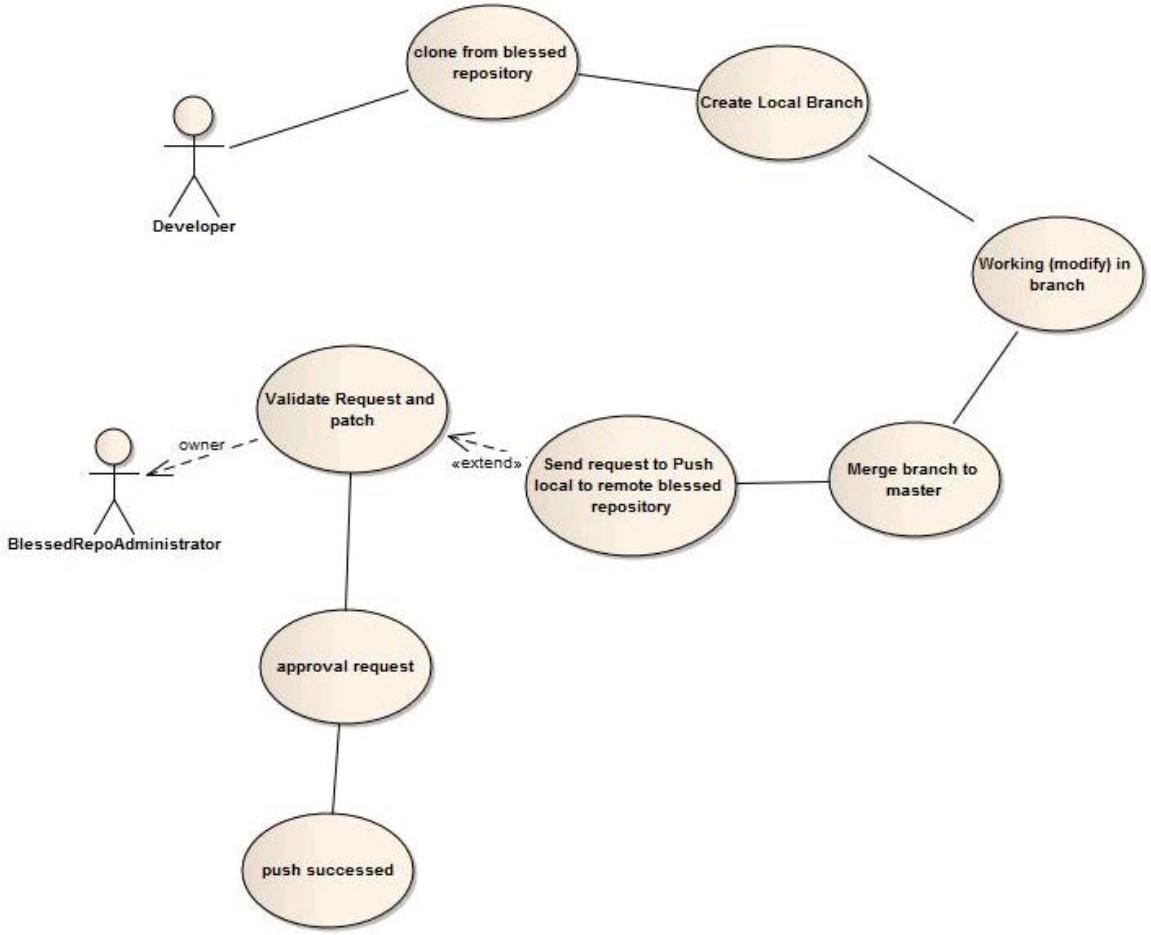
Git 的历史

Git 是 Linux 的创始人 Linus Tatvatds 开发的开源和免费的版本管理系统，也称源代码管理系统(Source Code Manage,SCM). 它的官方网站是 Http://git-scm.com

2005 年的时候 Linux 核心开发小组和当时的他们的版本管理系统提供商产生分歧。他们不能再使用原有的版本管理系统了。当时 Linus 环顾宇宙之内竟然没有一个能满足自己需求的版本管理系统可用。于是他毅然决定自己动手开发一个！这就是 Git。时至如今，Git 已经成为许多著名系统的版本管理系统。比如 Linux 核心，Eclipse，Android，Ruby on Rails，PostgreSQL，jQuery……http://en.wikipedia.org/wiki/Git_(software)

Linus 对这个版本控制开发时候的思考（特征）：
Speed 速度（用 C 写的）
Simple design 简单设计
Strong support for non-linear development(thousands of parallel branches) 上千个分支
Fully distributed 完全分布式
Able to handle large projects like the Linux kernel efficiently (speed and data size)

Git 开发人员使用流程



一般开发者，我就是 Google Source Code 使用者

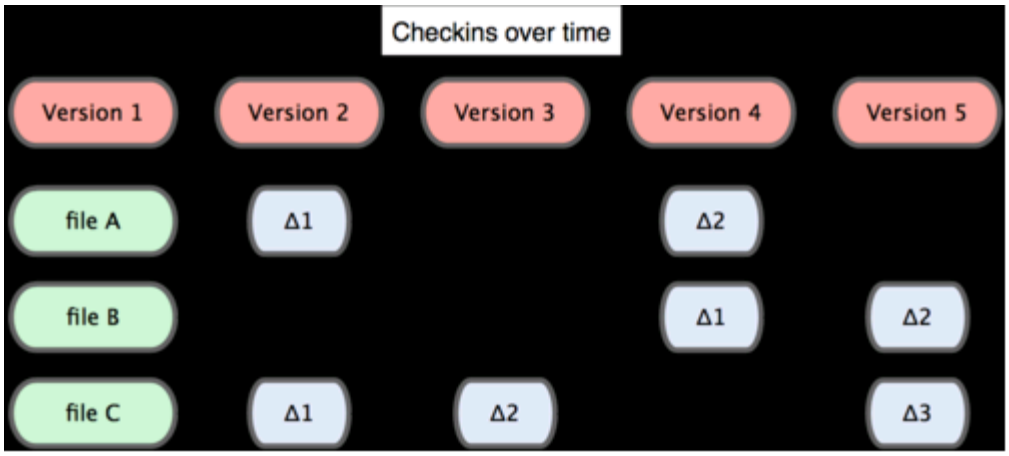
a) 从服务器上克隆数据库（包括代码和版本信息）到单机上。
b) 在自己的机器上创建分支，修改代码。
c) 在单机上自己创建的分支上提交代码。
d) 在单机上合并分支。
e) 新建一个分支，把服务器上最新版的代码 fetch 下来，然后跟自己的主分支合并。
f) 生成补丁（patch），把补丁发送给主开发者。
g) 看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突（他们之间可以合作解决的冲突），就会要求他们先解决冲突，然后再由其中一个人提交。如果主开发者可以自己解决，或者没有冲突，就通过。
h) 一般开发者之间解决冲突的方法，开发者之间可以使用 pull 命令解决冲突，解决完冲突之后再向主开发者提交补丁。

主开发者的角度（假设主开发者不用开发代码）：我就是 Google Source code 管理员

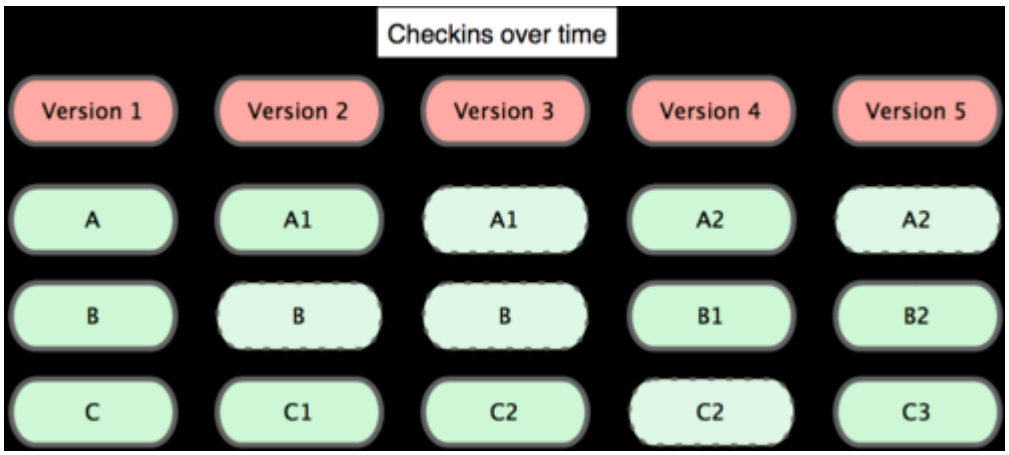
a) 查看邮件或者通过其它方式查看一般开发者的提交状态。
b) 打上补丁，解决冲突（可以自己解决，也可以要求开发者之间解决以后再重新提交，如果是开源项目，还要决定哪些补丁有用，哪些不用）。
c) 向公共服务器提交结果，然后通知所有开发者

Git 的记录方式

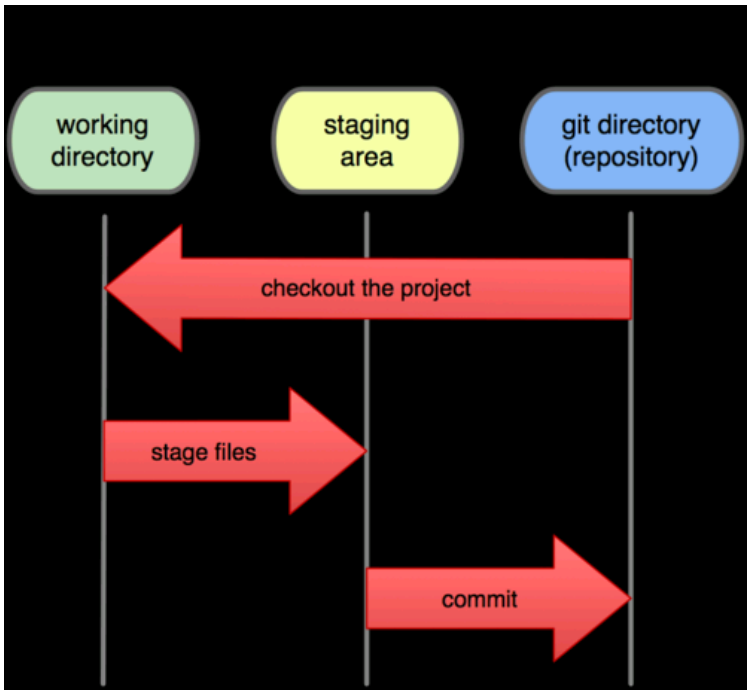
Git 是比快照而不是比不同
其他版本控制系统：



Git,更像一个小型的文件系统：

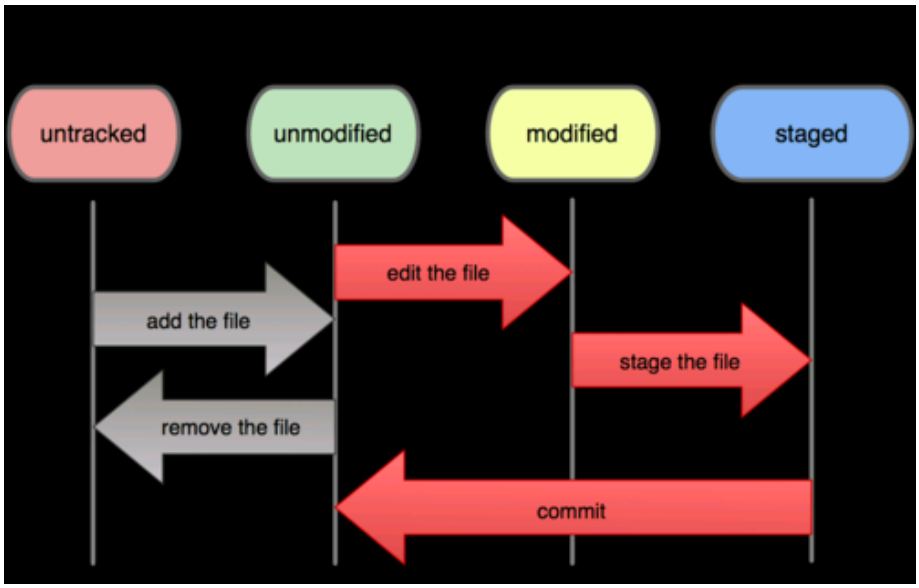


Git 文件的三个状态



Committed:文件安全地存储在你的本地
Modified:你修改了文件，但还未提交到你的仓库
Staged:已经标记了一个已修改文件到下一个版本的快照

Git 文件状态的生命周期



Untracked 相当于 SVN 中图标为紫色问号那个状态，也就是还没和版本控制没什么关系的
Unmodified 相当于 SVN 中图标为绝色打勾那个状态
modified 相当于 SVN 中图标为红色感叹号那个状态

下载安装

git 最新版是 Git-1.7.11-preview20120710.exe
下载地址：http://msysgit.github.com/或者 http://code.google.com/p/msysgit/downloads/list
安装就一路 next 就行了，在这里就不做说明。
打开命令行窗口 Git Bash，输入“git help git”即可打开 git 的帮助文档

```
$ git help git
Launching default browser to display HTML ...
```

D:/Program/Git/doc/git/html/git.html

Git 客户端 tortoisegit，现在的最新版本是 TortoiseGit-1.7.12.0-32bit.msi，下载地址：
http://code.google.com/p/tortoisegit/,这个安装也一路 NEXT，不作讲解

初步了解 Git

配置项

1. 了解系统环境变量
   /etc/gitconfig(D:/Program/Git/etc/gitconfig)
   .gitconfig
2. 设置身份
   $ git config --global user.name "yineng huang"
   $ git config --global user.email huangyineng@656463.com
3. 设置编辑器（可选）
   $ git config –global core.editor emacs
4. 设置你的比较工具（可选）
   $ git config –global merge.tool vimdiff
5. 检查你的配置（可选）
   $ git config –list

```
$ git config --list
core.symlinks=false
core.autocrlf=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
pack.packsizelimit=2g
help.format=html
http.sslcainfo=/bin/curl-ca-bundle.crt
sendemail.smtpserver=/bin/msmtp.exe
```

```
diff.astextplain.textconv=astextplain
rebase.autosquash=true
user.name=yineng huang
user.email=huangyineng@656463.com
```

6. 帮助

$ git help <verb>

$ git <verb> --help

如：$ git help git

注，怎么如何以上配置是用什么命令呢?请参阅帮助文档：
file:///D:/Program/Git/doc/git/html/git-config.html

几个区域

blessed(remote) repository:远程仓库
local repository:本地仓库
stage area:缓冲区-->git 目录下的 index 文件
work area:工作区

我们现在用实例来描述一下这几个区域

```
huangyineng@HUANGYINENG-PC ~
$ mkdir gittest1

huangyineng@HUANGYINENG-PC ~
$ cd gittest1/

huangyineng@HUANGYINENG-PC ~/gittest1
$ git init
Initialized empty Git repository in c:/Users/huangyineng/gittest1/.git/

huangyineng@HUANGYINENG-PC ~/gittest1 (master)
$ echo "hello" >> hello

huangyineng@HUANGYINENG-PC ~/gittest1 (master)
$ git add .
warning: LF will be replaced by CRLF in hello.
The file will have its original line endings in your working directory.

huangyineng@HUANGYINENG-PC ~/gittest1 (master)
$ git commit hello -m "init hello"
warning: LF will be replaced by CRLF in hello.
The file will have its original line endings in your working directory.
```

```
[master (root-commit) 4d850bf] init hello
warning: LF will be replaced by CRLF in hello.
The file will have its original line endings in your working directory.
 1 file changed, 1 insertion(+)
 create mode 100644 hello

huangyineng@HUANGYINENG-PC ~/gittest1 (master)
$
```

以上命令的操作过程是：创建 gittest1 目录，进入该目录，把该目录初始化为仓库。这个仓库的具体位置是：C:\Users\huangyineng\gittest1\.git。创建创建一个文件（hello），把当然目录下的所有文件添加到缓冲区中，然后提交把 hello 文件提交到本地仓库中。

工作区就是 gittest1 目录（C:\Users\huangyineng\gittest1）
缓冲区存放在.git 的 index 文件中（C:\Users\huangyineng\gittest1\.git\index）
本地仓库会存放在.git 的 objects 目录下（C:\Users\huangyineng\gittest1\.git\objects）

基本操作

初始化和建立项目

有两种方式，一种是 init,另外一种是 clone
init 在上面的例子中已经用过了，也就是进入项目所在的目录，用$ git init 即可。
Clone 一般是从远程服务器克隆一个已有的版本仓库到本机，命令如下：

```
$ git clone git://github.com/git/hello-world.git
Cloning into 'hello-world'...
remote: Counting objects: 158, done.
remote: Compressing objects: 100% (79/79), done.
remote: Total 158 (delta 54), reused 154 (delta 54)
Receiving objects: 100% (158/158), 15.63 KiB, done.
Resolving deltas: 100% (54/54), done.
error: unable to create file brainf*ck.bf (Invalid argument)
```

查看远程服务器：

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git remote
origin

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git remote -v
origin    git://github.com/git/hello-world.git (fetch)
origin    git://github.com/git/hello-world.git (push)
```

添加与提交

所用到的命令是 add、commit 和 status.

1. 创建一个名为 helloworld.naxsu 的文件.
2. 用 git status 查看当前目录文件的提交状态
   brainf*ck.bf 是刚才克隆的时候，没法克隆下来，这里显示是删除了
   helloworld.naxsu 是刚创建的文件，提示用"git add"添加的缓冲区中或者用"git commit -a"添加并提交
3. $ git add helloworld.naxsu 进行添加到缓冲区
   添加当前目录下的所有文件
   $ git add .
   添加以.c 为后缀的文件
   $ git add *.c
   添加指定文件
   $ git add index.jsp
4. $ git commit helloworld.naxsu -m "init helloworld.naxsu"提交到本地仓库

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ echo "hello world" >> helloworld.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ ls *.naxsu
helloworld.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git status
# On branch master
# Changes not staged for commit:
#     (use "git add/rm <file>..." to update what will be committed)
#     (use "git checkout -- <file>..." to discard changes in working directory)
#
#           deleted:       brainf*ck.bf
#
# Untracked files:
#     (use "git add <file>..." to include in what will be committed)
#
#           helloworld.naxsu
no changes added to commit (use "git add" and/or "git commit -a")

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git add helloworld.naxsu
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.
```

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git status
# On branch master
# Changes to be committed:
#     (use "git reset HEAD <file>..." to unstage)
#
#           new file:     helloworld.naxsu
#
# Changes not staged for commit:
#     (use "git add/rm <file>..." to update what will be committed)
#     (use "git checkout -- <file>..." to discard changes in working directory)
#
#           deleted:      brainf*ck.bf
#

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git commit helloworld.naxsu -m "init helloworld.naxsu"
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.
[master 8c17395] init helloworld.naxsu
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.
 1 file changed, 1 insertion(+)
 create mode 100644 helloworld.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes not staged for commit:
#     (use "git add/rm <file>..." to update what will be committed)
#     (use "git checkout -- <file>..." to discard changes in working directory)
#
#           deleted:      brainf*ck.bf
#
no changes added to commit (use "git add" and/or "git commit -a")

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$
```

忽略某些文件

Java 文件编译成.class 文件，他是自动生成的，我们没必要用版本控制它，所以提交的时候可以用忽略。
创建文件 class1.class、java1.java，创建.gitignore，并把 class1.class 添加到.gitignore 中,同时用 vim 编辑.gitignore，把他自己也添加到里面，用$ cat .gitignore 命令可以查看.gitignore 的内容。接下来用 add,commit,你就会发现 class1.class 是不会提交到仓库中的。

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ echo "class1" > class1.class

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ echo "java1" > java1.java

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ echo "class1.class" >.gitignore

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ vim .gitignore

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ cat .gitignore
class1.class
.gitignore

$ git status
$ git add .
$ git status
$ git commit -a -m "ignore test"
……
```

比较文件的不同

$ git diff(默认是$ git diff --staged)
$ git diff --staged:比较 workspace VS staged
$ git diff --cached:比较 staged VS local repo
演示思路：修改 helloworld.naxsu，用 git diff 查看不同，把他 add 之后再查看他们的不同，然后 commit 后，又一次查看他们的不同。

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ vim helloworld.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
```

```
$ git diff
diff --git a/helloworld.naxsu b/helloworld.naxsu
index 3b18e51..6d05489 100644
--- a/helloworld.naxsu
+++ b/helloworld.naxsu
@@ -1 +1,2 @@
  hello world
+add something
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git add helloworld.naxsu
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git diff --cached
diff --git a/helloworld.naxsu b/helloworld.naxsu
index 3b18e51..6d05489 100644
--- a/helloworld.naxsu
+++ b/helloworld.naxsu
@@ -1 +1,2 @@
  hello world
+add something
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git commit helloworld.naxsu -m "modified helloworld.naxsu"
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.
[master warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.
6e7814d] modified helloworld.naxsu
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working directory.
  1 file changed, 1 insertion(+)

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git diff

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git diff --cached
```

文件的移动和删除

移动=删除+添加

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ ls *naxsu
helloworld.naxsu    test.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git rm test.naxsu
rm 'test.naxsu'

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 6 commits.
#
# Changes to be committed:
#     (use "git reset HEAD <file>..." to unstage)
#
#           deleted:       test.naxsu
#

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git reset head test.naxsu
Unstaged changes after reset:
D           test.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 6 commits.
#
# Changes not staged for commit:
#     (use "git add/rm <file>..." to update what will be committed)
#     (use "git checkout -- <file>..." to discard changes in working directory)
#
#           deleted:       test.naxsu
#
no changes added to commit (use "git add" and/or "git commit -a")
```

如果没提交还可以 checkout 进行恢复

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
```

```
$ git checkout -- test.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ ls *.naxsu
helloworld.naxsu    test.naxsu
```

如果 commit 了之后，就不能 checkout 了

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git rm test.naxsu
rm 'test.naxsu'

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git commit -a -m "delete test.naxsu"
[master 46d28af] delete test.naxsu
  1 file changed, 1 deletion(-)
  delete mode 100644 test.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git checkout -- test.naxsu
error: pathspec 'test.naxsu' did not match any file(s) known to git.
```

移动用 mv 命令，具体参考$ git mv --help

查看操作记录

git log 显示所有的提交（commit）记录

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git log
commit 46d28afa27a90678c7391fc0bc5549db345f3c7d
Author: yineng huang <huangyineng@656463.com>
Date:      Fri Aug 17 23:28:34 2012 +0800

       delete test.naxsu
……
```

git whatchanged

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git whatchanged
commit 46d28afa27a90678c7391fc0bc5549db345f3c7d
Author: yineng huang <huangyineng@656463.com>
Date:      Fri Aug 17 23:28:34 2012 +0800

       delete test.naxsu

:100644 000000 77608b6... 0000000... D    test.naxsu
……
```

git-whatchanged 显示的信息比 git-log 更详细一些，可以显示具体的文件名。

共享及更新项目

1. 了解 remote
   remote 是显示远程仓库信息的命令
   显示远程仓库

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git remote
origin
```

   显示远程仓库地址

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git remote -v
origin    git://github.com/git/hello-world.git (fetch)
origin    git://github.com/git/hello-world.git (push)
```

2. fetch 从远程拉下来，不与本地的合并，并建立一个分支

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git fetch origin
```

3. pull 从远程拉下来，和本地的合并

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git pull origin
Already up-to-date.
```

4. push 提交到远程服务器

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git push origin master
fatal: remote error:
   You can't push to git://github.com/git/hello-world.git
   Use git@github.com:git/hello-world.git
```

   这个是没有权限提交的。

分支管理与合并

显示所有分支

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git branch
* master
```

增加分支

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git branch b1
```

切换分支

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git checkout b1
Switched to branch 'b1'

huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$
```

切换到不同的分支对 helloworld.naxsu 这个文件进行修改并提交，并查看他们的内容

原先 helloworld.naxsu 的内容是

```
huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$ cat helloworld.naxsu
hello world
add something
```

在 b1 分支修改并提交，然后查看文件内容

```
huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$ vim helloworld.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$ git commit -a -m "b1 update helloworld.naxsu"
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working
[b1 warning: LF will be replaced by CRLF in helloworld.naxsu
The file will have its original line endings in your working
b9c5de1] b1 update helloworld.naxsu
warning: LF will be replaced by CRLF in helloworld.naxsu.
The file will have its original line endings in your working
 1 file changed, 1 insertion(+)

huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$ git status
# On branch b1
nothing to commit (working directory clean)

huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$ cat helloworld.naxsu
hello world
add something
b1 branch add something
```

切换到 master 分支，进行和上面一样的操作

```
huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$ git checkout master
Switched to branch 'master'
```

```
Your branch is ahead of 'origin/master' by 7 commits.

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ vim helloworld.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git commit -a -m "master update helloworld.naxsu"
[master d9f15c9] master update helloworld.naxsu
  1 file changed, 1 insertion(+)

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ cat helloworld.naxsu
hello world
add something
master branch add something
```

从中可以看出两个分支是没有影响的。

列出各分支之间的信息

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git show-branch
! [b1] b1 update helloworld.naxsu
  * [master] master update helloworld.naxsu
--
  * [master] master update helloworld.naxsu
+    [b1] b1 update helloworld.naxsu
+* [master^] delete test.naxsu
```

比较 master 和 b1 这两个分支文件的不同

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git diff master b1
diff --git a/helloworld.naxsu b/helloworld.naxsu
index 9e73a56..7929722 100644
--- a/helloworld.naxsu
+++ b/helloworld.naxsu
@@ -1,3 +1,3 @@
  hello world
  add something
-master branch add something
+b1 branch add something
```

分支合并 merge

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git merge "merge" HEAD b1
Auto-merging helloworld.naxsu
```

```
CONFLICT (content): Merge conflict in helloworld.naxsu
Automatic merge failed; fix conflicts and then commit the result.
```

显示自动合并失败，我们看一下状态，可以看到 helloworld.naxsu 被两个分支修改过了

```
huangyineng@HUANGYINENG-PC ~/hello-world (master|MERGING)
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 8 commits.
#
# Unmerged paths:
#   (use "git add/rm <file>..." as appropriate to mark resolution)
#
#       both modified:      helloworld.naxsu
#
no changes added to commit (use "git add" and/or "git commit -a")
```

我们来看一下 helloworld.naxsu 的内容

```
huangyineng@HUANGYINENG-PC ~/hello-world (master|MERGING)
$ cat helloworld.naxsu
hello world
add something
<<<<<<< HEAD
master branch add something
=======
b1 branch add something
>>>>>>> b1
```

把们用 vim 编辑工具人为的把冲突去掉，然后保存、提交，切换到其他分支，也可以看到没有冲突了

```
huangyineng@HUANGYINENG-PC ~/hello-world (master|MERGING)
$ vim helloworld.naxsu

huangyineng@HUANGYINENG-PC ~/hello-world (master|MERGING)
$ git commit -a -m "merging"
[master 34908f4] merging

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ cat helloworld.naxsu
hello world
add something
master branch add something
b1 branch add something

huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git checkout b1
Switched to branch 'b1'
```

```
huangyineng@HUANGYINENG-PC ~/hello-world (b1)
$ cat helloworld.naxsu
hello world
add something
b1 branch add something
```

另一种合并的做法，自行去查帮助文档：

$ git checkout master

$ git pull . b1

标签

git 跟其它版本控制系统一样，可以打标签(tag)，作用是标记一个点为一个版本号，如 0.1.3, v0.1.7, ver_0.1.3.在程序开发到一个阶段后，我们需要打个标签，发布一个版本，标记的作用显而易见。

下面介绍一下打标签，分享标签，移除标签的操作命令。

打标签

```
huangyineng@HUANGYINENG-PC ~/hello-world (master)
$ git tag -a 0.1.3 -m "Release version 0.1.3"
```

详解：git tag 是命令,-a 0.1.3 是增加名为 0.1.3 的标签,-m 后面跟着的是标签的注释

分享提交标签到远程服务器上

git push origin master

git push origin –tags

–tags 参数表示提交所有 tag 至服务器端，普通的 git push origin master 操作不会推送标签到服务器端。

删除标签的命令

git tag -d 0.1.3

删除远端服务器的标签

git push origin :refs/tags/0.1.3

查看本地标签

huangyineng@HUANGYINENG-PC ~/hello-world (master)

$ git tag -l

0.1.3

Android 源码下载

待续

建立我们自己的 Git 开源项目

待续