Git原理与实践

youngsterxyf

众成技术聚乐部

1. 简介
2. 实践(上)
3. 原理
4. 实践(下)
5. 结束

1 简介

2 实践(上)

3 原理

4 实践(下)

5 结束

背景

Linux内核版本管理

- 2002 – 2005: BitKeeper
- 2005 – : → Git

Git的设计目标

- 速度
- 简单的设计
- 对非线性开发模式的强力支持（允许上千个并行开发的分支）
- 完全分布式
- 有能力高效管理类似Linux内核一样的超大规模项目（速度和数据量）

与SVN的区别

- 分布式
- 直接记录快照，而非差异比较
- 分支模型
- 权限管理
- ...

1 简介

2 实践(上)

3 原理

4 实践(下)

5 结束

基本用法

新建代码库

1. git init
2. git add *
3. git commit -m 'bbb'
4. git remote add origin http://xxx.yyy
5. git push origin master

变更提交

1. git add xxx / git rm yyy / git mv xxx yyy
2. git commit -m 'aaa'
3. git push origin master

更新代码

1. git fetch origin master
2. git merge origin/master

或合并为

- git pull origin master

基本用法(续)

克隆代码库

- git clone http://xxx.yyy

分支操作

1. git branch test /* 创建一个名为test的分支*/
2. git checkout test /* 切换到test分支*/
3. git push origin test /* 将test分支推送到远程服务器，即创建远程test分支*/
4. git checkout master /* 切换回master主分支*/
5. git branch -d|-D test /* 删除test分支*/
6. git push origin :test /* 删除远程test分支*/
7. git branch [-v] [-r] /* 查看分支列表，-v：详细信息，-r：远程分支列表*/
8. git checkout -b local-branch origin/remote-branch /* 迁出远程分子和remote-branch到一个本地分支local-branch */

步骤1、2可合并为：

- git checkout -b test /* 创建一个名为test的分支，并切换到该分支*/

步骤8实际可分为两个步骤：

1. git branch local-branch origin/remote-branch
2. git checkout local-branch

基本用法(续)

标签操作

- git tag v0.0.1 6030bd2cc /* 在commit 6030bd2cc上打个名为v0.0.1的轻量级标签*/
- git tag -a v0.0.1 -m 'xxx' /* 在当前HEAD指向的commit上打个名为v0.0.1的带附注的标签*/
- git show v0.0.1 /* 查看标签内容*/
- git push origin v0.0.1 /* 将标签推送到远程服务器*/
- git tag -d v0.0.1 /* 从本地仓库删除标签v0.0.1 */
- git push origin :v0.0.1 /* 从远程服务器删除标签v0.0.1 */

其它常用命令

- git status /* 检查当前文件状态*/
- git remote -v /* 查看当前代码库所关联的远程服务器地址*/
- git branch -v /* 查看当前代码库有哪些分支，当前分支是哪个*/
- git config ... /* 配置Git */
- git help / git help subcommand /* 查看文档，很明显，这是最重要的命令*/
- git checkout * / git checkout targetfile /* 撤销未提交的变更*/
- git log /* 日志的输出格式可以使用一些参数进行灵活的定制*/
- git diff /* 不带其他选项，则显示相比暂存区，工作区的变更，另外使用不同参数，可查看不同分支、不同提交之间的差异*/

1 简介

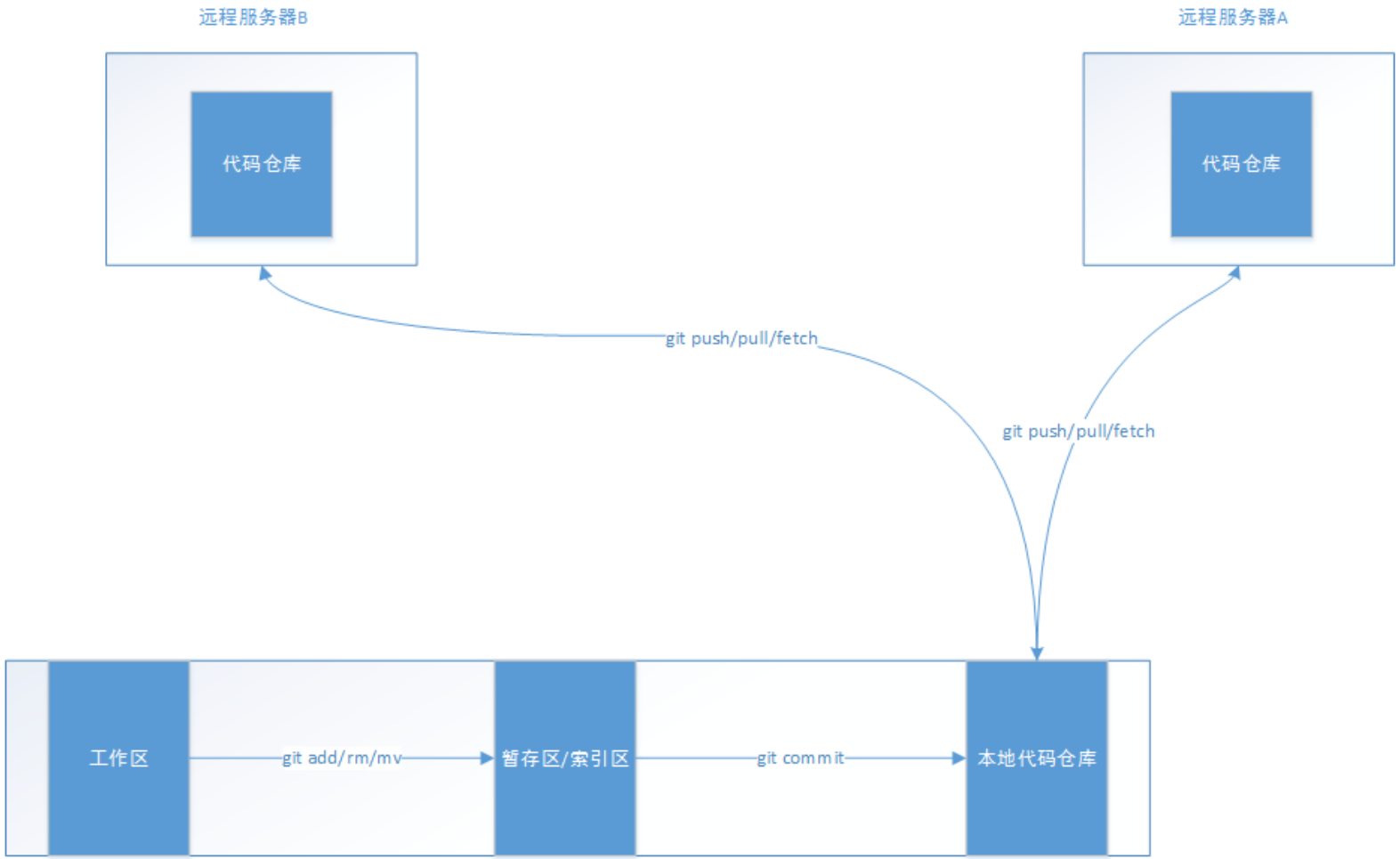
2 实践(上)

3 原理

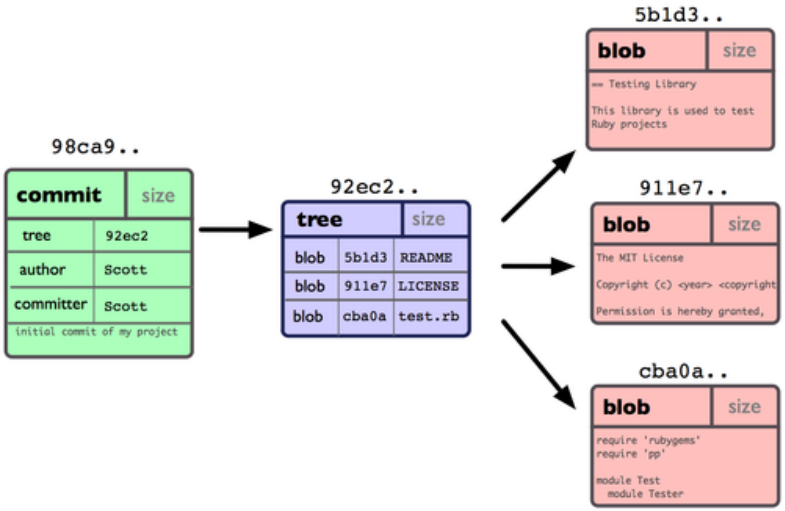
4 实践(下)

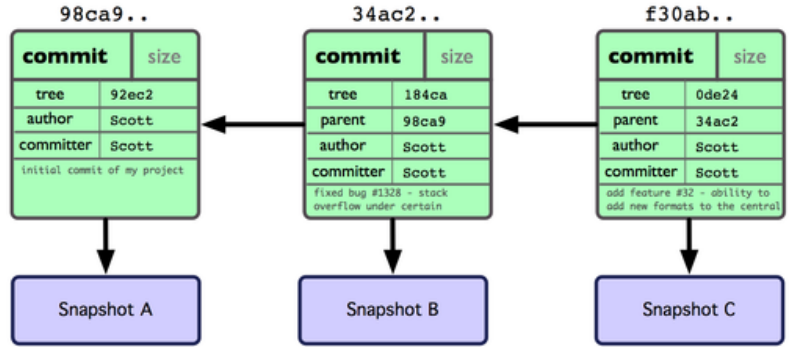
5 结束

基本操作流程


远程服务器B
代码仓库
远程服务器A
代码仓库
git push/pull/fetch
git push/pull/fetch
工作区
git add/rm/mv
暂存区/索引区
git commit
本地代码仓库

存储模型


98ca9..
commit
size
tree
92ec2
author
Scott
committer
Scott
initial commit of my project
92ec2..
tree
size
blob
5b1d3
README
blob
911e7
LICENSE
blob
cba0a
test.rb
5b1d3..
blob
size
== Testing Library
This library is used to test Ruby projects
911e7..
blob
size
The MIT License
Copyright (c) <year> <copyright
Permission is hereby granted,
cba0a..
blob
size
require 'rubygems'
require 'pp'
module Test
module Tester


98ca9..
commit
size
tree
92ec2
author
Scott
committer
Scott
initial commit of my project
Snapshot A
34ac2..
commit
size
tree
184ca
parent
98ca9
author
Scott
committer
Scott
fixed bug #1328 - stack overflow under certain
Snapshot B
f30ab..
commit
size
tree
0de24
parent
34ac2
author
Scott
committer
Scott
add feature #32 - ability to add new formats to the central
Snapshot C

存储模型(续)

Git以一种类似UNIX文件系统但更简单的方式来存储内容。所有内容以tree或blob对象存储，其中tree对象对应于UNIX中的目录，blob对象则大致对应于inodes或文件内容。一个单独的tree对象包含一条或多条tree记录，每一条记录含有一个指向blob或子tree对象的SHA-1指针，并附有该对象的权限模式(mode)、类型和文件名信息。

commit 对象有格式很简单：指明了该时间点项目快照的顶层树对象、作者/提交者信息(从Git设置的user.name和user.email中获得)以及当前时间戳、一个空行，以及提交注释信息。

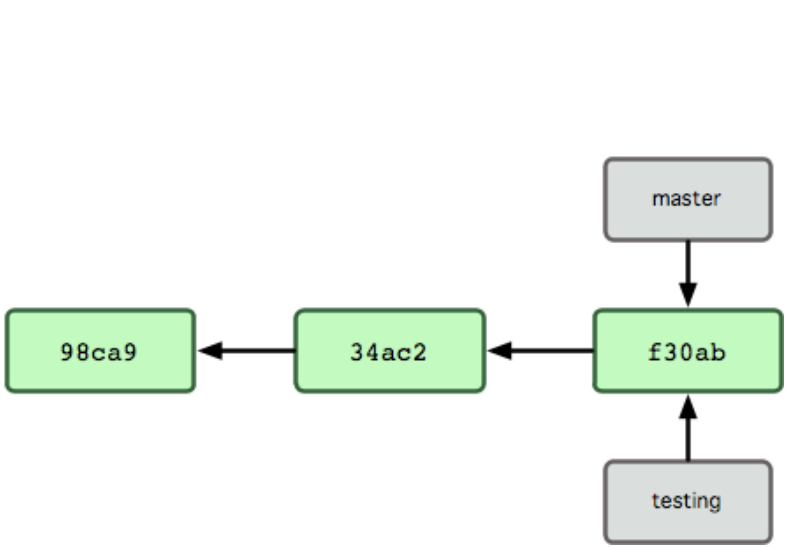
对象存储

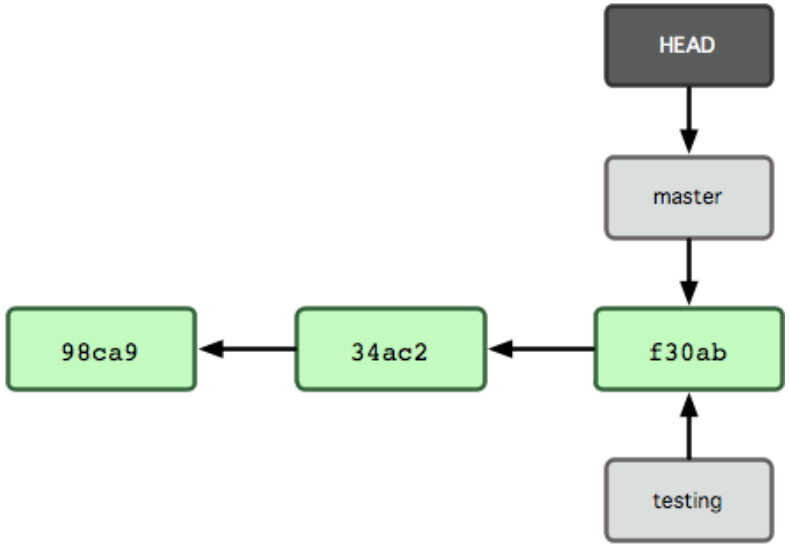
1. Git以对象类型为起始内容构造一个文件头，然后添加一个空格，接着是数据内容的长度，最后是一个空字节(null byte)。
2. 然后将构造出来的文件头与原始数据内容拼接起来，计算拼接后的新内容的SHA-1校验和
3. 使用zlib对拼接后的内容进行压缩
4. 将压缩后的内容写入磁盘，文件路径为：以SHA-1值的头两个字符作为子目录名称，剩余38个字符作为文件名保存至该子目录中

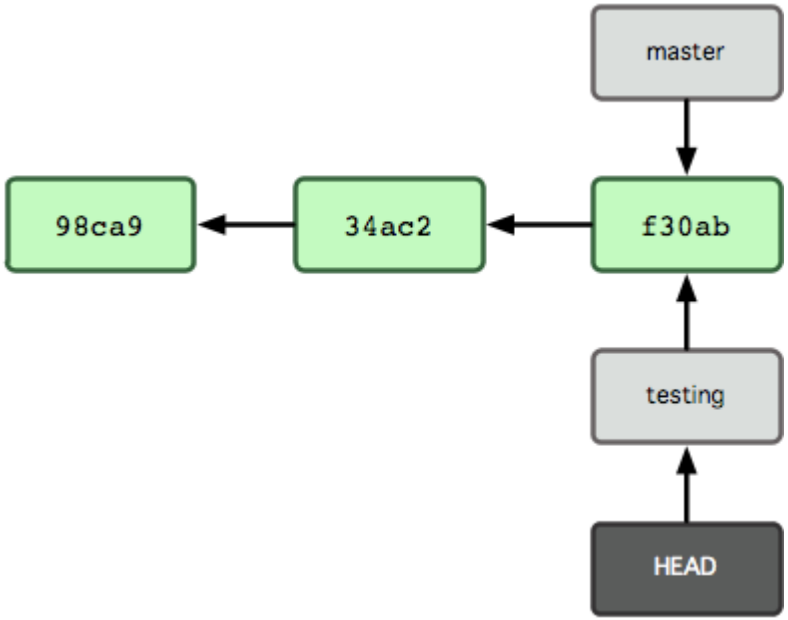
优化

Git往磁盘保存对象时默认使用的格式叫松散对象(loose object)格式。Git会不定期地将这些对象打包至一个叫packfile的二进制文件以节省空间并提高效率。当仓库中有太多的松散对象，或是手工调用git gc命令，或推送至远程服务器时，Git都会这样做。

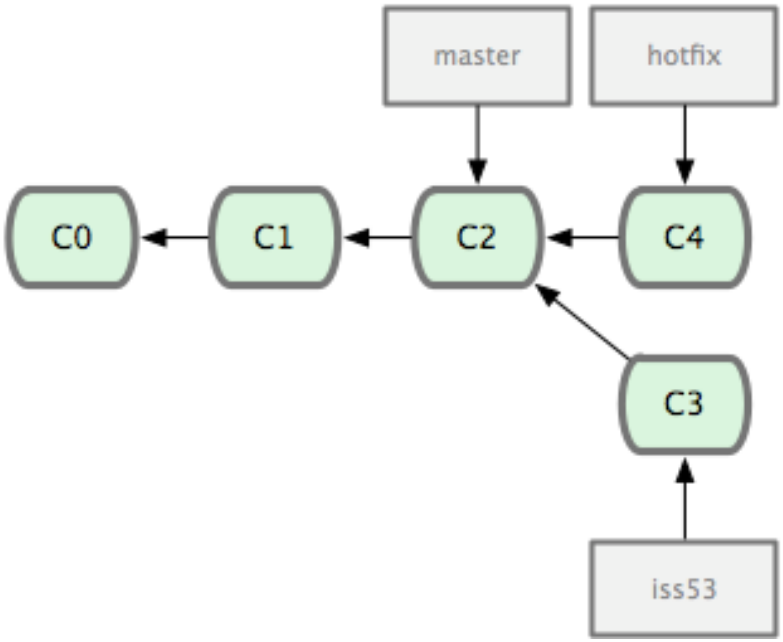
分支模型- 分支引用

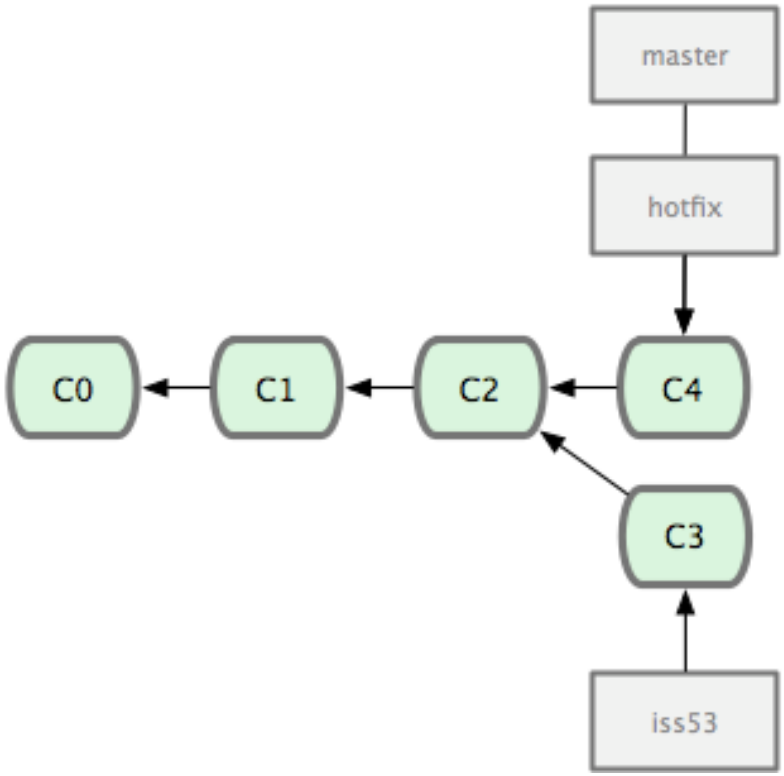

master
98ca9
34ac2
f30ab
testing


HEAD
master
98ca9
34ac2
f30ab
testing

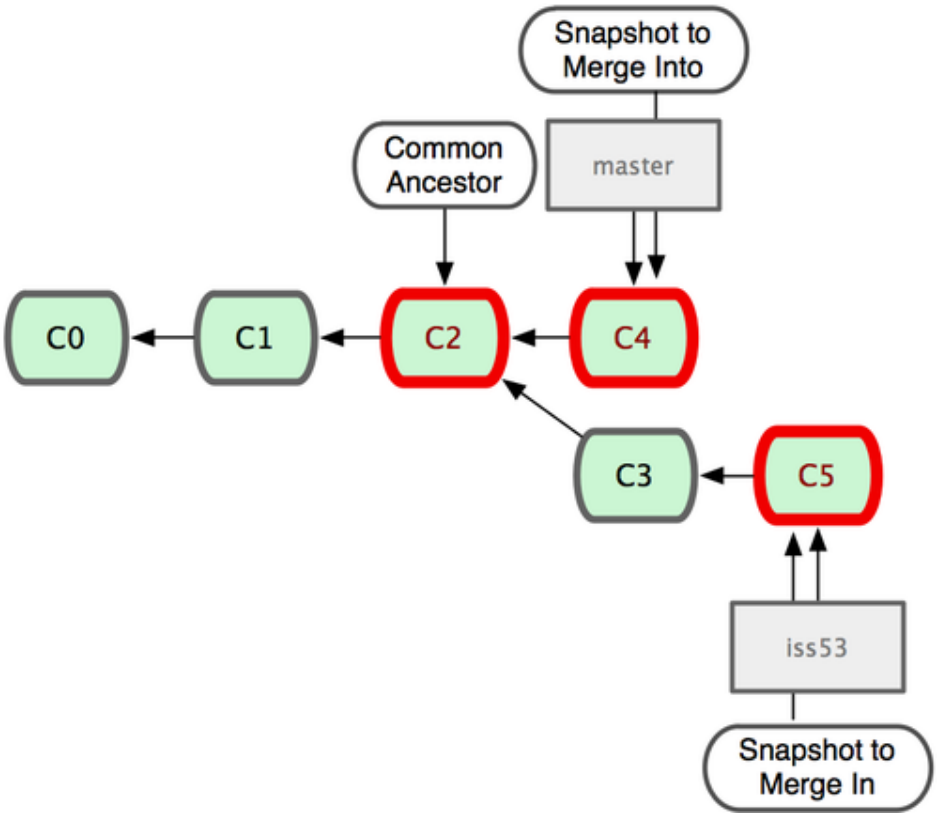

master
98ca9
34ac2
f30ab
testing
HEAD

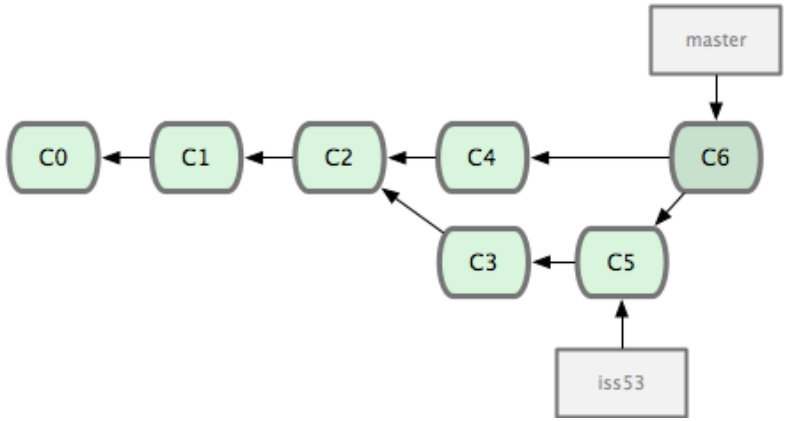
分支模型- 快进(Fast forward)合并





分支模型- 三方合并



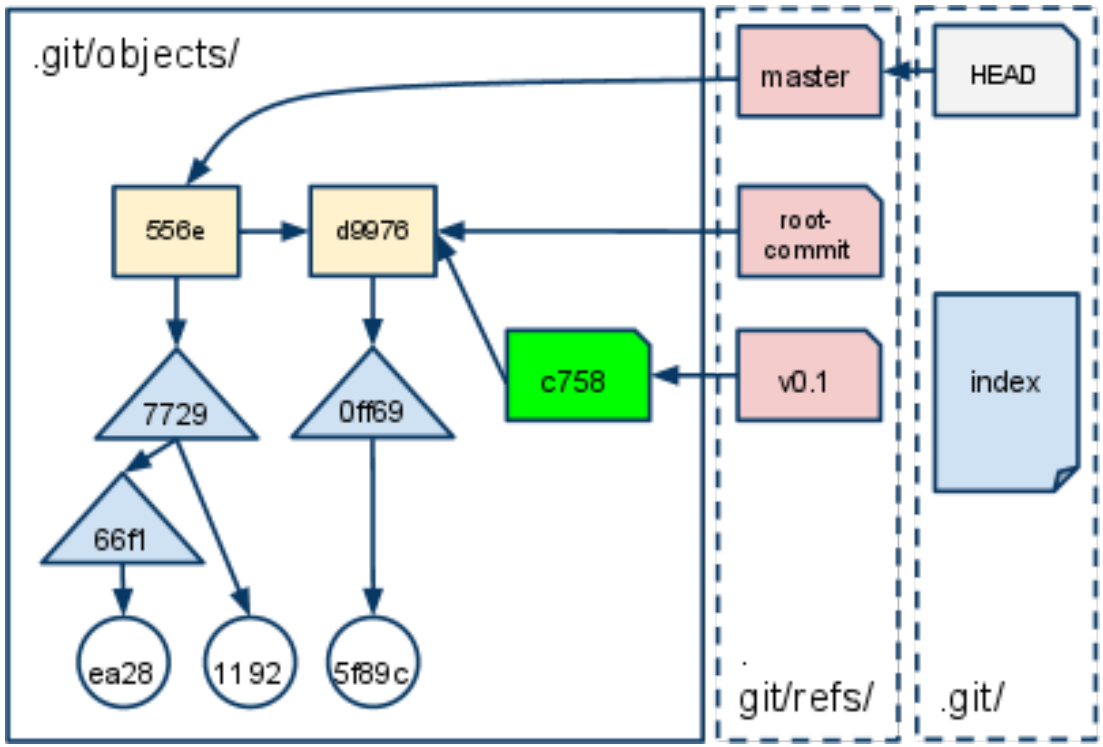


合并冲突

如果不同的分支中都修改了同一个文件的同一部分，Git自动合并就会失败，需要你进行人工合并。要看看哪些文件在合并时发生冲突，可以用git status查阅。在手动编辑解决了所有文件里的所有冲突后，运行git add将把它们标记为已解决状态。

标签

- 标签(tag)与分支引用类似
- 标签分为两种：轻量级标签与含附注的标签
- 轻量级标签其实就是一个指向指定commit的指针
- 含附注的标签会有一个额外的标签对象，对象中包含“打标签的人”、“日期”、“附注”以及“指向commit的SHA-1值”



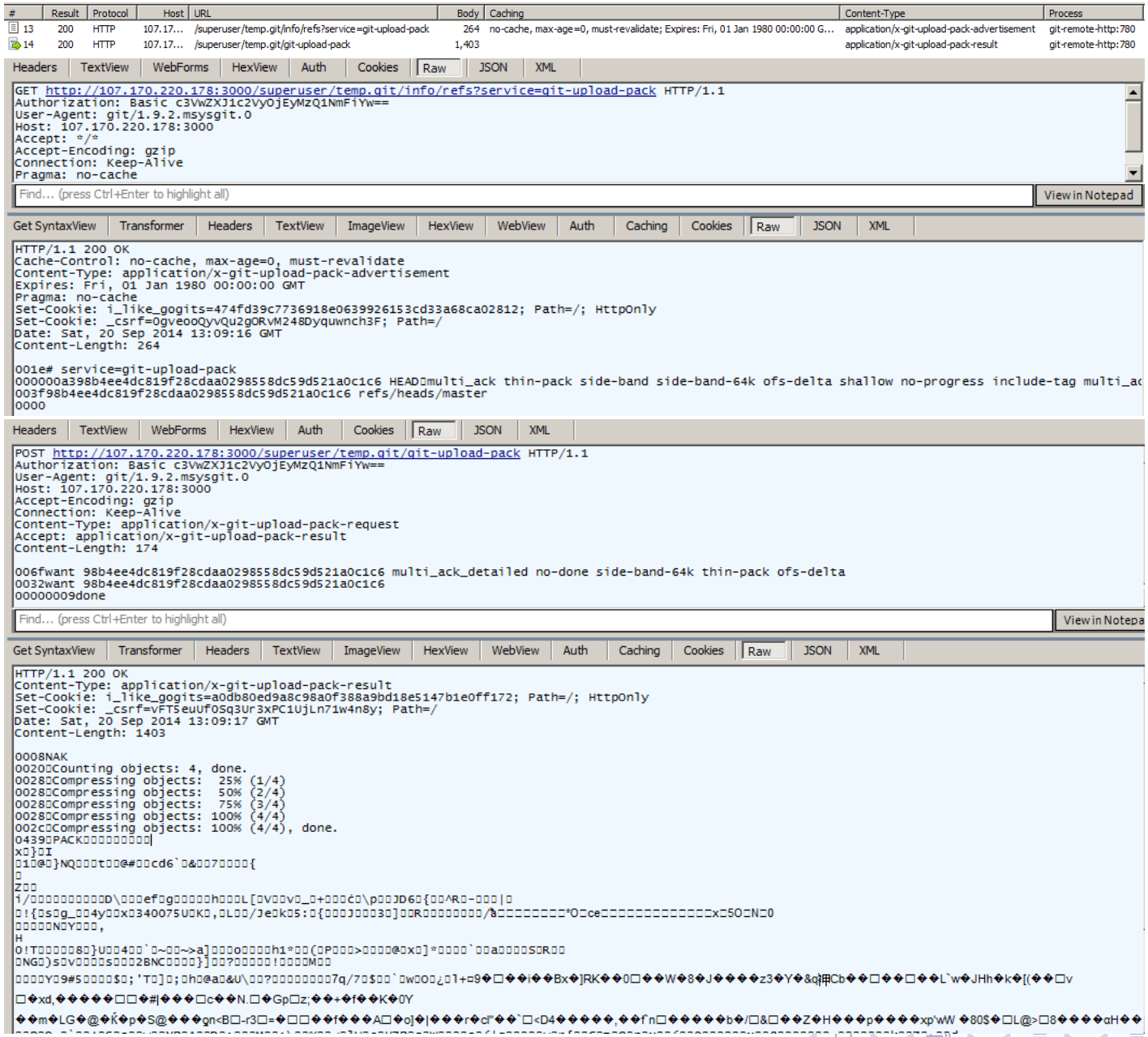
Git代码库目录结构

.git目录即代码仓库，包含了所有数据。

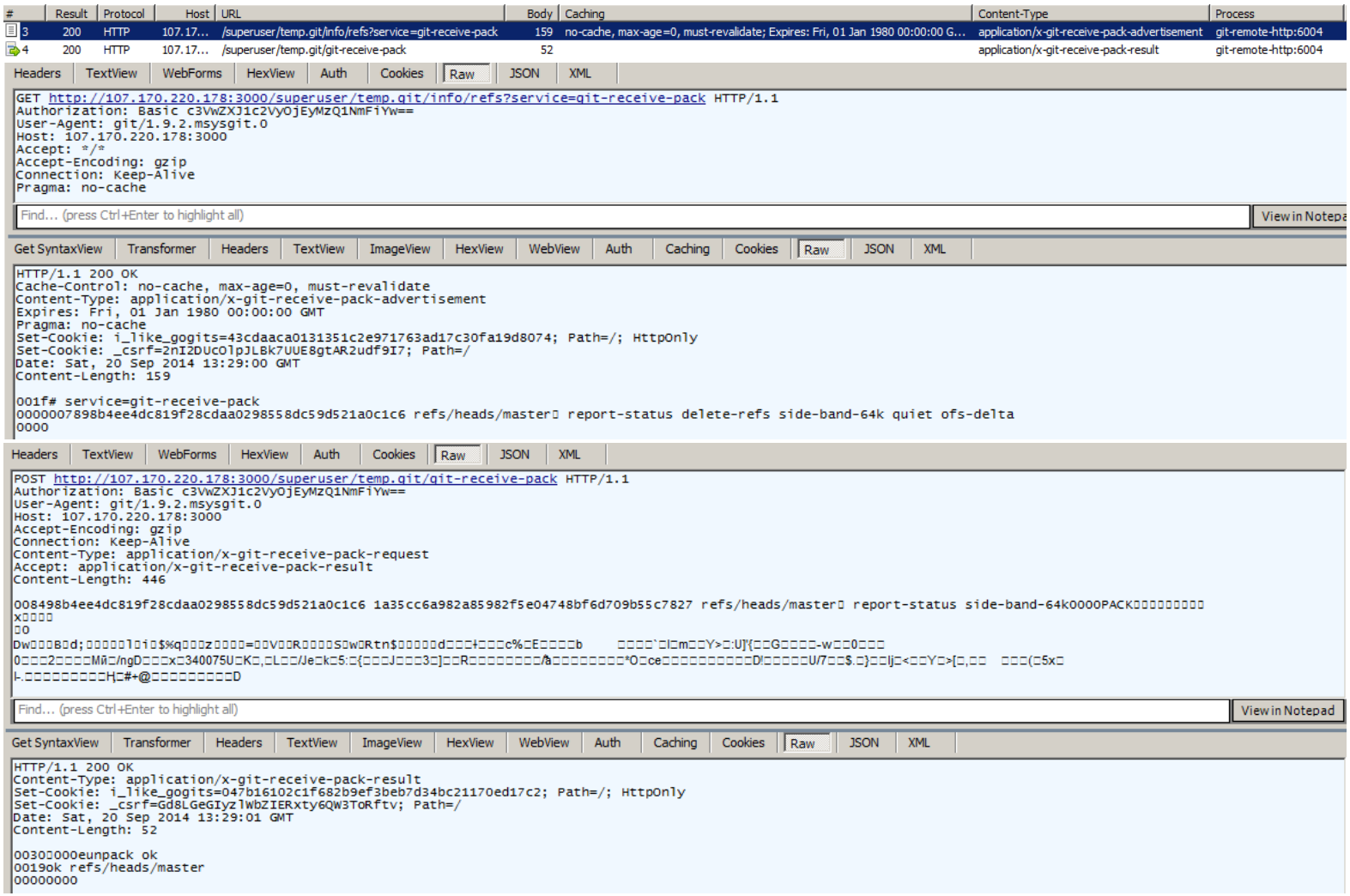
```
.git
|-- branches
|-- COMMIT_EDITMSG
|-- config
|-- description
|-- HEAD
|-- hooks
|   |-- applypatch-msg.sample
|   |-- commit-msg.sample
|   |-- post-update.sample
|   |-- pre-applypatch.sample
|   |-- pre-commit.sample
|   |-- prepare-commit-msg.sample
|   |-- pre-push.sample
|   |-- pre-rebase.sample
|   `-- update.sample
|-- index
|-- info
|   `-- exclude
|-- logs
|   |-- HEAD
|   `-- refs
|       |-- heads
|       |   `-- master
|       `-- remotes
|           `-- origin
|               `-- HEAD
|-- objects
|   |-- 54
|   |   `-- d2b7c108224090be63aa80ed6daebb689d9b06
|   |-- 5a
|   |   `-- d28e22767f979da2c198dc6c1003b25964e3da
|   |-- bb
|   |   `-- d8ec4f286b52e7402b79c2ac27b34e11bfc3fa
|   |-- fc
|   |   `-- 59679f6b6dced71b8ff102de09adcbf765a157
|   |-- info
|   `-- pack
|       |-- pack-430a2e9369984c284a6dff2e0ed8f1b813d1734f.idx
|       `-- pack-430a2e9369984c284a6dff2e0ed8f1b813d1734f.pack
```

```
|-- packed-refs
`-- refs
    |-- heads
    |   `-- master
    |-- remotes
    |   `-- origin
    |       `-- HEAD
    `-- tags
```

与远程服务器交互- git clone



与远程服务器交互- git push



#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process
3	200	HTTP	107.17...	/superuser/temp.git/info/refs?service=git-receive-pack	159	no-cache, max-age=0, must-revalidate; Expires: Fri, 01 Jan 1980 00:00:00 G...	application/x-git-receive-pack-advertisement	git-remote-http:6004
4	200	HTTP	107.17...	/superuser/temp.git/git-receive-pack	52		application/x-git-receive-pack-result	git-remote-http:6004

Headers | TextView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML

```
GET http://107.170.220.178:3000/superuser/temp.git/info/refs?service=git-receive-pack HTTP/1.1
Authorization: Basic c3VwZXJ1c2VyOjEyMzQ1NmFiYw==
User-Agent: git/1.9.2.msysgit.0
Host: 107.170.220.178:3000
Accept: */*
Accept-Encoding: gzip
Connection: Keep-Alive
Pragma: no-cache
```

Find... (press Ctrl+Enter to highlight all) | View in Notepa

Get SyntaxView | Transformer | Headers | TextView | ImageView | HexView | WebView | Auth | Caching | Cookies | Raw | JSON | XML

```
HTTP/1.1 200 OK
Cache-Control: no-cache, max-age=0, must-revalidate
Content-Type: application/x-git-receive-pack-advertisement
Expires: Fri, 01 Jan 1980 00:00:00 GMT
Pragma: no-cache
Set-Cookie: i_like_gogits=43cdaaca0131351c2e971763ad17c30fa19d8074; Path=/; HttpOnly
Set-Cookie: _csrf=2nI2DUcOlpJL8k7UUE8gtAR2udf9I7; Path=/
Date: Sat, 20 Sep 2014 13:29:00 GMT
Content-Length: 159

001f# service=git-receive-pack
0000007898b4ee4dc819f28cdaa0298558dc59d521a0c1c6 refs/heads/master□ report-status delete-refs side-band-64k quiet ofs-delta
0000
```

Headers | TextView | WebForms | HexView | Auth | Cookies | Raw | JSON | XML

```
POST http://107.170.220.178:3000/superuser/temp.git/git-receive-pack HTTP/1.1
Authorization: Basic c3VwZXJ1c2VyOjEyMzQ1NmFiYw==
User-Agent: git/1.9.2.msysgit.0
Host: 107.170.220.178:3000
Accept-Encoding: gzip
Connection: Keep-Alive
Content-Type: application/x-git-receive-pack-request
Accept: application/x-git-receive-pack-result
Content-Length: 446

008498b4ee4dc819f28cdaa0298558dc59d521a0c1c6 1a35cc6a982a85982f5e04748bf6d709b55c7827 refs/heads/master□ report-status side-band-64k0000PACK□□□□□□□□□
x□□□□
□O
Dw□□□B□d;□□□□□l□i□$%q□□□z□□□□=□□V□□R□□□□S□w□Rtn$□□□□□d□□□+□□□c%□E□□□□b      □□□□`□I□m□□Y>□:U]'{□□G□□□□-w□□0□□□
0□□□2□□□□Mü□/ngD□□□x□340075U□K□,□L□□/Je□k□5:□{□□□J□□□3□]□□R□□□□□□□□/à□□□□□□□□*O□ce□□□□□□□□□□□D/□□□□□U/7□□$.□}□□ij□<□□Y□>[□,□□  □□□(□5x□
⊢.□□□□□□□□□H¦□#+@□□□□□□□□□D
```

Find... (press Ctrl+Enter to highlight all) | View in Notepad

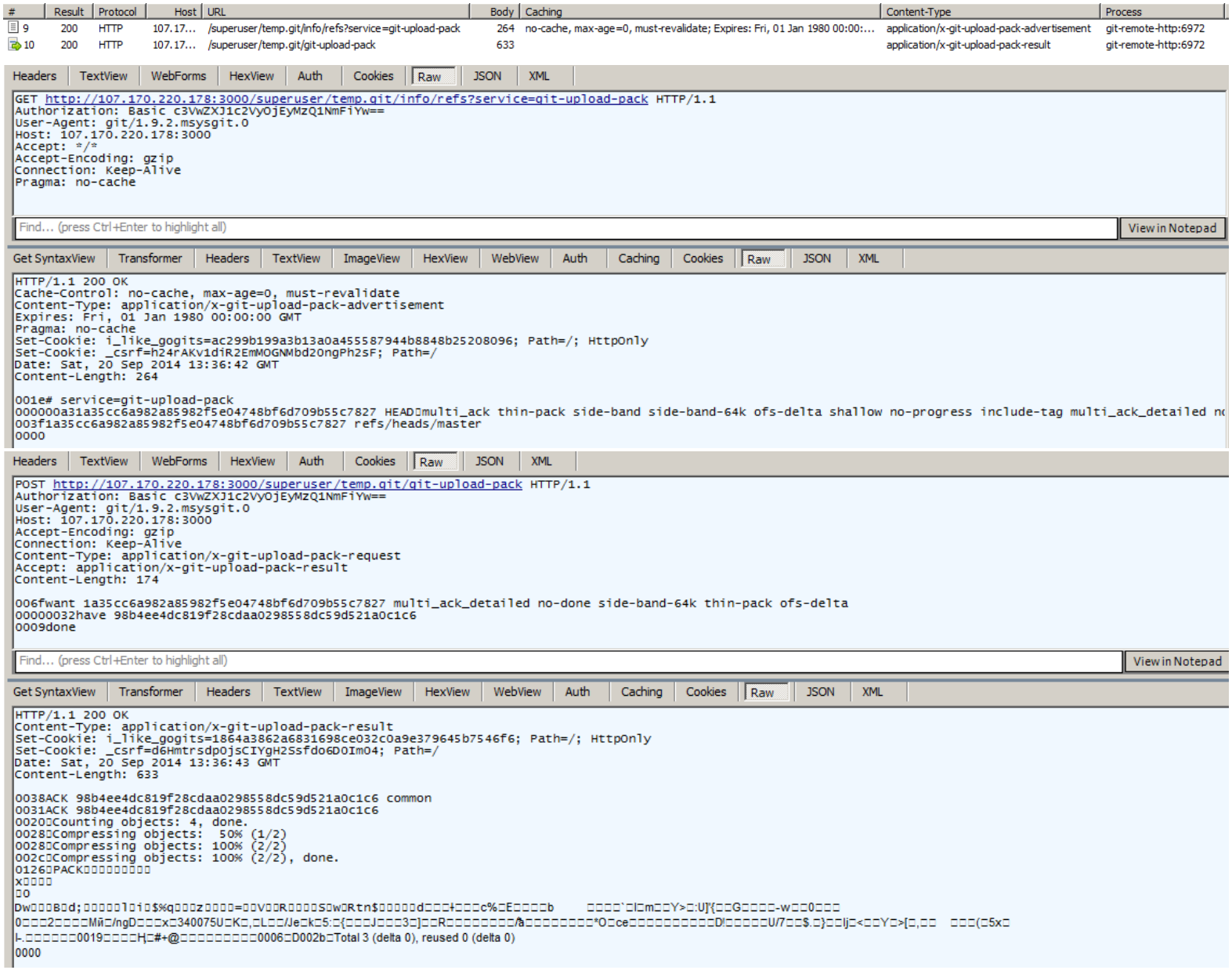
Get SyntaxView | Transformer | Headers | TextView | ImageView | HexView | WebView | Auth | Caching | Cookies | Raw | JSON | XML

```
HTTP/1.1 200 OK
Content-Type: application/x-git-receive-pack-result
Set-Cookie: i_like_gogits=047b16102c1f682b9ef3beb7d34bc21170ed17c2; Path=/; HttpOnly
Set-Cookie: _csrf=Gd8LGeGIyzlWbZIERxty6QW3ToRftv; Path=/
Date: Sat, 20 Sep 2014 13:29:01 GMT
Content-Length: 52

0030□000eunpack ok
0019ok refs/heads/master
00000000
```

git push可能会失败？谁来判定是否失败?

与远程服务器交互- git fetch



1 简介

2 实践(上)

3 原理

4 实践(下)

5 结束

配置

配置文件

- Git的配置项有三个级别：系统级、用户级、项目级。同一个配置项，后一级覆盖前一级。对应配置文件有三处：
  - Linux上，/etc/gitconfig，$HOME/.gitconfig，.git/config
  - Windows上，Git安装目录/etc/gitconfig，$HOME/.gitconfig，.git/config
- Git配置修改有两种方式：
  - 直接修改配置文件
  - 使用git config命令
- git config的两个选项–system、–global对应“系统级”和“用户级”，不提供这两个选项则默认为项目级。
- 查看配置信息：git config –list，git config user.name

配置- 常用配置项

- git config –global user.name "John Doe"
- git config –global user.email johndoe@example.com
- git config –global core.editor d:/softwares/Vim/vim74/gvim.exe
- git config –global color.ui true // 为Git命令输出使用默认着色方案
- git config –global http.proxy 127.0.0.1:8087 // HTTP代理
- git config –global https.proxy 127.0.0.1:8087 // HTTPS代理
- git config –global http.sslVerify false
- git config –global http.postBuffer 524288000

.gitignore

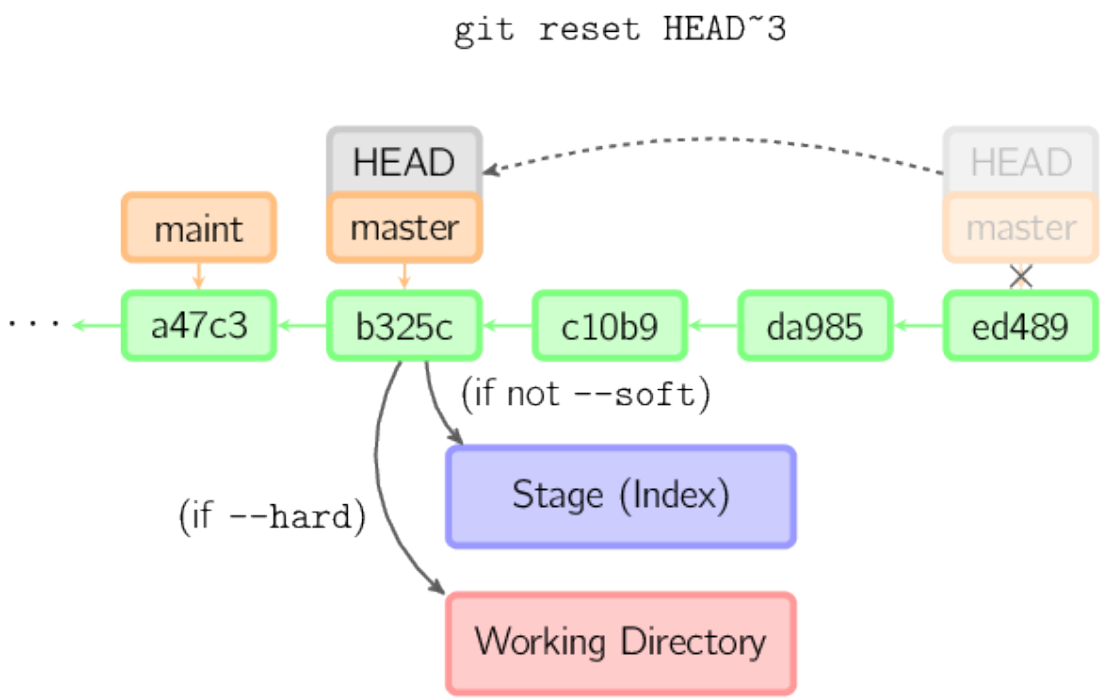
格式规范

- 所有空行或者以注释符号#开头的行都会被Git忽略
- 可以使用标准的glob模式匹配（所谓的glob模式是指shell所使用的简化了的正则表达式。星号(*)匹配零个或多个任意字符；[*abc*]匹配任何一个列在方括号中的字符问号(?)只匹配一个任意字符；如果在方括号中使用短划线分隔两个字符，表示所有在这两个字符范围内的都可以匹配(比如[0 – 9]表示匹配所有0到9的数字)）
- 匹配模式最后跟反斜杠(/)说明要忽略的是目录
- 要忽略指定模式以外的文件或目录，可以在模式前加上惊叹号(!)取反

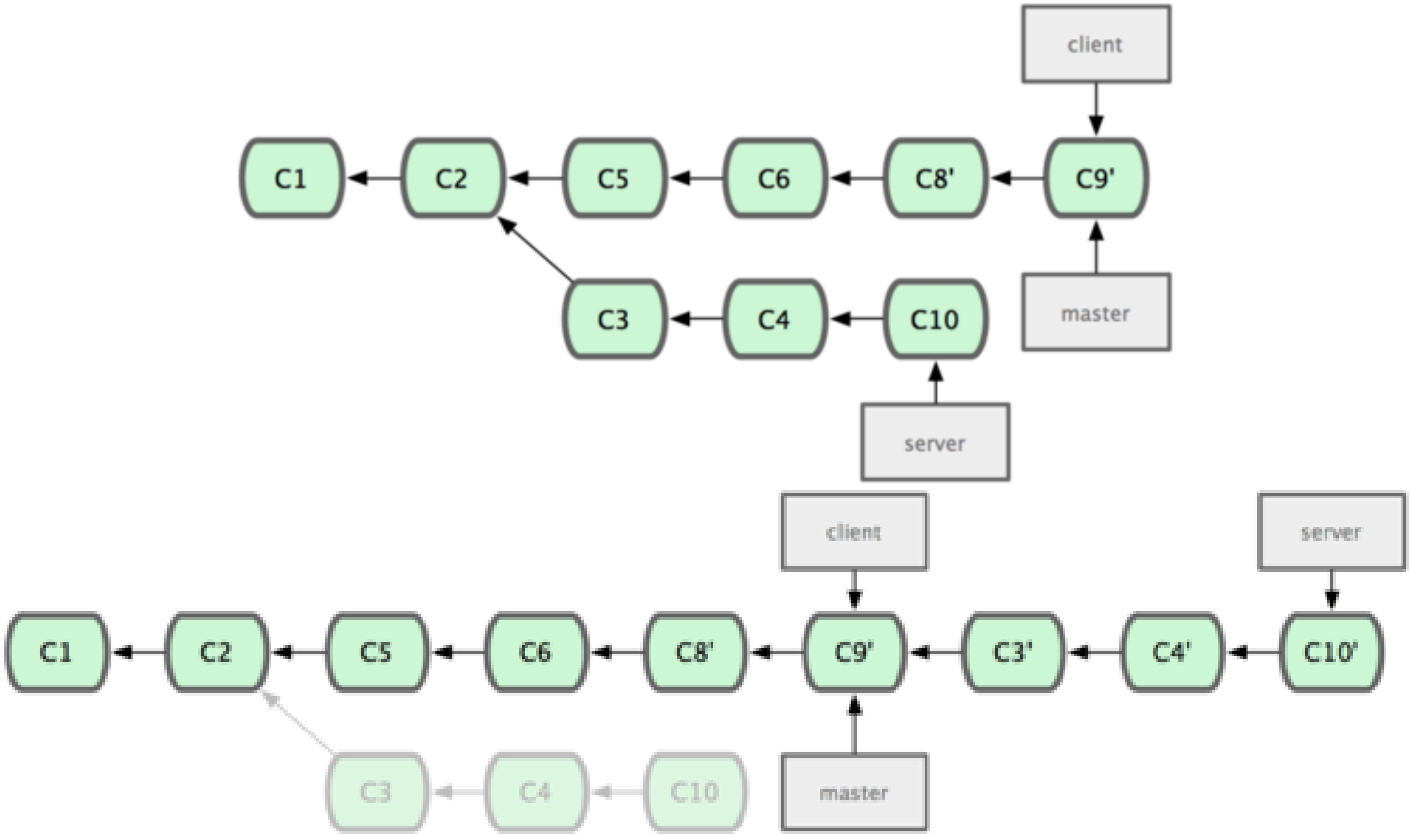
注意：已加入版本控制的文件、目录无法被ignore掉
.gitignore模板集合：https://github.com/github/gitignore

撤销操作

- git reset – files 用来撤销最后一次git add files，你也可以用git reset撤销所有暂存区域文件
- git checkout – files 把文件从暂存区域复制到工作目录，用来丢弃本地修改



分支衍合(rebase)



1. git checkout server
2. git rebase master
3. git checkout master
4. git merge server

衍合的优点？问题？

Stashing

当你正在进行项目中某一部分的工作，里面的东西处于一个比较杂乱的状态，而你想转到其他分支上进行一些工作。问题是，你不想提交进行了一半的工作，否则以后你无法回到这个工作点。解决这个问题的办法就是**git stash**命令。

- git stash /* 对为加入版本控制的文件不起作用，因为分支切换不会影响到这些文件*/
- git stash list
- git stash apply

高级话题

- 子模块：当你在一个项目上工作时，你需要在其中使用另外一个项目。也许它是一个第三方开发的库或者是你独立开发和并在多个父项目中使用的。这个场景下一个常见的问题产生了：你想将两个项目单独处理但是又需要在其中一个中使用另外一个。
- 重写历史
- git svn
- hook

1 简介

2 实践(上)

3 原理

4 实践(下)

5 结束

Git图形化工具推荐

- Git Extensions
- Github for Windows
- SourceTree

参考资料&推荐阅读

- Pro Git
- 通过示例学习Git内部构造（译）

谢谢！